UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UM COMPONENTE PARA TRANSFERÊNCIA DE ARQUIVOS EM AMBIENTES DE ALTO DESEMPENHO HETEROGÊNEOS

IZAIAS DE FARIA

FLORIANÓPOLIS

2011

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

IZAIAS DE FARIA

DESENVOLVIMENTO DE UM COMPONENTE PARA TRANSFERÊNCIA DE ARQUIVOS EM AMBIENTES DE ALTO DESEMPENHO HETEROGÊNEOS

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Mário Antônio Ribeiro Dantas

FLORIANÓPOLIS

2011

IZAIAS DE FARIA

DESENVOLVIMENTO DE UM COMPONENTE PARA TRANSFERÊNCIA DE ARQUIVOS EM AMBIENTES DE ALTO DESEMPENHO HETEROGÊNEOS

Trabalho de conclusão de curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Mário Antônio Ribeiro Dantas

Banca examinadora

_____

Prof. Dr. Mário Antônio Ribeiro Dantas

_____

Prof. Dr. Carina Friedrich Dorneles

_____

Prof. Dr. Lúcia Helena Martins Pacheco

“Dedico este trabalho a minha família que me apoiou e incentivou incondicionalmente ao longo de todo o curso na busca incessante pelo conhecimento”.

RESUMO

Desde o final da década de 90, temos observado um grande avanço na tecnologia de Computação de Grade ou *Grid Computing*. Atualmente, diferentes tipos de grades vêm sido utilizadas em ambientes onde é necessário grande capacidade de processamento e armazenamento de dados. Neste contexto, foram observados diferentes gerenciadores de grades que utilizam diferentes protocolos de comunicação. Independente do gerenciador se faz necessário, na maioria dos casos, o envio de arquivos para a grade. Para este processo, o grande número de diferentes protocolos em utilização, em especial os privados, representa um impasse a transferência de dados. O objetivo deste trabalho é desenvolver um componente capaz de se comunicar com diferentes gerenciadores, tornando possível a transferência de arquivos independendo dos protocolos utilizados. O Resultado deste trabalho é um componente capaz de se comunicar e transferir arquivos para diferentes ambientes grades. Por fim, para validação deste componente, serão realizados testes de transferência e considerado seu desempenho.

Palavras-chave: *Grid computing***.**

Lista de Figuras

Lista de Abreviaturas

LAN : Local Area Network

SMP : Symmetric MultiProcessing

PC : Personal COmputer

PDA : Personal digital assistant

SSI : Single System Image

FTP : File Transfer Protocol

API : Application Program Interface

OV : Organização Virtual

HAS : Hardware as a service

SAS : Software as a service

DAS : Data as a service

PAS : Plataform as a service

QoS : Quality of Service

CPU : Central Processing Unit

SOA : Service Oriented Architecture

E/S : Entrada/Saída

SRB : Storage Resource Broker

HBDP : High Bandwidth Delay Product

BDP : Bandwidth Delay Product

TCP : Transmission Control Protocol

UDP : User Datagram Protocol

API : Application Programming Interface

ANL: Argnone National Laboratory

RSL: Resource Specification Language

GRIS: Grid Resource Information Service

GIIS: Grid Index Information Service

CA: Certificate Authority

SSL: Secure Socket Layer4

Sumário

1 Introdução

Sistemas de grade científica são ferramentas importantes para pesquisadores analisarem, armazenarem e compartilharem dados. A maioria destes sistemas fornece não apenas os serviços computacionais, mas também serviços de dados. Entre os sistemas existentes de grade científica, muitos dos serviços computacionais são construídos em Globus Toolkit (FOSTER, I., 2003b), ou suas variantes, tais como gLite (CERN, 2011b). Alguns outros grupos adotam Condor (2011), um sistema de agendamento diferente, que pode funcionar com Globus via Condor-G. No entanto, o mecanismo usado para armazenamento de dados e acesso não é único.

Alguns sistemas de dados da rede são projetados para serem sistemas de arquivos virtuais, oferecendo espaço de armazenamento a pesquisadores sem que usuários saibam o exato lugar de dados, como SRB (BARU, C., 1998) e seu sucessor, iRODS (RAJASEKAR, A., 2006), bem como dCache (ERNST, M., 2001). A maneira de acessar esses sistemas de grades de dados varia. Por exemplo, dCache fornece uma interface GridFTP e uma interface HTTP, de modo que sistemas de submissão a grade possam facilmente acessar dados a partir dele. Alguns outros, como iRODS, implementam os seus próprios protocolos proprietários e sistemas de envio de trabalhos a grade não têm como obter dados diretamente fora dele.

Como a quantidade de dados está aumentando, torna-se um requisito comum transferir dados entre fontes de dados heterogêneas. Especialmente quando um novo protocolo é desenvolvido, existe um trabalho considerável a ser feito para integrá-lo no sistema existente. Normalmente, os usuários necessitam usar um cliente para cópia de dados da fonte para um espaço intermediário, como um espaço de armazenamento local, e então utilizar outro cliente para mover dados a partir do espaço intermediário para o destino. Este é um método errado e ineficiente. Por existir uma etapa extra, o usuário deve se sentar frente a um terminal ou escrever um script para automatizar todo o processo. Se algo inesperado ocorrer, o usuário necessita realizar depuração do trabalho para descobrir o erro.

Nesse contexto temos um cenário onde é necessário cada vez mais que a grade esteja preparada para transferir arquivos potencialmente grandes para uma grande possibilidade de clientes, seja utilizando um cliente com protocolo

proprietário, ou no pior dos casos, para outra grade com gerenciador diferente no utilizado na fonte. O segundo caso representa um grande problema para a comunidade desenvolvedora que alimenta a transferência de arquivos entre grades computacionais, uma vez que as grades são dependentes fixamente dos protocolos utilizados para seu desenvolvimento.

O trabalho proposto envolve a pesquisa e desenvolvimento de um componente para transferência de arquivos em arquiteturas de grades computacionais heterogêneas. Atualmente as ferramentas que auxiliam a transferência de arquivos a grades são dependentes dos gerenciadores de recursos do mesmo. Neste trabalho pretende-se desenvolver um componente capaz de adaptar-se a diferentes tipos de fontes de dados, para tal será utilizado um protocolo padrão para comunicação com grades, o GridFTP. Após desenvolvimento do componente o mesmo foi testado e seu desempenho avaliado comprovando sua viabilidade com relação a alternativas utilizadas atualmente no mercado.

1.2 Objetivos

Objetivos Gerais

O objetivo geral desse trabalho será a pesquisa e posterior desenvolvimento de um componente para transferência de arquivos em grades computacionais heterogêneos. Para isso o componente deve ser capaz de se comunicar com diferentes gerenciadores de recursos e ainda identificar o sistema de arquivos em utilização na grade computacional.

Objetivos específicos

- Levantamento do estado da arte em transferência de arquivos em ambientes de alto desempenho heterogêneos;
- Adaptação de um componente para transferência de arquivos em ambientes de alto desempenho heterogêneos;

- Testes de desempenho do componente desenvolvido, comparando com outras tecnologias existentes.

1.3 Estrutura do Trabalho

Este trabalho está dividido na seguinte forma. O Capítulo 2 trata da fundamentação teórica do trabalho abrangendo a conceituação de computação distribuída e das diversas configurações encontradas, tais como Clusters, Multi-Clusters, Grids e Clouds. O Capítulo 3 aborda o estado da arte do cenário de transferência de arquivos nos ambientes identificados no Capítulo 2.

No Capítulo 4 é apresentada a proposta de trabalho, ou seja, o componente desenvolvido, a configuração física, e a aplicação executada. O Capítulo 5 serve como comparativo dos resultados das aplicações dos testes explicados no capítulo anterior. Por fim, o Capítulo 6 trata das conclusões após observação dos resultados e propõe trabalhos futuros que poderão ser desenvolvidos a partir deste.

2 Sistemas Distribuídos

2.1 Computação Distribuída

Sistemas computacionais distribuídos surgiram em meados da década de 80, com o desenvolvimento de microprocessadores e redes locais de alta velocidade ou LANs (Local Area Networks), ganhando popularização por sua vantagem em custo/benefício com relação a seus antecessores, os mainframes, máquinas muito grandes e caras.

Segundo Coulouris (2005), um sistema distribuído é aquele no qual seus componentes de hardware ou software, localizados em computadores interligados por uma rede de interconexão, se comunicam e coordenam suas ações por meio de troca de mensagens. Outra definição para o termo é um conjunto de máquinas sem memória compartilhada (SMP), ou seja, independentes, que se apresentam a seus usuários como um sistema único e coerente (TANENBAUM, 2007).

Assim sendo, podemos destacar que um sistema distribuído consiste em um conjunto de recursos computacionais que podem estar espalhados geograficamente, seja em países diferentes, no mesmo prédio ou na mesma sala, e realizam troca de mensagens para comunicação e definição de ações, e ainda se apresentam ao usuário com uma única imagem, ilustrado na figura 2.1.

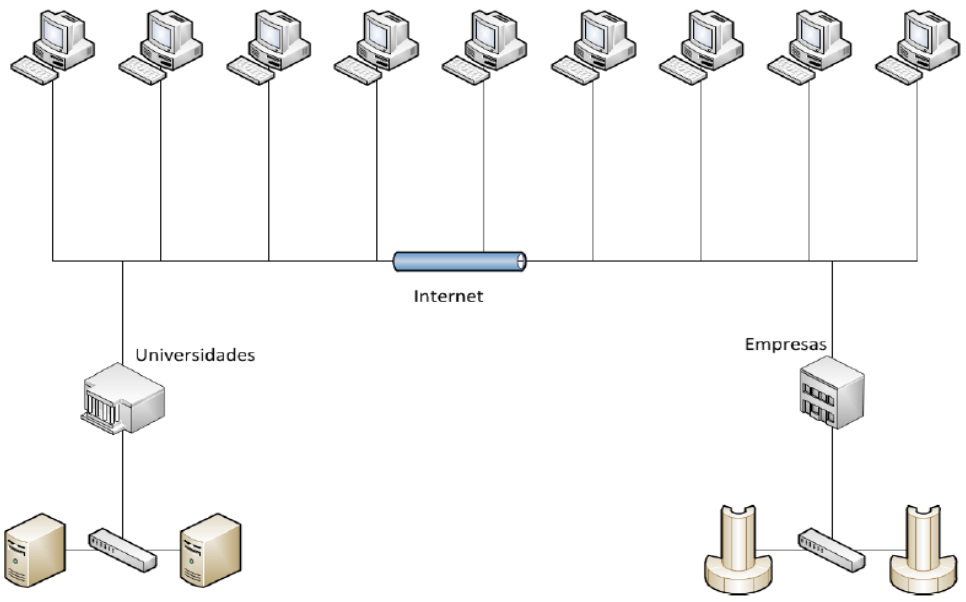

FIGURA 2.1: Sistema distribuído

Tanenbaum (1992) define alguns aspectos que necessitam ser tratados em um projeto de sistema distribuído, são eles:

- ***Transparência***: trata da obtenção da imagem única do sistema; um sistema distribuído pode ser transparente **quanto à localização**, à **migração**, à **replicação**, à **concorrência** e ao **paralelismo**;
- ***Flexibilidade***: com vários processadores trabalhando em conjunto devem existir mecanismos para permitir adaptações dinâmicas durante a execução do sistema. Podem acontecer situações não previstas e deve ser possível contornar isso de forma simples; além disso, deve ser fácil incluir ou excluir processadores do sistema sem que para isso seja necessário modificar todos os componentes.
- ***Confiabilidade:*** a ideia básica por trás de confiabilidade é o fato de que caso algumas máquinas do sistema saiam do ar, as demais sustentariam o sistema; neste caso podemos destacar ainda outros aspectos tais como: **disponibilidade**, **segurança** e **tolerância a falhas**;
- ***Desempenho*****:** várias métricas podem ser utilizadas para avaliar o desempenho do sistema, tais como **tempo de resposta** e **throughput** (número de jobs por hora). A utilização do sistema e a quantidade de recursos consumidos da rede também são utilizadas na medida da performance. O problema da performance é também muito influenciado pela comunicação; no caso de sistemas distribuídos, isso ocorre por meio de troca de mensagens, problema este não presente em sistemas com um só processador. Outro aspecto diz respeito a **granularidade** do processamento;
- ***Escalabilidade*****:** quando se trabalhando com sistemas distribuídos é necessário considerar gargalos potenciais em sistemas muito grandes, que podem comprometer a escalabilidade do mesmo são estes: componentes centralizados, tabelas centralizadas e algoritmos centralizados.

Considerando as vantagens que sistemas distribuídos oferecem, existe um paradigma voltado para o alto desempenho, denominado de computação distribuída de alto desempenho. Clusters, Multi-Clusters, Grids e Clouds são atualmente, os representantes, sob o aspecto de arquitetura, dessa categoria computacional. Essas configurações podem ter como característica uma grande heterogeneidade com relação aos recursos nelas disponibilizados. Podem ser citados como exemplos: PC’s, notebooks, supercomputadores, PDA’s, celulares, etc.

2.2 Configurações distribuídas

2.2.1 Cluster

Um cluster, também conhecido por agregado computacional, é uma coleção de estações de trabalho ou computadores pessoais que estão interconectados via alguma tecnologia de rede. Para propósitos de computação paralela, um cluster irá geralmente ser constituído por estações de trabalho de alta performance ou computadores convencionais conectados a uma rede de alta velocidade. Um cluster trabalha como uma coleção integrada de recursos e possui uma imagem única do sistema (SSI) abrangendo todo o sistema voltado a um domínio específico (BUYYA, 2003). Observando que um nó do cluster pode ser composto por uma máquina com um único processador, ou por um sistema multi-processado. Desta forma, é possível alcançar processamento semelhante aos super-computadores, contudo, por um custo mais baixo.

Uma característica importante para a formação das configurações dos clusters é a forma de interligação física e utilização dos diversos computadores através de um dispositivo de interconexão de rede. Conforme observado por Dantas (2005) os agregados computacionais podem ser interligados de duas maneiras, de forma não-dedicada ou dedicada.

Configurações não-dedicadas normalmente se referem a ambientes em que o equipamento de rede local (hub ou switch) representa o dispositivo de interconexão da configuração. Neste tipo de configuração não existe apriori a figura de um pacote central que faça o controle lógico do ambiente, ou seja, embora exista a conectividade dos computadores através da rede, não existe a interoperabilidade dos recursos e aplicações do ambiente. Para que o agregado seja criado é preciso que existam pacotes de software instalados em todos os computadores constituintes do cluster. Para a formação de clusters nesse tipo de ambiente normalmente são utilizados pacotes de software semelhantes ao Condor (2011) ou OpenMosix (2011). A Figura 2.2 exemplifica um cluster não-dedicado dentro de uma rede local.

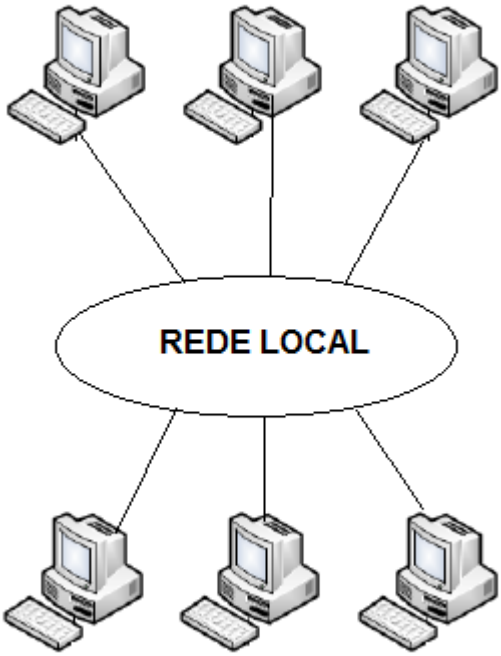

FIGURA 2.2: Ambiente de cluster não-dedicado

Apesar de as configurações não-dedicadas representarem um excelente custo-desempenho, uma vez que é possível aproveitar recursos ociosos já disponíveis, as **configurações dedicadas** podem ser consideradas como opção mais recomendada para execução de aplicações críticas de uma organização. A falta de uma política única e bem definida acaba por demandar enorme esforço do software de middleware utilizado para descoberta de características de dispositivos e comportamentos funcionais. A Figura 2.3 exemplifica um cluster dedicado dentro de uma rede corporativa.

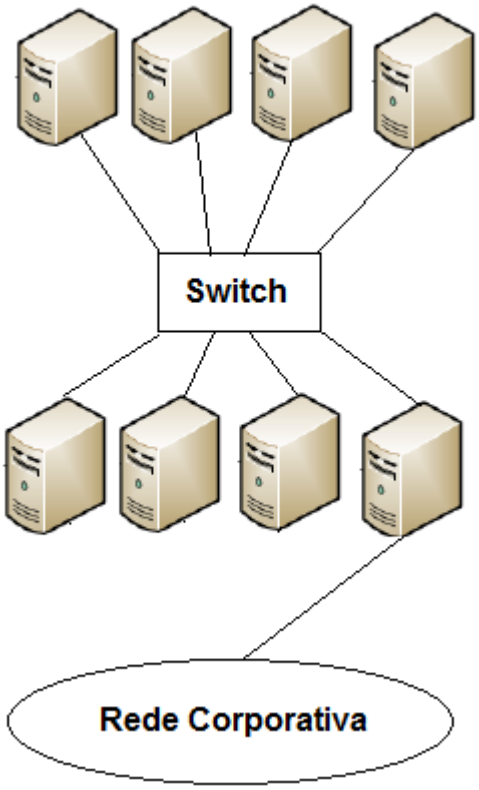

FIGURA 2.3: Ambiente de cluster dedicado ligado direto à rede corporativa

Com um estabelecimento de políticas de uso, o pacote de middleware da configuração pode:

- efetuar uma melhor escolha na distribuição das aplicações para os processadores disponíveis, uma vez que se conhece como a configuração deve funcionar e quais aplicativos devem ter prioridade;
- potencializar a função de alto desempenho do ambiente, uma vez que é possível definir quais nodos devem ter outro computador como espelho (imagem);
- a escalabilidade do ambiente é maior, pois é possível a criação de perfis típicos de nós escravos. Estes perfis podem ser aproveitados para novos nodos inseridos no cluster, o que gera um ganho de tempo;
- o pacote de imagem única, que possibilita a criação de um elemento centralizador para efetuar cópias de segurança de processos submetidos aos processadores e ainda cópias de backups;

Segundo Pereira (2002), o objetivo desejado em um cluster resume-se em:

- ***Alta Disponibilidade***, quando se deseja que o cluster forneça determinados serviços, sendo que estes devem estar sempre (ou quase sempre) disponíveis para receber solicitações. Esta probabilidade do serviço estar apto a receber solicitações é um fator dependente do cluster.
- ***Alto Processamento***, quando se deseja que o cluster execute determinadas tarefas, sendo que elas são divididas (na sua íntegra ou em frações de uma mesma tarefa) e processadas separadamente em vários nós, a fim de a velocidade de processamento seja incrementada.

É possível também unir as duas características acima em um único Cluster formando um terceiro tipo híbrido, aumentando assim a disponibilidade e escalabilidade de serviços e recursos. Este tipo de configuração de cluster é bastante utilizado em servidores de *web, mail, news ou ftp*.

Conforme visto nesta seção, clusters referem-se a ambientes de um único domínio. Quando necessário agregar clusters de domínios diferentes, utiliza-se o conceito de multi-cluster.

2.2.2 Multi-Cluster

Uma configuração multi-cluster permite o compartilhamento de recursos em diferentes domínios, conectando múltiplos ambientes de cluster geograficamente dispersos. Esta configuração forma um sistema integrado de imagem única, ou seja, o usuário tem uma visão global dos recursos e os utiliza como se estivessem em uma única máquina, independente de onde estão fisicamente associados (YEO et al. 2006).

Associar clusters de diferentes domínios permite um melhor aproveitamento dos recursos do sistema como um todo. Isto porque máquinas ociosas de um domínio podem ser utilizadas por usuários de diferentes domínios, e não só por usuários do domínio local. Desta forma, usuários que ficariam em fila em seus domínios, aproveitam os recursos de outros domínios e, além disso, usuários que precisam de recursos específicos obtém uma maior quantidade de opções dentro do

sistema. Entretanto, a administração destes recursos em um ambiente multi-cluster é limitada a um único tipo de gerenciador, ou seja, o gerenciador de cada cluster local deve ser o mesmo para todos os domínios do sistema. A Figura 2.4 mostra um exemplo de ambiente Multi-cluster.

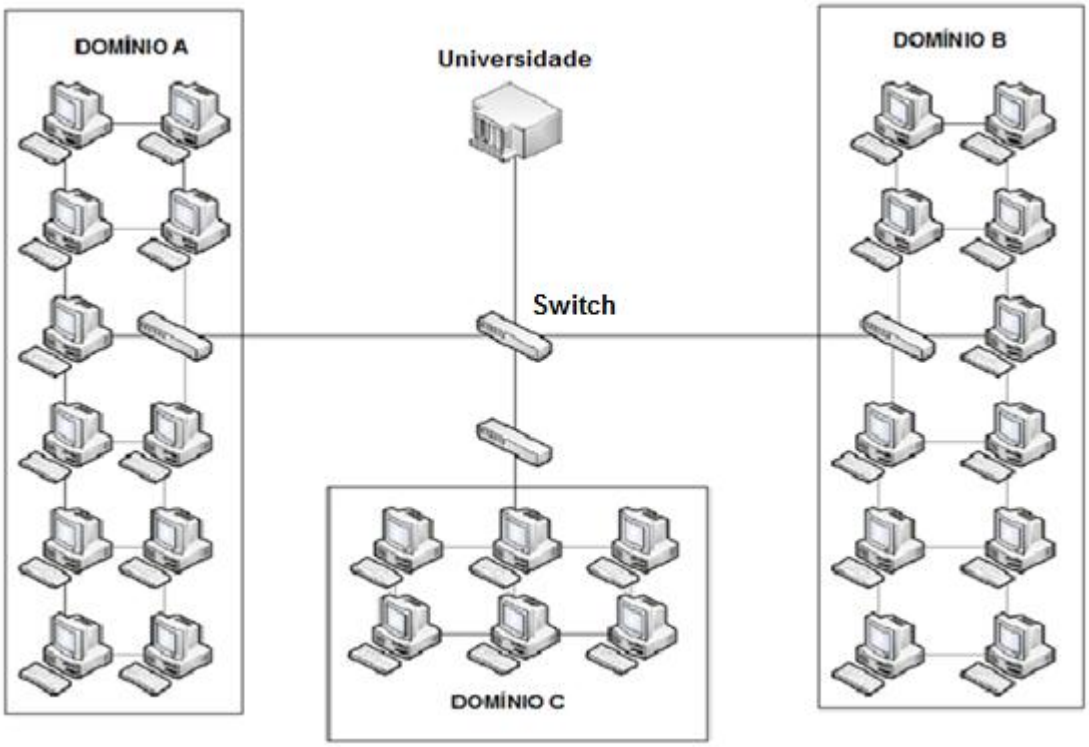

FIGURA 2.4: Exemplo de ambiente Multi-Cluster

Considera-se também que cada cluster componente trabalha com sua própria fila de submissão, arquivos de registro e unidades funcionais, ou seja, as instruções executadas em um cluster foram submetidas pelo nodo mestre dele, e apenas elas podem ler/escrever fisicamente nele. Entretanto, as instruções são obtidas de uma cache única e globalmente acessível.

Há consideráveis benefícios no uso de um multi-cluster. Basicamente, cada cluster realiza menos operações de leitura/escrita por ciclo, por isso ocorrem menos acessos aos arquivos de registro. Com menos operações de leitura/escrita há a necessidade de uma lógica de escalonamento de instruções menos complexa.

Além disso, o multi-cluster trabalha sobre políticas similares de uso de recursos e escalonamento, ou seja, há a necessidade de um gerenciador global de recursos, que por sua vez, deve ser capaz de compreender as diversas formas de descrição do que cada organização ou domínio disponibiliza para o ambiente.

2.2.3 Grades Computacionais

O conceito de grade computacional, do inglês *computational grid*, tem sua origem no termo *electrical power grid* que denota uma rede de energia elétrica (geração, transmissão e distribuição) (FOSTER, 2003). Esta rede é a infra-estrutura que possibilita o uso da energia elétrica (o recurso, neste caso) de forma consistente, generalizada, barata e confiável

A comparação de um ambiente grade com uma rede elétrica vem do fato que uma rede elétrica não se precisa importar com a fonte geradora de energia, o que ocorre igualmente em um ambiente de grade. Portanto, não devemos nos preocupar com a origem dos recursos provenientes da grade e sim se eles estão disponíveis para serem usados pelas aplicações.

Uma grade pode ser vista como um ambiente de componentes heterogêneos e geograficamente dispersos e com uma quantidade de recursos e serviços superior a de outras configurações distribuídas. Isso torna a questão da consistência do sistema ainda mais importante, dado que não existe um nó mestre coordenando a submissão, escalonamento e verificação dos processos, uma exemplificação de grade pode ser vista na Figura 2.5.

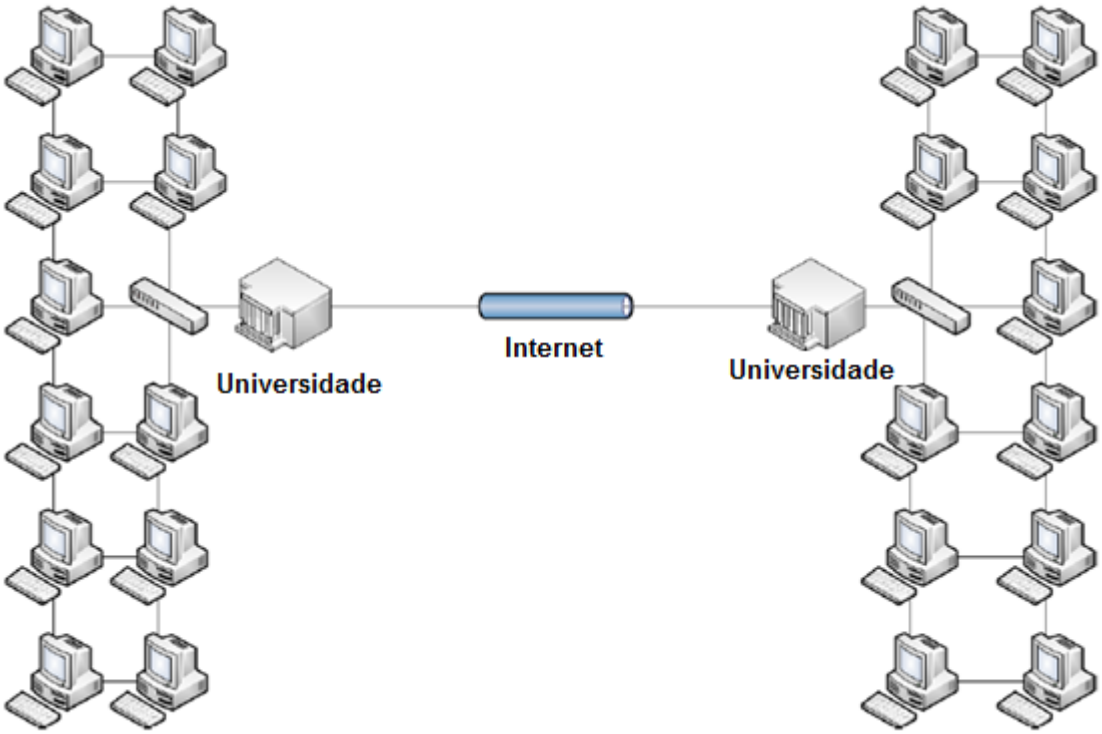

FIGURA 2.5: Exemplo de Grade Computacional

Segundo Foster (2001)**,** o termo *Grid Computing* foi estabelecido em meados da década de 90 para denotar *"uma proposta de infraestrutura computacional distribuída para engenharia e ciências avançadas".* Baker (2000) considera que a popularização da Internet e a disponibilidade de computadores poderosos e redes de alta velocidade a baixo custo fornecem a oportunidade tecnológica de usar as redes de computadores como um recurso computacional unificado. Assim, devido ao seu foco em compartilhamentos inter-organizacionais e dinâmicos, as tecnologias de grade complementam ao invés de competir com as tecnologias de computação distribuídas existentes (FOSTER et al, 2001).

Outra boa definição sobre grade computacional é apresentada por Krauter (2002): *"um grid é um sistema computacional de rede que pode escalar para ambientes do tamanho da Internet com máquinas distribuídas através de múltiplas organizações e domínios administrativos. Neste contexto, um sistema computacional de rede distribuído e um computador virtual formado por um conjunto de máquinas heterogêneas ligadas por uma rede que concordam em compartilhar seus recursos e locais com os outros."*

Uma das tarefas a serem tratadas pela infraestrutura da grade é o gerenciamento de dados, estejam eles armazenados em arquivos ou em bases de dados. Como estas infraestruturas ainda estão em desenvolvimento, ainda há muito a ser feito nesta questão. Inicialmente, cada usuário resolvia o tratamento dos dados através de soluções específicas, muitas vezes recorrendo a arquivos de lote (shell scripts) ou a transferências de arquivo manuais via protocolo FTP. Vários trabalhos vêm se preocupando principalmente com os dados em forma de arquivos, até por ser relativamente simples extrair dados armazenados em um banco de dados para o formato de arquivo texto. Existem propostas desde protocolos eficientes de transferência de dados como o GridFTP (ALLCOCK, 2001) do Projeto Globus até sistemas de arquivos como o LegionFS (WHITE, 2001).

Segundo Foster (2002), um ambiente de grade deve coordenar os recursos disponíveis que não estão sob um gerenciamento centralizado, o que faz com que muitas das soluções hoje disponíveis na verdade servem para a estruturação de clusters, pois são compostas por gerenciadores locais. Deve usar protocolos e interfaces padronizados, abertos e de propósito geral, já que deve integrar diferentes domínios de máquinas, com suas próprias políticas de disponibilidade de recursos, autenticação e autorização. Por fim, uma grade deve entregar diferentes qualidades

de serviços em diferentes áreas, seja tempo de resposta, *throughput*, disponibilidade e outros, para que a utilidade da soma dos componentes seja maior do que a deles individualmente.

Alguns pesquisadores consideram a própria Internet como sendo um sistema distribuído, mais precisamente uma grade global com uma vasta gama de recursos disponíveis, sejam eles dedicados, ou apenas o tempo ocioso de máquinas pessoais. Esse último é conhecido também como configuração oportunista, que em toda amplitude da malha, usa apenas os recursos ociosos dos computadores que a formam para reduzir o custo da computação, como os projetos SETI@Home e Folding@Home.

As aplicações clientes das grades oportunistas costumam ser muito pequenas e transparentes ao usuário, e respeitam o poder maior que, no caso das grades oportunistas, é do usuário local. Esse poder de controle se deve ao fato de que, diferentemente aos clusters, cada nodo da grade possui seu próprio gerenciador de recursos, que prioriza as aplicações locais.

Apesar dessa visão otimista, os pontos levantados pelo artigo de Foster mostram que a Internet ainda não pode ser vista como uma grade computacional, pois apesar de usar protocolos abertos e de propósito gerais para acessar recursos distribuídos, não permite o uso coordenado destes para obter qualidades de serviço interessantes.

Tal como os clusters, não existe uma classificação clara dos ambientes de grade, mas uma das características observadas é a topologia. Vários clusters dentro de uma mesma instituição, formam uma intragrid. Quando mais de uma instituição comunicam suas grades, há uma extragrid, e quando se extrapola para muitas instituições, tem-se uma intergrid (DANTAS, 2005).

Devido à complexidade inerente de se trabalhar com as grades, um esforço tem sido feito no desenvolvimento dos chamados *portais de grid*, que tem como objetivo servir de ponto de acesso único às informações e recursos disponíveis, além esconder os detalhes complexos do ambiente. Eles podem ter um enfoque mais genérico, sendo então chamados de portais de usuário, ou então serem aplicados em áreas específicos, os portais científicos.

Além destes, há um grande trabalho na área de ferramentas de unificação de grades, como os projetos Globus e Condor, que lidam com a parte de segurança e

comunicação entre as diferentes organizações que formam a grade, de forma que os processos sejam encaminhados aos nós mais apropriados para sua execução.

Segundo GridData (2011), Existem três principais aspectos que caracterizam grades computacionais:

- ***Heterogeneidade*** *(heterogeneity):* uma grade envolve uma multiplicidade de recursos que são heterogêneos por natureza e que podem estar dispersos por numerosos domínios administrativos através de grandes distâncias geográficas;
- ***Escalabilidade*** *(scalability):* uma grade pode crescer de poucos recursos para milhões. Isto levanta o problema da potencial degradação do desempenho a medida que o tamanho de uma grade cresce. Consequentemente, aplicações que requerem um grande número de recursos dispersos geograficamente devem ser projetados para serem extremamente tolerantes a latência;
- ***Dinamicidade*** *ou* ***adaptabilidade*** *(dynamicity or adaptability):* em uma grade a falha de um recurso é a regra, e não a exceção. De fato, com tantos recursos, a probabilidade de que algum recurso falhe é naturalmente alta. Os gerenciadores de recursos ou aplicações devem adaptar o seu comportamento dinamicamente a fim de extrair o máximo de desempenho a partir dos recursos e serviços disponíveis.

Objetivando uma arquitetura extensível e de estrutura aberta, a partir da qual são fornecidas soluções, para os requisitos chaves desempenhados pela OV; foi apresentada por Foster (2001) uma comparação funcional entre a arquitetura para ambientes de grade e a arquitetura de protocolos da Internet, como apresentada na Figura 2.6. A equivalência entre as duas arquiteturas é interessante, pois auxilia o entendimento funcional das camadas da grade e também demonstra a preocupação de uma padronização (DANTAS, 2005).

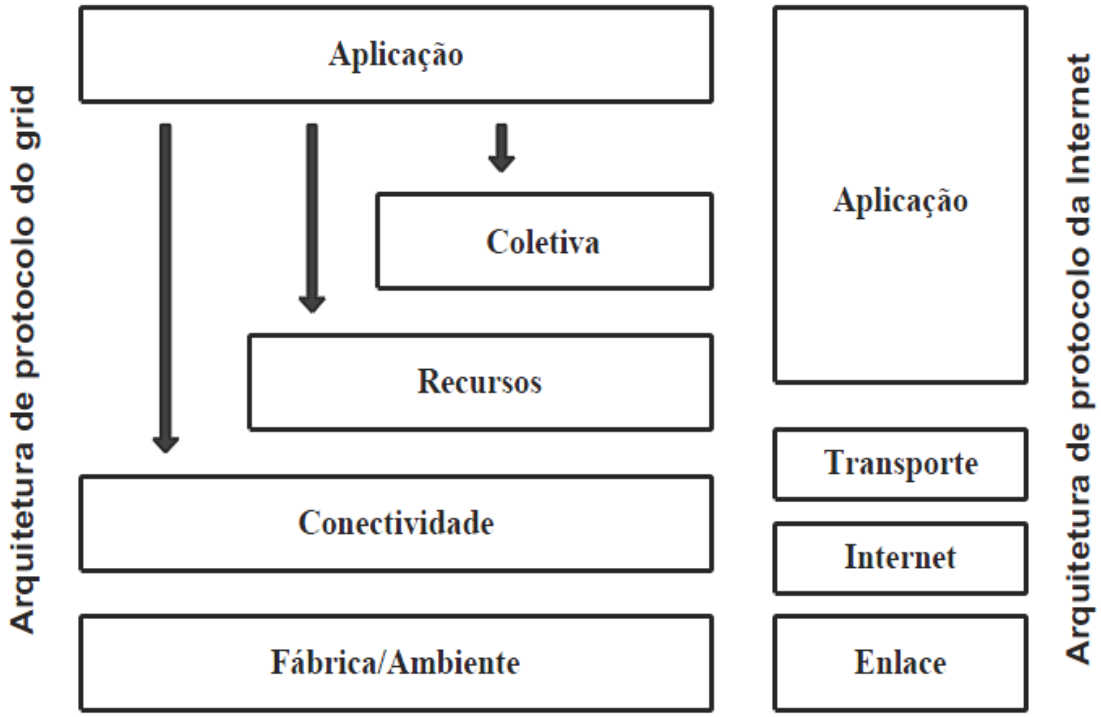

FIGURA 2.6: Comparação funcional entre a arquitetura da grade e da Internet

- **Ambiente**: especifica determinadas operações locais que ocorrem em cada recurso como resultado do compartilhamento nas camadas superiores. Há mecanismos de negociação, para obter informações sobre a estrutura, o estado e as possibilidades dos recursos. E há mecanismos de gerenciamento, para monitorar a qualidade de serviço;
- **Conectividade**: define os protocolos básicos para transações de rede específicas da grade;
- **Recursos**: a camada define protocolos e APIs (Application Program Interface) que forneçam segurança na negociação, iniciação, monitoramento, controle, geração de relatórios;
- **Coletiva**: atuam nas interações entre coleções de recursos.
- **Aplicação**: esta camada compreende as aplicações dos usuários que são executados no ambiente da OV. Para a correta execução dessas aplicações, as camadas inferiores proveem serviços essenciais para esta camada.

É importante deixar claro as diferenças entre cluster e grades. No trabalho de Colvero (2005) foi desenvolvida uma comparação entre estes tipos de ambientes. Nele, é mostrado que clusters referem-se a ambientes de um único domínio, com

homogeneidade em sistema operacional. Já grades podem integrar múltiplos domínios heterogêneos. Além disso, atendem problemas de granularidade muito maior que clusters, e o custo para solução é compartilhado entre os domínios, em vez de se concentrar em uma única organização.

2.2.4 Nuvens Computacionais

Nuvem computacional é uma denominação relativamente recente, baseada nos conceitos de virtualização, computação distribuída, *utility computing*, redes, web e serviços de software. Ainda não existe uma definição estabelecida, muitas discordâncias são encontradas na literatura. Segundo Wang (2008), computação nas nuvens se define por um conjunto de serviços disponíveis na rede, provendo escalabilidade, garantia de qualidade de serviço, com plataformas computacionais de baixo custo, que podem ser acessadas de forma simples e pervasiva.

Nuvens computacionais podem disponibilizar, como serviço, uma plataforma computacional integrada. Para que isso ocorra, é necessário que serviços de acesso a hardware, software e dados estejam disponíveis ao usuário. De forma mais clara:

- O sistema Hardware-as-a-service (HaaS) é uma modalidade que permite que usuários utilizem recursos de hardware de forma virtual, flexível, escalável e configurável atendendo as necessidades do usuário (HAAS, 2006).
- No Software-as-a-service (SaaS) as aplicações são disponibilizadas ao usuário de forma virtual, através da internet. Assim, não há necessidade, por parte do usuário, de instalar e rodar a aplicação no seu próprio computador (WANG et al., 2008).
- Já no Data-as-a-service (DaaS) dados de diferentes fontes e em vários formatos, podem ser acessados pelos usuários através da rede, em outras palavras, o usuário faria a manipulação dos dados remotos como se estivessem em seu disco local (WANG et al., 2008).

Através dos mecanismos HaaS, SaaS e DaaS, a nuvem pode oferecer, para os usuários, uma plataforma integrada como serviço, chamada na literatura de Platform-as-a-service (PaaS). Assim, cada usuário pode escolher uma plataforma

que melhor supre suas necessidades quanto a configuração de hardware, softwares instalados e acesso a dados. A Figura 2.7 apresenta um esquema dessas funcionalidades na nuvem.

Alguns dos conceitos que envolvem a computação nas nuvens a distingue de outros paradigmas computacionais. Os recursos e serviços disponibilizados para os usuários são sob demanda, dessa forma, usuários podem personalizar seu ambiente computacional. As configurações computacionais, providas pela nuvem, podem garantir qualidade de serviço (QoS) para os usuários, como velocidade de CPU, tamanho de memória, ou seja, performance em hardware. Outra característica das nuvens computacionais é sua autonomicidade. Hardware, software e dados dentro das nuvens podem ser automaticamente reconfigurados, apresentando uma nova plataforma ao usuário.

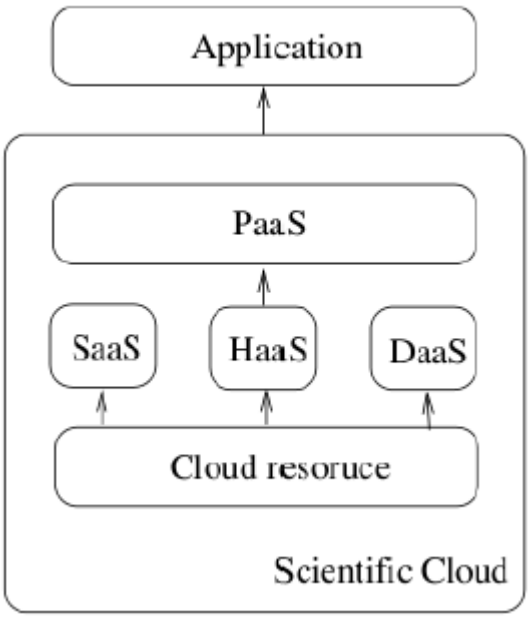

FIGURA 2.7: Nuvem Científica

Outros dois aspectos de suma importância quanto à caracterização de nuvens computacionais são a escalabilidade e a flexibilidade. Os serviços e as plataformas computacionais oferecidas pela nuvem devem ser disponibilizados em diferentes regiões geográficas, com diferentes configurações de hardware e software, sendo que essas plataformas devem ser flexíveis para que se adaptem aos mais diversos pedidos de um potencial grande número de usuários.

Tecnologias de virtualização como VMware (2011), a Web 2.0 (2011), serviços Web, arquitetura orientada a serviços (SOA), sistemas de armazenamento distribuído, todos esses conceitos e tecnologias orquestram a computação nas nuvens, de forma que a partir deles é viável possuir flexibilidade e escalabilidade em serviços de hardware, bem como uma melhor interconectividade e iteratividade de aplicações Web.

As tecnologias móveis já fazem parte dos sistemas distribuídos, e a sua utilização nesses ambientes está em constante crescimento. São encontrados alguns desafios para a inserção desses dispositivos, como limitação de processamento e armazenamento e fonte de energia finita. Porém, uma vez concretizada, a computação móvel distribuída será uma ótima ferramenta para solução de problemas tanto de âmbito acadêmico como comercial.

3 Sistemas de Arquivos Distribuídos

3.1 Armazenamento de dados em Grades

Segundo Bergua (2008), muitas aplicações em grades exigem acesso a grandes quantidades de dados. Para este tipo de aplicações, a maioria dos esforços de gerenciamento de dados em grades tem sido focada na replicação de dados. Réplicas de arquivos são cópias múltiplas de um arquivo espalhados por toda a grade, utilizadas para melhorar o acesso a dados. No entanto todos os esquemas de replicação de dados fornecidos na literatura falham em fornecer um sistema de arquivos global para acesso a arquivos em ambientes grades de dados. Um importante desafio para a computação em grade é um sistema de arquivos verdadeiramente global para aplicações de grade em execução em diferentes domínios administrativos.

No contexto de sistemas de arquivos da grade, o Open Grid Forum (2011) define um sistema de arquivos Grade como um namespace de recursos legíveis por humanos de forma hierárquica voltada à gestão de recursos heterogêneos de dados distribuídos que podem ser divididos entre múltiplos domínios administrativos autônomos. Além disso, um sistema de arquivos da grade deve fornecer alto desempenho de acesso a dados. E/S paralelas têm sido a técnica tradicional para melhorar acesso a dados em clusters e multiprocessadores. Paralelismo em sistemas de arquivos é obtido usando vários servidores independentes e distribuindo dados entre esses nós para permitir o acesso aos arquivos de forma paralela. Existem muitos sistemas de arquivos paralelos para este tipo de plataformas, no entanto muito poucos sistemas de arquivos paralelos têm sido fornecidos a ambientes grade.

Existem alguns sistemas de arquivos de grade, a maioria deles se une a outros sistemas de arquivos oferecendo assim namespaces comuns lógicos. Exemplos deste tipo de sistemas de arquivos são: Storage Resource Broker (SRB) (Nirvana Storage, 2011), Avaki (Sybase Inc, 2011) e SlashGrid (GridPP, 2011). Um exemplo de sistema de arquivos grade, que oferece E/S paralela é o GFARM(TATEBE, 2004). GFARM é um sistema de arquivos distribuídos e paralelos da grade que permite o acesso a dados distribuídos por diferentes domínios

administrativos. O principal problema da GFARM é que ele implementa um servidor de E / S que deve ser implantado e executado em cada nó do sistema e não usa os serviços de grade padrão. Outro sistema de arquivos de grade que oferece E/S paralela é o GFAL (CERN, 2011), que faz parte do gLite (CERN, 2011b).

3.2 Transferência de arquivos

A exploração científica em várias disciplinas, como Física de Altas Energias, Modelagem Climática e Ciências da Vida, requer o processamento de coleções enormes de dados (ANGLANO, 2005). Por exemplo, a quantidade de dados que devem ser processados em muitos experimentos de física de alta energia é da ordem de Terabytes (e às vezes até Petabytes) (GriPhyN, 2001), (LHC, 2001), (PPDG, 2011), enquanto tipicamente as aplicações de modelagem do clima geram conjuntos de dados de saída cujo tamanho está na ordem de centenas de Gigabytes (ALLCOCK, 2002). Para estas aplicações, a criação de Grades de Dados (CHERVENAK, 2001), que são um conjunto de recursos geograficamente distribuídos de armazenamento e computação, parece uma solução promissora. A fim de possibilitar um desempenho satisfatório, aplicações intensivas de dados grade requerem a disponibilidade de um sistema capaz de transferir arquivos potencialmente enormes na quantidade de tempo mais curto possível (ALLCOCK, 2002). Por uma questão de fato, o tempo típico de conclusão de aplicações intensivas de dados é dada pela soma de seu tempo de execução e do tempo necessário para transferir os dados de que necessitam (RANGANATHAN, 2003), e é muitas vezes dominada pelo tempo de transferência de dados.

Um modo aparentemente simples de reduzir o tempo de transferência de dados é aumentar a largura de banda da interconexão das redes dos recursos em uma grade. Os recentes avanços na tecnologia de comunicação estão, na verdade permitindo a implantação de redes Gigabit-por-segundo em escala geográfica, a um custo razoável, e alguns conjuntos de teste deste tipo já vêm sendo realizados. Estas redes, que são muitas vezes chamadas de redes *High Bandwidth Delay Product (HBDP)*, são caracterizadas por altas latências por causa das distâncias percorridas pelos seus links.

No entanto, a evidência empírica e analítica existe demonstrando as limitações intrínsecas do protocolo TCP, o padrão de fato para transporte de pacotes na Internet, que tem se mostrado incapaz de utilizar plenamente a capacidade disponível fornecida por redes HBDP. Isso ocorre porque os mecanismos de janelas TCP impõe um limite na quantidade de dados que vai enviar antes de esperar por uma confirmação. Assim, os atrasos típicos de redes de HBDP significam que o TCP irá gastar uma enorme quantidade de tempo de espera por *aknowledgments*, que por sua vez significa que a transmissão de dados do cliente nunca vai chegar a capacidade disponível da rede.

Portanto, simplesmente aumentando a largura de banda disponível não necessariamente se traduz em melhorias proporcionais ao desempenho de transferência de dados, a menos que versões otimizadas TCP sejam utilizadas. A solução tradicional para este problema é ajustar os tamanhos das janelas TCP e buffer para coincidir com a BDP da rede, mas nem sempre esta operação pode ser executada por usuários com privilégios padrão. Por exemplo, em muitas variantes Unix, o tamanho máximo de buffer é um valor de todo o sistema que só pode ser modificada pelo superusuário (root). Para superar essas limitações intrínsecas do TCP, outras abordagens estão sendo investigadas, e novas técnicas de transferência de dados vem consequentemente, sendo desenvolvidas. Estas técnicas funcionam, tanto ao nível do sistema quanto ao nível de aplicação. Soluções de nível de sistema, que normalmente requerem modificações no sistema operacional das máquinas, dos equipamentos de rede, ou de ambos, incluem ajuste automático de tamanhos de janela TCP (AutoTCP, 2011), às vezes complementado por outros mecanismos sofisticados (Web100, 2011), e desenvolvimento de novas versões do TCP (por exemplo, Seletive Acknowledment TCP (MATHIS, 2011), High Speed TCP (FLOYD, 2003b), e Scalable TCP (KELLY, 2003)). Esta abordagem pode render um desempenho muito bom, mas geralmente requer atualizações significativas na infra-estrutura da rede. Por outro lado, as soluções de nível de aplicação não requerem qualquer modificação na infra-estrutura da rede, mas tentam contornar problemas TCP explorando técnicas não necessitando de qualquer privilégio especial. Estas técnicas são baseadas em TCP ou UDP. soluções baseadas em TCP (por exemplo, GridFTP (GridFTP, 2011), bbFTP (BBFTP, 2011), e bbcp (HANUSHEVSKY, 2001)) usam conexões TCP para mover dados, e tentam contornar os problemas de tamanho da janela do TCP

usando fluxos paralelos, de modo que uma janela de congestionamento agregado, combinando o BDP no caminho de rede usado para realizar a transferência, é obtida. Por outro lado, as soluções baseadas em UDP (por exemplo, FOBS (DICKENS, 2003), Tsunami (WALLACE, 2003), UDT (GU, 2003), e SABUL (GU, 2003b), utilizam UDP para mover dados, e empregam algoritmos de controle baseados em taxa para coincidir a velocidade de transmissão com a largura de banda disponível, as vezes juntamente com os algoritmos de controle de fluxo e congestionamento, a fim de manter o mais baixo possível a taxa de perda durante as transmissões para controle de congestionamento.

Apesar de um número relativamente grande de ferramentas de transferência de alto desempenho, a questão sobre a eficácia de cada uma deles surge naturalmente. No entanto, segundo Anglano (2005), apesar de protótipos de muitos dos sistemas acima estiveram em foco por algum tempo, uma comparação experimental ainda está faltando na literatura.

3.3 Abordagens de transferência de arquivos

3.3.1 TCP

Transmission Control Protocol (TCP) tem sido amplamente utilizado como um protocolo de comunicação em nível de transporte na Internet (POSTEL, 1981). GridFTP foi concebido para utilizar TCP como seu protocolo de comunicação a nível de transporte (ALLCOCK,2003).No entanto, o TCP é um protocolo de comunicação muito antigo que foi projetado na década de 1970. Vários problemas foram relatados sobre TCP, tais como a sua incapacidade para suportar a velocidade rapidamente crescente de redes recentes. Como exemplo, o atual TCP Reno (TCP versão Reno) não consegue detectar o congestionamento em uma rede até que ocorra perda de pacotes, portanto, um grande número de pacotes é perdido. Com as velocidades mais rápidas de redes e tamanhos maiores de buffer de roteadores em uma rede, a quantidade de pacotes perdidos e o *throughput* do TCP deterioram significativamente. Para resolver os problemas existentes no TCP, GridFTP tem características como o estabelecimento de múltiplas conexões TCP em paralelo para acelerar o início lento na fase inicial do TCP e negociando o tamanho do *TCP*

socket buffer entre o servidor e o cliente GridFTP de acordo com o atraso da largura de banda de um rede (ALLCOCK, 2003).

No entanto, a eficácia desses recursos não foi totalmente investigada. Em outras palavras, as configurações ideais para o número de conexões TCP paralelas e tamanho do *TCP socket buffer* não foram investigados. Houve vários trabalhos relacionados em tamanho de *TCP socket buffer* e conexões TCP paralelas (SEMKE, 1998), (LU, 2005).

Em (SEMKE, 1998), (DUNIGAN, 2002), os mecanismos de ajuste automático de *TCP socket buffer* foram propostos. No entanto, ambas as abordagens requerem algumas modificações para um socket API e/ou uma pilha de protocolo TCP. Em Grades computacionais, recursos de computação heterogêneos são integrados. Por isso, tais modificações de sistemas operacionais são irrealistas, ou seja, a otimização de tamanho do *TCP socket buffer* não deve depender de qualquer informação que não obtida pelo middleware da grade. Em (THULASIDASAN, 2003), uma extensão do protocolo GridFTP para negociação automática do tamanho do *TCP socket buffer* foi proposta. No entanto, o mecanismo proposto é simples e não ideal, ou seja, ele simplesmente atribui o dobro do BDP para cada conexão TCP. Para alocação de memória eficiente para *TCP socket buffer*, mecanismos mais inteligentes devem ser incorporados.

Pelo contrário, a eficácia de conexões TCP paralelas tem sido estudada por muitos pesquisadores (SIVAKUMAR, 2000), (LU, 2005). Em (SIVAKUMAR, 2000), (LU, 2005), o desempenho de conexões TCP paralelas é investigado utilizando experimentos de simulação. No entanto, para aperfeiçoar o número de conexões TCP paralelas, simulações baseadas em abordagens são inadequadas, ou seja, é muito difícil ou, na maioria dos casos, impossível de aplicar os resultados da simulação para uma otimização de parâmetro. Para otimizar o número de conexões TCP paralelas, algumas dicas no efeito de conexões TCP paralelas sobre o seu desempenho é necessário. Em (HACKER, 2001), (LU, 2005), modelos analíticos simples de conexões TCP paralelas foram apresentadas. No entanto, os modelos analíticos não são aplicáveis para otimizar o número de conexões TCP paralelas, uma vez que não captam os trade-offs na transferência de dados paralelos, como explicado em (ITO, 2006).

3.3.2 Computação Virtual

Os sistemas transferência de arquivos citados na sessão 3.3 são aplicações que se apoiam em protocolos de nível de transporte, ou seja, TCP e UDP, para transferir dados de sua origem para o destino. Embora estes sistemas explorem técnicas altamente sofisticadas para aumentar sua taxa de transferência, eles sofrem de uma desvantagem comum, nomeadamente a sua dependência sobre os protocolos de roteamento IP, que tem como consequência a fato de que o *throughput* que eles conseguem é limitado pela largura de banda disponível no caminho de rede escolhido pela camada de roteamento IP.

Infelizmente, os protocolos de roteamento IP notoriamente produzem rotas sub-ótimas (SAVAGE, 1999), (SAVAGE, 1999b), (TANGMUNARUNKIT, 2001), uma vez que sua escolha de caminhos da rede não é guiada por considerações de desempenho como estão principalmente preocupados com a troca de conectividade da informação. Como consequência, não é raro o caso onde os tempos mais curtos de transferência podem ser obtidos escolhendo um caminho de rede diferente do escolhido pelos algoritmos de roteamento IP. Por exemplo, (SAVAGE, 1999b) observa que para 30 a 80 por cento dos caminhos escolhidos pelos algoritmos de roteamento IP entre pares de hosts da Internet, tomadas a partir de um conjunto relativamente grande de máquinas, foi possível encontrar caminhos alternativos com melhores características de desempenho.

Outra consequência da dependência em algoritmos de roteamento IP é a quantidade, possivelmente, alta de tempo necessário para recuperação de falhas de links. Por uma questão de fato, mecanismos de recuperação de falhas utilizados pelos protocolos de roteamento típicos da Internet por vezes, levam muitos minutos a convergir a uma forma consistente (LABOVILZ, 200), e há momentos em que o caminho pode levar a interrupções significativas na comunicação com duração de até dezenas de minutos ou mais (CHANDRA, 2001), (PAXSON, 1996), (PAXSON, 1997). Como consequência, as transferências ao longo de caminhos com defeito pode ser atrasada por um tempo muito longo, assim a largura de banda efetiva obtida nestas situações é reduzida abaixo de qualquer valor razoável.

Em (ANGLANO, 2004) é descrito o *File Mover*, um software que atende os problemas acima através da exploração de uma arquitetura *overlay network* (Rede de sobreposição). Uma rede de sobreposição é uma rede virtual, em camadas em

cima da Internet existente, cujos membros nós são colocados nas bordas da rede física subjacente e se comunica por meio de um protocolo de nível de transporte (por exemplo, TCP ou UDP). Cada par de nós membros de uma rede sobreposta se comunica por meio de um link virtual, que corresponde ao caminho de rede escolhido pela camada IP para transferência de dados de um membro para o outro. Os Relays de uma rede de sobreposição concorda em frente um do outro tráfego ao longo de um ou mais links virtuais, até que o host de destino é atingido. A Figura 3.1 apresenta esquematicamente uma possível configuração do File Mover em que uma rede de sobreposição compreendendo quatro Relays (A, B, C e D) é assumida.

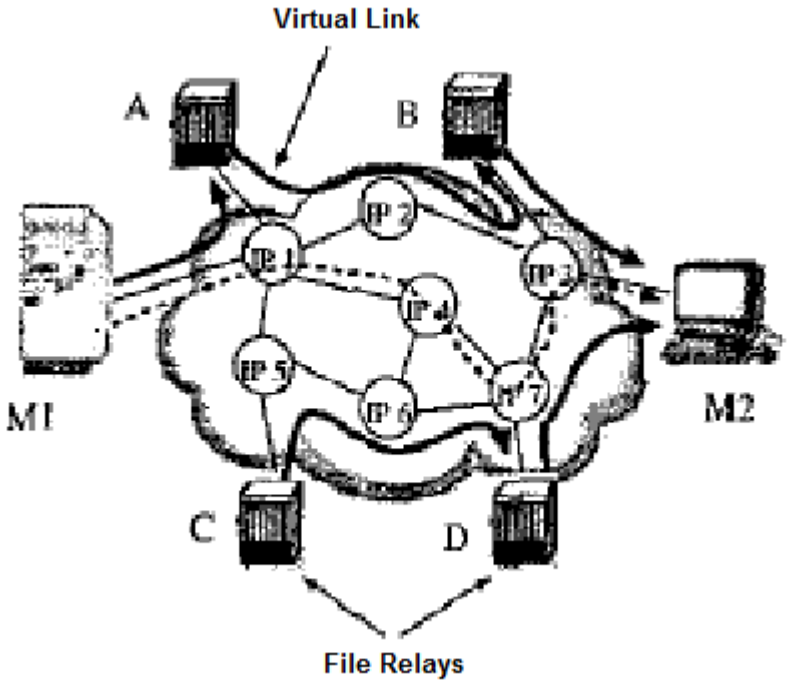

FIGURA 3.1: Exemplo de Rede de Sobreposição

As vantagens do *File Mover* com relação a soluções convencionais baseadas em IP para transferência de arquivos são muitas. Primeiro de tudo, caminhos de rede resultando em melhor *throughput* do que um escolhido pelos algoritmos de roteamento IP que pode ser explorado por interposição, entre a fonte de arquivo e o destino do arquivo, um conjunto adequado de *Relays*. Por exemplo, considerando a situação descrita na figura 3.1, assumindo que o caminho da rede escolhido por IP para conectar máquina M1 a máquina M2 (denotado por uma seta pontilhada) atravessa roteadores IP1, IP4, IP7, e IP3. Se o caminho atravessando roteadores

IP1, IP2, e IP3 tem um melhor *throughput*, o *File Mover* é capaz de usá-lo transferindo o arquivo sobre os links virtuais conectando M1 a A, A a B, B a M2. Inversamente, uma solução convencional baseada em IP (por exemplo, FTP) seria forçada a usar o caminho de rede menos eficiente.

Segundo, se mais caminhos estão disponíveis entre uma fonte e uma máquina de destino, eles podem ser usados simultaneamente para transferir partes diferentes do mesmo arquivo em paralelo, de modo que o tempo de download é reduzido, como por exemplo proposto em (RAO, 2001) para reduzir a latência end-to-end. Por exemplo, na situação retratada na figura 3.1, é possível transferir uma parte do arquivo no caminho IP 1, IP 2, IP 3 (usando os *Relays* A e 6) e outra parte no caminho IP 1, IP 5, IP 6, IP 7, IP 3 (usando C e D). Transferências por múltiplos caminhos também podem ser utilizados para aumentar o *throughput* entre uma máquina origem e um destino transferindo arquivos diferentes ao mesmo tempo quando um lote de arquivos é solicitado para o mesmo servidor. Em outras palavras, o *File Mover* é capaz de agregar vários caminhos de rede para aumentar a largura de banda disponível e, consequentemente, a taxa de transferência, entre a fonte de dados e o destino de dados. Em contraste, o download paralelo de um arquivo ou lote de arquivos com um sistema convencional de transferência de arquivos exige o uso de múltiplas sessões de transferência simultâneas entre o mesmo par de hosts. Estas sessões, no entanto, devem usar o mesmo caminho de rede, de modo que o *throughput* agregado que pode ser alcançado é limitado pela taxa de transferência caminho.

Terceiro, no caso de uma falha de um link, um tempo menor de recuperação pode ser obtida usando uma nova sequência de links virtuais que não inclui links com defeito físicos ou roteadores. Em contraste, como já mencionado, os sistemas baseados em roteamento IP devem esperar até que os protocolos de roteamento IP convergem para uma configuração consistente, e isso pode levar alguns minutos.

3.4 Protocolo GridFTP

GridFTP é um protocolo de transferência de dados, que é projetado para transferir efetivamente grande volume de dados em *Grid Computing* (Globus, 2003),

(ALLCOCK, 2003), (Globus, 2002). GridFTP é uma extensão de FTP (File Transfer Protocol) (POSTEL, 1985; ELZ, 1997; HETHMON, 1998) que tem sido amplamente utilizado e esta atualmente sob a normalização do GGF (Global Grid forum) (GGF, 2011). GridFTP, que usa TCP como seu protocolo de comunicação na camada de transporte, foi concebido para resolver vários problemas do TCP. Por exemplo, além das características dos atuais FTP, ele tem funcionalidades adicionais, tais como negociação automática do tamanho do *TCP socket buffer*, transferência de dados paralela, controle de transferência de arquivos de terceiros, transferência de arquivos parciais, segurança, e transferência de dados confiável (ALLCOCK, 2003). A maioria destas características específicas de GridFTP são realizadas por um novo modo de transferência chamado *Modo de Bloqueio Estendida* (ALLCOCK, 2003).

Atualmente, o servidor GridFTP e software cliente em conformidade com GridFTP versão 1 (GridFTPv1) é implementado no Globus Toolkit (Globus, 2011b), que é o middleware padrão de fato para a computação em Grid. No entanto, nesta implementação GridFTP específica, o recurso da negociação automática do tamanho do *TCP socket buffer* não é implementado e, portanto, um usuário deve especificar manualmente o número de conexões TCP paralelas para transferência de dados paralelo. Além disso, GGF tem discutido os problemas com GridFTP v1 e também realizado um estudo sobre GridFTP v2 (versão 2) como uma solução para esses problemas (MANDRICHENKO, 2005). Recursos adicionais foram incorporados ao GridFTP v2. Estas características atenuam várias limitações do modo de bloqueio estendido do GridFTP v1, tanto que a transferência de dados é restrito a uma única direção e incapaz de abrir / fechar canais de dados no meio da transferência de dados. No entanto, como configurar o Parâmetros de controle do GridFTP sobre conexões TCP paralelas não foi abordado ainda na discussão sobre GridFTP v2. (ITO, 2006) apresenta um resumo de duas principais características do GridFTP, auto-negociação do tamanho do tamanho do *TCP socket buffer* e transferência paralela de dados e problemas não resolvidos da GridFTP é apresentado.

Auto-Negociação do tamanho do *TCP Socket buffer*

Em GridFTP, o tamanho do *TCP socket buffer do servidor* pode ser explicitamente configurado pelo cliente com o comando SBUF (Buffer Size Set).

Além disso, o tamanho do *TCP socket buffer* pode ser configurado através da negociação entre servidor e cliente GridFTP usando o comando ABUF (Auto-Negociar Tamanho Buffer). Quase todas as implementações TCP existentes alocam um tamanho fixo (por exemplo, 64 [Kbyte]) para o *TCP socket buffer*, assim pode ser esperado uma melhora no *throughput* quando o tamanho do *TCP socket buffer* está devidamente configurado de acordo com o produto de largura de banda-atraso da rede.

No entanto, não foi suficientemente estudado como o comando ABUF deve ser implementado. Por exemplo, o comando ABUF não foi implementado na implementação GridFTP incluídos em um Globus Toolkit (Globus, 2011b). Exemplos da aplicação de comando ABUF são discutidos em (ALLCOCK, 2003; THULASIDASAN, 2003), que medem um tempo de ida e volta e largura de banda disponível de uma rede de medição de tráfego gerados entre os servidor GridFTP e o cliente.

Entretanto, mais investigação sobre a execução de comandos ABUF é necessário. Em uma rede real, a largura de banda disponível de uma rede varia com o tempo, e uma grande quantidade de tráfego de medição devem ser gerados para medir com precisão a largura de banda disponível. Devido a estas razões, abordagens de medição ativa como discutido em (ALLCOCK, 2003; THULASIDASAN, 2003) que configuram o tamanho do *TCP socket buffer* baseado simplesmente na medida do produto largura de banda-atraso, são inadequadas para fins práticos.

Transferência paralela de dados

GridFTP pode estabelecer múltiplas conexões TCP em paralelo usando OPTS RETR ou OPTS comandos STOR. Assim, um único arquivo pode ser transferido via múltiplas conexões TCP de/para um simples ou múltiplos servidores GridFTP. Pode ser esperado um maior *throughput* do que com uma única conexão TCP através da agregação de múltiplas conexões TCP (SIVAKUMAR, 2000; QIU, 1999).

Isto pode ser explicado por três razões seguintes. Em primeiro lugar, uma maior largura de banda pode ser adquirida através da agregação de múltiplas conexões TCP ao competir com outras conexões TCP na fase de evitar o

congestionamento TCP (FLOYD, 2003). Isso ocorre porque o controle de fluxo da janela AIMD é adotado na fase de evasão de congestionamento do TCP e transferência de dados pode ser melhor realizada através da agregação de múltiplas conexões TCP em uma rede com uma baixa probabilidade de perda de pacotes. Segundo, o tamanho total do *TCP socket buffer* que pode ser usado em uma transferência de arquivos torna-se maior agregando múltiplas conexões TCP. Isso ocorre porque o total de tamanhos de *TCP socket buffer* é N vezes maior, agregando N conexões TCP. Terceiro, o arranque inicial da taxa de transferência é acelerado na fase inicial lenta do TCP, agregando múltiplas conexões TCP. Na fase de início lenta, a janela de congestionamento dobra para cada tempo de retorno. Assim, o start-up para a taxa de transferência é N vezes mais rápido através da agregação de N conexões TCP.

No entanto, se o número N de conexões TCP agregado é muito grande, resulta em diminuição da taxa de transferência pelas seguintes razões. Primeiro, o tamanho da janela diminui por conexão TCP, e timeout TCP são mais prováveis de ocorrer. Segundo, o overhead necessário para o servidor GridFTP e cliente para processar a pilha do protocolo TCP aumenta. Assim, o valor ideal para o número de conexões TCP agregadas, N, deve ser determinada de acordo com as condições de rede diversas. No entanto, não foi totalmente investigado e ainda é um problema não resolvido como determinar o número de conexões TCP paralelas N em vários ambientes de rede. A Figura 3.2 ilustra o processo.

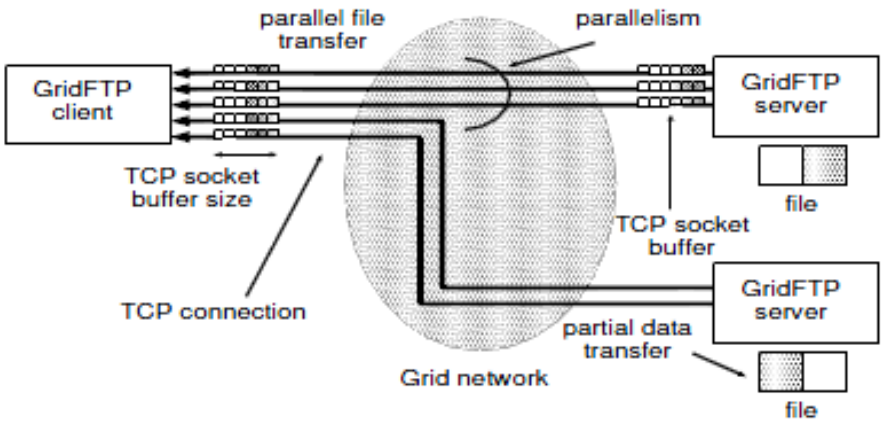

FIGURA 3.2: Transferência paralela em GridFTP

3.5 Ferramenta

Existem algumas ferramentas disponíveis hoje para o desenvolvimento de aplicativos que fazem uso da arquitetura de grades computacionais, além de implementações comerciais, mantidas por algumas empresas. Alguns projetos têm alcançado grande popularidade, principalmente por serem desenvolvidos a partir de padrões abertos e mantidos por comunidades de programadores dentre eles podemos destacar o projeto Globus Toolkit.

Dentro desse contexto temos também algumas ferramentas desenvolvidas para transferência de arquivo em ambiente grade, que assim sendo, fazem uso de tal ferramenta para estabelecer comunicação com a grade.

3.5.1 Globus Toolkit

Globus Toolkit (Globus, 2003) é um conjunto de ferramentas e bibliotecas de software que dão suporte à arquitetura e às aplicações em Grade. É um projeto desenvolvido pelo Argnone National Laboratory (ANL) e University of Southern California. É mantido por uma comunidade de programadores e é baseada em arquiteturas e códigos abertos. Com ele é possível implementar segurança, busca de informações, gerenciamento de recursos e de dados, comunicação, detecção de falhas e portabilidade (DANTAS, 2003).

Segundo (FOSTER, 2003b), os serviços no globus são baseados em um modelo conhecido como Ampulheta. Nesta analogia, os serviços locais encontram-se na parte inferior da ampulheta, enquanto que os serviços globais estão na parte superior. Os 21 serviços fornecidos pelo Globus ficam localizados no gargalo da ampulheta.

Os componentes do conjunto de ferramentas Globus podem ser usados tanto independentemente quanto em conjunto no desenvolvimento de aplicações e de ferramentas de programação de grade. Para cada componente existe uma API2 em linguagem C ou Java definida para facilitar o uso pelos desenvolvedores.

A Figura 3.3 ilustra a estrutura do Globus Toolkit.

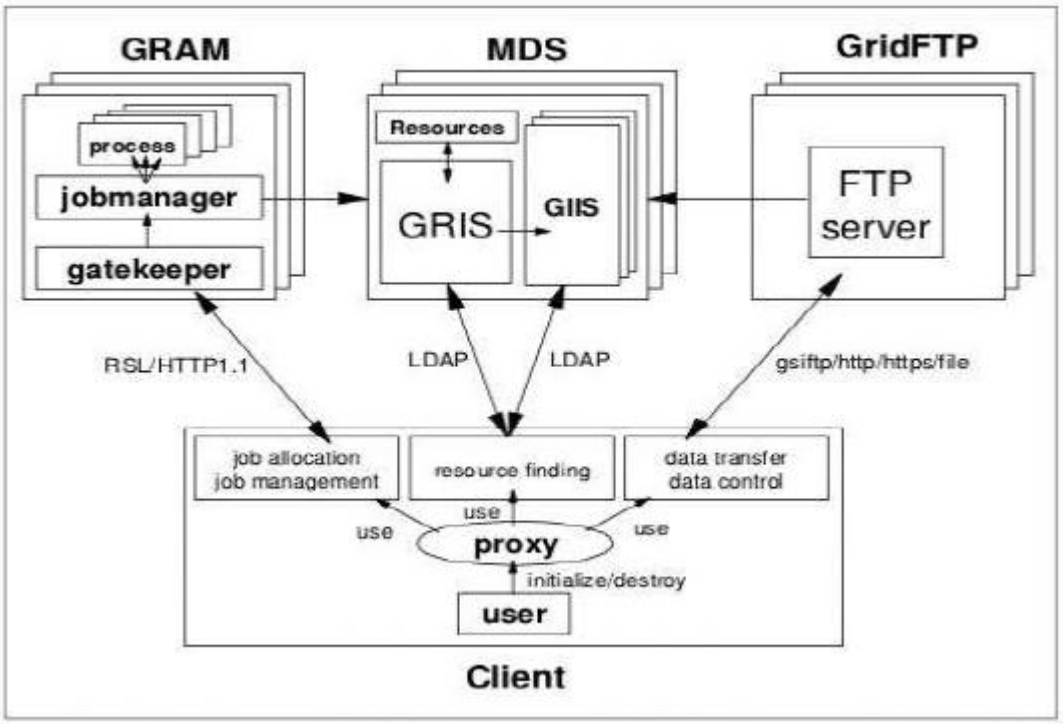

FIGURA 3.3: Estrutura do Globus Toolkit (FERREIRA, 2003)

Segundo (FOSTER, 2003c) e (FERREIRA, 2003), seus componentes mais importantes são:

Grid Resource Allocation and Management (GRAM)

Opera como uma interface entre serviços locais e globais, traduzindo requisições de recursos genéricos em comandos que são específicos para um sistema local particular. Uma grade construída sobre as ferramentas globus pode conter vários GRAMs, sendo cada um responsável por um conjunto local de recursos em particular (PITANGA, 2003).

O software gatekeeper pode ser considerado como uma interface entre o usuário e o GRAM, pois é responsável pela autenticação do usuário e sua associação com uma conta no computador local. Também provê segurança, confiabilidade, criação e gerenciamento de serviços.

Outra função do gatekeeper é iniciar o processo job manager após ter autenticado o usuário. O job manager, por sua vez, é responsável pela comunicação com usuários locais, alocação de recursos, execução de tarefas e por desalocar os recursos após a conclusão da tarefa (PITANGA, 2003).

O GRAM também conta com uma linguagem utilizada para comunicar requisições de recursos. Conhecida como Globus Resource Specification Language (RSL), é uma linguagem extensível e simples, composta por uma lista de expressões de recursos logicamente combinados com alguns operadores lógicos.

Metacomputing Directory Service (MDS)

Fornece um conjunto de ferramentas para descobrir e acessar configurações e informações de sistemas, de sua utilização, de gerenciamento e com relação a rede. Como exemplo de informação, pode-se citar carga da rede, o estado de um recurso, processos sendo executados, arquitetura e sistema operacional de determinada máquina, carga do sistema, entre outros. É baseado em LDAP3 e composto de duas partes, o Grid Resource Information Service (GRIS) e o Grid Index Information Service (GIIS).

Grid Security Infrastructure (GSI)

Provê serviço de autenticação do tipo *single sign on*, com suporte a controle local, delegação e mapeamento de credenciais. É usado para autenticação, autorização e delegação de credenciais para computações remotas. Emprega o protocolo de autenticação ITU X.509 (PITANGA, 2003), que especifica a utilização de certificados. A segurança baseada em certificados, credenciais que provêm identidade, conta exclusivamente com a confiança da Certificate Authority (CA), que provê a cada entidade seu próprio certificado (PITANGA, 2003). Após a autenticação ter sido realizada a comunicação do globus é desempenhada utilizando o protocolo Secure Socket Layer4 (SSL), que cifra as mensagens.

GridFTP

Componente chave para a transferência de arquivos de modo seguro e com alta performance. A palavra GridFTP pode se referir a um protocolo, um servidor ou a um conjunto de ferramentas. O protocolo é baseado no protocolo FTP5, mas o extende de modo que suporte as características da arquitetura da grade e outras ferramentas do Globus como o GSI.

Biblioteca Nexus

Biblioteca responsável pelos serviços de comunicação no conjunto de ferramentas Globus. Esta define uma interface de programação para suportar alguns paradigmas de programação importantes, como passagem de mensagens, chamadas de procedimento remoto, entre outras (DANTAS, 2003).

4 Proposta

Conforme o aumento de dados presentes em grades computacionais, têm se tornado bastante comum a prática de transferência de dados entre fontes de dados heterogêneas, especialmente quando novos protocolos são desenvolvidos. Muito trabalho vem sido aplicado para integrar tais fontes de dados. Tipicamente, usuários precisam utilizar um cliente para copiar dados de uma fonte para uma fonte intermediária, como, por exemplo, uma fonte de dados local, e em seguida utilizar outro cliente para mover os dados a partir da fonte intermediária para a fonte destino. Isso significa um passo extra na transferência de dados, o que é errado e ineficiente, considerando que o usuário necessita permanecer na frente do computador para efetuar a transferência.

A proposta de Zhang (2010) descreve uma abordagem diferente, que é capaz de converter qualquer interface de fonte de dados para a interface do GridFTP, que por sua vez é compatível com sistemas grade, e é capaz de potencializar as vantagens do protocolo com relação a transferência de dados. Ele resolve dois problemas com uma única solução. Primeiro, ele possibilita sistemas grade acessarem dados de qualquer fonte de dados. Segundo, transferir arquivos entre duas fontes de dados com protocolos distintos se torna possível sem a utilização de um ponto intermediário. A Figura 4.1 representa o funcionamento da mesma.

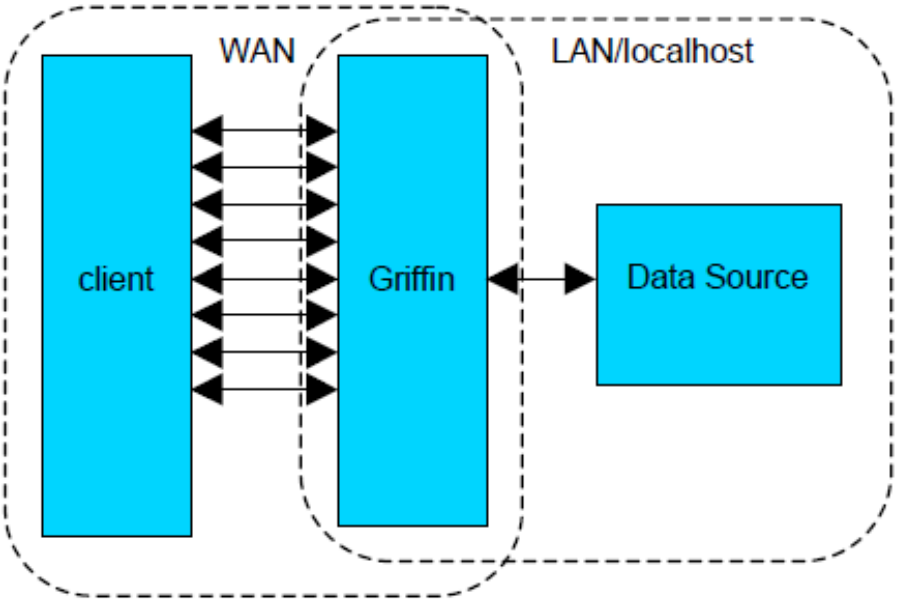

FIGURA 4.1: Modelo lógico do Griffin

A proposta deste trabalho consiste na modificação do código fonte do *Griffin* (Zhang S., 2010) de forma que a adição de adaptadores seja o mais simples possível. Conforme pode ser observado na Figura 4.2, a proposta modifica a arquitetura do *Griffin* adicionando uma nova camada de abstração ao seu código denominada *Service Adaptor*, cuja função é tornar o Griffin o mais flexível possível, ainda podemos destacar que tal camada implementa a interface *Generic file system framework* (Framework de sistemas de arquivos genérico) e os adaptadores de fontes de dados.

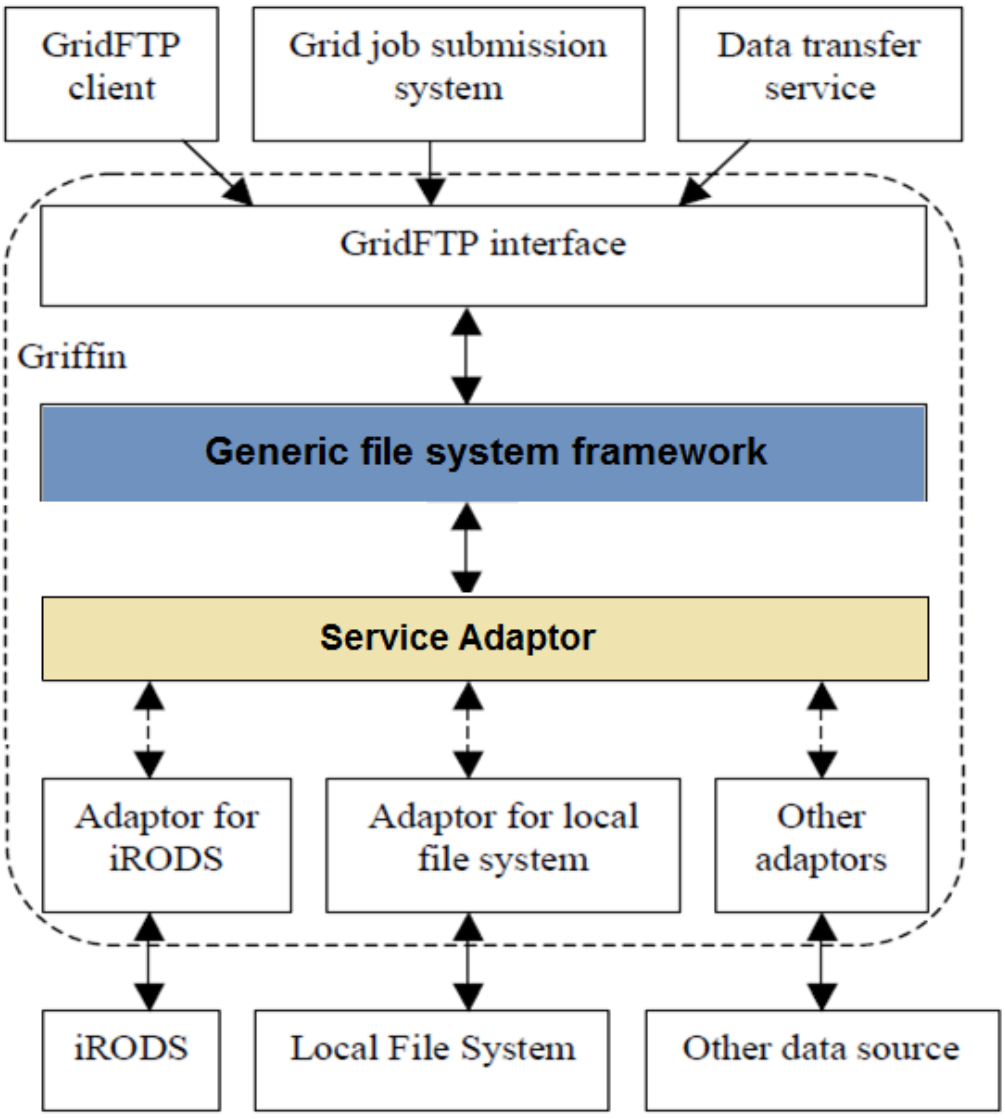

FIGURA 4.2 Arquitetura do Griffin

A implementação, ilustrada na Figura 4.2, foi desenvolvida em Java e baseado no framework Spring (ARTHUR, 2005), que oferece uma arquitetura para desenvolvedores facilmente desenvolver aplicações modulares. A interface do GridFTP é a base da aplicação. Ele gerencia todas as conexões do cliente com um parser para analisar comandos GridFTP e invocá-los com um classe de comando relevantes. A implementação da autenticação GSI é baseada em jGlobus (LASZEWSKI, 2001). Adaptadores para fontes de dados dependem da biblioteca de cliente relevante. Por exemplo, o adaptador para iRODS depende da biblioteca Java do iRODS, a *Jargon*. Tendo o Framework genérico torna-se possível desenvolver adaptadores para bibliotecas de outros sistemas de arquivos, como commons-vfs e JSAGA, de modo que o Griffin pode ser usado para acessar todas as fontes de dados que são suportados por essas bibliotecas. O binário de Griffin é leve, stand-alone e autônomo, com todas as bibliotecas java dependentes. Por isso, não depende de quaisquer componentes Globus, o que torna fácil de instalar e manter. Além disso, sendo uma aplicação Java, é possível ser executado na maioria dos sistemas operacionais sem recompilação. Por exemplo, se um usuário quiser enviar um job de grade para analisar os dados que estão localizados em uma área de trabalho em um computador desktop, uma instância de Griffin poderia ser executada neste ambiente de trabalho da máquina, não importa se é Windows ou Linux. Os Jobs submetidos pelo Globus podem então estagiar dados entrantes do desktop e estagiar dados de saída para o desktop, sem a necessidade de copiar dados de e para algum lugar que é acessível pelo nó de submissão de Jobs do Globus, como um servidor dedicado GridFTP, ou um espaço que é montado para os nós trabalhadores.

Com o design modular e com a ajuda do framework de injeção de dependências Spring (FOWLER, 2004), mudando o sistema de dados subjacente requer apenas uma mudança em um arquivo XML de configuração, sem a necessidade de recompilar todo o do sistema. Figura 4.3 é uma amostra de uma parte do arquivo de configuração do *Griffin* com o adaptador iRODS. Para executar Griffin com um sistema de arquivos local, só é preciso substituir a parte <*bean id = "filesystem"*> parte com a Figura 4.4, e reiniciar o Griffin.

```
<bean id="server"
class="au.org.arcs.griffin.server.impl.GsiFtpServer"
singleton="true">
   <property name="name" value="GSI FTP Server" />
   <property name="options" ref="options" />
   <property name="resources" value="griffin-resources"/>
   <property name="fileSystem" ref="fileSystem" />
</bean>
<bean id="fileSystem"
class="au.org.arcs.griffin.filesystem.impl.jargon.JargonFileSystemI
mpl" singleton="true">
   <property name="serverName" value="localhost" />
   <property name="serverPort" value="1247" />
   <property name="serverType" value="irods" />
</bean>
```

FIGURA 4.3 Exemplo de configuração com adaptador para iRODS

```
<bean id="fileSystem"
class="au.org.arcs.griffin.filesystem.impl.localfs.LocalFileSystemI
mpl" singleton="true">
   <property name="rootPath" value="/data/data-fabric" />
   <property name="userManager" ref="userManager" />
</bean>
```

FIGURA 4.4 Exemplo de configuração com adaptador para sistema de arquivos local

Para acessar sistemas de arquivos arbitrários de uma maneira uniforme, foi criada a interface *Generic file system framework* e em seguida a camada que implementa seus métodos a *Service Adaptor* que funciona como back-end para a interface GridFTP. Por meio da camada *Service Adaptor*, qualquer pedido proveniente da interface GridFTP será traduzido para uma operação padrão do sistema de arquivos. Para torná-lo genérico, a camada é leve e simples, suportando apenas operações básicas e comumente implementadas pela interface GridFTP. A concepção desta camada é específica para as necessidades do protocolo GridFTP, especialmente recursos avançados para transferência de dados, a fim de minimizar a sobrecarga entre a interface front-end do GridFTP e o sistema de dados no back-end. Como o protocolo GridFTP é usado principalmente para transferência de dados, esta camada não requer outras funções, tais como manipulação de meta-dados.

A estrutura top-down simples é ilustrada na Figura. 4.5, com quatro grandes objetos. O superior é o único caso no sistema que correspondente à fonte de dados, com todas as configurações e controles a nível de fonte de dados. Cada usuário, uma vez autenticado, será associado a um objeto de conexão, que é o segundo

objeto a partir do topo. O objeto conexão mantém todas as informações do usuário, pois diferentes usuários possuem uma visão diferente da fonte de dados, com diferentes permissões. Assim, o objeto de conexão pode determinar se um objeto de arquivo, o terceiro de cima para baixo, pode ser criado para um determinado caminho, e que as funções podem ser chamadas, com base nas permissões do usuário. Uma vez que um objeto de arquivo é criado, é usado para recuperar informações do arquivo de destino, ou fazer algumas operações sobre ele. O último objeto é usado principalmente para acessar o conteúdo de um objeto de arquivo, desde que o objeto de arquivo tenha sido instanciado com permissões adequadas. Um adaptador para um sistema de grade de dados deve implementar todas as quatro interfaces.

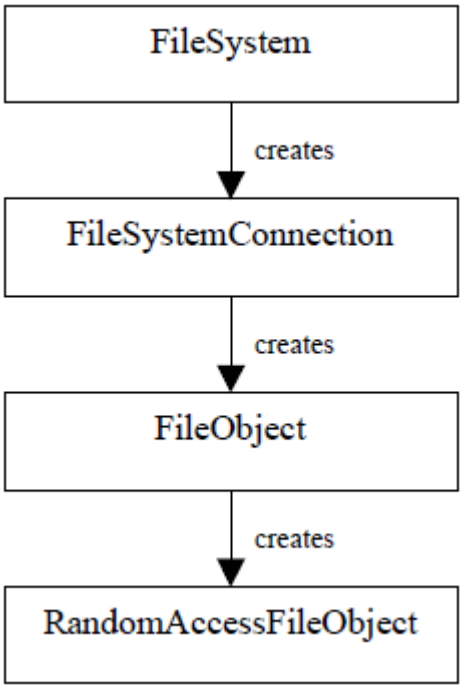

FIGURA 4.5 A estrutura da camada Service Adaptor

O objeto *FileSystem* é a raiz da hierarquia. Ele possui os métodos mostrados na Figura. 4.6. Este objeto apresenta o sistema subjacente de dados na interface GridFTP e é chamado na inicialização e desligamento. Em particular, o método init () é chamado para examinar as configurações e verificar a conectividade com a fonte de dados durante a inicialização. Se houver um erro, por exemplo, a configuração não está correta ou o sistema subjacente de dados não está acessível, uma exceção será lançada, assim o servidor GridFTP pode decidir se ele pode continuar a ser

executado. Da mesma forma, o método *exit()* é chamado durante o desligamento para liberar recursos e encerrar com o sistema de dados se houver. O método *GetSeparator()* retorna o separador usado em caminhos já que ele pode ser diferente em alguns sistemas, como a barra invertida no Windows e barra em Linux. O Método *createFileSystemConnection()* será executado quando um usuário se conectar à interface GridFTP e uma sessão de GridFTP criada para esse usuário. Correspondentemente, uma conexão com o sistema de dados subjacente será criada, de acordo com a credencial GSI. Este método retorna um objeto *FileSystemConnection*, que representa uma sessão para o sistema de dados. Se o sistema de dados subjacentes suporta autenticação GSI, a mesma credencial GSI utilizada para autenticar com a interface GridFTP será usada para autenticar com o sistema de dados subjacente. No entanto, se o sistema de dados subjacente não suportar à autenticação GSI, algum tipo de mecanismo de mapeamento será empregado.

```
public String getSeparator();
public void init() throws IOException;
public FileSystemConnection
    createFileSystemConnection(GSSCredential credential) throws
    FtpConfigException, IOException;
public void exit();
```

FIGURA 4.6 Interface FileSystem

FileSystemConnection mantém uma conexão com o sistema de dados subjacente para o usuário. Ela consiste de alguns métodos para todo o sistema e relacionadas ao usuário. *getHomeDir()* e *getFreeSpace()* são auto-descritivos. *getUser()* retorna o nome de usuário do sistema subjacente associado a este sessão. *isConnected()* indica se a conexão está aberta, enquanto *close()* irá encerrar a conexão atual. O Método *getFileObject()* gera um FileObject para um caminho particular no contexto do sistema GridFTP. Todos os métodos podem ser conferidos na Figura 4.7.

```
public FileObject getFileObject(String path);
public String getHomeDir();
public String getUser();
public void close() throws IOException;
public boolean isConnected();
public long getFreeSpace(String path);
```

FIGURA 4.7 Interface FileSystemConnection

FileObject representa um objeto de dados na fonte de dados. Ele é projetado para refletir a classe *java.io.File*, fornecendo métodos para obter o nome do arquivo, caminho, caminho canônico, comprimento, hora da última modificação, o seu pai ou seus filhos. Além disso, ele permite aos usuários verificar se esse arquivo existe, ou se este objeto é um arquivo ou um diretório. Os usuários também podem usar esse objeto para fazer um novo diretório, apagar-se, mudar o nome em si, ou definir a hora da última modificação. O método *GetPermission()* retorna um número abstrato de permissão, que só pode ser sem acesso (0), o privilégio ler (1), o privilégio escrever (2) ou de leitura/escrita (3). Como o protocolo GridFTP não requer a manipulação de permissões, todos os tipos de permissões são selecionadas para a permissão do usuário atual. O que é exigido no *Service Adaptor* é descobrir as permissões para o usuário associado à conexão atual. Assim, retornando uma permissão simples numerada é a maneira mais simples e genérica para abordar esta questão, e esconder as complexidades de diferentes sistemas dados de grade. O Método *getRandomAccessFileObject()* retorna um *RandomAccessFileObject*, como mostrado na Figura 4.9, para acessar o conteúdo deste objeto arquivo. Todos os métodos podem ser encontrados na Figura 4.8.

RandomAccessFileObject fornece métodos para ler e escrever conteúdo no *FileObject* correspondente. O design é semelhante ao *java.IO.RandomAccessFile*, para os desenvolvedores. Vários métodos *read()* e *write()* constituem a maior parte deste objeto para ler o conteúdo do arquivo para um array de bytes ou escrever o conteúdo de um array de bytes para o arquivo. Outros métodos incluem *seek()* para saltar para um determinado ponto no objeto de dados e *length()* para retornar o comprimento do objeto de dados. Ao contrário dos objetos streaming, que podem apenas ler ou escrever em seqüência, o *RandomAccessFileObject* oferece um caminho para o front-end da interface GridFTP que permite facilmente ler ou gravar dados de qualquer posição no arquivo, que é útil para o modo de bloqueio estendido. Consequentemente, a fonte de dados subjacente deve suportar acesso aleatório para que ele possa ser conectado a este framework.

```
public String getName();
public String getPath();
public boolean exists();
public boolean isFile();
public boolean isDirectory();
public int getPermission();
public String getCanonicalPath() throws IOException;
public FileObject[] listFiles();
public long length();
public long lastModified();
public RandomAccessFileObject getRandomAccessFileObjec(String
    type) throws IOException;
public boolean delete();
public FileObject getParent();
public boolean mkdir();
public boolean renameTo(FileObject file);
public boolean setLastModified(long t);
```

FIGURA 4.8 Interface FileObject

```
public void seek(long offset) throws IOException;
public int read() throws IOException;
public int read(byte[] b) throws IOException;
public int read(byte[] b, int off, int len) throws IOException;
public void close() throws IOException;
public String readLine() throws IOException;
public void write(int b) throws IOException;
public void write(byte[] b) throws IOException;
public void write(byte[] b, int off, int len) throws IOException;
public long length() throws IOException;
```

FIGURA 4.9 Interface RandomAccessFileObject

5 Experimentos

5.1 Ambiente Experimental

O ambiente de testes montado consiste em máquina desktop (PC) e um netbook; para fins do experimento no desktop foi instalado o ubuntu server versão 9.04, um simulador de ambiente grade, o GriSim (2010), e rodando sobre ele o Globus Toolkit. Os testes foram conduzidos em um ambiente controlado, e as transferências realizadas em uma rede residencial via intranet, com link de velocidade de 100,00 MBytes. A configuração completa das máquinas pode ser observada a seguir:

- **Ubuntu Server (9.04)**
  - Processador: Core intel i5, 2.90GHz
  - Disco Rígido: 500 GB
  - Memória RAM: 4 GB
- **Ubuntu Client (11.04)**
  - Processador: Atom 1,33GHz de CPU
  - Disco Rígido: 320 GB
  - Memória RAM: 2GB

5.2 Execução e Resultados

Estes experimentos têm por objetivo demonstrar que o desempenho das transferências não é afetado pela utilização do *Griffin*. No primeiro teste, foi comparado a taxa de transferência do protocolo original com GridFTP via *Griffin*, ilustrado na Figura 5.1.

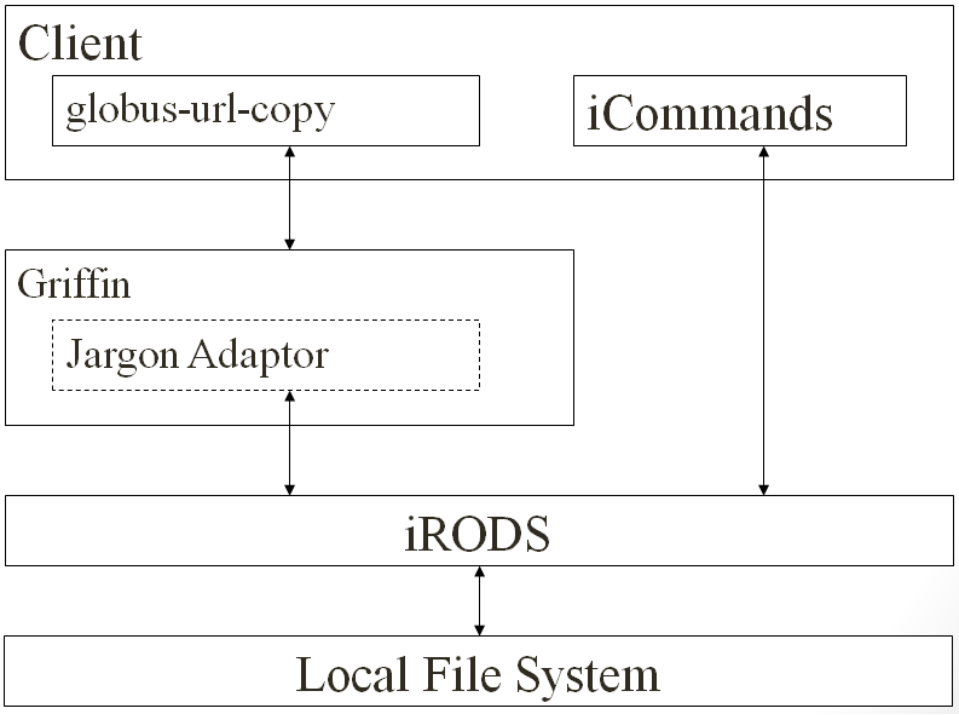

FIGURA 5.1 Arquitetura para experimento (Griffin x iCommands)

IRODS 2.1 foi instalado como fonte de dados. *Griffin* foi configurado no servidor com 786MB de tamanho de pilha como a interface GridFTP para este iRODS. Com esta configuração, os dados em iRODS podem ser acessado via interface nativa iRODS ou interface GridFTP. Foram comparamos a taxa de upload e taxa de download entre Servidor e cliente, com tamanhos de arquivo de 256MB, 512MB, 1GB, 2GB, 4GB, 8GB a 16GB, e oito threads são usadas tanto para transferências de iCommands quanto *Griffin*.

Tendo cada teste sido executado 5 vezes, o resultado na Figura 5.2 mostra a taxa de transferência média em Kbytes por segundo. A partir da figura, ao fazer upload de um arquivo, iCommands e *Griffin* executam a uma taxa semelhante, para arquivos menores que 8 GB, no entanto, se o arquivo é de 8GB ou mais, *Griffin* é mais rápido que iCommands.

Ao fazer o download, *Griffin* funciona de forma constante para os diversos tamanhos no teste, enquanto iCommands executa mais rápido para arquivos pequenos, mas fica mais lento conforme o tamanho do arquivo aumenta.

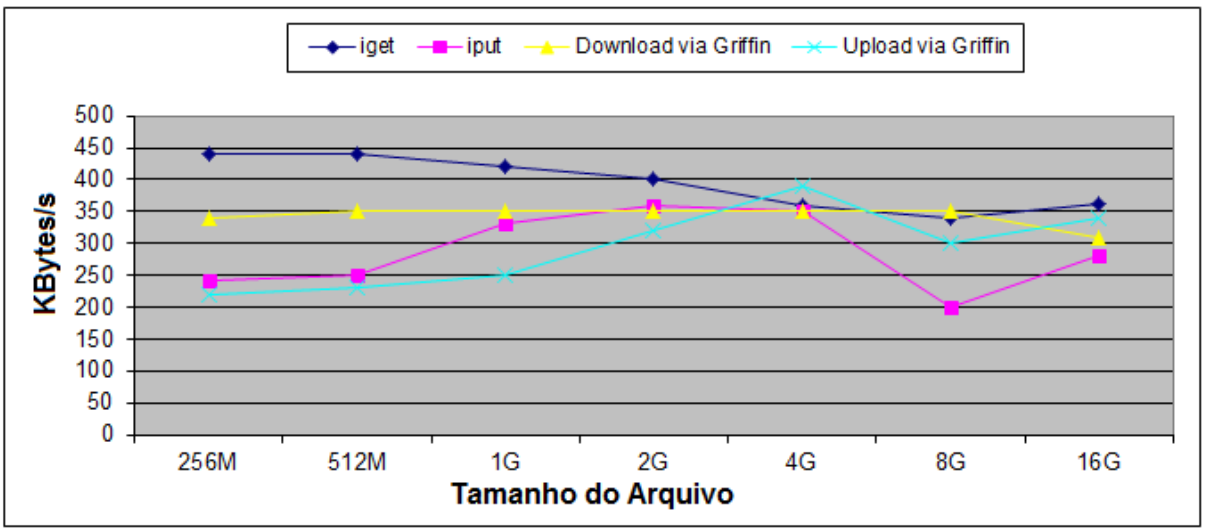

FIGURA 5.2 Comparação entre iCommands e Griffin

No segundo teste, ilustrado pela Figura 5.3, é comparar *Griffin* com o servidor Globus GridFTP, no mesmo teste ambiente, como os testes acima. O servidor Globus GridFTP foi instalado no servidor a partir VDT 1.10.1. Griffin foi reconfigurado com um adaptador de sistema de arquivos local, de modo que lê e grava dados no sistema de arquivos local.

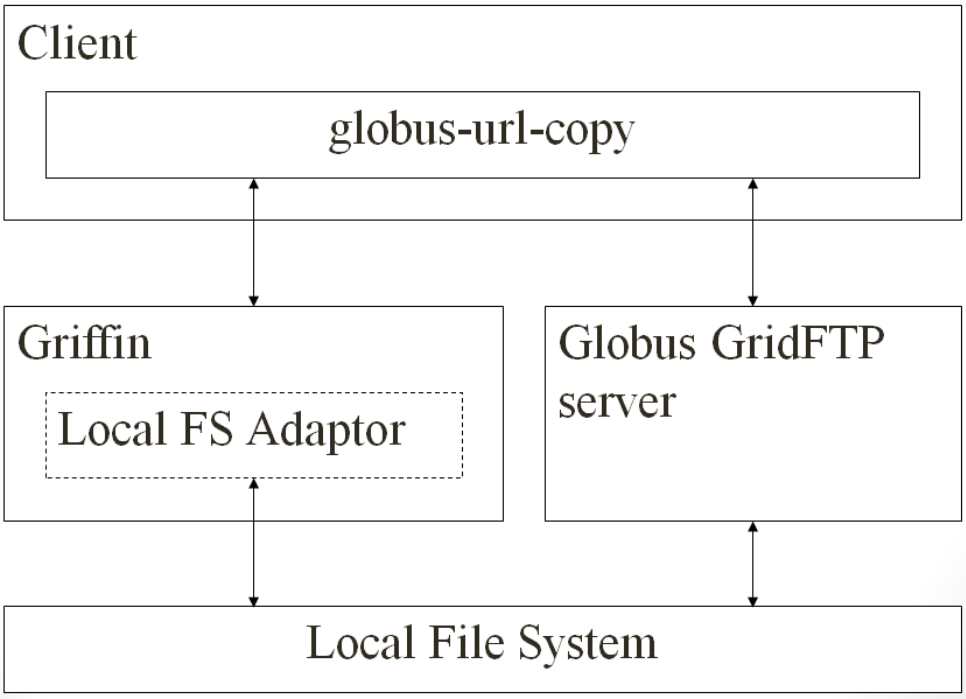

FIGURA 5.3 Arquitetura para experimento (Griffin x Globus GridFTP)

Neste teste, os dados de testes são armazenados na mesma partição que o recurso iRODS. Semelhante ao primeiro teste, foram transferidos arquivos com tamanhos de 256M, 512M, 1G, 2G, 4G, 8G e 16G, via Griffin e Globus GridFTP, com

16 threads. Cada teste de transferência foi feito 5 vezes e os valores médios são mostrados na Figura 5.4. O resultado mostra que Griffin executa muito próximo do servidor Globus GridFTP em upload e download de dados.

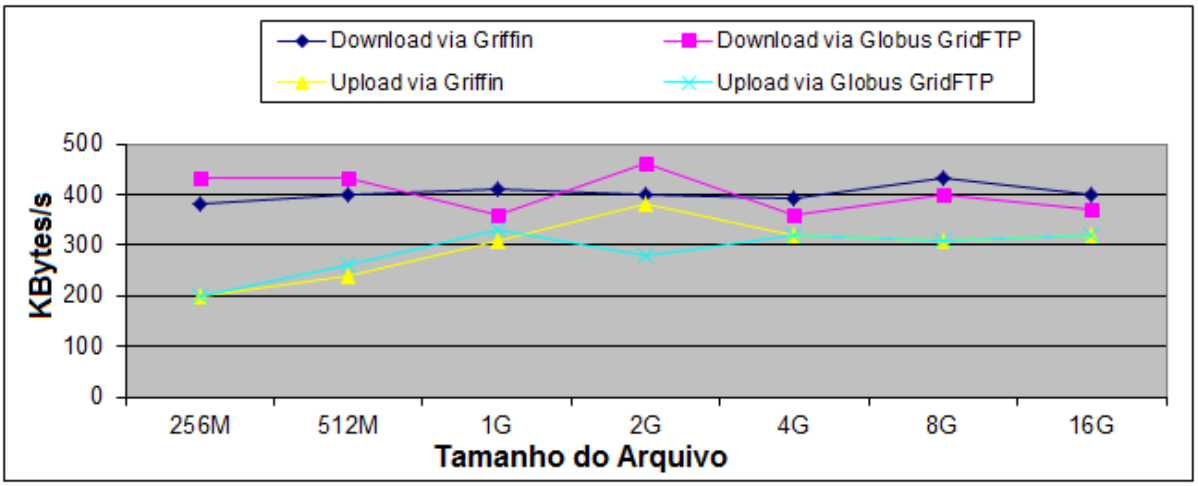

FIGURA 5.4 Comparação entre Servidor Globus GridFTP e Griffin

6 Conclusão e Trabalhos Futuros

A abordagem utilizada neste trabalho é relativamente nova, sendo que a mesma foi apresentada em outubro de 2010 para a comunidade científica. Sendo uma abordagem recente ainda há muito em que se trabalhar, este trabalho, por exemplo, modificou o protótipo inicial para facilitar a criação e novos adaptadores.

Neste trabalho foi identificado um impasse na forma como dados são transferidos entre ambientes heterogêneos de alto desempenho, e durante o levantamento do estado da arte foi-se identificada a proposta do *Griffin*.

O *Griffin* representa uma abordagem genérica, leve e fácil de usar para se conectar a uma fonte de dados arbitrários para a grade, usando um quadro genérico de sistema de arquivo com uma interface GridFTP. Ele permite que qualquer fonte de dados possa ser acessada pela grade, permite ainda a transferência de grandes conjuntos de dados, tem muito pouco impacto sobre o desempenho para o transporte de dados, conforme observado nos experimentos, e torna possível mover dados entre fontes de dados com diferentes protocolos facilmente.

A implementação é baseada em Java sendo uma alternativa para o servidor Globus GridFTP; ela é autônoma, não depende da pilha de software Globus, e pode ser executado na maioria dos sistemas operacionais. É adequado para fontes de dados que ofereçam uma API Java.

No futuro, suporte UDT será adicionado ao *Griffin*, e mecanismos de verificação serão implementados para verificar a integridade de arquivos copiados. Além disso, múltiplos fluxos do Griffin para o fonte de dados será investigada em casos onde a fonte de dados suporte múltiplos fluxos. Striping (segmentação) de transferências é outra recurso útil a se ter no futuro. Ele permite o envio de um enorme arquivo a partir de vários servidores GridFTP com cada um enviando uma parte do arquivo para obter melhor desempenho do que o uso de apenas um servidor GridFTP.

Referências Bibliográficas

ALLCOCK, W.; BESTER, J.; BRESNAHAN, J; CHERVENAK, A.; LIMING, L.; TUECKE, ST. GridFTP: Protocol extensions to FTP for the grid. Technical report, ANL, March 2001. Internet Draft. Available at http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf.

ALLCOCK, B.; BESTER, J.; BRESNAHAN, J.; CHERVENAK, A.L.; FOSTER, I.; KESSELMAN, C.; MEDER, S.; NEFEDOVA, V.; QUESNEL, D.; TUECKE, S. Data Management and Transfer in High-Performance Computational Grid Environments. Parallel Computing, 28(5):749–771, May 2002.

ALLCOCK, W. et al., “GridFTP: Protocol extensions to FTP for the Grid,” GGF Document Series GFD.20, Apr. 2003. Disponível em http://www.gridforum.org/GFD.20.pdf.

ANGLANO, C.; CANONICO, M. The File Mover: an efficient data transfer system for Grid applications, 27 set. 2004.

ANGLANO, C.; CANONICO, M. A comparative evaluation of high-performance file transfer systems for data-intensive grid applications, 14 jan. 2005.

ARTHUR, J.; AZADEGAN, S., "Spring Framework for rapid open source J2EE Web Application Development: A case study," in Proc of the Sixth Int Conf on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing(SNPD’05), 2005.

Automatic TCP Window Tuning and Applications. Disponível em http://dast.nlanr.net/Projects/Autobuf v.10/autotcp.html. Acessado em Junho de 2011.

BAKER, M.; BUYYA, R.; LAFORENZA, D. The grid: International efforts in global computing. In International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000), Rome, Italy, July 31 – August 6 2000. Also available at http://www.csse.monash.edu.au/rajkumar/papers/TheGrid.pdf.

BARU, C.; MOORE, R.; RAJASEKAR, A.; et al., "The SDSC Storage Resource Broker," in Proc of the Centre for Advanced Studies on Collaborative Research (CASCON), 1998.

BBFTP. The bbFTP – Large Files Transfer ProtocolsWeb Site. Disponível em http://doc.in2p3.fr/bbftp. Acessado em Junho de 2011.

BERGUA, B.; GARCIA-CARBALLEIRA, F.; CALDERON, A.; SANCHEZ, L.M.; CARRETERO, J. Comparing grid data transfer technologies in the Expand parallel file system, 25 fev. 2008.

BUYYA, R. High Performance Cluster Computing: Architectures and Systems. [S.l.]: Axcel Books, 2003.

CERN. GFAL: Grid File Access Library. Disponível em http://griddeployment.web.cern.ch/griddeployment/gis/GFAL/GFALindex.html. Acessado em Junho de 2011.

CERN. glite. Disponível em http://glite.web.cern.ch/glite/. Acessado em Junho de 2011b.

CHANDRA, B. et al. End-to-End WAN Service Availability. In Proc. of 3rd Usenix Symp. on Inremet Technologies and Systems (USITS), pages 97-108, San Francisco, CA, Februaly 2001.

CHERVENAK, A.; FOSTER, I.; KESSELMAN, C.; SALISBURY, C.; TUECKE, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. Journal of Network and Computer Applications, 23:187–200, 2001.

COLVERO, T. A.; DANTAS, M.; CUNHA, D. P. da. Ambientes de clusters e grids computacionais: Características, facilidades e desafios. UNESC - Criciúma, 2005.

Condor. Disponível em http://www.cs.wisc.edu/condor/. Acessado em Maio de 2011.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. Distributed Systems: Concepts and Design. 4a . ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321263545.

DANTAS, M.; ALLEMAND, J.; PASSOS, L. An Evaluation of globus and Legion Software Environments. Disponível em: http://hpcs2003.ccs.usherbrooke.ca/papers/Dantas_01.pdf. Acesso em: Junho de 2011.

DANTAS, M. A. R. Computação Distribuídas de Alto Desempenho: Redes, Clusters e Grids Computacionais. Rio de Janeiro, RJ, Brasil: Axcel Books, 2005. ISBN 8573232404.

DICKENS, P.M. FOBS: A Lightweight Communication Protocol for Grid Computing. In Proc. of Europar 2003, Klagenfurt, Austria, August 2003.

DUNIGAN, T.; MATHIS, M.; Tierney, B. “A TCP tuning daemon,” in Proceedings of SuperComputing: High-Performance Networking and Computing, Nov. 2002.

ELZ, R.; HETHMON, P. FTP security extensions. Request for Comments (RFC) 2228, Oct. 1997.

ERNST, M.; FUHRMANN, P.; GASTHUBER, M.; et al., "dCache, a distributed data storage caching system," in Computing in High Energy and nuclear Physics (CHEP2001), Beijing, 2001.

FERREIRA, L., BERTIS, V., ARMSTRONG, J., KENDZIERSKI, M., et al. Introduction to Grid Computing with Globus. Disponível em:

http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf Acesso em: Dezembro de 2003.

FLOYD, S. “Highspeed TCP for large congestion windows,” Internet Draft draft-ietf-tsvwg-highspeed-01.txt, Aug. 2003.

FLOYD, S. HighSpeed TCP for Large Congestion Windows. RFC 3649, Dec. 2003b.

FOSTER, I. What is the grid? a three point checklist. p. 22–25, 2002.

FOSTER, I.; KESSELMAN, C. The Grid 2: Blueprint for a new Computing Infrastructure. [S.l.]: John Wiley and Sons Ltd, 2003.

FOSTER, I., KESSELMAN, C. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Aplications 11, 2, 115-128. Disponível em: ftp://ftp.globus.org/pub/globus/papers/globus.pdf. Acesso em: Novembro de 2003b.

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of Supercomputer Applications, 15(3), 2001. Disponível em http://www.globus.org/research/papers/anatomy.pdf. Acessado em Junho de 2011.

FOSTER, I., KESSELMAN, C., TUECKE, S. e NICK, J. The Physiology of the Grid: An Open Grid Services Architeture for Distributed Systems Integration. Disponível em: http://www.globus.org/research/papers/ogsa.pdf. Acesso em: Novembro de 2003c.

FOWLER, M.(2004). Inversion of Control Containers and the Dependency Injection pattern.(29 April 2010). http://www.martinfowler.com/articles/injection.html.

Global Grid Forum. Disponível em http://www.ggf.org/. Acessado em Junho de 2011.

Globus Toolkit. Disponível em http://www.globus.org/. Acessado em Junho de 2011b.

GridData. Disponível em www.cos.ufrj.br/~kayser/cos833/gridData.pdf. Acessado em Maio de 2011.

GridFTP: Universal Data Transfer for the Grid. Disponível em http://www.globus.org. Acessado em Junho de 2011.

GriPhyN .The Grid Physics Networks project. Disponível em http://www.griphyn.org. Acessado em Junho de 2011.

GridPP and PPARC. Slashgrid. . Disponível em http://www.gridsite.org/slashgrid/. Acessado em Junho de 2011.

GridSim, Disponível em http://www.buyya.com/papers/gridsim.pdf. Acessado em Outubro de 2011.

GU, Y.; GROSSMAN, R.L. Using UDP for Reliable Data Transfer over High Bandwidth-Delay Product Networks. Disponível em http://www.dataspaceweb.net/papers.html, 2003. Submitted for publication.

GU, Y.; HONG, X.; MAZZUCCO, M.; GROSSMAN, R.L. SABUL: A High Performance Data Transfer Protocol. Disponível em http://www.dataspaceweb.net/papers.htm, 2003b. Submitted for publication.

HACKER, T. J.; ATHEY, B. D. “The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network,” in Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS), Aug. 2001.

HANUSHEVSKY, A.; TRUNOV, A.; COTTRELL, L. Peer-to-Peer Computing for Secure High Performance Data Copying. In Proc. of the 2001 Int. Conf. on Computing in High Energy and Nuclear Physics (CHEP 2001), Beijng, China, September 2001.

HE, E. ; LEIGH, J.; YU, O.; DEFANTI, T. Reliable Blast UDP: Predictable High Performance Bulk Data Transfer. In Proc. of 5th Int. Conf. on Cluster Computing, 2002.

HETHMON, P.; ELZ, R. Feature negotiation mechanism for the file transfer protocol. Request for Comments (RFC) 2389, Aug. 1998.

ITO, T. OSHSAKI, H. IMASE, M. On parameter tuning of data transfer protocol GridFTP for wide-area grid computing, 13 fev. 2006.

KELLY, T. Scalable TCP: Improving Performance in HighspeedWide Area Networks. In Proc. of First Int. Workshop on Protocols for Fast Long-Distance Networks, CERN, Geneva, Switzerland, February 2003.

KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. Software: Practice and Experience (SPE) Journal, 32(2), February 2002. Also available at http://www.csse.monash.edu.au/rajkumar/papers/gridtaxonomy.pdf.

LABOVILZ, C.; AHUJA, A.; BOSE, A.; JAHANIAN, F. Delayed Internet Routing Convergence. In Pmc. of ACM SIGCOMM. pages 175-187, Stockolm. Sweden, September 2000.

LASZEWSKI, G. v.; FOSTER, I.; GAWOR, J., et al., "A Java Commodity Grid Toolkit," Concurrency: Practice and Experience, vol. 13, 2001.

LHC. The Large Hadron Collider project. Disponível em http://lhc.web.cern.ch/lhc. Acessado em Junho de 2011.

LU, D.; QUAO, Y.; DINDA, P.; BUSTAMENTE, F. “Modeling and taming parallel TCP on the wide area network,” in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Apr. 2005.

MANDRICHENKO, I.; ALLCOCK, W.; PERELMUTOV, T. GridFTP v2 protocol description. GGF Document Series GFD.47, May 2005. Disponível em http://www.ggf.org/documents/GFD.47.pdf.

MATHIS, M.; MAHDAVI, J.; FLOYD, S.; Romanow, A. TCP Selective Acknowledgment. RFC 2018. Disponível em http://www.faqs.org/rfcs/rfc2018.html. Acessado em Junho de 2011.

Nirvana Storage. Storage resource broker. Disponível em http://www.nirvanastorage.com. Acessado em Junho de 2011.

Open Grid Forum. Disponível em: http://www.ogf.org. Acessado em Junho de 2011.

OpenMosix, Disponível em: http://sourceforge.net/projects/openmosix/. Acessado em Maio de 2011.

PAXSON, V. End-to-End Routing Behavior in the Internet. In Pmc. of ACM SIGCOMM, pages 2s-38, Stanford, CA, Ausus1 1996.

PAXSON, V. end-to-End Internet Packet Dynamics. In Proc. ACM SIGCOMM, pages 139-152, Canncs. France, September 1997.

PEREIRA, N. "*Linux, Clusters e Alta Disponibilidade**", São Paulo, 2002. Dissertação de Mestrado,*** Universidade de São Paulo.

PITANGA, M. Computação em Cluster – O Estado da Arte em Computação. Rio de Janeiro: Editora Brasport, 2003.

PPDG. The Particle Physics Data Grid Project. Disponível em http://www.cacr.caltech/edu/ppdg. Acessado em Junho de 2011.

POSTEL, J. "Transmission control protocol," Request for Comments (RFC) 793, Sept. 1981.

POSTEL, J.; REYNOLDS, J. File transfer protocol (FTP). Request for Comments (RFC) 959, Oct. 1985.

QIU, L.; ZHANG, Y.; KESHAV, S. "On individual and aggregate TCP performance," in Proceedings of Internetl Conference on Network Protocols, Oct. 1999, pp. 203–212.

RAJASEKAR, A.; WAN, M.; MOORE, R.; et al., "A Prototype Rule-based Distributed Data Management System," in High Performance Parallel and Distributed Computing (HPDC), Paris, França, 2006.

RANGANATHAN, K.; FOSTER, I. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. Journal of Grid Computing, 1(1):53–62, 2003.

RAO, N.S.V. Multiple Paths for End-to-End Delay Minimization in Distributed Computing Over Internet. In Proc. of Supercomputing 2001,2001.

SAVAGE, S. et al. Detour: A Case for Informed Internet Routing and Transpon. IEEE Micro, 19(1):50-59, Jan. 1999.

SAVAGE, S. et al. The End-to-End Effects of Intemet Path Selection. In Proc. of ACM SIGCOMM, pages 289-299, Boston, MA, 1999b.

SEMKE, J.; MAHDAVI, J.; MATHIS, M. “Automatic TCP buffer tuning,” in Proceedings of ACM SIGCOMM ’98, vol. 28, Oct. 1998.

SIVAKUMAR, H.; BAiLEY, S.; GROSSMAN, R. L. “PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks,” in Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, Nov. 2000.

Sybase Inc. Avaki. Disponível em http://www.sybase.com. Acessado em Junho de 2011.

TANENBAUM, A. S. Modern Operating Systems. Englewood Cliffs: Prentice Hall, 1992.

TANENBAUM, A. S.; STEEN, M. V. Distributed Systems: Principles and Paradigms. 2a . ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007. ISBN 0132392275.

TANGMUNARUNKIT, H.; GOVINDAN, R.; SHERKER, S.; ESLRIN, D. The Impact of Routing Policy on Internet Paths. In Proc. IEEE Infocom '01, Anchorage, Alaska, April 2001.

TATEBE, O.; SODA, N.; MORITA, Y.; MATSUUOKA, S.; SEKIGUCHI, S. Gfarm v2: A grid file system that supports high-performance distributed and parallel data computing. In Proceedings of the 2004 Computing in High Energy and Nuclear Physics (CHEP04), Switzerland, Sept. 2004.

The Globus Project. GridFTP update January 2002, 2002. Disponível em: http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf.

The Globus Project. GridFTP: universal data transfer for the Grid. White Paper, Sept. 2003. Diisponível em: http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf.

THULASIDASAN, S.; FENG, W.; GARDNER, M. K. “Optimizing GridFTP through dynamic right-sizing,” in Proceedings of IEEE International Symposium on High Performance Distributed Computing, June 2003.

VMWARE. 2011. Acessado em: Maio de 2011. Disponível em: http://www.vmware.com.

WALLACE, S. Tsunami File Transfer Protocol. In Proc. of First Int. Workshop on Protocols for Fast Long-Distance Networks, CERN, Geneva, Switzerland, February 2003.

WANG, L. et al. Scientific cloud computing: Early definition and experience. In: IEEE. High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on. [S.l.], 2008. p. 825–830.

Web100. The Web100 Project. Disponível em http://www.web100.org/. Acessado em Junho de 2011.

WEB2.0. 2011. Acessado em: agosto de 2011. Disponível em: http://en.wikipedia.org/wiki-/Web 2.0.

WHITE, B. S.; WALKER, M.; HUMPHREY, M.; GRIMSHAW, A. S.. Legionfs: A secure and scalable le system supporting crossdomain high-performance applications. In Supercomputing 2001, Denver, USA, November 2001. Disponível em http://legion.virginia.edu/papers/SC2001.pdf. Acessado em Junho de 2011.

WU, R.X.; Chien, A. GTP: Group Transport Protocol for Lambda-Grids. In Proc. of 4th ACM/IEEE Int. Symp. on Cluster Computing and the Grid, Chicago,USA, 2004.

YEO, C. S. et al. Cluster computing: High-performance, high-availability, and high-throughput processing on a network of computers. 2006.

ZHANG, S.; CODDINGTON, P.; WENDELBORN, A.; Connecting arbitrary data resources to the grid. In Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on Oct. 2010.

Apêndices

APÊNDICE A – Artigo sobre o TCC

DESENVOLVIMENTO DE UM COMPONENTE PARA TRANSFERÊNCIA DE ARQUIVOS EM AMBIENTES DE ALTO DESEMPENHO HETEROGÊNEOS

Izaias de Faria, M.A R. Dantas

Departamento de Informática e Estatística (INE)

Universidade Federal de Santa Catarina (UFSC)

{izaiasdf, mario}@inf.ufsc.br

Abstract. *Currently, Different types of grids have been used in environments where you need large processing power and data storage. In this context, we observed different grid managers that use different communication protocols. Regardless of the manager in use it is required to send files to the grid. For this process, the large number of different protocols in use, especially private ones, represents a transfer of data deadlock. The objective of this work is to develop a component that can communicate with different managers, making it possible to transfer files regardless of the protocol used.*

Resumo. *Atualmente, diferentes tipos de grades vêm sido utilizadas em ambientes onde é necessário grande capacidade de processamento e armazenamento de dados. Neste contexto, foram observados diferentes gerenciadores de grades que utilizam diferentes protocolos de comunicação. Independente do gerenciador se faz necessário enviar arquivos para a grade. Para este processo, o grande número de diferentes protocolos em utilização, em especial os privados, representa um impasse à transferência de dados. O objetivo deste trabalho é desenvolver um componente capaz de se comunicar com diferentes gerenciadores, tornando possível a transferência de arquivos independendo do protocolo utilizado.*

1. Introdução

Sistemas de grade científica são ferramentas importantes para pesquisadores analisarem, armazenarem e compartilharem dados. A maioria destes sistemas fornece não apenas os serviços computacionais, mas também serviços de dados. Entre os sistemas existentes de grade científica, muitos dos serviços computacionais são construídos em Globus Toolkit [Foster 2003], ou suas variantes, tais como gLite [Cern, 2011]. Alguns outros grupos adotam Condor (2011), um sistema de agendamento diferente, que pode funcionar com Globus via Condor-G. No entanto, o mecanismo usado para armazenamento de dados e acesso não é único.

Alguns sistemas de dados da rede são projetados para serem sistemas de arquivos virtuais, oferecendo espaço de armazenamento a pesquisadores sem que usuários saibam o exato lugar de dados, como SRB [Baru 1998] e seu sucessor, iRODS [Rajasekar 2006], bem como dCache [Ernst 2001]. A maneira de acessar esses sistemas de grades de dados varia. Por exemplo, dCache fornece uma interface GridFTP e uma interface HTTP, de modo que sistemas de submissão a grade possam facilmente acessar dados a partir dele. Alguns outros, como iRODS, implementam os seus próprios protocolos proprietários e sistemas de envio de trabalhos a grade não têm como obter dados diretamente fora dele.

Como a quantidade de dados está aumentando, torna-se um requisito comum transferir dados entre fontes de dados heterogêneas. Especialmente quando um novo protocolo é desenvolvido, existe um trabalho considerável a ser feito para integrá-lo no sistema existente. Normalmente, os usuários necessitam usar um cliente para cópia de dados da fonte para um espaço intermediário, como um espaço de armazenamento local, e então utilizar outro cliente para mover dados a partir do espaço intermediário para o destino. Este é um método errado e ineficiente. Por existir uma etapa extra, o usuário deve se sentar frente a um terminal ou escrever um script para automatizar todo o processo. Se algo inesperado ocorrer, o usuário necessita realizar depuração do trabalho para descobrir o erro.

Nesse contexto temos um cenário onde é necessário cada vez mais que a grade esteja preparada para transferir arquivos potencialmente grandes para uma grande possibilidade de clientes, seja utilizando um cliente com protocolo proprietário, ou no pior dos casos, para outra grade com gerenciador diferente no utilizado na fonte. O segundo caso representa um grande problema para a comunidade desenvolvedora que alimenta a transferência de arquivos entre grades computacionais, uma vez que as grades são dependentes fixamente dos protocolos utilizados para seu desenvolvimento.

Este artigo apresenta um componente para transferência de arquivos em arquiteturas de grades computacionais heterogêneas. O componente é capaz de adaptar-se a diferentes tipos de fontes de dados, para tal utiliza o protocolo padrão para comunicação com grades, o GridFTP. Após desenvolvimento do componente o mesmo foi testado e seu desempenho avaliado comprovando sua viabilidade com relação a alternativas utilizadas atualmente no mercado.

2. Trabalhos Relacionados

TCP

Transmission Control Protocol (TCP) tem sido amplamente utilizado como um protocolo de comunicação em nível de transporte na Internet [Postel 1981]. GridFTP foi concebido para utilizar TCP como seu protocolo de comunicação a nível de transporte [Allcock 2003]. No entanto, o TCP é um protocolo de comunicação muito antigo que foi projetado na década de 1970. Vários problemas foram relatados sobre TCP, tais como a sua incapacidade para suportar a velocidade rapidamente crescente de redes recentes. Como exemplo, o atual TCP Reno (TCP versão Reno) não consegue detectar o congestionamento em uma rede até que ocorra perda de pacotes, portanto, um grande número de pacotes é perdido. Com as velocidades mais rápidas de redes e tamanhos maiores de buffer de roteadores em uma rede, a quantidade de pacotes perdidos e o throughput do TCP deterioram significativamente. Para resolver os problemas existentes no TCP, GridFTP tem características como o estabelecimento de múltiplas conexões TCP em paralelo para acelerar o início lento na fase inicial do TCP e negociando o tamanho do TCP socket buffer entre o servidor e o cliente GridFTP de acordo com o atraso da largura de banda de um rede [Allcock 2003].

No entanto, a eficácia desses recursos não foi totalmente investigada. Em outras palavras, as configurações ideais para o número de conexões TCP paralelas e tamanho do TCP socket buffer não foram investigados. Houve vários trabalhos relacionados em tamanho de TCP socket buffer e conexões TCP paralelas [Semke 1998, Lu 2005].

Em Semke (1998) e Dunigan (2002), os mecanismos de ajuste automático de TCP socket buffer foram propostos. No entanto, ambas as abordagens requerem algumas modificações para um socket API e/ou uma pilha de protocolo TCP. Em Grades computacionais, recursos de computação heterogêneos são integrados. Por isso, tais modificações de sistemas operacionais são irrealistas, ou seja, a otimização de tamanho do TCP socket buffer não deve depender de qualquer informação que não obtida pelo middleware da grade. Em Thulasidasan (2003), uma extensão do protocolo GridFTP para negociação automática do tamanho do TCP socket buffer foi proposta. No entanto, o mecanismo proposto é simples e não ideal, ou seja, ele simplesmente atribui o dobro do BDP para cada conexão TCP. Para alocação de memória eficiente para TCP socket buffer, mecanismos mais inteligentes devem ser incorporados.

Pelo contrário, a eficácia de conexões TCP paralelas tem sido estudada por muitos pesquisadores [Sivakumar 2000, Lu 2005]. Em Sivakumar (2000) e Lu (2005), o desempenho de conexões TCP paralelas é investigado utilizando experimentos de simulação. No entanto, para aperfeiçoar o número de conexões TCP paralelas, simulações baseadas em abordagens são inadequadas, ou seja, é muito difícil ou, na maioria dos casos, impossível de aplicar os resultados da simulação para uma otimização de parâmetro. Para otimizar o número de conexões TCP paralelas, algumas dicas no efeito de conexões TCP paralelas sobre o seu desempenho é necessário. Em Hacker (2001) e Lu (2005), modelos analíticos simples de conexões TCP paralelas foram apresentadas. No entanto, os modelos analíticos não são aplicáveis para otimizar o número de conexões TCP paralelas, uma vez que não captam os trade-offs na transferência de dados paralelos, como explicado por Ito (2006).

Computação Virtual

Os sistemas de transferência de arquivos mais utilizados atualmente são aplicações que se apoiam em protocolos de nível de transporte, ou seja, TCP e UDP, para transferir dados de sua origem para o destino. Embora estes sistemas explorem técnicas altamente sofisticadas para aumentar sua taxa de transferência, eles sofrem de uma desvantagem comum, nomeadamente a sua dependência sobre os protocolos de roteamento IP, que tem como consequência a fato de que o throughput que eles conseguem é limitado pela largura de banda disponível no caminho de rede escolhido pela camada de roteamento IP.

Infelizmente, os protocolos de roteamento IP notoriamente produzem rotas sub-ótimas [Savage 1999, Savage 1999b, Tangmunarunkit 2001], uma vez que sua escolha de caminhos da rede não é guiada por considerações de desempenho como estão principalmente preocupados com a troca de conectividade da informação. Como consequência, não é raro o caso onde os tempos mais curtos de transferência podem ser obtidos escolhendo um caminho de rede diferente do escolhido pelos algoritmos de roteamento IP. Por exemplo, Savage (1999b) observa que para 30 a 80 por cento dos caminhos escolhidos pelos algoritmos de roteamento IP entre pares de hosts da Internet, tomadas a partir de um conjunto relativamente grande de máquinas, foi possível encontrar caminhos alternativos com melhores características de desempenho.

Outra consequência da dependência em algoritmos de roteamento IP é a quantidade, possivelmente, alta de tempo necessário para recuperação de falhas de links. Por uma questão de fato, mecanismos de recuperação de falhas utilizados pelos protocolos de roteamento típicos da Internet por vezes, levam muitos minutos a convergir a uma forma consistente

[Labovilz 2000], e há momentos em que o caminho pode levar a interrupções significativas na comunicação com duração de até dezenas de minutos ou mais [Chandra 2001, Paxson 1996, Paxson 1997]. Como consequência, as transferências ao longo de caminhos com defeito pode ser atrasada por um tempo muito longo, assim a largura de banda efetiva obtida nestas situações é reduzida abaixo de qualquer valor razoável.

Anglano (2004) descreve o File Mover, um software que atende os problemas acima através da exploração de uma arquitetura overlay network (Rede de sobreposição). Uma rede de sobreposição é uma rede virtual, em camadas em cima da Internet existente, cujos membros nós são colocados nas bordas da rede física subjacente e se comunica por meio de um protocolo de nível de transporte (por exemplo, TCP ou UDP). Cada par de nós membros de uma rede sobreposta se comunica por meio de um link virtual, que corresponde ao caminho de rede escolhido pela camada IP para transferência de dados de um membro para o outro. Os Relays de uma rede de sobreposição concorda em frente um do outro tráfego ao longo de um ou mais links virtuais, até que o host de destino é atingido.

Protocolo GridFTP

GridFTP é um protocolo de transferência de dados, que é projetado para transferir efetivamente grande volume de dados em Grid Computing [Globus 2003, Allcock 2003, Globus 2002]. GridFTP é uma extensão de FTP (File Transfer Protocol) [Postel 1985, Elz 1997, Hethmon 1998] que tem sido amplamente utilizado e esta atualmente sob a normalização do GGF (Global Grid Forum) (GGF, 2011). GridFTP, que usa TCP como seu protocolo de comunicação na camada de transporte, foi concebido para resolver vários problemas do TCP. Por exemplo, além das características dos atuais FTP, ele tem funcionalidades adicionais, tais como negociação automática do tamanho do TCP socket buffer, transferência de dados paralela, controle de transferência de arquivos de terceiros, transferência de arquivos parciais, segurança, e transferência de dados confiável [Allcock 2003]. A maioria destas características específicas de GridFTP são realizadas por um novo modo de transferência chamado Modo de Bloqueio Estendida [Allcock 2003].

Atualmente, o servidor GridFTP e software cliente em conformidade com GridFTP versão 1 (GridFTPv1) é implementado no Globus Toolkit [Globus 2011], que é o middleware padrão de fato para a computação em Grid. No entanto, nesta implementação GridFTP específica, o recurso da negociação automática do tamanho do TCP socket buffer não é implementado e, portanto, um usuário deve especificar manualmente o número de conexões TCP paralelas para transferência de dados paralelo. Além disso, GGF tem discutido os problemas com GridFTP v1 e também realizado um estudo sobre GridFTP v2 (versão 2) como uma solução para esses problemas [Mandrichenko 2005]. Recursos adicionais foram incorporados ao GridFTP v2. Estas características atenuam várias limitações do modo de bloqueio estendido do GridFTP v1, tanto que a transferência de dados é restrito a uma única direção e incapaz de abrir / fechar canais de dados no meio da transferência de dados. No entanto, como configurar o Parâmetros de controle do GridFTP sobre conexões TCP paralelas não foi abordado ainda na discussão sobre GridFTP v2. (ITO, 2006) apresenta um resumo de duas principais características do GridFTP, auto-negociação do tamanho do tamanho do TCP socket buffer e transferência paralela de dados e problemas não resolvidos da GridFTP é apresentado.

3. Abordagem Desenvolvida

Conforme o aumento de dados presentes em grades computacionais, têm se tornado bastante comum a prática de transferência de dados entre fontes de dados heterogêneas, especialmente

quando novos protocolos são desenvolvidos. Muito trabalho vem sido aplicado para integrar tais fontes de dados. Tipicamente, usuários precisam utilizar um cliente para copiar dados de uma fonte para uma fonte intermediária, como, por exemplo, uma fonte de dados local, e em seguida utilizar outro cliente para mover os dados a partir da fonte intermediária para a fonte destino. Isso significa um passo extra na transferência de dados, o que é errado e ineficiente, considerando que o usuário necessita permanecer na frente do computador para efetuar a transferência.

A proposta de Zhang (2010) descreve uma abordagem diferente, que é capaz de converter qualquer interface de fonte de dados para a interface do GridFTP, que por sua vez é compatível com sistemas grade, e é capaz de potencializar as vantagens do protocolo com relação a transferência de dados. Ele resolve dois problemas com uma única solução. Primeiro, ele possibilita sistemas grade acessarem dados de qualquer fonte de dados. Segundo, transferir arquivos entre duas fontes de dados com protocolos distintos se torna possível sem a utilização de um ponto intermediário. A Figura 1 representa o funcionamento da mesma.

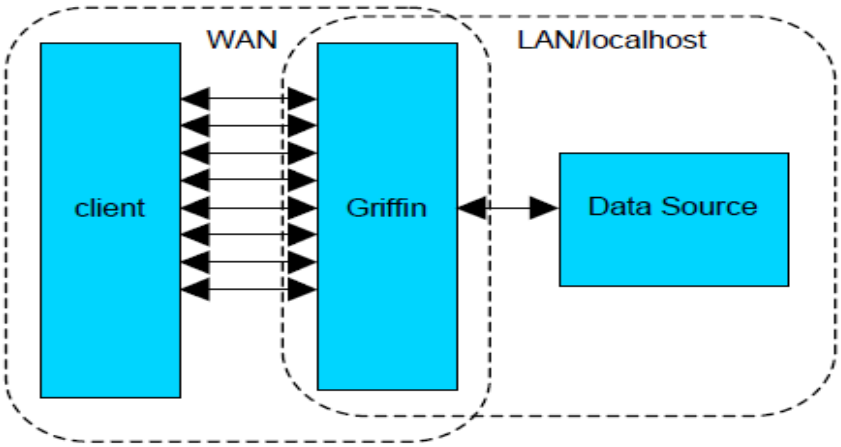

Figura 1. Modelo lógico do Griffin

A proposta deste trabalho consiste na modificação do código fonte do Griffin [Zhang 2010] de forma que a adição de adaptadores seja o mais simples possível. Conforme pode ser observado na Figura 2, a proposta modifica a arquitetura do Griffin adicionando uma nova camada de abstração ao seu código denominada Service Adaptor, cuja função é tornar o Griffin o mais flexível possível, ainda podemos destacar que tal camada implementa a interface Generic file system framework (Framework de sistemas de arquivos genérico) e os adaptadores de fontes de dados.

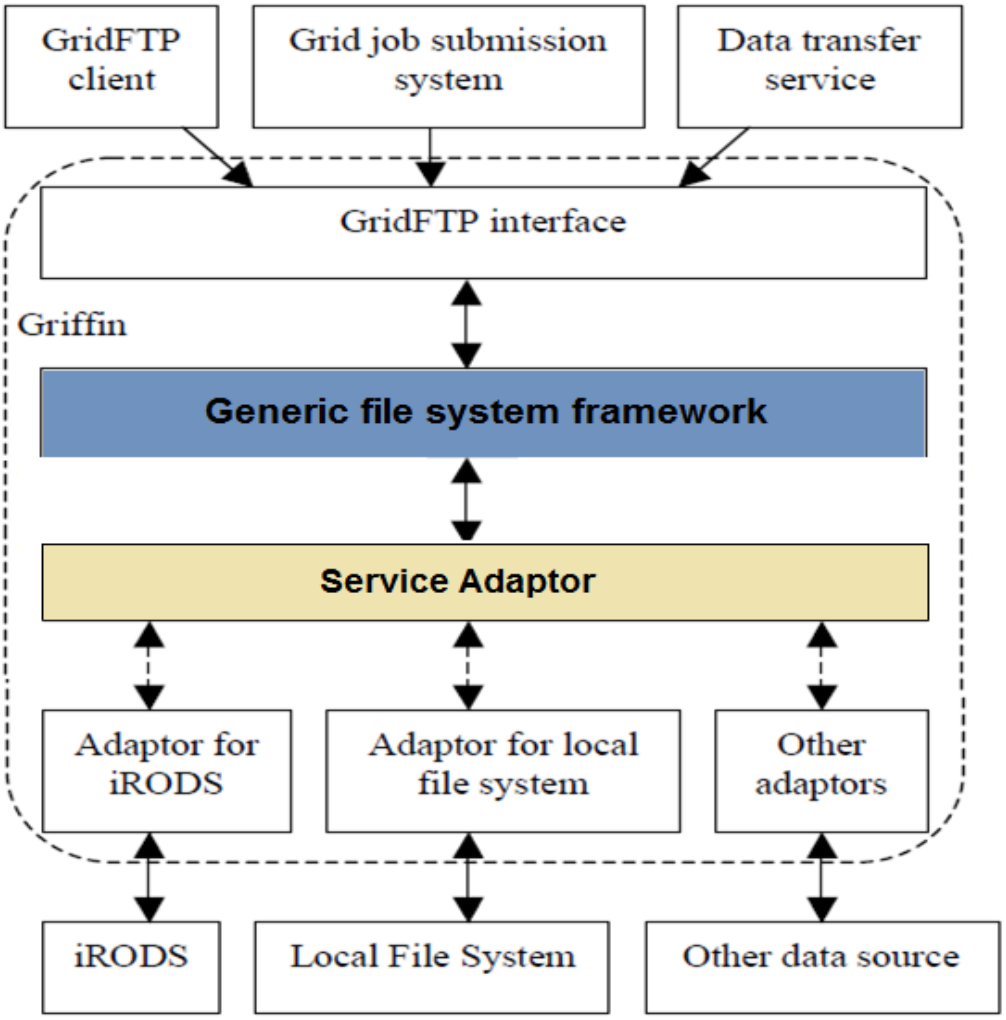

Figura 2. Arquitetura do Griffin

A implementação, ilustrada na Figura 2, foi desenvolvida em Java e baseado no framework Spring [Arthur 2005], que oferece uma arquitetura para desenvolvedores facilmente desenvolver aplicações modulares. A interface do GridFTP é a base da aplicação. Ele gerencia todas as conexões do cliente com um parser para analisar comandos GridFTP e invocá-los com um classe de comando relevantes. A implementação da autenticação GSI é baseada em jGlobus [Laszewski 2001]. Adaptadores para fontes de dados dependem da biblioteca de cliente relevante. Por exemplo, o adaptador para iRODS depende da biblioteca Java do iRODS, a Jargon. Tendo o Framework genérico torna-se possível desenvolver adaptadores para bibliotecas de outros sistemas de arquivos, como commons-vfs e JSAGA, de modo que o Griffin pode ser usado para acessar todas as fontes de dados que são suportados por essas bibliotecas. O binário de Griffin é leve, stand-alone e autônomo, com todas as bibliotecas java dependentes. Por isso, não depende de quaisquer componentes Globus, o que torna fácil de instalar e manter. Além disso, sendo uma aplicação Java, é possível ser executado na maioria dos sistemas operacionais sem recompilação. Por exemplo, se um usuário quiser enviar um job de grade para analisar os dados que estão localizados em uma área de trabalho em um computador desktop, uma instância de Griffin poderia ser executada neste ambiente de trabalho da máquina, não importa se é Windows ou Linux. Os Jobs submetidos pelo Globus podem então estagiar dados entrantes do desktop e estagiar dados de saída para o desktop, sem a necessidade de copiar dados de e para algum lugar que é acessível pelo nó de submissão de Jobs do Globus, como um servidor dedicado GridFTP, ou um espaço que é montado para os nós trabalhadores.

Com o design modular e com a ajuda do framework de injeção de dependências Spring [Fowler 2004], mudando o sistema de dados subjacente requer apenas uma mudança em um arquivo XML de configuração, sem a necessidade de recompilar todo o do sistema. A Figura 3 é uma amostra de uma parte do arquivo de configuração do Griffin com o adaptador iRODS. Para executar Griffin com um sistema de arquivos local, só é preciso substituir a parte <bean id = "filesystem"> parte com a Figura 4, e reiniciar o Griffin.

```
<bean id="server"
class="au.org.arcs.griffin.server.impl.GsiFtpServer"
singleton="true">
   <property name="name" value="GSI FTP Server" />
   <property name="options" ref="options" />
   <property name="resources" value="griffin-resources"/>
   <property name="fileSystem" ref="fileSystem" />
</bean>
<bean id="fileSystem"
class="au.org.arcs.griffin.filesystem.impl.jargon.JargonFileSystemI
mpl" singleton="true">
   <property name="serverName" value="localhost" />
   <property name="serverPort" value="1247" />
   <property name="serverType" value="irods" />
</bean>
```

Figura 3. Exemplo de configuração com adaptador para iRODS

```
<bean id="fileSystem"
class="au.org.arcs.griffin.filesystem.impl.localfs.LocalFileSystemI
mpl" singleton="true">
   <property name="rootPath" value="/data/data-fabric" />
   <property name="userManager" ref="userManager" />
</bean>
```

Figura 4. Exemplo de configuração com adaptador para sistema de arquivos local

Para acessar sistemas de arquivos arbitrários de uma maneira uniforme, foi criada a interface Generic file system framework e em seguida a camada que implementa seus métodos a Service Adaptor que funciona como back-end para a interface GridFTP. Por meio da camada Service Adaptor, qualquer pedido proveniente da interface GridFTP será traduzido para uma operação padrão do sistema de arquivos. Para torná-lo genérico, a camada é leve e simples, suportando apenas operações básicas e comumente implementadas pela interface GridFTP. A concepção desta camada é específica para as necessidades do protocolo GridFTP, especialmente recursos avançados para transferência de dados, a fim de minimizar a sobrecarga entre a interface front-end do GridFTP e o sistema de dados no back-end. Como o protocolo GridFTP é usado principalmente para transferência de dados, esta camada não requer outras funções, tais como manipulação de meta-dados.

A estrutura top-down simples é ilustrada na Figura 5, com quatro grandes objetos. O superior é o único caso no sistema que correspondente à fonte de dados, com todas as configurações e controles em nível de fonte de dados. Cada usuário, uma vez autenticado, será associado a um objeto de conexão, que é o segundo objeto a partir do topo. O objeto conexão mantém todas as informações do usuário, pois diferentes usuários possuem uma visão diferente da fonte de dados, com diferentes permissões. Assim, o objeto de conexão pode determinar se um objeto de arquivo, o terceiro de cima para baixo, pode ser criado para um determinado caminho, e que as funções podem ser chamadas, com base nas permissões do usuário. Uma vez que um objeto de arquivo é criado, é usado para recuperar informações do

arquivo de destino, ou fazer algumas operações sobre ele. O último objeto é usado principalmente para acessar o conteúdo de um objeto de arquivo, desde que o objeto de arquivo tenha sido instanciado com permissões adequadas. Um adaptador para um sistema de grade de dados deve implementar todas as quatro interfaces.

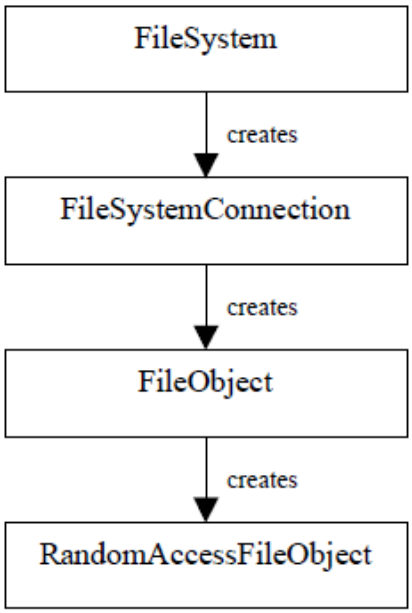

Figura 5. A estrutura da camada Service Adaptor

O objeto FileSystem é a raiz da hierarquia. Este objeto apresenta o sistema subjacente de dados na interface GridFTP e é chamado na inicialização e desligamento. Em particular, o método init () é chamado para examinar as configurações e verificar a conectividade com a fonte de dados durante a inicialização. Se houver um erro, por exemplo, a configuração não está correta ou o sistema subjacente de dados não está acessível, uma exceção será lançada, assim o servidor GridFTP pode decidir se ele pode continuar a ser executado. Da mesma forma, o método exit() é chamado durante o desligamento para liberar recursos e encerrar com o sistema de dados se houver. O método GetSeparator() retorna o separador usado em caminhos já que ele pode ser diferente em alguns sistemas, como a barra invertida no Windows e barra em Linux. O Método createFileSystemConnection() será executado quando um usuário se conectar à interface GridFTP e uma sessão de GridFTP criada para esse usuário. Correspondentemente, uma conexão com o sistema de dados subjacente será criada, de acordo com a credencial GSI. Este método retorna um objeto FileSystemConnection, que representa uma sessão para o sistema de dados. Se o sistema de dados subjacentes suporta autenticação GSI, a mesma credencial GSI utilizada para autenticar com a interface GridFTP será usada para autenticar com o sistema de dados subjacente. No entanto, se o sistema de dados subjacente não suportar à autenticação GSI, algum tipo de mecanismo de mapeamento será empregado.

O objeto FileSystemConnection mantém uma conexão com o sistema de dados subjacente para o usuário. Ela consiste de alguns métodos para todo o sistema e relacionadas ao usuário. getHomeDir() e getFreeSpace() são auto-descritivos. getUser() retorna o nome de usuário do sistema subjacente associado a este sessão. isConnected() indica se a conexão está aberta, enquanto close() irá encerrar a conexão atual. O Método getFileObject() gera um FileObject para um caminho particular no contexto do sistema GridFTP.

O objeto FileObject representa um objeto de dados na fonte de dados. Ele é projetado para refletir a classe java.io.File, fornecendo métodos para obter o nome do arquivo, caminho, caminho canônico, comprimento, hora da última modificação, o seu pai ou seus filhos. Além

disso, ele permite aos usuários verificar se esse arquivo existe, ou se este objeto é um arquivo ou um diretório. Os usuários também podem usar esse objeto para fazer um novo diretório, apagar-se, mudar o nome em si, ou definir a hora da última modificação. O método GetPermission() retorna um número abstrato de permissão, que só pode ser sem acesso (0), o privilégio ler (1), o privilégio escrever (2) ou de leitura/escrita (3). Como o protocolo GridFTP não requer a manipulação de permissões, todos os tipos de permissões são selecionadas para a permissão do usuário atual. O que é exigido no Service Adaptor é descobrir as permissões para o usuário associado à conexão atual. Assim, retornando uma permissão simples numerada é a maneira mais simples e genérica para abordar esta questão, e esconder as complexidades de diferentes sistemas dados de grade. O Método getRandomAccessFileObject() retorna um RandomAccessFileObject para acessar o conteúdo deste objeto arquivo.

O objeto RandomAccessFileObject fornece métodos para ler e escrever conteúdo no FileObject correspondente. O design é semelhante ao java.IO.RandomAccessFile, para os desenvolvedores. Vários métodos read() e write() constituem a maior parte deste objeto para ler o conteúdo do arquivo para um array de bytes ou escrever o conteúdo de um array de bytes para o arquivo. Outros métodos incluem seek() para saltar para um determinado ponto no objeto de dados e length() para retornar o comprimento do objeto de dados. Ao contrário dos objetos streaming, que podem apenas ler ou escrever em seqüência, o RandomAccessFileObject oferece um caminho para o front-end da interface GridFTP que permite facilmente ler ou gravar dados de qualquer posição no arquivo, que é útil para o modo de bloqueio estendido. Consequentemente, a fonte de dados subjacente deve suportar acesso aleatório para que ele possa ser conectado a este framework.

4. Ambiente e Resultados Experimentais

O ambiente de testes montado consiste em máquina desktop (PC) e um netbook; para fins do experimento no desktop foi instalado o ubuntu server versão 9.04 (Processador: Core intel i5, 2,90 GHz, Disco Rígido: 500 GB, Memória RAM: 4 GB), um simulador de ambiente grade, o GridSim (2010), e rodando sobre ele o Globus Toolkit. Enquanto que no netebook foi instalada a versão mais atual do ubuntu a 11.10 (Processador: Atom 1,33GHz de CPU, Disco Rígido: 320 GB, Memória RAM: 2 GB) Os testes foram conduzidos em um ambiente controlado, e as transferências realizadas em uma rede residencial via intranet, com link de velocidade de 100,00 MBytes.

Estes experimentos têm por objetivo demonstrar que o desempenho das transferências não é afetado pela utilização do Griffin. No primeiro teste, foi comparado a taxa de transferência do protocolo original com GridFTP via Griffin.

IRODS 2.1 foi instalado como fonte de dados. Griffin foi configurado no servidor com 786MB de tamanho de pilha como a interface GridFTP para este iRODS. Com esta configuração, os dados em iRODS podem ser acessado via interface nativa iRODS ou interface GridFTP. Foram comparamos a taxa de upload e taxa de download entre Servidor e cliente, com tamanhos de arquivo de 256MB, 512MB, 1GB, 2GB, 4GB, 8GB a 16GB, e oito threads são usadas tanto para transferências de iCommands quanto Griffin.

Tendo cada teste sido executado 5 vezes, o resultado na Figura 6 mostra a taxa de transferência média em Kbytes por segundo. A partir da figura, ao fazer upload de um arquivo, iCommands e Griffin executam a uma taxa semelhante, para arquivos menores que 8 GB, no entanto, se o arquivo é de 8GB ou mais, Griffin é mais rápido que iCommands.

Ao fazer o download, Griffin funciona de forma constante para os diversos tamanhos no teste, enquanto iCommands executa mais rápido para arquivos pequenos, mas fica mais lento conforme o tamanho do arquivo aumenta.

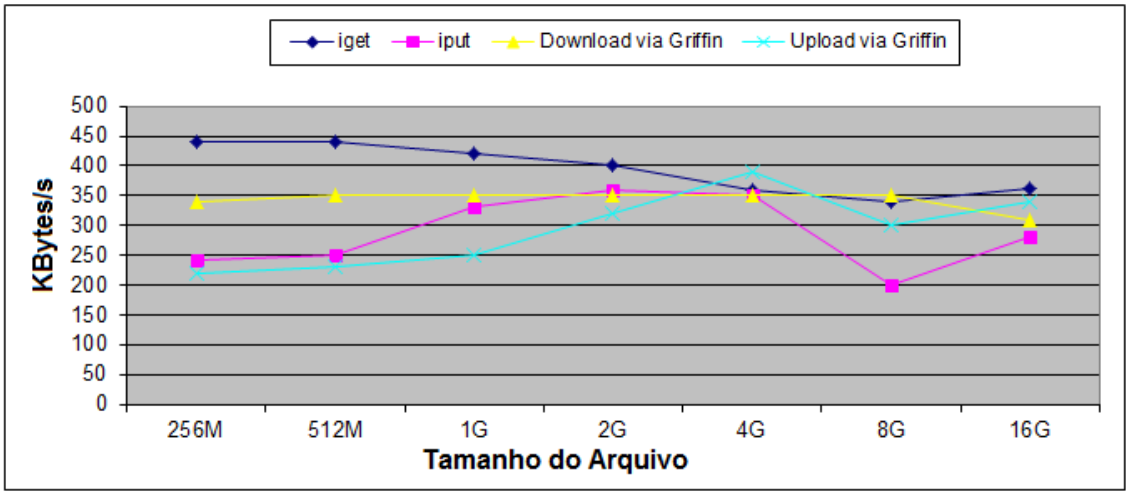

Figura 6. Comparação entre iCommands e Griffin

No segundo teste, é comparado o Griffin com o servidor Globus GridFTP, no mesmo teste ambiente, como os testes acima. O servidor Globus GridFTP foi instalado no servidor a partir VDT 1.10.1. Griffin foi reconfigurado com um adaptador de sistema de arquivos local, de modo que lê e grava dados no sistema de arquivos local.

Neste teste, os dados de testes são armazenados na mesma partição que o recurso iRODS. Semelhante ao primeiro teste, foram transferidos arquivos com tamanhos de 256M, 512M, 1G, 2G, 4G, 8G e 16G, via Griffin e Globus GridFTP, com 16 threads. Cada teste de transferência foi feito 5 vezes e os valores médios são mostrados na Figura 7. O resultado mostra que Griffin executa muito próximo do servidor Globus GridFTP em upload e download de dados.

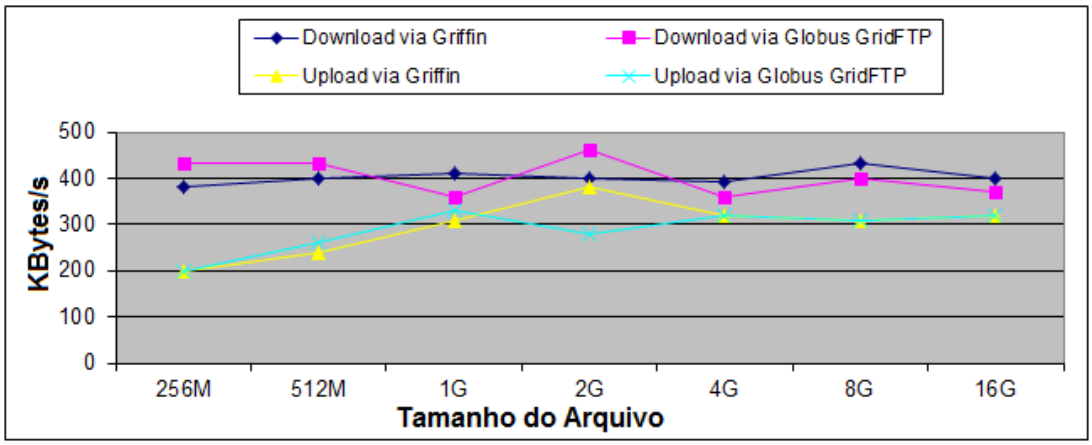

Figura 7. Comparação entre Servidor Globus GridFTP e Griffin

5. Conclusão e Trabalhos Futuros

A abordagem utilizada neste trabalho é relativamente nova, sendo que a mesma foi apresentada em outubro de 2010 para a comunidade científica. Sendo uma abordagem recente ainda há muito em que se trabalhar, este trabalho, por exemplo, modificou o protótipo inicial para facilitar a criação e novos adaptadores. Neste trabalho foi identificado um impasse na

forma como dados são transferidos entre ambientes heterogêneos de alto desempenho, e durante o levantamento do estado da arte foi-se identificada a proposta do Griffin.

O Griffin representa uma abordagem genérica, leve e fácil de usar para se conectar a uma fonte de dados arbitrários para a grade, usando um quadro genérico de sistema de arquivo com uma interface GridFTP. Ele permite que qualquer fonte de dados possa ser acessada pela grade, permite ainda a transferência de grandes conjuntos de dados, tem muito pouco impacto sobre o desempenho para o transporte de dados, conforme observado nos experimentos, e torna possível mover dados entre fontes de dados com diferentes protocolos facilmente.

A implementação é baseada em Java sendo uma alternativa para o servidor Globus GridFTP; ela é autônoma, não depende da pilha de software Globus, e pode ser executado na maioria dos sistemas operacionais. É adequado para fontes de dados que ofereçam uma API Java.

No futuro, suporte UDT será adicionado ao Griffin, e mecanismos de verificação serão implementados para verificar a integridade de arquivos copiados. Além disso, múltiplos fluxos do Griffin para o fonte de dados será investigada em casos onde a fonte de dados suporte múltiplos fluxos. Striping (segmentação) de transferências é outro recurso útil a se ter no futuro. Ele permite o envio de um enorme arquivo a partir de vários servidores GridFTP com cada um enviando uma parte do arquivo para obter melhor desempenho do que o uso de apenas um servidor GridFTP.

Referências

Allcock, W. (2003) “GridFTP: Protocol extensions to FTP for the Grid,” GGF Document Series GFD.20, Apr. 2003. Disponível em http://www.gridforum.org/GFD.20.pdf.

Anglano, C., Canonico, M. (2004). The File Mover: an efficient data transfer system for Grid applications, 27 set. 2004.

Arthur, J., Azadegan, S. (2005). "Spring Framework for rapid open source J2EE Web Application Development: A case study," in Proc of the Sixth Int Conf on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing(SNPD’05), 2005.

Baru, C., Moore, R., Rajasekar, A. (1998) "The SDSC Storage Resource Broker," in Proc of the Centre for Advanced Studies on Collaborative Research (CASCON), 1998.

Cern. (2011) glite. Disponível em http://glite.web.cern.ch/glite/. Acessado em Junho de 2011.

Chandra, B. (2001). End-to-End WAN Service Availability. In Proc. of 3rd Usenix Symp. on Inremet Technologies and Systems (USITS), pages 97-108, San Francisco, CA, Februaly 2001.

Condor.(2011). Disponível em http://www.cs.wisc.edu/condor/. Acessado em Maio de 2011.

Dunigan, T., Mathis, M., Tierney, B. (2002) “A TCP tuning daemon,” in Proceedings of SuperComputing: High-Performance Networking and Computing, Nov. 2002.

Elz, R., Hethmon, P. (1997). FTP security extensions. Request for Comments (RFC) 2228, Oct. 1997.

Ernst, M., Fuhrmann, P., Gasthuber, M. (2001) "dCache, a distributed data storage caching system," in Computing in High Energy and nuclear Physics (CHEP2001), Beijing, 2001.

Foster, I., Kesselman, C. (2003). Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Aplications 11, 2, 115-128. Disponível em: ftp://ftp.globus.org/pub/globus/papers/globus.pdf. Acesso em: Novembro de 2003.

Fowler, M. (2004). Inversion of Control Containers and the Dependency Injection pattern.(29 April 2010). http://www.martinfowler.com/articles/injection.html.

Globus, The Project. (2003). GridFTP: universal data transfer for the Grid. White Paper, Sept. 2003. Diisponível em: http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf.

Globus, The Project. (2002). GridFTP update January 2002, 2002. Disponível em: http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf.

Globus Toolkit. (2011). Disponível em http://www.globus.org/. Acessado em Junho de 2011.

GridSim. (2011). Disponível em http://www.buyya.com/papers/gridsim.pdf. Acessado em Outubro de 2011.

Hacker, T. J., Athey, B. D. (2001). "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS), Aug. 2001.

Hethmon, P., Elz, R. (1998). Feature negotiation mechanism for the file transfer protocol. Request for Comments (RFC) 2389, Aug. 1998.

Ito, T. Oshsaki, H. Imase, M. (2006). On parameter tuning of data transfer protocol GridFTP for wide-area grid computing, 13 fev. 2006.

Labovilz, C., Ahuja, A., Bose, A., Jahanian, F. (2000). Delayed Internet Routing Convergence. In Pmc. of ACM SIGCOMM. pages 175-187, Stockolm. Sweden, September 2000.

Laszewski, G. V., Foster, I., Gawor, J. (2001). "A Java Commodity Grid Toolkit," Concurrency: Practice and Experience, vol. 13, 2001.

Lu, D.; Quao, Y., Dinda, P., Bustamente, F. (2005) "Modeling and taming parallel TCP on the wide area network," in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Apr. 2005.

Mandrichenko, I., Allcock, W., Perelmutov, T. (2005). GridFTP v2 protocol description. GGF Document Series GFD.47, May 2005. Disponível em http://www.ggf.org/documents/GFD.47.pdf.

Paxson, V. (1996). End-to-End Routing Behavior in the Internet. In Pmc. of ACM SIGCOMM, pages 2s-38, Stanford, CA, Ausus1 1996.

Paxson, V. (1997). End-to-End Internet Packet Dynamics. In Proc. ACM SIGCOMM, pages 139-152, Canncs. France, September 1997.

Postel, J. (1981) "Transmission control protocol," Request for Comments (RFC) 793, Sept. 1981.

Postel, J., Reynolds, J. (1985). File transfer protocol (FTP). Request for Comments (RFC) 959, Oct. 1985.

Rajasekar, A., Wan, M., Moore, R. (2006) "A Prototype Rule-based Distributed Data Management System," in High Performance Parallel and Distributed Computing (HPDC), Paris, França, 2006.

Savage, S. (1999). Detour: A Case for Informed Internet Routing and Transpon. IEEE Micro, 19(1):50-59, Jan. 1999.

Savage, S. (1999b). The End-to-End Effects of lntemet Path Selection. In Proc. of ACM SIGCOMM, pages 289-299, Boston, MA, 1999b.

Semke, J., Mahdavi, J., Mathis, M. (1998) “Automatic TCP buffer tuning,” in Proceedings of ACM SIGCOMM ’98, vol. 28, Oct. 1998.

Sivakumar, H., Bailey, S., Grossman, R. L. (2000). “PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks,” in Proceedings of the 2000 ACM/IEEE Conference on Supercomputing, Nov. 2000.

Tangmunarunkit, H., Govindan, R., Sherker, S., Eslrin, D. (2001) The Impact of Routing Policy on Internet Paths. In Proc. IEEE Infocom '01, Anchorage, Alaska, April 2001.

Thulasidasan, S., Feng, W., Gardner, M. K. (2003) “Optimizing GridFTP through dynamic right-sizing,” in Proceedings of IEEE International Symposium on High Performance Distributed Computing, June 2003.

Zhang, S., Coddington, P., Wendelborn, A. (2010). Connecting arbitrary data resources to the grid. In Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on Oct. 2010.

APÊNDICE B – Código Fonte do Griffin

Griffin.au.org.arcs.griffin.FtpServerStarter.java

```
/*
2         * ----------------------------------------------------------------------------
3         * Hermes FTP Server
4         * Copyright (c) 2005-2007 Lars Behnke
5         * ----------------------------------------------------------------------------
6         *
7         * This file is part of Hermes FTP Server.
8         *
9         * Hermes FTP Server is free software; you can redistribute it and/or modify
10        * it under the terms of the GNU General Public License as published by
11        * the Free Software Foundation; either version 2 of the License, or
12        * (at your option) any later version.
13        *
14        * Hermes FTP Server is distributed in the hope that it will be useful,
15        * but WITHOUT ANY WARRANTY; without even the implied warranty of
16        * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17        * GNU General Public License for more details.
18        *
19        * You should have received a copy of the GNU General Public License
20        * along with Hermes FTP Server; if not, write to the Free Software
21        * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22        * ----------------------------------------------------------------------------
23        */
24
25       package au.org.arcs.griffin;
26
27       import java.io.File;
28       import java.io.PrintStream;
29       import java.net.InetAddress;
```

```
30  import java.security.NoSuchAlgorithmException;
31  import java.util.ArrayList;
32  import java.util.List;
33  import java.util.Set;
34
35  import org.apache.commons.logging.Log;
36  import org.apache.commons.logging.LogFactory;
37  import org.springframework.context.ApplicationContext;
38  import org.springframework.context.support.ClassPathXmlApplicationContext;
39  import org.springframework.context.support.FileSystemXmlApplicationContext;
40
41  import au.org.arcs.griffin.common.BeanConstants;
42  import au.org.arcs.griffin.common.FtpConstants;
43  import au.org.arcs.griffin.common.FtpServerOptions;
44  import au.org.arcs.griffin.server.FtpServer;
45  import au.org.arcs.griffin.utils.IOUtils;
46  import au.org.arcs.griffin.utils.LoggingOutputStream;
47  import au.org.arcs.griffin.utils.NetUtils;
48  import au.org.arcs.griffin.utils.SecurityUtil;
49
50  /**
51   * Griffin FTP application.
52   *
53   * @author Lars Behnke
54   * @author Shunde Zhang
55   */
56  public final class FtpServerStarter {
57
58      //private static final int THREAD_ALIVE_CHECK_INTERVAL = 1000;
59
60      private static final int PASSWORD_ARG_COUNT         = 3;
61
62      private static Log       log                        = LogFactory.getLog(FtpServerStarter.class);
63
64      /**
65       * Constructor.
66       */
67      public FtpServerStarter() {
68          super();
69      }
70
71      /**
72       * Entry point of the application.
73       *
74       * @param args Optionally the bean resource file can be passed.
75       */
76      public static void main(String[] args) {
77          // TODO Use commons-cli
78          if (args.length > 0 && args[0].trim().equalsIgnoreCase("-password")) {
79              generatePassword(args);
80          } else {
81              log.info("Starting Griffin FTP Server...");
82              PluginManager.startApplication(FtpServerStarter.class.getName(), "startServer", args);
83              log.info("Griffin FTP Server ready.");
84          }
85      }
86
87      private void redirectOutErr(){
88          // now rebind stdout/stderr to logger
89          Log log;
```

```
90              LoggingOutputStream los;
91
92              log = LogFactory.getLog("stdout");
93              los = new LoggingOutputStream(log, "stdout");
94              System.setOut(new PrintStream(los, true));
95
96              log = LogFactory.getLog("stderr");
97              los= new LoggingOutputStream(log, "stderr");
98              System.setErr(new PrintStream(los, true));
99
100         }
101
102         private static void generatePassword(String[] args) {
103             if (args.length != PASSWORD_ARG_COUNT) {
104                 System.err
105                   .println("Please adhere to the following synthax: FtpServerApp password <password>
<algorithm>");
106                 return;
107             }
108             String password = args[1];
109             String algorithm = args[2];
110             try {
111                 String hash = SecurityUtil.encodePassword(password, algorithm);
112                 System.out.print("Hash: " + hash + "\n");
113             } catch (NoSuchAlgorithmException e) {
114                 System.err.println("ERROR: " + e);
115             }
116         }
117
118         /**
119          * Starts the FTP servers(s).
120          *
121          * @param args The arguments passed with main method.
122          */
123         public void startServer(String[] args) {
124     //        if (!NetUtils.isSSLAvailable()) {
125     //            System.exit(1);
126     //        }
127
128             redirectOutErr();
129     //      System.out.println("test redirection - out");
130     //      System.err.println("test redirection - err");
131
132             String beanRes = args.length > 0 ? args[0] : FtpConstants.DEFAULT_BEAN_RES;
133             File file = new File(beanRes);
134
135             logPaths(file);
136             if (System.getProperty(FtpConstants.GRIFFIN_HOME)==null)
137                   System.setProperty(FtpConstants.GRIFFIN_HOME,file.getParent());
138
139             /* Prepare three main threads */
140             ApplicationContext appContext = getApplicationContext(beanRes, file);
141             FtpServer svr = (FtpServer) appContext.getBean(BeanConstants.BEAN_SERVER);
142     //        FtpServer sslsvr = (FtpServer)
appContext.getBean(BeanConstants.BEAN_SSL_SERVER);
143     //        ConsoleServer console = (ConsoleServer)
appContext.getBean(BeanConstants.BEAN_CONSOLE);
144
145             /* Log settings */
146
```

```
147             logOptions(svr.getOptions());
148
149             /* Check local ip addresses */
150             InetAddress addr = NetUtils.getMachineAddress(true);
151             if (addr == null) {
152                 log.error("No local network ip address available.");
153                 System.exit(1);
154             }
155             log.info("Local ip address: " + addr);
156
157             /* Start servers */
158             Thread svrThread;
159     //        Thread sslSvrThread;
160             try {
161                 svrThread = new Thread(svr);
162                 svrThread.start();
163     //            sslSvrThread = new Thread(sslsvr);
164     //            sslSvrThread.start();
165
166                 /* Start web console */
167     //            if (svr.getOptions().getBoolean("console.enabled", false)) {
168     //                console.start();
169     //            }
170
171                 /* Register Shutdown Hook */
172                 List<FtpServer> serverList = new ArrayList<FtpServer>();
173                 serverList.add(svr);
174     //            serverList.add(sslsvr);
175                 addShutdownHook(serverList);
176
177             } catch (Exception e) {
178                 log.error("Unexpected error", e);
179             }
180         }
181
182         /**
183          * Add shutdown hook.
184          */
185         private static void addShutdownHook(final List<FtpServer> servers) {
186
187             Runnable shutdownHook = new Runnable() {
188                 public void run() {
189                     for (FtpServer ftpServer : servers) {
190                         log.info("Stopping server '" + ftpServer.getName() + "'.");
191                         ftpServer.abort();
192                     }
193                     log.info("All servers down.");
194                 }
195             };
196             Runtime runtime = Runtime.getRuntime();
197             runtime.addShutdownHook(new Thread(shutdownHook));
198         }
199
200         private void logPaths(File file) {
201     //        log.info("Hermes Home: " + IOUtils.getHomeDir());
202
203             log.info("Application context: " + file);
204             if (file != null && file.getParent() != null) {
205                 System.setProperty("griffin.ctx.dir", file.getParent());
206                 log.info("Application context path: " + file.getParent());
```

```
207             }
208         }
209
210         private static ApplicationContext getApplicationContext(String beanRes, File file) {
211             ApplicationContext appContext;
212             if (file.exists()) {
213                 appContext = new FileSystemXmlApplicationContext("file:"+beanRes);
214             } else {
215                 log.error("Griffin FTP application context not found: " + file
216                         + ". Trying to read context from classpath...");
217                 appContext = new ClassPathXmlApplicationContext(
218                     new String[] {"/" + FtpConstants.DEFAULT_BEAN_RES});
219             }
220             return appContext;
221         }
222
223         private static void logOptions(FtpServerOptions aOptions) {
224             log.info(aOptions.getAppTitle());
225             log.info("Version " + aOptions.getAppVersion());
226             log.info("Build info: " + aOptions.getAppBuildInfo());
227
228             log.info("Ftp server options:");
229             Set<Object> keyset = aOptions.getProperties().keySet();
230             for (Object key : keyset) {
231                 String value = aOptions.getProperty(key.toString());
232                 log.info("    " + key + ": " + value);
233             }
234         }
235     }
```

Griffin.au.org.arcs.griffin.PluginManager.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin;
26
27      import java.io.File;
28      import java.io.FilenameFilter;
```

```
29  import java.io.IOException;
30  import java.lang.reflect.Method;
31  import java.net.MalformedURLException;
32  import java.net.URL;
33  import java.net.URLClassLoader;
34  import java.util.ArrayList;
35  import java.util.HashSet;
36  import java.util.List;
37  import java.util.Set;
38  import java.util.jar.JarEntry;
39  import java.util.jar.JarInputStream;
40
41
42  import org.apache.commons.lang.StringUtils;
43  import org.apache.commons.logging.Log;
44  import org.apache.commons.logging.LogFactory;
45
46  import au.org.arcs.griffin.common.FtpConstants;
47
48  /**
49   * Utility class that manages the plug-in classpath and class loading.
50   *
51   * @author Lars Behnke.
52   * @author Shunde Zhang
53   */
54  public final class PluginManager {
55
56      private static final String   PATH_SEPARATOR  = "path.separator";
57
58      private static final String   JAVA_CLASS_PATH = "java.class.path";
59
60      private static final String   JAR_EXT         = ".jar";
61
62      private static Log            log             = LogFactory.getLog(PluginManager.class);
63
64      private static DynClassLoader classLoader;
65
66      static {
67          List<URL> jars = new ArrayList<URL>();
68
69          StringBuffer cpExtension = new StringBuffer();
70          File[] jarFiles = collectJars(new String[] {getPluginDir().getPath()});
71
72          for (int i = 0; i < jarFiles.length; i++) {
73              try {
74                  File file = jarFiles[i];
75                  jars.add(file.toURI().toURL());
76                  cpExtension.append(System.getProperty(PATH_SEPARATOR));
77                  cpExtension.append(file.toString());
78              } catch (MalformedURLException e) {
79                  log.error(e);
80              }
81          }
82          String cp = System.getProperty(JAVA_CLASS_PATH) + cpExtension.toString();
83          System.setProperty(JAVA_CLASS_PATH, cp);
84          if (log.isDebugEnabled()) {
85              log.debug("Classpath: " + cp);
86          }
87          classLoader = new DynClassLoader((URL[]) jars.toArray(new URL[jars.size()]));
88
```

```
89              /*
90               * Necessary for Spring to find plugin classes. Don't do this in servlet/ejb container!!
91               */
92              Thread.currentThread().setContextClassLoader(classLoader);
93          }
94
95          /**
96           * Hidden constructor.
97           */
98          private PluginManager() {
99          }
100
101         private static File[] collectJars(String[] paths) {
102             Set<File> jarList = new HashSet<File>();
103             for (int i = 0; i < paths.length; i++) {
104                 File dir = new File(paths[i]);
105                 if (log.isWarnEnabled() && !dir.exists()) {
106                     log.warn("JAR folder not found: " + dir);
107                 }
108                 if (dir.exists() && dir.isDirectory()) {
109                     File[] files = dir.listFiles(new JarFileFilter());
110                     for (int j = 0; j < files.length; j++) {
111                         jarList.add(files[j]);
112                     }
113                 }
114             }
115             return (File[]) jarList.toArray(new File[jarList.size()]);
116         }
117
118         /**
119          * Adds a resource to the classpath. Note that the resource is not available before the
update
120          * method of the classloader is called.
121          *
122          * @param jarOrPath The resource to add.
123          */
124         public static void addResource(File jarOrPath) {
125             try {
126                 classLoader.addURL(jarOrPath.toURI().toURL());
127             } catch (MalformedURLException e) {
128                 log.error(e);
129             }
130         }
131
132         /**
133          * Get the directory where plugins are installed.
134          *
135          * @return the directory where plugins are installed.
136          */
137         public static File getPluginDir() {
138             String dir = System.getProperty(FtpConstants.GRIFFIN_HOME);
139             if (StringUtils.isEmpty(dir)) {
140                 dir = System.getProperty("user.dir");
141             }
142             File file = new File(dir, "plugins");
143             try {
144                 file = file.getCanonicalFile();
145             } catch (IOException e) {
146                 log.debug("No canonical path: " + file);
147             }
```

```
148             return file;
149         }
150
151         /**
152         * Need to be invoked from the application's main in order to utilize the dynamic class loader.
153          *
154          * @param mainClassName Main class.
155          * @param startMethod Start method that accepts the main arguments.
156          * @param args Array of optional arguments
157          */
158         public static void startApplication(String mainClassName, String startMethod, String[] args) {
159             try {
160                 classLoader.update();
161                 Class<?> clazz = classLoader.loadClass(mainClassName);
162                 Object instance = clazz.newInstance();
163                 Method startup = clazz.getMethod(startMethod, new Class[] {(new
String[0]).getClass()});
164                 startup.invoke(instance, new Object[] {args});
165             } catch (Exception e) {
166                 log.error(e, e);
167             }
168         }
169
170         /**
171          * Filter for JAR files.
172          *
173          * @author developer
174          */
175         private static final class JarFileFilter implements FilenameFilter {
176             public boolean accept(File f, String name) {
177                 return name.endsWith(JAR_EXT);
178             }
179         }
180
181         /**
182          * Class loader that is capable of adding new resources on the fly. In contrast to the default
183          * class loader the addURL method is public. The approach was inspired by the Apache
JMeter
184          * project.
185          *
186          * @author developer
187          */
188         private static class DynClassLoader extends URLClassLoader {
189
190             private static final int EXTENSION_LENGTH = 6;
191
192             public DynClassLoader(URL[] urls) {
193                 super(urls);
194             }
195
196             /**
197              * @see java.net.URLClassLoader#addURL(java.net.URL)
198              */
199             public void addURL(URL url) {
200                 super.addURL(url);
201             }
202
203             /**
204              * Loads all class files known to the class loader.
205              */
```

```
206             public void update() {
207                 try {
208                     loadClasses(classLoader.getURLs());
209                 } catch (IOException e) {
210                     log.error("Loading classes failed: " + e);
211                 }
212             }
213
214             private void loadClasses(URL[] urls) throws IOException {
215                 for (int i = 0; i < urls.length; i++) {
216                     JarInputStream jis = new JarInputStream(urls[0].openStream());
217                     JarEntry entry = jis.getNextJarEntry();
218                     int loadedCount = 0;
219                     int totalCount = 0;
220
221                     while ((entry = jis.getNextJarEntry()) != null) {
222                         String name = entry.getName();
223                         if (name.endsWith(".class")) {
224                             totalCount++;
225                             name = name.substring(0, name.length() - EXTENSION_LENGTH);
226                             name = name.replace('/', '.');
227
228                             try {
229                                 classLoader.loadClass(name);
230                                 log.debug("Plugin class " + name + "\t- loaded");
231                                 loadedCount++;
232                             } catch (Throwable e) {
233                                 log.debug("Plugin class " + name + "\t- not loaded - " + e);
234                             }
235
236                         }
237
238                     }
239                     log.debug("Classes loaded: " + loadedCount);
240                 }
241             }
242         }
243
244     }
```

Griffin.au.org.arcs.griffin.utils. AbstractAppAwareBean.java

```
1       /*
2        * --------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * --------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
```

```
19        * You should have received a copy of the GNU General Public License
20        * along with Hermes FTP Server; if not, write to the Free Software
21        * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22        * ----------------------------------------------------------------------------
23        */
24
25       package au.org.arcs.griffin.utils;
26
27
28       import org.springframework.context.ApplicationContext;
29       import org.springframework.context.ApplicationContextAware;
30
31       import au.org.arcs.griffin.common.BeanConstants;
32
33       /**
34        * Provides a basic, application context aware class that can be subclassed.
35        *
36        * @author Lars Behnke
37        */
38       public abstract class AbstractAppAwareBean implements ApplicationContextAware,
BeanConstants {
39
40          private ApplicationContext applicationContext;
41
42          /**
43           * {@inheritDoc}
44           */
45          public void setApplicationContext(ApplicationContext applicationContext) {
46             this.applicationContext = applicationContext;
47
48          }
49
50          /**
51           * Getter method for the java bean <code>applicationContext</code>.
52           *
53           * @return Returns the value of the java bea <code>applicationContext</code>.
54           */
55          public ApplicationContext getApplicationContext() {
56             return applicationContext;
57
58          }
59
60       }
```

Griffin.au.org.arcs.griffin.utils. IOUtils.java

```
1        /*
2         * ----------------------------------------------------------------------------
3         * Hermes FTP Server
4         * Copyright (c) 2005-2007 Lars Behnke
5         * ----------------------------------------------------------------------------
6         *
7         * This file is part of Hermes FTP Server.
8         *
9         * Hermes FTP Server is free software; you can redistribute it and/or modify
10        * it under the terms of the GNU General Public License as published by
11        * the Free Software Foundation; either version 2 of the License, or
12        * (at your option) any later version.
13        *
14        * Hermes FTP Server is distributed in the hope that it will be useful,
```

```
15        * but WITHOUT ANY WARRANTY; without even the implied warranty of
16        * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17        * GNU General Public License for more details.
18        *
19        * You should have received a copy of the GNU General Public License
20        * along with Hermes FTP Server; if not, write to the Free Software
21        * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22        * ----------------------------------------------------------------------------
23        */
24
25       package au.org.arcs.griffin.utils;
26
27       import java.io.File;
28       import java.io.IOException;
29       import java.io.InputStream;
30       import java.io.InputStreamReader;
31       import java.io.StringWriter;
32       import java.lang.reflect.InvocationTargetException;
33       import java.lang.reflect.Method;
34       import java.text.DateFormat;
35       import java.text.MessageFormat;
36       import java.text.SimpleDateFormat;
37       import java.util.Date;
38       import java.util.Locale;
39       import java.util.Properties;
40       import java.util.StringTokenizer;
41
42
43       import org.apache.commons.lang.StringUtils;
44       import org.springframework.beans.BeanUtils;
45
46       import au.org.arcs.griffin.filesystem.FileObject;
47
48       /**
49        * This class contains some functionallity concerning I/O communication not covered by
50        * {@link org.apache.commons.io.IOUtils}.
51        *
52        * @author Lars Behnke
53        */
54       public final class IOUtils {
55
56           private static final int       FILE_SIZE_LENGTH_UNIX = 11;
57
58           private static final DateFormat DATE_FORMAT_UNIX      = new SimpleDateFormat("MMM
dd HH:mm", Locale.US);
59
60           private static final String     APP_PROPERTIES        = "/app.properties";
61
62           private static Properties       appProperties;
63
64           /**
65            * Constructor hidden.
66            */
67           private IOUtils() {
68               super();
69           }
70
71           /**
72            * Closes a file, socket, stream, writer etc. without throwing exceptions. Any exception is
```

```
73          * catched and logged. In contrast to the "Commons IO" method
<code>closeQuietly()</code> this
74          * method closes any object that has a <code>close()</code>-method and is not restricted
to
75          * readers, writers and streams.
76          *
77          * @param o The object to be closed.
78          * @return True, if object could be closed.
79          */
80         public static boolean closeGracefully(Object o) {
81             boolean result;
82             try {
83                 Method closeMethod = BeanUtils.findMethod(o.getClass(), "close", null);
84                 closeMethod.invoke(o, (Object[]) null);
85                 result = true;
86             } catch (IllegalArgumentException e) {
87                 result = false;
88             } catch (IllegalAccessException e) {
89                 result = false;
90             } catch (InvocationTargetException e) {
91                 result = false;
92             } catch (NullPointerException e) {
93                 result = false;
94             }
95             return result;
96         }
97
98         /**
99          * Returns a line formated directory entry. The permissions are set as follows: The passed
read
100         * flag is relevant for owner, group and others. The passed write flag is only relevant for the
101         * owner.
102         *
103         * @param file The file to be formatted.
104         * @param read True if readable.
105         * @param write True if writable.
106         * @return The formatted line.
107         */
108        public static String formatUnixFtpFileInfo(String user, FileObject file, boolean read, boolean
write) {
109            long size;
110            StringBuffer sb = new StringBuffer();
111            String wFlag = write ? "w" : "-";
112            String rFlag = read ? "r" : "-";
113            String permflags;
114            if (file.isDirectory()) {
115                permflags = MessageFormat.format("d{0}{1}x{0}-x{0}-x", new Object[] {rFlag, wFlag});
116                size = 0;
117            } else {
118                permflags = MessageFormat.format("-{0}{1}-{0}--{0}--", new Object[] {rFlag, wFlag});
119                size = file.length();
120            }
121            Date date = new Date(file.lastModified());
122            sb.append(permflags);
123            sb.append(" 1 ").append(user).append(" nobody ");
124            sb.append(StringUtils.leftPad("" + size, FILE_SIZE_LENGTH_UNIX));
125            sb.append(" ");
126            sb.append(DATE_FORMAT_UNIX.format(date));
127            sb.append(" ");
128            sb.append(file.getName());
```

```
129             return sb.toString();
130         }
131
132         /**
133          * Reads an arbitrary text resource from the class path.
134          *
135          * @param name The name of the resource.
136          * @param encoding The text encoding.
137          * @return The text.
138          * @throws IOException Error on accessing the resource.
139          */
140         public static String loadTextResource(String name, String encoding) throws IOException {
141             String result = null;
142             StringWriter sw = null;
143             InputStreamReader isr = null;
144             if (encoding == null) {
145                 encoding = "UTF-8";
146             }
147             try {
148                 InputStream is = IOUtils.class.getResourceAsStream(name);
149                 isr = new InputStreamReader(is, encoding);
150                 sw = new StringWriter();
151                 int c;
152                 while ((c = isr.read()) != -1) {
153                     sw.write(c);
154                 }
155                 result = sw.getBuffer().toString();
156             } finally {
157                 closeGracefully(isr);
158                 closeGracefully(sw);
159             }
160             return result;
161         }
162
163         /**
164          * Returns application properties (app.properties).
165          *
166          * @return The properties object.
167          */
168         public static Properties getAppProperties() {
169             if (appProperties == null) {
170                 appProperties = new Properties();
171                 InputStream is = IOUtils.class.getResourceAsStream(APP_PROPERTIES);
172                 try {
173                     appProperties.load(is);
174                 } catch (IOException e) {
175                     e.printStackTrace(System.out);
176                 }
177             }
178             return appProperties;
179         }
180
181         /**
182          * Tries to figure out the application's home directory.
183          *
184          * @return The directory.
185          */
186         public static File getHomeDir() {
187             File result = null;
188             String cp = System.getProperty("java.class.path");
```

```
189             StringTokenizer tokenizer = new StringTokenizer(cp, File.pathSeparator);
190             if (tokenizer.countTokens() == 1) {
191                 File jar = new File(tokenizer.nextToken());
192                 try {
193                     result = jar.getCanonicalFile().getParentFile().getParentFile();
194                 } catch (IOException e) {
195                 }
196             } else {
197                 File userDir = new File(System.getProperty("user.dir"));
198                 result = userDir.getAbsoluteFile().getParentFile();
199             }
200             return result;
201         }
202 
203         /**
204          * Checks, if UNC file naming is supported.
205          *
206          * @return True, if UNC supported.
207          */
208         public static boolean isUNCSupported() {
209             return System.getProperty("os.name").startsWith("Windows");
210         }
211     }
```

Griffin.au.org.arcs.griffin.utils. LoggingOutputStream.java

```
1       package au.org.arcs.griffin.utils;
2   
3       import java.io.ByteArrayOutputStream;
4       import java.io.IOException;
5   
6       import org.apache.commons.logging.Log;
7   
8   
9       public class LoggingOutputStream extends ByteArrayOutputStream{
10          private String lineSeparator;
11  
12          private Log log;
13          private String level;
14  
15          /**
16           * Constructor
17           * @param logger Logger to write to
18           * @param level Level at which to write the log message
19           */
20          public LoggingOutputStream(Log log, String level) {
21              super();
22              this.log = log;
23              this.level = level;
24              lineSeparator = System.getProperty("line.separator");
25          }
26  
27          /**
28           * upon flush() write the existing contents of the OutputStream
29           * to the logger as a log record.
30           * @throws java.io.IOException in case of error
31           */
32          public void flush() throws IOException {
33  
34              String record;
```

```
35              synchronized(this) {
36                  super.flush();
37                  record = this.toString();
38                  super.reset();
39
40                  if (record.length() == 0 || record.equals(lineSeparator)) {
41                      // avoid empty records
42                      return;
43                  }
44
45                  if (level!=null&&level.equalsIgnoreCase("stdout"))
46                      log.info(record);
47                  else if (level!=null&&level.equalsIgnoreCase("stderr"))
48                      log.error(record);
49      //              logger.logp(level, "", "", record);
50              }
51          }
52
53      }
```

Griffin.au.org.arcs.griffin.utils. LoggingReader.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.io.BufferedReader;
28      import java.io.IOException;
29      import java.io.Reader;
30
31      import org.apache.commons.logging.Log;
32      import org.apache.commons.logging.LogFactory;
33
34      /**
35       * A writer that adds logging capability to a passed writer. This mechanism corresponds the
36       * decorator pattern.
37       *
38       * @author Lars Behnke
```

```
39       *
40       */
41      public class LoggingReader
42          extends BufferedReader {
43
44          private static Log log = LogFactory.getLog(LoggingReader.class);
45
46          private static final int LOG_LINE_LENGTH = 80;
47
48          /**
49           * Constructor.
50           *
51           * @param in A reader.
52           */
53          public LoggingReader(Reader in) {
54              super(in);
55          }
56
57          /**
58           * Constructor.
59           *
60           * @param in A reader.
61           * @param sz Input buffer size.
62           */
63          public LoggingReader(Reader in, int sz) {
64              super(in, sz);
65          }
66
67          /**
68           * {@inheritDoc}
69           */
70          public String readLine() throws IOException {
71              String result = super.readLine();
72              if (log.isDebugEnabled()) {
73                  String x;
74                  if (result != null && result.length() >= LOG_LINE_LENGTH) {
75                      x = result.substring(0, LOG_LINE_LENGTH) + " ["
76                          + (result.length() - LOG_LINE_LENGTH) + " more chars]";
77                  } else {
78                      x = result;
79                  }
80                  log.debug("<--: " + x);
81              }
82              return result;
83          }
84
85      }
```

Griffin.au.org.arcs.griffin.utils. LoggingWriter.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
```

```
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.io.OutputStream;
28      import java.io.PrintWriter;
29      import java.io.Writer;
30
31      import org.apache.commons.logging.Log;
32      import org.apache.commons.logging.LogFactory;
33
34      /**
35       * Helper class that logs server responses.
36       *
37       * @author Lars Behnke
38       */
39      public class LoggingWriter
40          extends PrintWriter {
41
42          private static final int LOG_LINE_LENGTH = 80;
43
44          private static Log log = LogFactory.getLog(LoggingWriter.class);
45
46          /**
47           * Constructor.
48           *
49           * @param out The output stream.
50           * @param flush Automatic flush
51           */
52          public LoggingWriter(OutputStream out, boolean flush) {
53              super(out, flush);
54          }
55
56          /**
57           * Constructor.
58           *
59           * @param out The output writer.
60           * @param flush Automatic flush
61           */
62          public LoggingWriter(Writer out, boolean flush) {
63              super(out, flush);
64          }
65
66          /**
67           * {@inheritDoc}
68           */
69          public void println(String text) {
70              if (log.isDebugEnabled()) {
```

```
71                  String x;
72                  if (text != null && text.length() >= LOG_LINE_LENGTH) {
73                      x = text.substring(0, LOG_LINE_LENGTH) + " [" + (text.length() -
LOG_LINE_LENGTH)
74                              + " more chars]";
75                  } else {
76                      x = text;
77                  }
78                  log.debug("-->: " + x);
79              }
80              super.print(text);
81              super.write(0x0d);
82              super.write(0x0a);
83          //      if (checkError()) {
84          //          log.debug("Writing to control stream failed.");
85          //      }
86          }
87      }
```

Griffin.au.org.arcs.griffin.utils. NetUtils.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.net.Inet6Address;
28      import java.net.InetAddress;
29      import java.net.NetworkInterface;
30      import java.net.SocketException;
31      import java.net.UnknownHostException;
32      import java.util.Enumeration;
33
34      import org.apache.commons.logging.Log;
35      import org.apache.commons.logging.LogFactory;
36
37      /**
38       * Utility methods that facilitate networking.
39       *
```

```
40       * @author Lars Behnke
41       */
42      public final class NetUtils {
43
44          private static final String REGEX_POINT   = "\\.";
45          private static final String REGEX_COLON   = ":";
46
47          private static final int    QUADRUPLE_LEN = 4;
48          private static final int    OCTUPLE_LEN = 8;
49
50          private static Log          log           = LogFactory
51                                                    .getLog(NetUtils.class);
52
53          /**
54           * Hidden constructor.
55           */
56          private NetUtils() {
57              super();
58          }
59
60          /**
61           * Returns the machine's network address.
62           *
63           * @param fallBackToLocalhost True if loopback address should be used if
64           *            there is no net.
65           * @return The ip address.
66           */
67          public static InetAddress getMachineAddress(boolean fallBackToLocalhost) {
68              InetAddress result = null;
69              try {
70                  Enumeration<NetworkInterface> nis = NetworkInterface
71                          .getNetworkInterfaces();
72                  while (nis.hasMoreElements()) {
73                      NetworkInterface ni = nis.nextElement();
74                      InetAddress ia = getMachineAddress(ni);
75                      if (ia != null) {
76                          result = ia;
77                          break;
78                      }
79                  }
80                  if (result == null) {
81                      result = InetAddress.getLocalHost();
82                  }
83
84              } catch (SocketException e) {
85                  log.error(e);
86              } catch (UnknownHostException e) {
87                  log.error(e);
88              }
89              return result;
90          }
91
92          /**
93           * Checks the IP protocol version (IPv4 or IPv6)
94           *
95           * @param addr The address to check.
96           * @return True, if IPv6.
97           */
98          public static boolean isIPv6(InetAddress addr) {
99              return addr instanceof Inet6Address;
```

```
100         }
101
102         /**
103          * Returns the network address of a particular network interface.
104          *
105          * @param ni The network interface.
106          * @return The machine address of a particular network interface.
107          */
108         public static InetAddress getMachineAddress(NetworkInterface ni) {
109             Enumeration<InetAddress> addrs = ni.getInetAddresses();
110             InetAddress mAddr = null;
111             while (addrs.hasMoreElements()) {
112                 InetAddress addr = addrs.nextElement();
113                 if (addr.isSiteLocalAddress()) {
114                     mAddr = addr;
115                 }
116             }
117             return mAddr;
118         }
119
120         /**
121          * Checks if SSL is available.
122          *
123          * @return True, if SSL is available.
124          */
125         public static boolean isSSLAvailable() {
126             try {
127                 Class.forName("com.sun.net.ssl.internal.ssl.Provider");
128                 log.info("JSSE installed.");
129                 return true;
130             } catch (ClassNotFoundException e) {
131                 log.error("JSSE is NOT installed correctly!");
132                 return false;
133             }
134
135         }
136
137         /**
138          * Checks if the passed IP address complies to a given pattern.
139          *
140          * @param ipTemplateList String list of patterns. Wild cards are allowed: 192.168.*.*,
127.0.0.1, !85.0.0.0
141          * @param ip The IP address to check.
142          * @return True, if the passed IP address matches at least one of the
143          *         patterns.
144          */
145         public static boolean checkIPMatch(String ipTemplateList, InetAddress addr) {
146             if (isIPv6(addr)) {
147                 return checkIPv6Match(ipTemplateList, addr.getHostAddress());
148             } else {
149                 return checkIPv4Match(ipTemplateList, addr.getHostAddress());
150             }
151         }
152
153         /**
154          * Checks if the passed IPv4 complies to a given pattern.
155          *
156          * @param ipTemplateList String list of patterns. Wild cards are allowed: 192.168.*.*,
127.0.0.1, !85.0.0.0
157          * @param ipStr The IP address to check.
```

```
158         * @return True, if the passed IP address matches at least one of the
159         *         patterns.
160         */
161        public static boolean checkIPv4Match(String ipTemplateList, String ipStr) {
162
163            String[] chk = ipStr.split(REGEX_POINT);
164            if (chk.length != QUADRUPLE_LEN) {
165                throw new IllegalArgumentException("Illegal IPv4 address: " + ipStr);
166            }
167
168            boolean inverse = false;
169            String[] ipTemplateArr = ipTemplateList.split(",");
170            for (int i = 0; i < ipTemplateArr.length; i++) {
171                String t;
172                if (ipTemplateArr[i].trim().startsWith("!")) {
173                    t = ipTemplateArr[i].substring(1).trim();
174                    inverse = true;
175                } else {
176                    t = ipTemplateArr[i].trim();
177                }
178                String[] tmpl = t.split(REGEX_POINT);
179                boolean match = true;
180                for (int j = 0; j < tmpl.length; j++) {
181                    match &= ("*".equals(tmpl[j].trim()))
182                            || (tmpl[j].trim().equals(chk[j].trim())) ^ inverse;
183                }
184                if (match) {
185                    return true;
186                }
187            }
188
189            return false;
190
191        }
192
193        /**
194         * Checks if the passed IPv6 complies to a given pattern.
195         *
196         * @param ipTemplateList String list of patterns. Wild cards are allowed.
197         * @param ipStr The IP address to check.
198         * @return True, if the passed IP address matches at least one of the
199         *         patterns.
200         */
201        public static boolean checkIPv6Match(String ipTemplateList, String ipStr) {
202            String[] chk = ipStr.split(REGEX_COLON);
203            if (chk.length != OCTUPLE_LEN) {
204                throw new IllegalArgumentException("Illegal IPv6 address: " + ipStr);
205            }
206
207            boolean inverse = false;
208            String[] ipTemplateArr = ipTemplateList.split(",");
209            for (int i = 0; i < ipTemplateArr.length; i++) {
210                String t;
211                if (ipTemplateArr[i].trim().startsWith("!")) {
212                    t = ipTemplateArr[i].substring(1).trim();
213                    inverse = true;
214                } else {
215                    t = ipTemplateArr[i].trim();
216                }
217                String[] tmpl = t.split(REGEX_COLON);
```

```
218                 boolean match = true;
219                 for (int j = 0; j < tmpl.length; j++) {
220                     match &= ("*".equals(tmpl[j].trim()))
221                             || (tmpl[j].trim().equals(chk[j].trim())) ^ inverse;
222                 }
223                 if (match) {
224                     return true;
225                 }
226             }
227
228             return false;
229
230         }
231     }
```

Griffin.au.org.arcs.griffin.utils.package.html

```
1       <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
2       <html>
3           <head>
4           </head>
5           <body bgcolor="white">
6
7               Contains utility classes used throughout the project.
8
9           </body>
10      </html>
```

Griffin.au.org.arcs.griffin.utils.SecurityUtil.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.io.FileInputStream;
28      import java.io.FileNotFoundException;
29      import java.io.IOException;
30      import java.io.InputStream;
```

```
31      import java.security.KeyManagementException;
32      import java.security.KeyStore;
33      import java.security.KeyStoreException;
34      import java.security.MessageDigest;
35      import java.security.NoSuchAlgorithmException;
36      import java.security.UnrecoverableKeyException;
37      import java.security.cert.CertificateException;
38
39      import javax.net.ssl.KeyManagerFactory;
40      import javax.net.ssl.SSLContext;
41
42
43      import org.apache.commons.codec.binary.Base64;
44
45      import au.org.arcs.griffin.exception.FtpConfigException;
46
47      /**
48       * Security related utility methods.
49       *
50       * @author Lars Behnke
51       */
52      public final class SecurityUtil {
53
54          private static final String ALG_END   = "}";
55
56          private static final String ALG_START = "{";
57
58          /**
59           * Constructor hidden.
60           */
61          private SecurityUtil() {
62              super();
63          }
64
65          /**
66           * Calculates based on the passed parameters an hash code and returns it as BASE64-
String. The
67           * used algorithm is prepended.
68           *
69           * @param password The password to encode.
70           * @param algorithm The alogrithm to use (MD5 e.g.)
71           * @return The encoded password as string.
72           * @throws NoSuchAlgorithmException Passed alogrith is not supported.
73           */
74          public static String encodePassword(String password, String algorithm) throws
NoSuchAlgorithmException {
75              if (password == null) {
76                  throw new IllegalArgumentException("No password passed");
77              }
78              String result = password.trim();
79              if (algorithm == null) {
80                  return result;
81              }
82              MessageDigest digest = MessageDigest.getInstance(algorithm);
83              digest.update(password.getBytes());
84              byte[] digestedPassword = digest.digest();
85              String base64password = new String(Base64.encodeBase64(digestedPassword));
86              return ALG_START + algorithm + ALG_END + base64password;
87          }
88
```

```
89          /**
90           * Checks the validity of the password password based on a given hashcode. The hashcode
to check
91           * against contains the used algorithm has prefix. Example:
{MD5}Cwz8B/yoHJVquRgdhXb0qA==.
92           *
93           * @param passwordHash The hash code to check against.
94           * @param password The password to check.
95           * @return True, if password is valid.
96           * @throws NoSuchAlgorithmException Algorithm (from hash prefix) is not supported.
97           */
98          public static boolean checkPassword(String passwordHash, String password) throws
NoSuchAlgorithmException {
99              if (passwordHash == null || password == null) {
100                 return false;
101             }
102             String algorithm = null;
103             int startIdx = passwordHash.indexOf(ALG_START);
104             int endIdx = passwordHash.indexOf(ALG_END);
105             if (startIdx == 0 && endIdx > startIdx) {
106                 algorithm = passwordHash.substring(startIdx + 1, endIdx);
107                 String hashStr = encodePassword(password.trim(), algorithm);
108                 return passwordHash.equals(hashStr);
109             } else {
110                 return passwordHash.trim().equals(password.trim());
111             }
112         }
113
114         /**
115          * Create the security context required for SSL communication.
116          *
117          * @param keyStoreFile The name of the keystore file.
118          * @param keyStorePassword The password for the keystore.
119          * @return The context.
120          * @throws FtpConfigException Thrown on error in configuration.
121          */
122         public static SSLContext createSslContext(String keyStoreFile, char[] keyStorePassword)
123                 throws FtpConfigException {
124             SSLContext sslContext = null;
125             try {
126                 /* Get keystore file and password */
127                 InputStream ksInputStream = getKeyStoreInputStream(keyStoreFile);
128
129                 /*
130                  * Get the java keystore object an key manager. A keystore is where keys and
131                  * certificates are kept.
132                  */
133                 KeyStore keystore = KeyStore.getInstance("JKS");
134                 keystore.load(ksInputStream, keyStorePassword);
135                 KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");
136                 kmf.init(keystore, keyStorePassword);
137
138                 /*
139                  * An SSLContext is an environment for implementing JSSE. It is used to create a
140                  * ServerSocketFactory
141                  */
142                 sslContext = SSLContext.getInstance("SSL");
143                 sslContext.init(kmf.getKeyManagers(), null, null);
144             } catch (KeyManagementException e) {
145
```

```
146                 throw new SecurityException("A key management authorization problem occurred.");
147             } catch (FileNotFoundException e) {
148                 throw new SecurityException("The key store file could not be found.");
149             } catch (KeyStoreException e) {
150                 throw new SecurityException("A key store problem occurred.");
151             } catch (NoSuchAlgorithmException e) {
152                 throw new SecurityException("The hash algorithm is not supported.");
153             } catch (CertificateException e) {
154                 throw new SecurityException("Certificate could not be loaded.");
155             } catch (UnrecoverableKeyException e) {
156                 throw new SecurityException("Key store cannot be recovered.");
157             } catch (IOException e) {
158                 throw new SecurityException("Reading the key store failed.");
159             }
160             return sslContext;
161         }
162
163         private static InputStream getKeyStoreInputStream(String ksFile) throws
FileNotFoundException {
164             if (ksFile == null) {
165                 throw new FileNotFoundException("Keystore file not defined.");
166             }
167             return new FileInputStream(ksFile);
168
169         }
170
171     }
```

Griffin.au.org.arcs.griffin.utils.StringUtils.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.util.ArrayList;
28      import java.util.List;
29      import java.util.regex.Pattern;
30
```

```
31      /**
32       * Utilities for handling strings.
33       *
34       * @author Behnke
35       */
36      public final class StringUtils {
37
38          /**
39           * Hidden Constructor.
40           */
41          private StringUtils() {
42          }
43
44          /**
45           * Converts the passed list to a comma separated string.
46           *
47           * @param list The list to be converted.
48           * @return The string.
49           */
50          public static String convertLstToString(List<String> list) {
51              StringBuffer sb = new StringBuffer();
52              for (String s : list) {
53                  if (sb.length() > 0) {
54                      sb.append(", ");
55                  }
56                  sb.append(s);
57              }
58              return sb.toString();
59          }
60
61          /**
62           * Validates the syntax of the passed email.
63           *
64           * @param email The email.
65           * @return True, if syntax ok.
66           */
67          public static boolean validateEmail(String email) {
68              String patternStr = "^[a-zA-Z][\\w\\.-]*[a-zA-Z0-9]@"
69                      + "[a-zA-Z0-9][\\w\\.-]*[a-zA-Z0-9]\\.[a-zA-Z][a-zA-Z\\.]*[a-zA-Z]$";
70              return Pattern.matches(patternStr, email);
71          }
72
73          /**
74          * Quotes the special characters of a given regular expression string. Example: <blockquote>
75           *
76           * <pre>
77           *     This.is.an.example  --&gt;  This\.is\.an\.example
78           * </pre>
79           *
80           * </blockquote>
81           *
82           * @param s The unencoded string.
83           * @return The encoded string.
84           */
85          public static String encodeRegex(String s) {
86              StringBuffer sb = new StringBuffer();
87              for (int i = 0; i < s.length(); i++) {
88                  char c = s.charAt(i);
89                  if (".-\\[]{}()".indexOf(c) > -1) {
90                      sb.append("\\");
```

```
91                  }
92                  sb.append(c);
93              }
94              return sb.toString();
95          }
96
97          /**
98           * Parses a string encoded list of integer values. The list may include ranges. Example:
99           *
100          * <pre>
101          *    1000-1011,1022,1023,2000-3000
102          * </pre>
103          *
104          * @param portListStr Encoded list of integer values.
105          * @return List of alle Integers defined by the passed list.
106          */
107         public static Integer[] parseIntegerList(String portListStr) {
108
109             if (portListStr == null) {
110                 return null;
111             }
112             List<Integer> portList = new ArrayList<Integer>();
113             String[] allowedPorts = portListStr.split(",");
114             if (allowedPorts != null && allowedPorts.length > 0 && allowedPorts[0] != null
115                     && allowedPorts[0].trim().length() > 0) {
116                 for (int i = 0; i < allowedPorts.length; i++) {
117                     String[] portRange = allowedPorts[i].split("\\-");
118                     if (portRange.length > 1) {
119                         int portMin = Integer.parseInt(portRange[0].trim());
120                         int portMax = Integer.parseInt(portRange[1].trim());
121                         for (int port = portMin; port <= portMax; port++) {
122                             portList.add(new Integer(port));
123                         }
124                     } else {
125                         int port = Integer.parseInt(portRange[0].trim());
126                         portList.add(new Integer(port));
127                     }
128                 }
129             }
130             return (Integer[]) portList.toArray(new Integer[portList.size()]);
131
132         }
133     }
```

Griffin.au.org.arcs.griffin.utils.TransferMonitor.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
```

```
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * -----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.text.DecimalFormat;
28
29      import au.org.arcs.griffin.cmd.AbstractFtpCmd;
30
31      /**
32       * Controls the upload/download bandwidth.
33       *
34       * @author Behnke
35       */
36      public class TransferMonitor {
37
38          private static final int SLEEP_INTERVAL = 100;
39
40          private double          maxRate;
41
42          private long            startTime;
43
44          private long            transferredBytes;
45
46          private AbstractFtpCmd cmd;
47
48          private static final int PERF_MARKER_INTERVAL = 5000;
49
50          private long lastPerfMarkerTime;
51
52          private DecimalFormat decFormatter = new DecimalFormat("###.#");
53
54              private boolean showPerfMarker;
55
56          /**
57           * Constructor.
58           */
59          public TransferMonitor() {
60              this(-1);
61          }
62
63          /**
64           * Constructor.
65           *
66           * @param maxRate KB per second.
67           */
68          public TransferMonitor(double maxRate) {
69              this.maxRate = maxRate;
70          }
71
72          /**
73           * Initializes the object.
74           *
```

```
75          * @param maxRate The maximum transfer rate.
76          */
77         public void init(double maxRate, AbstractFtpCmd cmd) {
78            this.maxRate = maxRate;
79            startTime = System.currentTimeMillis();
80            transferredBytes = 0;
81            this.cmd=cmd;
82         }
83
84         public void sendPerfMarker(){
85            if (!showPerfMarker) return;
86            lastPerfMarkerTime=System.currentTimeMillis();
87            StringBuffer perf=new StringBuffer("112-Perf Marker\r\n");
88            perf.append(" Timestamp:  "+decFormatter.format(lastPerfMarkerTime/1000d)+"\r\n");
89            perf.append(" Stripe Index: 0\r\n");
90            perf.append(" Stripe Bytes Transferred: "+transferredBytes+"\r\n");
91            perf.append(" Total Stripe Count: 1\r\n");
92            perf.append("112 End.");
93            cmd.out(perf.toString());
94         }
95
96         /**
97          * Returns the current transfer rate.
98          *
99          * @return The transfer rate in KB per seconds.
100         */
101        public double getCurrentTransferRate() {
102           double seconds = (System.currentTimeMillis() - startTime) / 1024d;
103           if (seconds <= 0) {
104              seconds = 1;
105           }
106           return (transferredBytes / 1024d) / seconds;
107
108        }
109
110        /**
111         * Updates the transfer rate statistics. If the maximum rate has been exceeded, the method
112         * pauses.
113         *
114         * @param byteCount The number of bytes previously transfered.
115         */
116        public void execute(long byteCount) {
117           transferredBytes += byteCount;
118           if (System.currentTimeMillis()-lastPerfMarkerTime>PERF_MARKER_INTERVAL)
sendPerfMarker();
119     //       while (maxRate >= 0 && getCurrentTransferRate() > maxRate) {
120     //          try {
121     //             Thread.sleep(SLEEP_INTERVAL);
122     //          } catch (InterruptedException e) {
123     //             break;
124     //          }
125     //       }
126        }
127
128        public long getTransferredBytes(){
129           return this.transferredBytes;
130        }
131
132           public void hidePerfMarker() {
133                 showPerfMarker = false;
```

```
134
135             }
136
137             public void showPerfMarker() {
138                     showPerfMarker = true;
139
140             }
141
142     }
```

Griffin.au.org.arcs.griffin.utils.VarMerger.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.utils;
26
27      import java.util.Properties;
28      import java.util.regex.Matcher;
29      import java.util.regex.Pattern;
30
31      /**
32       * Replaces all placeholders in a given text document with the values of a properties-Object.
33       * Example: The text "Hello ${name}" becomes "Hello world" provided that passed properties
object
34       * contains the entry name=world.
35       * <p>
36       * <code>
37       *   VarMerger vm = new VarMerger("The ${dog} chases the ${cat}");
38       *   Properties p = new Properties();
39       *   p.put("dog", "fox");
40       *   p.put("cat", "mouse");
41       *   vm.merge(p);
42       *   String neuerSatz = vm.getText();
43       * </code>
44       *
45       * @author Lars Behnke
46       */
47      public class VarMerger {
```

```
48
49          private static final String VAR_PATTERN = "(\\$\\{)([\\w\\.]*)(\\})";
50
51          private String              text;
52
53          /**
54           * @param text Der Text containing placeholders.
55           */
56          public VarMerger(String text) {
57              setText(text);
58          }
59
60          /**
61           * Replaces the placeholders with the passed values.
62           *
63           * @param props The values.
64           */
65          public void merge(Properties props) {
66              StringBuffer sb = new StringBuffer();
67              Pattern pattern = Pattern.compile(VAR_PATTERN);
68              Matcher m = pattern.matcher(getText());
69              boolean hasNext = m.find();
70              while (hasNext) {
71                  String key = m.group(2);
72                  String val = props.getProperty(key);
73                  if (val != null) {
74                      val = quoteReplacement(val);
75                      m.appendReplacement(sb, val);
76                  }
77                  hasNext = m.find();
78              }
79              m.appendTail(sb);
80              text = sb.toString();
81          }
82
83          /**
84           * Returns a literal replacement <code>String</code> for the specified
<code>String</code>.
85           * This method produces a <code>String</code> that will work use as a literal replacement
86           * <code>s</code> in the <code>appendReplacement</code> method of the {@link
Matcher} class.
87           * The <code>String</code> produced will match the sequence of characters in
<code>s</code>
88           * treated as a literal sequence. Slashes ('\') and dollar signs ('$') will be given no special
89           * meaning.
90           *
91           * @param s The string to be literalized
92           * @return A literal string replacement
93           */
94          private static String quoteReplacement(String s) {
95              if ((s.indexOf('\\') == -1) && (s.indexOf('$') == -1)) {
96                  return s;
97              }
98              StringBuffer sb = new StringBuffer();
99              for (int i = 0; i < s.length(); i++) {
100                 char c = s.charAt(i);
101                 if (c == '\\') {
102                     sb.append('\\');
103                     sb.append('\\');
104                 } else if (c == '$') {
```

```
105                 sb.append('\\');
106                 sb.append('$');
107             } else {
108                 sb.append(c);
109             }
110         }
111         return sb.toString();
112     }
113
114     /**
115      * Sets the text containing placeholders.
116      *
117      * @param text The text.
118      */
119     public void setText(String text) {
120         if (text == null) {
121             throw new IllegalArgumentException("NULL-Values can not be processed.");
122         }
123         this.text = text;
124     }
125
126     /**
127      * @return Der Text mit den ersetzten Platzhaltern.
128      */
129     public String getText() {
130         return text;
131     }
132
133     /**
134      * Prï¿œft, ob der Text noch Platzhalter enthï¿œlt, die noch nicht ersetzt wurden.
135      *
136      * @return True, falls der Text keine Platzhalter mehr enthï¿œlt.
137      */
138     public boolean isReplacementComplete() {
139         Pattern pattern = Pattern.compile(VAR_PATTERN);
140         boolean found = pattern.matcher(getText()).find();
141         return !found;
142     }
143
144 }
```

Griffin.au.org.arcs.griffin.usermanager.package.html

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
2   <html>
3   <head>
4   </head>
5   <body bgcolor="white">
6
7   Provides for the basic interface implemented by user managers.
8
9   </body>
10  </html>
```

Griffin.au.org.arcs.griffin.usermanager.UserManager.java

```
1   /*
2    * ------------------------------------------------------------------------
3    * Hermes FTP Server
4    * Copyright (c) 2005-2007 Lars Behnke
```

```
5         * --------------------------------------------------------------------------
6         *
7         * This file is part of Hermes FTP Server.
8         *
9         * Hermes FTP Server is free software; you can redistribute it and/or modify
10        * it under the terms of the GNU General Public License as published by
11        * the Free Software Foundation; either version 2 of the License, or
12        * (at your option) any later version.
13        *
14        * Hermes FTP Server is distributed in the hope that it will be useful,
15        * but WITHOUT ANY WARRANTY; without even the implied warranty of
16        * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17        * GNU General Public License for more details.
18        *
19        * You should have received a copy of the GNU General Public License
20        * along with Hermes FTP Server; if not, write to the Free Software
21        * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22        * --------------------------------------------------------------------------
23        */
24
25       package au.org.arcs.griffin.usermanager;
26
27       import java.util.List;
28       import java.util.Map;
29
30       import au.org.arcs.griffin.common.FtpSessionContext;
31       import au.org.arcs.griffin.exception.FtpConfigException;
32       import au.org.arcs.griffin.exception.FtpQuotaException;
33       import au.org.arcs.griffin.usermanager.model.GroupDataList;
34       import au.org.arcs.griffin.usermanager.model.UserData;
35
36
37       /**
38        * Generic description of user management classes.
39        *
40        * @author Lars Behnke
41        */
42       public interface UserManager {
43
44          /**
45           * Registers the current user's resource consumption. That is, downloaded or uploaded
bytes or
46           * files.
47           *
48           * @param user The user name.
49           * @param limitName The name of the consumption type (resource limit).
50           * @param value The consumed resources.
51           * @throws FtpQuotaException Thrown if resource limit has been reached.
52           */
53          void updateIncrementalStatistics(String user, String limitName, long value) throws
FtpQuotaException;
54
55          /**
56           * Registers the current user's transfer rate. A mean value is calculated.
57           *
58           * @param user The user name.
59           * @param avgKeyName The key of the transfer rate (resource limit).
60           * @param value The consumed resources.
61           */
62          void updateAverageStatistics(String user, String avgKeyName, long value);
```

```
63
64          /**
65           * Checks the resource consumption of the passed users. Only the passed limits are
condidered.
66           *
67           * @param user The user name.
68           * @param limitNames The resource limits to condider.
69           * @throws FtpQuotaException Thrown if at least one limit has been reached.
70           */
71          void checkResourceConsumption(String user, String[] limitNames) throws
FtpQuotaException;
72
73          /**
74           * Returns the resource consumption statistics for a given user.
75           *
76           * @param user The user.
77           * @return The statistics.
78           */
79          Map<String, Long> getUserStatistics(String user);
80
81          /**
82           * Returns the logged statistics for all user and all available dates (since the server was
83           * started).
84           *
85           * @return The statistics.
86           */
87          Map<String, Map<String, Long>> getAllStatistics();
88
89          /**
90           * Returns object representations of all registered users.
91           *
92           * @param username The user's name.
93           * @return The user data.
94           * @throws FtpConfigException Error in configuration.
95           */
96          UserData getUserData(String username) throws FtpConfigException;
97
98          /**
99           * Returns object representations of all registered users.
100          *
101          * @return The users.
102          * @throws FtpConfigException Error in configuration.
103          */
104         List<UserData> getUserDataList() throws FtpConfigException;
105
106         /**
107          * Returns object representations of all groups the passed user belongs to.
108          *
109          * @param username The user's name.
110          * @return The group data.
111          * @throws FtpConfigException Error in configuration.
112          */
113         GroupDataList getGroupDataList(String username) throws FtpConfigException;
114
115         /**
116          * Validates the passed user credentials.
117          *
118          * @param user The username.
119          * @param password The password
120          * @param ctx The context of the current session.
```

```
121         * @return True, if credentials are valid.
122         * @throws FtpConfigException Error on reading or processing a configuration file.
123         */
124        boolean authenticate(String user, String password, FtpSessionContext ctx) throws
FtpConfigException;
125        UserData authenticate(String dn) throws FtpConfigException;
126
127        /**
128         * (Re)loads the configuration.
129         *
130         * @throws FtpConfigException Error on reading or processing a configuration file.
131         */
132        void load() throws FtpConfigException;
133
134        /**
135         * Checks if the configuration is loaded.
136         *
137         * @return True, if configuration has already been loaded.
138         * @throws FtpConfigException Error on reading or processing a configuration file.
139         */
140        boolean isLoaded() throws FtpConfigException;
141
142    }
```

Griffin.au.org.arcs.griffin.usermanager.impl.package.html

```
1       <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
2       <html>
3       <head>
4       </head>
5       <body bgcolor="white">
6
7       Implements a concrete user manager that is based on xml configuration files.
8       Also related classes are included in this package.
9
10      </body>
11      </html>
```

Griffin.au.org.arcs.griffin.usermanager.impl. XmlFileReader.java

```
1       /*
2        * ------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
```

```
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.usermanager.impl;
26
27      import java.io.BufferedReader;
28      import java.io.File;
29      import java.io.FileReader;
30      import java.io.IOException;
31      import java.io.InputStream;
32      import java.io.InputStreamReader;
33      import java.util.List;
34
35
36      import org.apache.commons.logging.Log;
37      import org.apache.commons.logging.LogFactory;
38      import org.dom4j.Document;
39      import org.dom4j.DocumentException;
40      import org.dom4j.Element;
41      import org.dom4j.io.SAXReader;
42
43      import au.org.arcs.griffin.exception.FtpConfigException;
44      import au.org.arcs.griffin.usermanager.model.GroupData;
45      import au.org.arcs.griffin.usermanager.model.MappingData;
46      import au.org.arcs.griffin.usermanager.model.PermissionData;
47      import au.org.arcs.griffin.usermanager.model.UserData;
48      import au.org.arcs.griffin.usermanager.model.UserManagerData;
49
50      /**
51       * Reads the user management configuration data from a file.
52       *
53       * @author Lars Behnke
54       */
55      public class XmlFileReader {
56
57          private static Log         log                       = LogFactory.getLog(XmlFileReader.class);
58
59          private static final String DEFAULT_GRIFFIN_USERS_FILE   = "griffin-users.xml";
60
61          private static final String ATTR_PATH                 = "path";
62
63          private static final String XPATH_PERMISSIONS           = "permissions/permission";
64
65          private static final String ATTR_VALUE                = "value";
66
67          private static final String ATTR_FLAG                 = "flag";
68
69          private static final String XPATH_LIMITS              = "limits/limit";
70
71          private static final String XPATH_GROUPS              = "/user-manager/groups/group";
72
73          private static final String ATTR_NAME                 = "name";
74
75          private static final String ELEM_GROUP_REF            = "group-ref";
76
77          private static final String ATTR_DIR                  = "dir";
78
79          private static final String ATTR_PASSWORD             = "password";
```

```
80
81          private static final String ATTR_FULLNAME               = "fullname";
82
83          private static final String ATTR_ADMINROLE              = "adminrole";
84
85          private static final String ATTR_UID                    = "uid";
86
87          private static final String ELEM_USER                   = "user";
88
89          private static final String ATTR_DEFAULT_DIR            = "default-dir";
90
91          private static final String XPATH_USERS                 = "/user-manager/users";
92
93          private static final String XPATH_MAPPINGS              = "/user-manager/mappings";
94
95          private static final String ELEM_MAPPING                = "mapping";
96
97          private static final String ATTR_DN                     = "dn";
98
99          private String              filename;
100
101         /**
102          * Getter method for the java bean <code>filename</code>.
103          *
104          * @return Returns the value of the java bean <code>filename</code>.
105          */
106         public String getFilename() {
107             if (filename == null || filename.length() == 0) {
108                 String ctxDir = System.getProperty("griffin.ctx.dir");
109                 File file;
110                 if (ctxDir != null) {
111                     file = new File(ctxDir, DEFAULT_GRIFFIN_USERS_FILE);
112                 } else {
113                     file = new File(DEFAULT_GRIFFIN_USERS_FILE);
114                 }
115                 filename = file.getAbsolutePath();
116             }
117             log.info("User configuration file: " + filename);
118             return filename;
119         }
120
121         /**
122          * Setter method for the java bean <code>filename</code>.
123          *
124          * @param filename The value of filename to set.
125          */
126         public void setFilename(String filename) {
127             this.filename = filename;
128         }
129
130         /**
131          * Reads the user management data from a file. If the file was not found the classpath is
132          * searched.
133          *
134          * @return The user management data.
135          * @throws FtpConfigException Error on reading or processing a configuration file.
136          */
137         public UserManagerData read() throws FtpConfigException {
138             UserManagerData result;
139             File file = null;
```

```
140             try {
141                 SAXReader reader = new SAXReader();
142                 file = new File(getFilename());
143                 BufferedReader br;
144                 if (file.exists()) {
145                     br = new BufferedReader(new FileReader(file));
146                 } else {
147                     InputStream is = getClass().getResourceAsStream("/" +
DEFAULT_GRIFFIN_USERS_FILE);
148                     br = new BufferedReader(new InputStreamReader(is));
149                 }
150                 Document doc = reader.read(br);
151                 result = process(doc);
152             } catch (IOException e) {
153                 throw new FtpConfigException("Reading " + getFilename() + " failed.");
154             } catch (DocumentException e) {
155                 log.error(e.toString());
156                 throw new FtpConfigException("Error while processing the configuration file " + file +
".");
157             }
158
159             return result;
160         }
161
162         private UserManagerData process(Document doc) {
163             UserManagerData result = new UserManagerData();
164             processUserData(doc, result);
165             processGroupData(doc, result);
166             processMappingData(doc, result);
167
168             return result;
169         }
170
171         private void processMappingData(Document doc, UserManagerData umd) {
172             Element mappingsElement = (Element) doc.selectSingleNode(XPATH_MAPPINGS);
173
174             List<Element> mappingElements = mappingsElement.selectNodes(ELEM_MAPPING);
175             for (Element mappingElement : mappingElements) {
176                 String uid = mappingElement.attributeValue(ATTR_UID);
177                 String dn = mappingElement.attributeValue(ATTR_DN);
178                 MappingData mappingData = new MappingData();
179                 mappingData.setDn(dn);
180                 mappingData.setUid(uid);
181                 umd.getMappingData().add(mappingData);
182
183             }
184             }
185
186             private void processUserData(Document doc, UserManagerData umd) {
187             Element usersElement = (Element) doc.selectSingleNode(XPATH_USERS);
188             String defaultDir = usersElement.attributeValue(ATTR_DEFAULT_DIR);
189
190             List<Element> userElements = usersElement.selectNodes(ELEM_USER);
191             for (Element userElement : userElements) {
192                 String uid = userElement.attributeValue(ATTR_UID);
193                 String fullName = userElement.attributeValue(ATTR_FULLNAME);
194                 String password = userElement.attributeValue(ATTR_PASSWORD);
195                 String adminrole = userElement.attributeValue(ATTR_ADMINROLE);
196                 String dir = userElement.attributeValue(ATTR_DIR);
197                 if (dir == null || dir.length() == 0) {
```

```
198                 dir = defaultDir;
199             }
200             UserData userData = new UserData();
201             userData.setFullName(fullName);
202             userData.setUid(uid);
203             userData.setPassword(password);
204             userData.setAdminRole(new Boolean(adminrole).booleanValue());
205             userData.setDir(dir);
206             List<Element> groupRefElements = userElement.selectNodes(ELEM_GROUP_REF);
207             for (Element element : groupRefElements) {
208                 String groupRefName = element.attributeValue(ATTR_NAME);
209                 if (groupRefName != null) {
210                     userData.addGroupName(groupRefName.trim());
211                 }
212             }
213             umd.getUserData().add(userData);
214
215         }
216     }
217
218     private void processGroupData(Document doc, UserManagerData umd) {
219         List<Element> groupElements = doc.selectNodes(XPATH_GROUPS);
220         for (Element groupElement : groupElements) {
221             String name = groupElement.attributeValue(ATTR_NAME);
222             GroupData groupData = new GroupData();
223             groupData.setName(name);
224
225             List<Element> limitElements = groupElement.selectNodes(XPATH_LIMITS);
226             for (Element limitElement : limitElements) {
227                 String limitName = limitElement.attributeValue(ATTR_NAME);
228                 String limitValue = limitElement.attributeValue(ATTR_VALUE);
229                 if (limitName != null && limitValue != null) {
230                     Long limitLong = new Long(limitValue);
231                     groupData.getLimits().put(limitName, limitLong);
232                 }
233             }
234             List<Element> permissionElements =
groupElement.selectNodes(XPATH_PERMISSIONS);
235             for (Element permissionElement : permissionElements) {
236                 String path = permissionElement.attributeValue(ATTR_PATH);
237
238                 /* value attribute is supported for backward compatibility */
239                 String value = permissionElement.attributeValue(ATTR_VALUE);
240                 String flag = permissionElement.attributeValue(ATTR_FLAG);
241                 int permissionValue = getPermissionValue(value, flag);
242                 PermissionData pd = new PermissionData();
243                 pd.setPermission(permissionValue);
244                 pd.setTemplate(path);
245                 groupData.getPermissions().add(pd);
246             }
247             umd.getGroupData().addGroup(groupData);
248         }
249     }
250
251     /**
252      * Decides on the permission value based on the passed arguments. Value attribute is
supported
253      * for backward compatibility
254      *
255      * @param value The permission value 1=read, 3=read/write.
```

```
256         * @param flag The permission flag R=read, RW=read/write.
257         * @return The permission value.
258         */
259        private int getPermissionValue(String value, String flag) {
260            flag = flag == null ? "" : flag.toUpperCase().trim();
261            int permissionValue = 0;
262            if (value != null && value.length() > 0) {
263                permissionValue = Integer.parseInt(value);
264            }
265            if (flag.indexOf('R') >= 0) {
266                permissionValue |= 1;
267            }
268            if (flag.indexOf('W') >= 0) {
269                permissionValue |= 2;
270            }
271            return permissionValue;
272        }
273
274    }
```

Griffin.au.org.arcs.griffin.usermanager.impl.XmlFileUserManager.java

```
1       /*
2        * ---------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ---------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ---------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.usermanager.impl;
26
27      import java.security.NoSuchAlgorithmException;
28      import java.text.DateFormat;
29      import java.text.SimpleDateFormat;
30      import java.util.Collections;
31      import java.util.Date;
32      import java.util.HashMap;
33      import java.util.List;
34      import java.util.Map;
35
36
37      import org.apache.commons.logging.Log;
38      import org.apache.commons.logging.LogFactory;
```

```
39
40      import au.org.arcs.griffin.common.FtpSessionContext;
41      import au.org.arcs.griffin.exception.FtpConfigException;
42      import au.org.arcs.griffin.exception.FtpQuotaException;
43      import au.org.arcs.griffin.usermanager.UserManager;
44      import au.org.arcs.griffin.usermanager.model.GroupData;
45      import au.org.arcs.griffin.usermanager.model.GroupDataList;
46      import au.org.arcs.griffin.usermanager.model.MappingData;
47      import au.org.arcs.griffin.usermanager.model.UserData;
48      import au.org.arcs.griffin.usermanager.model.UserManagerData;
49      import au.org.arcs.griffin.utils.SecurityUtil;
50      import au.org.arcs.griffin.utils.StringUtils;
51
52      /**
53       * @author Lars Behnke
54       */
55      public class XmlFileUserManager implements UserManager {
56
57          private static Log log =
LogFactory.getLog(XmlFileUserManager.class);
58
59          private XmlFileReader fileReader;
60
61          private UserManagerData userManagerData;
62
63          private Map<String, Map<String, Long>> resourceConsumption = Collections
64                                                  .synchronizedMap(new HashMap<String,
Map<String, Long>>());
65
66          private DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
67
68          /**
69           * {@inheritDoc}
70           */
71          public synchronized boolean authenticate(String user, String password, FtpSessionContext
ctx)
72                  throws FtpConfigException {
73              boolean result = false;
74              UserData userData = userManagerData.getUserData(user);
75              if (userData == null) {
76                  return result;
77              }
78              if (userData.getPassword() == null) {
79                  result = password != null && StringUtils.validateEmail(password);
80              } else {
81                  try {
82                      result = SecurityUtil.checkPassword(userData.getPassword(), password);
83                  } catch (NoSuchAlgorithmException e) {
84                      throw new FtpConfigException("Algorithm not supported: " +
userData.getPassword());
85                  }
86              }
87
88              return result;
89          }
90
91          public synchronized UserData authenticate(String dn) throws FtpConfigException {
92              MappingData mappingData = userManagerData.getMappingData(dn);
93              if (mappingData==null){
94                  throw new FtpConfigException("Could not find a mapping for the given DN.");
```

```
95              }
96              UserData userData = userManagerData.getUserData(mappingData.getUid());
97              if (userData == null) {
98                  throw new FtpConfigException("Could not find the mapped user.");
99              }
100             return userData;
101         }
102
103         /**
104          * {@inheritDoc}
105          */
106         public synchronized GroupDataList getGroupDataList(String username) throws
FtpConfigException {
107             UserData userData = userManagerData.getUserData(username);
108             if (userData == null) {
109                 throw new FtpConfigException("User " + username + " not configured.");
110             }
111             GroupDataList groupList = new GroupDataList();
112             for (String groupName : userData.getGroupNames()) {
113                 GroupData groupData = userManagerData.getGroupData(groupName);
114                 groupList.addGroup(groupData);
115             }
116             return groupList;
117
118         }
119
120         /**
121          * {@inheritDoc}
122          */
123         public synchronized List<UserData> getUserDataList() throws FtpConfigException {
124             return userManagerData.getUserData();
125         }
126
127         /**
128          * {@inheritDoc}
129          */
130         public synchronized void load() throws FtpConfigException {
131             userManagerData = fileReader.read();
132         }
133
134         /**
135          * Getter method for the java bean <code>fileReader</code>.
136          *
137          * @return Returns the value of the java bean <code>fileReader</code>.
138          */
139         public XmlFileReader getFileReader() {
140             return fileReader;
141         }
142
143         /**
144          * Setter method for the java bean <code>fileReader</code>.
145          *
146          * @param fileReader The value of fileReader to set.
147          */
148         public void setFileReader(XmlFileReader fileReader) {
149             this.fileReader = fileReader;
150         }
151
152         /**
153          * {@inheritDoc}
```

```
154         */
155        public void checkResourceConsumption(String user, String[] limitNames) throws
FtpQuotaException {
156            Map<String, Long> userConsumptions = getUserStatistics(user);
157            for (String limitName : userConsumptions.keySet()) {
158                Long consumptionObj = userConsumptions.get(limitName);
159                long consumption = consumptionObj == null ? 0 : consumptionObj.longValue();
160                long limit;
161                try {
162                    GroupDataList list = getGroupDataList(user);
163                    limit = list.getUpperLimit(limitName);
164                } catch (FtpConfigException e) {
165                    log.error(e);
166                    limit = 0;
167                }
168                if (consumption >= limit) {
169                   throw new FtpQuotaException(createQuotaMessage(limitName, consumption, limit));
170                }
171            }
172
173        }
174
175        /**
176         * {@inheritDoc}
177         */
178        public void updateIncrementalStatistics(String user, String limitName, long value)
179                throws FtpQuotaException {
180            Map<String, Long> userConsumptions = getUserStatistics(user);
181            Long consumptionObj = (Long) userConsumptions.get(limitName);
182            long consumption = consumptionObj == null ? 0 : consumptionObj.longValue();
183            consumption += value;
184            long limit;
185            try {
186                GroupDataList list = getGroupDataList(user);
187                limit = list.getUpperLimit(limitName);
188            } catch (FtpConfigException e) {
189                log.error(e);
190                limit = 0;
191            }
192            if (consumption > limit) {
193                throw new FtpQuotaException(createQuotaMessage(limitName, consumption, limit));
194            }
195            userConsumptions.put(limitName, new Long(consumption));
196
197        }
198
199        /**
200         * {@inheritDoc}
201         */
202        public void updateAverageStatistics(String user, String avgKey, long value) {
203            String countKey = "Sample count (" + avgKey + ")";
204            Map<String, Long> userConsumptions = getUserStatistics(user);
205            Long prevAvgObj = (Long) userConsumptions.get(avgKey);
206            long prevAvg = prevAvgObj == null ? 0 : prevAvgObj.longValue();
207            Long prevCountObj = (Long) userConsumptions.get(countKey);
208            long prevCount = prevCountObj == null ? 0 : prevCountObj.longValue();
209            long currentAvg = (prevAvg * prevCount + value) / (prevCount + 1);
210            userConsumptions.put(avgKey, new Long(currentAvg));
211            userConsumptions.put(countKey, new Long(prevCount + 1));
212        }
```

```
213
214         /**
215          * {@inheritDoc}
216          */
217         public Map<String, Long> getUserStatistics(String user) {
218             String userAndDate = getUserAndDateKey(user);
219             return getUserResourceConsumptions(userAndDate);
220         }
221
222         /**
223          * {@inheritDoc}
224          */
225         public Map<String, Map<String, Long>> getAllStatistics() {
226             return resourceConsumption;
227         }
228
229         /**
230          * {@inheritDoc}
231          */
232         public boolean isLoaded() throws FtpConfigException {
233             return userManagerData != null;
234
235         }
236
237         /**
238          * {@inheritDoc}
239          */
240         public synchronized UserData getUserData(String username) throws FtpConfigException {
241             return userManagerData.getUserData(username);
242         }
243
244         private String getUserAndDateKey(String user) {
245             String userAndDate = dateFormat.format(new Date()) + " " + user;
246             return userAndDate;
247         }
248
249         private Map<String, Long> getUserResourceConsumptions(String userAndDate) {
250             Map<String, Long> userConsumption = resourceConsumption.get(userAndDate);
251             if (userConsumption == null) {
252                 userConsumption = Collections.synchronizedMap(new HashMap<String, Long>());
253                 resourceConsumption.put(userAndDate, userConsumption);
254             }
255             return userConsumption;
256         }
257
258         private String createQuotaMessage(String limitName, long consumption, long limit) {
259             return limitName + " exceed limit of " + limit + " (current consumption is " + consumption +
")";
260         }
261
262     }
```

Griffin.au.org.arcs.griffin.usermanager.model.package.html

```
1       <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
2       <html>
3       <head>
4       </head>
5       <body bgcolor="white">
6
```

```
7       Provides for the internal data model of the user manager configuration.
8
9       </body>
10      </html>
```

Griffin.au.org.arcs.griffin.usermanager.model.GroupData.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.usermanager.model;
26
27      import java.util.ArrayList;
28      import java.util.HashMap;
29      import java.util.List;
30      import java.util.Map;
31
32      import au.org.arcs.griffin.common.FtpConstants;
33      import au.org.arcs.griffin.exception.FtpConfigException;
34
35
36      /**
37       * Model data of a user group including the configured limits and path permissions. Note that
the
38       * order of the configured permission entries is important, since the first path match provides
the
39       * permission value.
40       *
41       * @author Lars Behnke
42       */
43      public class GroupData {
44
45          /**
46           * Control code for unlimited.
47           */
48          public static final long     UNLIMITED = -1;
49
50          private String               name;
51
```

```
52          private Map<String, Long>    limits;
53
54          private List<PermissionData> permissions;
55
56          /**
57           * Getter method for the java bean <code>limits</code>.
58           *
59           * @return Returns the value of the java bean <code>limits</code>.
60           */
61          public Map<String, Long> getLimits() {
62              if (limits == null) {
63                  limits = new HashMap<String, Long>();
64              }
65              return limits;
66          }
67
68          /**
69           * Convenience method for returning the limit specified by the passed name.
70           *
71           * @param name The limit name.
72           * @return The value.
73           */
74          public long getLimit(String name) {
75              Long limit = (Long) getLimits().get(name);
76              if (limit == null || limit.longValue() == UNLIMITED) {
77                  return Long.MAX_VALUE;
78              } else {
79                  return limit.longValue();
80              }
81          }
82
83          /**
84           * Getter method for the java bean <code>permissions</code>.
85           *
86           * @return Returns the value of the java bean <code>permissions</code>.
87           */
88          public List<PermissionData> getPermissions() {
89              if (permissions == null) {
90                  permissions = new ArrayList<PermissionData>();
91              }
92              return permissions;
93          }
94
95          /**
96           * Returns the group permission on the passed path.
97           *
98           * @param path The path to check.
99           * @param ftproot The FTP root folder.
100          * @param user The user's name.
101          * @return The permission.
102          * @throws FtpConfigException Error on reading or processing a configuration file.
103          */
104         public int getPermission(String path, String ftproot, String user) throws FtpConfigException {
105             for (PermissionData permission : getPermissions()) {
106                 if (permission.matches(path, ftproot, user)) {
107                     return permission.getPermission();
108                 }
109             }
110             return FtpConstants.PRIV_NONE;
111         }
```

```
112
113         /**
114          * Getter method for the java bean <code>name</code>.
115          *
116          * @return Returns the value of the java bean <code>name</code>.
117          */
118         public String getName() {
119             return name;
120         }
121
122         /**
123          * Setter method for the java bean <code>name</code>.
124          *
125          * @param name The value of name to set.
126          */
127         public void setName(String name) {
128             this.name = name;
129         }
130
131     }
```

Griffin.au.org.arcs.griffin.usermanager.model.GroupDataList.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.usermanager.model;
26
27      import java.util.ArrayList;
28      import java.util.Collections;
29      import java.util.List;
30
31      import au.org.arcs.griffin.common.FtpConstants;
32      import au.org.arcs.griffin.exception.FtpConfigException;
33
34
35      /**
36       * Data of a user group.
37       *
```

```
38       * @author Behnke
39       */
40      public class GroupDataList {
41
42          private List<GroupData> list = Collections.synchronizedList(new ArrayList<GroupData>());
43
44          /**
45           * Adds group data.
46           *
47           * @param data group data.
48           */
49          public void addGroup(GroupData data) {
50              list.add(data);
51          }
52
53          /**
54           * Clears all group data from the list.
55           */
56          public void clear() {
57              list.clear();
58          }
59
60          /**
61           * Returns the group of a given name.
62           *
63           * @param name The group name.
64           * @return The group or null.
65           */
66          public GroupData getGroup(String name) {
67              if (name == null) {
68                  return null;
69              }
70              for (GroupData data : list) {
71                  if (data.getName().equalsIgnoreCase(name)) {
72                      return data;
73                  }
74              }
75              return null;
76          }
77
78          /**
79           * The upper limit of the constraints named by the passed key.
80           *
81           * @param key The name of the constraint.
82           * @return The value.
83           */
84          public long getUpperLimit(String key) {
85              long limit = -1;
86              for (GroupData data : list) {
87                  long l = data.getLimit(key);
88                  if (l < 0) {
89                      return l;
90                  }
91                  limit = Math.max(l, limit);
92
93              }
94              return limit;
95          }
96
97          /**
```

```
98          * The permission on a given path.
99          *
100         * @param path The path to check.
101         * @param username The user that wants to access the path.
102         * @param ftproot The absolute ftp root directory.
103         * @return The permission constant.
104         * @throws FtpConfigException Error in configuration.
105         */
106        public int getPermission(String path, String username, String ftproot) throws
FtpConfigException {
107            int result = FtpConstants.PRIV_NONE;
108            for (GroupData groupData : list) {
109                int permission = groupData.getPermission(path, ftproot, username);
110                result = Math.max(result, permission);
111            }
112            return result;
113        }
114    }
```

Griffin.au.org.arcs.griffin.usermanager.model.MappingData.java

```
1       package au.org.arcs.griffin.usermanager.model;
2
3       public class MappingData {
4               private String dn;
5               private String uid;
6               public String getDn() {
7                       return dn;
8               }
9               public void setDn(String dn) {
10                      this.dn = dn;
11              }
12              public String getUid() {
13                      return uid;
14              }
15              public void setUid(String uid) {
16                      this.uid = uid;
17              }
18      }
```

Griffin.au.org.arcs.griffin.usermanager.model.PermissionData.java

```
1       /*
2        ------------------------------
3        Hermes FTP Server
4        Copyright (c) 2006 Lars Behnke
5        ------------------------------
6
7        This file is part of Hermes FTP Server.
8
9        Hermes FTP Server is free software; you can redistribute it and/or modify
10       it under the terms of the GNU General Public License as published by
11       the Free Software Foundation; either version 2 of the License, or
12       (at your option) any later version.
13
14       Foobar is distributed in the hope that it will be useful,
15       but WITHOUT ANY WARRANTY; without even the implied warranty of
16       MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       GNU General Public License for more details.
18
```

```
19          You should have received a copy of the GNU General Public License
20          along with Foobar; if not, write to the Free Software
21          Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
22          */
23
24          package au.org.arcs.griffin.usermanager.model;
25
26          import java.util.Properties;
27
28          import org.apache.commons.io.FilenameUtils;
29
30          import org.springframework.util.AntPathMatcher;
31
32          import au.org.arcs.griffin.exception.FtpConfigException;
33          import au.org.arcs.griffin.utils.VarMerger;
34
35          /**
36           * Represents the permission on one or more file pathes. The paths a configured using ant
style path
37           * naming conventions.
38           *
39           * @author Lars Behnke
40           */
41          public class PermissionData {
42
43              private String template;
44
45              private int permission;
46
47              /**
48               * Getter method for the java bean <code>permission</code>.
49               *
50               * @return Returns the value of the java bean <code>permission</code>.
51               */
52              public int getPermission() {
53                  return permission;
54              }
55
56              /**
57               * Setter method for the java bean <code>permission</code>.
58               *
59               * @param permission The value of permission to set.
60               */
61              public void setPermission(int permission) {
62                  this.permission = permission;
63              }
64
65              /**
66               * Getter method for the java bean <code>template</code>.
67               *
68               * @return Returns the value of the java bean <code>template</code>.
69               */
70              public String getTemplate() {
71                  return template;
72              }
73
74              /**
75               * Setter method for the java bean <code>template</code>.
76               *
77               * @param template The value of template to set.
```

```
78          */
79         public void setTemplate(String template) {
80            this.template = template;
81         }
82
83         /**
84          * Fills the placeholders in the path template and checks if the passed path matches the
85          * template.
86          *
87          * @param checkPath The path to check.
88          * @param ftproot The ftp root folder.
89          * @param username The username.
90          * @return True, if the path matches the configured pattern.
91          * @throws FtpConfigException Error on reading or processing a configuration file.
92          */
93         public boolean matches(String checkPath, String ftproot, String username)
94               throws FtpConfigException {
95            if (checkPath == null) {
96               return false;
97            }
98            AntPathMatcher pathMatcher = new AntPathMatcher();
99            String antPath = replacePlaceholders(ftproot, username);
100           antPath = FilenameUtils.normalizeNoEndSeparator(antPath);
101           antPath = FilenameUtils.separatorsToUnix(antPath);
102           checkPath = FilenameUtils.normalizeNoEndSeparator(checkPath);
103           checkPath = FilenameUtils.separatorsToUnix(checkPath);
104           return pathMatcher.match(antPath, checkPath);
105        }
106
107        private String replacePlaceholders(String ftproot, String username) throws
FtpConfigException {
108           VarMerger varMerger = new VarMerger(getTemplate());
109           Properties props = new Properties();
110           props.setProperty("ftproot", FilenameUtils.separatorsToUnix(ftproot));
111           props.setProperty("user", username);
112           varMerger.merge(props);
113           if (!varMerger.isReplacementComplete()) {
114              throw new FtpConfigException("Unresolved placeholders in user configuration file
found.");
115           }
116           return varMerger.getText();
117        }
118     }
```

Griffin.au.org.arcs.griffin.usermanager.model.UserData.java

```
1        /*
2         * ---------------------------------------------------------------------------
3         * Hermes FTP Server
4         * Copyright (c) 2005-2007 Lars Behnke
5         * ---------------------------------------------------------------------------
6         *
7         * This file is part of Hermes FTP Server.
8         *
9         * Hermes FTP Server is free software; you can redistribute it and/or modify
10        * it under the terms of the GNU General Public License as published by
11        * the Free Software Foundation; either version 2 of the License, or
12        * (at your option) any later version.
13        *
14        * Hermes FTP Server is distributed in the hope that it will be useful,
```

```
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ---------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.usermanager.model;
26
27      import java.util.ArrayList;
28      import java.util.List;
29
30      /**
31       * Represents the configuration for a particular user; Constructor.
32       */
33      public class UserData {
34
35          private String       uid;
36
37          private String       fullName;
38
39          private String       password;
40
41          private List<String> groupNames;
42
43          private String       dir;
44
45          private boolean      adminRole;
46
47          /**
48           * Getter method for the java bean <code>uid</code>.
49           *
50           * @return Returns the value of the java bean <code>uid</code>.
51           */
52          public String getUid() {
53              return uid;
54          }
55
56          /**
57           * Setter method for the java bean <code>uid</code>.
58           *
59           * @param uid The value of uid to set.
60           */
61          public void setUid(String uid) {
62              this.uid = uid;
63          }
64
65          /**
66           * Getter method for the java bean <code>password</code>.
67           *
68           * @return Returns the value of the java bean <code>password</code>.
69           */
70          public String getPassword() {
71              return password;
72          }
73
74          /**
```

```
75          * Setter method for the java bean <code>password</code>.
76          *
77          * @param password The value of password to set.
78          */
79         public void setPassword(String password) {
80             this.password = password;
81         }
82
83         /**
84          * Getter method for the java bean <code>fullName</code>.
85          *
86          * @return Returns the value of the java bean <code>fullName</code>.
87          */
88         public String getFullName() {
89             return fullName;
90         }
91
92         /**
93          * Setter method for the java bean <code>fullName</code>.
94          *
95          * @param fullName The value of fullName to set.
96          */
97         public void setFullName(String fullName) {
98             this.fullName = fullName;
99         }
100
101        /**
102         * Getter method for the java bean <code>groupNames</code>.
103         *
104         * @return Returns the value of the java bean <code>groupNames</code>.
105         */
106        public List<String> getGroupNames() {
107            if (groupNames == null) {
108                groupNames = new ArrayList<String>();
109            }
110            return groupNames;
111        }
112
113        /**
114         * Adds the name of a group the user belongs to.
115         *
116         * @param name The group name.
117         */
118        public void addGroupName(String name) {
119            getGroupNames().add(name);
120        }
121
122        /**
123         * Setter method for the java bean <code>groupNames</code>.
124         *
125         * @param groupNames The value of groupNames to set.
126         */
127        public void setGroupNames(List<String> groupNames) {
128            this.groupNames = groupNames;
129        }
130
131        /**
132         * Getter method for the java bean <code>dir</code>.
133         *
134         * @return Returns the value of the java bean <code>dir</code>.
```

```
135          */
136         public String getDir() {
137             return dir;
138         }
139 
140         /**
141          * Setter method for the java bean <code>dir</code>.
142          *
143          * @param dir The value of dir to set.
144          */
145         public void setDir(String dir) {
146             this.dir = dir;
147         }
148 
149         /**
150          * Getter method for the java bean <code>adminRole</code>.
151          *
152          * @return Returns the value of the java bean <code>adminRole</code>.
153          */
154         public boolean isAdminRole() {
155             return adminRole;
156         }
157 
158         /**
159          * Setter method for the java bean <code>adminRole</code>.
160          *
161          * @param adminRole The value of adminRole to set.
162          */
163         public void setAdminRole(boolean adminRole) {
164             this.adminRole = adminRole;
165         }
166 
167     }
```

Griffin.au.org.arcs.griffin.usermanager.model.UserManagerData.java

```
1       /*
2        * ---------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ---------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ---------------------------------------------------------------------------
23       */
24 
```

```
25  package au.org.arcs.griffin.usermanager.model;
26
27  import java.util.ArrayList;
28  import java.util.Collections;
29  import java.util.List;
30
31  /**
32   * Data Model for the user manager configuration.
33   *
34   * @author Lars Behnke
35   */
36  public class UserManagerData {
37
38      private GroupDataList  groupDataList;
39
40      private List<UserData> userData;
41
42      private List<MappingData> mappingData;
43
44      /**
45       * Getter method for the java bean <code>groupData</code>.
46       *
47       * @return Returns the value of the java bean <code>groupData</code>.
48       */
49      public GroupDataList getGroupData() {
50          if (groupDataList == null) {
51              groupDataList = new GroupDataList();
52          }
53          return groupDataList;
54      }
55
56      /**
57       * Getter method for the java bean <code>userData</code>.
58       *
59       * @return Returns the value of the java bean <code>userData</code>.
60       */
61      public List<UserData> getUserData() {
62          if (userData == null) {
63              userData = Collections.synchronizedList(new ArrayList<UserData>());
64          }
65          return userData;
66      }
67
68
69      public List<MappingData> getMappingData() {
70          if (mappingData == null) {
71                  mappingData = Collections.synchronizedList(new ArrayList<MappingData>());
72          }
73          return mappingData;
74      }
75
76      /**
77       * Returns the user object by the username.
78       *
79       * @param username The user's name.
80       * @return The user object.
81       */
82      public UserData getUserData(String username) {
83          username = username.trim();
84          for (UserData user : getUserData()) {
```

```
85                  if (user.getUid().equals(username)) {
86                      return user;
87                  }
88              }
89              return null;
90          }
91
92          /**
93           * Returns the group object by the group name.
94           *
95           * @param groupname The group's name.
96           * @return The group object.
97           */
98          public GroupData getGroupData(String groupname) {
99              return groupDataList.getGroup(groupname.trim());
100         }
101
102         public MappingData getMappingData(String dn){
103             dn=dn.trim();
104             for (MappingData mapping : getMappingData()){
105                 if (mapping.getDn().equals(dn)){
106                     return mapping;
107                 }
108             }
109             return null;
110         }
111
112     }
```

Griffin.au.org.arcs.griffin.server.package.html

```
1       <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
2       <html>
3       <head>
4       </head>
5       <body bgcolor="white">
6
7       Contains interfaces and abstract base classes related to FTP server functionallity.
8
9       </body>
10      </html>
```

Griffin.au.org.arcs.griffin.server.AbstractFtpServer.java

```
1       /*
2        * ------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
17        * GNU General Public License for more details.
18        *
19        * You should have received a copy of the GNU General Public License
20        * along with Hermes FTP Server; if not, write to the Free Software
21        * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22        * -----------------------------------------------------------------------------
23        */
24
25       package au.org.arcs.griffin.server;
26
27       import java.io.File;
28       import java.io.IOException;
29       import java.net.InetAddress;
30       import java.net.ServerSocket;
31       import java.net.Socket;
32       import java.net.SocketTimeoutException;
33       import java.util.ArrayList;
34       import java.util.Date;
35       import java.util.List;
36
37
38       import org.apache.commons.logging.Log;
39       import org.apache.commons.logging.LogFactory;
40
41       import au.org.arcs.griffin.common.FtpConstants;
42       import au.org.arcs.griffin.common.FtpEventListener;
43       import au.org.arcs.griffin.common.FtpServerOptions;
44       import au.org.arcs.griffin.common.FtpSessionContext;
45       import au.org.arcs.griffin.filesystem.FileSystem;
46       import au.org.arcs.griffin.session.FtpSession;
47       import au.org.arcs.griffin.usermanager.UserManager;
48       import au.org.arcs.griffin.utils.AbstractAppAwareBean;
49       import au.org.arcs.griffin.utils.IOUtils;
50       import au.org.arcs.griffin.utils.NetUtils;
51
52       /**
53        * Ancestor class for FTP server implementations.
54        *
55        * @author Lars Behnke
56        */
57       public abstract class AbstractFtpServer extends AbstractAppAwareBean implements
FtpServer, FtpConstants,
58             FtpEventListener {
59
60          private static final int       DEFAULT_TIMEOUT          = 3000;
61
62          private static Log             log                      = LogFactory.getLog(FtpServer.class);
63
64          private List<FtpSession>       sessions                 = new ArrayList<FtpSession>();
65
66          private boolean                terminated;
67
68          private FtpServerOptions       options;
69
70          private String                 resources;
71
72          private String                 name;
73
74          private int                    status                   = SERVER_STATUS_UNDEF;
75
```

```
76          private List<FtpEventListener> ftpEventListeners        = new ArrayList<FtpEventListener>();
77
78          private int                 connectionCountHWMark;
79
80          private Date                connectionCountHWMarkDate = new Date();
81
82          private FileSystem                  fileSystem;
83
84          public FileSystem getFileSystem() {
85                  return fileSystem;
86              }
87
88              public void setFileSystem(FileSystem fileSystem) {
89                  this.fileSystem = fileSystem;
90              }
91
92              /**
93           * Creates a server socket. Depending on the server implementation this can be a SSL or a
94           * regular server socket.
95           *
96           * @return The server socket.
97           * @throws IOException Error on creating server socket.
98           */
99          protected abstract ServerSocket createServerSocket() throws IOException;
100
101         /**
102          * Creates the context object passed to the user session.
103          *
104          * @return The session context.
105          */
106         protected abstract FtpSessionContext createFtpContext();
107
108         /**
109          * Halts the server.
110          */
111         public void abort() {
112             this.terminated = true;
113         }
114
115         /**
116          * Getter method for the java bean <code>options</code>.
117          *
118          * @return Returns the value of the java bean <code>options</code>.
119          */
120         public FtpServerOptions getOptions() {
121             return options;
122         }
123
124         /**
125          * Setter method for the java bean <code>options</code>.
126          *
127          * @param options The value of options to set.
128          */
129         public void setOptions(FtpServerOptions options) {
130             this.options = options;
131         }
132
133         /**
134          * {@inheritDoc}
135          */
```

```
136         public void run() {
137             setStatus(SERVER_STATUS_INIT);
138             ServerSocket serverSocket = null;
139             try {
140                     getFileSystem().init();
141                 serverSocket = createServerSocket();
142                 serverSocket.setSoTimeout(DEFAULT_TIMEOUT);
143                 setStatus(SERVER_STATUS_READY);
144                 while (!isTerminated()) {
145                     Socket clientSocket;
146                     try {
147                         clientSocket = serverSocket.accept();
148                     } catch (SocketTimeoutException e) {
149                         continue;
150                     }
151 
152                     /* Check blacklisted IP v4 addresses */
153 
154                     InetAddress clientAddr = clientSocket.getInetAddress();
155                     InetAddress localAddr = clientSocket.getLocalAddress();
156 
157                     log.info("Client requests connection. ClientAddr.: " +  clientAddr + ", LocalAddr.: " +
localAddr);
158 
159                     String listKey = NetUtils.isIPv6(clientAddr) ?
FtpConstants.OPT_IPV6_BLACK_LIST : FtpConstants.OPT_IPV4_BLACK_LIST;
160                     String ipBlackList = getOptions().getString(listKey, "");
161 
162                     if (NetUtils.checkIPMatch(ipBlackList, clientAddr)) {
163                         log.info("Client with IP address " + clientAddr.getHostAddress() + " rejected
(blacklisted).");
164                         IOUtils.closeGracefully(clientSocket);
165                         continue;
166                     }
167 
168                     /* Initialize session context */
169                     FtpSessionContext ctx = createFtpContext();
170                     ctx.setCreationTime(new Date());
171                     ctx.setClientSocket(clientSocket);
172                     // set default reply type to be clear
173                     ctx.setReplyType("clear");
174                     FtpSession session = (FtpSession)
getApplicationContext().getBean(BEAN_SESSION);
175                     session.setFtpContext(ctx);
176 
177                     /* Start session */
178                     log.debug("Accepting connection to " + clientAddr.getHostAddress());
179                     session.start();
180                     registerSession(session);
181 
182                 }
183                 setStatus(SERVER_STATUS_HALTED);
184             } catch (IOException e) {
185                 setStatus(SERVER_STATUS_UNDEF);
186                 log.error(e, e);
187             } finally {
188                 terminateAllClientSessions();
189                 IOUtils.closeGracefully(serverSocket);
190             }
191             getFileSystem().exit();
```

```
192         }
193
194         private void registerSession(FtpSession session) {
195             synchronized (this) {
196                 sessions.add(session);
197                 int sessionCount = getConnectionCount();
198                 if (sessionCount >= connectionCountHWMark) {
199                     connectionCountHWMark = sessionCount;
200                     connectionCountHWMarkDate = new Date();
201                 }
202             }
203         }
204
205         /**
206          * Getter method for the java bean <code>connectionCount</code>.
207          *
208          * @return Returns the value of the java bean <code>connectionCount</code>.
209          */
210         public int getConnectionCount() {
211             synchronized (this) {
212                 cleanUpSessions();
213                 return sessions.size();
214             }
215         }
216
217         /**
218          * {@inheritDoc}
219          */
220         public void cleanUpSessions() {
221             List<FtpSession> newList = new ArrayList<FtpSession>();
222             for (FtpSession session : sessions) {
223                 if (!session.isTerminated()) {
224                     newList.add(session);
225                 }
226             }
227             sessions = newList;
228         }
229
230         private void terminateAllClientSessions() {
231             for (FtpSession session : sessions) {
232                 session.abort();
233             }
234         }
235
236         /**
237          * Convenience method for accessing the application properties.
238          *
239          * @param name The name of the requested property.
240          * @return The property.
241          */
242         public String getOption(String name) {
243             return getOptions().getProperty(name);
244         }
245
246         /**
247          * Getter method for the java bean <code>resources</code>.
248          *
249          * @return Returns the value of the java bean <code>resources</code>.
250          */
251         public String getResources() {
```

```
252             return resources;
253         }
254
255         /**
256          * Setter method for the java bean <code>resources</code>.
257          *
258          * @param resources The value of resources to set.
259          */
260         public void setResources(String resources) {
261             this.resources = resources;
262         }
263
264         /**
265          * Getter method for the java bean <code>terminated</code>.
266          *
267          * @return Returns the value of the java bean <code>terminated</code>.
268          */
269         public boolean isTerminated() {
270             return terminated;
271         }
272
273         /**
274          * Getter method for the java bean <code>status</code>.
275          *
276          * @return Returns the value of the java bean <code>status</code>.
277          */
278         public int getStatus() {
279             return status;
280         }
281
282         /**
283          * Setter method for the java bean <code>status</code>.
284          *
285          * @param status The value of status to set.
286          */
287         public void setStatus(int status) {
288             this.status = status;
289         }
290
291         /**
292          * {@inheritDoc}
293          */
294         public void addFtpEventListener(FtpEventListener lstnr) {
295             this.ftpEventListeners.add(lstnr);
296         }
297
298         /**
299          * {@inheritDoc}
300          */
301         public void downloadPerformed(String clientId, File file) {
302             for (FtpEventListener listener : ftpEventListeners) {
303                 listener.downloadPerformed(clientId, file);
304             }
305             log.debug("Download event delegated to listeners.");
306         }
307
308         /**
309          * {@inheritDoc}
310          */
311         public void loginPerformed(String clientId, boolean successful) {
```

```
312             for (FtpEventListener listener : ftpEventListeners) {
313                 listener.loginPerformed(clientId, successful);
314             }
315             log.debug("Login event delegated to listeners.");
316         }
317 
318         /**
319          * {@inheritDoc}
320          */
321         public void uploadPerformed(String clientId, File file) {
322             for (FtpEventListener listener : ftpEventListeners) {
323                 listener.uploadPerformed(clientId, file);
324             }
325             log.debug("Upload event delegated to listeners.");
326         }
327 
328         /**
329          * {@inheritDoc}
330          */
331         public void sessionOpened(Object sessionObj) {
332             for (FtpEventListener listener : ftpEventListeners) {
333                 listener.sessionOpened(sessionObj);
334             }
335             log.debug("Session opened event delegated to listeners.");
336         }
337 
338         /**
339          * {@inheritDoc}
340          */
341         public void sessionClosed(Object sessionObj) {
342             synchronized (this) {
343                 sessions.remove(sessionObj);
344             }
345             for (FtpEventListener listener : ftpEventListeners) {
346                 listener.sessionOpened(sessionObj);
347             }
348             log.debug("Session closed event delegated to listeners.");
349         }
350 
351         /**
352          * {@inheritDoc}
353          */
354         public List<FtpSession> getSessions() {
355             return sessions;
356         }
357 
358         /**
359          * {@inheritDoc}
360          */
361         public int getConnectionCountHWMark() {
362             return connectionCountHWMark;
363         }
364 
365         /**
366          * {@inheritDoc}
367          */
368         public Date getConnectionCountHWMarkDate() {
369             return connectionCountHWMarkDate;
370         }
371 
```

```
372         /**
373          * Setter methode for property <code>connectionCountHWMark</code>.
374          *
375          * @param connectionCountHWMark Value for <code>connectionCountHWMark</code>.
376          */
377         public void setConnectionCountHWMark(int connectionCountHWMark) {
378             this.connectionCountHWMark = connectionCountHWMark;
379         }
380
381         /**
382          * Setter methode for property <code>connectionCountHWMarkDate</code>.
383          *
384          * @param connectionCountHWMarkDate Value for
<code>connectionCountHWMarkDate</code>.
385          */
386         public void setConnectionCountHWMarkDate(Date connectionCountHWMarkDate) {
387             this.connectionCountHWMarkDate = connectionCountHWMarkDate;
388         }
389
390
391         public String getName() {
392             return name;
393         }
394
395
396         public void setName(String name) {
397             this.name = name;
398         }
399
400     }
```

Griffin.au.org.arcs.griffin.server.FtpServer.java

```
1       /*
2        * ----------------------------------------------------------------------------
3        * Hermes FTP Server
4        * Copyright (c) 2005-2007 Lars Behnke
5        * ----------------------------------------------------------------------------
6        *
7        * This file is part of Hermes FTP Server.
8        *
9        * Hermes FTP Server is free software; you can redistribute it and/or modify
10       * it under the terms of the GNU General Public License as published by
11       * the Free Software Foundation; either version 2 of the License, or
12       * (at your option) any later version.
13       *
14       * Hermes FTP Server is distributed in the hope that it will be useful,
15       * but WITHOUT ANY WARRANTY; without even the implied warranty of
16       * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
17       * GNU General Public License for more details.
18       *
19       * You should have received a copy of the GNU General Public License
20       * along with Hermes FTP Server; if not, write to the Free Software
21       * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
22       * ----------------------------------------------------------------------------
23       */
24
25      package au.org.arcs.griffin.server;
26
27      import java.util.Date;
```

```
28  import java.util.List;
29
30  import au.org.arcs.griffin.common.FtpEventListener;
31  import au.org.arcs.griffin.common.FtpServerOptions;
32  import au.org.arcs.griffin.filesystem.FileSystem;
33  import au.org.arcs.griffin.session.FtpSession;
34  import au.org.arcs.griffin.usermanager.UserManager;
35
36
37  /**
38   * Contract for FTP server implementations.
39   *
40   * @author Lars Behnke
41   */
42  public interface FtpServer extends Runnable {
43
44      /**
45       * Returns the name of the FTP server.
46       *
47       * @return The name.
48       */
49      String getName();
50
51      /**
52       * Returns the number of FTP connections currently active.
53       *
54       * @return The number of connections.
55       */
56      int getConnectionCount();
57
58      /**
59       * Maximum number of connections since server started.
60       *
61       * @return The high water mark.
62       */
63      int getConnectionCountHWMark();
64
65      /**
66       * Date when number of connections reached its maximum.
67       *
68       * @return The high water mark date.
69       */
70      Date getConnectionCountHWMarkDate();
71
72      /**
73       * Halts the server.
74       */
75      void abort();
76
77      /**
78       * Returns the server status.
79       *
80       * @return The status code (0 = undefined, 1 = initializing, 2 = ready).
81       */
82      int getStatus();
83
84      /**
85       * Returns the list of active sessions.
86       *
87       * @return The list.
```

```
88          */
89         List<FtpSession> getSessions();
90
91         /**
92          * Removes closed sessions from memory.
93          */
94         void cleanUpSessions();
95
96         /**
97          * Returns the application properties.
98          *
99          * @return The properties.
100         */
101        FtpServerOptions getOptions();
102
103        /**
104         * Adds an external listerner that wants to get informed about FTP events.
105         *
106         * @param lstnr The listener.
107         */
108        void addFtpEventListener(FtpEventListener lstnr);
109
110        FileSystem getFileSystem();
111
112    }
```