UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Matheus Anversa Viera

Uma Abordagem para Reserva Antecipada de Recursos em Ambientes de Grades Computacionais Móveis

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Mario Antônio Ribeiro Dantas, Dr., UFSC
(Orientador)

Florianópolis, Agosto de 2011

Uma Abordagem para Reserva Antecipada de Recursos em Ambientes de Grades Computacionais Móveis

Matheus Anversa Viera

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Computação Paralela e Distribuída e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Mario Antônio Ribeiro Dantas, Dr. (Coordenador)

Banca Examinadora

Prof. Mario Antônio Ribeiro Dantas, Dr., UFSC
(Orientador)

Profa. Alba Cristina Magalhães Alves de Melo, Dra., UNB

Prof. Carlos Barros Montez, Dr., UFSC

Prof. João Bosco Mangueira Sobral, Dr., UFSC

"Bom mesmo é ir à luta com determinação e abraçar a vida com paixão, perder com classe e vencer com ousadia, pois o triunfo pertence a quem mais se atreve e a vida é muito para ser insignificante."

Charles Chaplin

Agradecimentos

Gostaria de agradecer, primeiramente, ao meu orientador e grande amigo Mario Dantas por ter me confiado diversas responsabilidades e projetos, contribuindo com o seu conhecimento acadêmico e transmitindo valores que levarei para o resto da vida. E ainda, por ter auxiliado no processo que me oportunizou uma viagem para o Canadá, por um período de 5 meses, em um intercâmbio acadêmico na UWO (*University of Western Ontario*) - instituição a que também sou grato por ter me acolhido.

Agradeço ao professor Michael A. Bauer, meu orientador durante o intercâmbio, por contribuir com ideias para minha pesquisa e por disponibilizar a SHARCNET (*Shared Hierarchical Academic Research Computing Network*) para a realização de alguns experimentos. Agradeço também aos grandes amigos que fiz, Lucio e Elisa, por terem me acolhido em sua casa durante este intercâmbio.

Também agradeço aos membros do LaPeSD, em especial ao professor Frank Siqueira e, novamente, ao professor Mario Dantas, por permitirem minha participação em um dos projetos do laboratório, o qual influenciou diretamente neste trabalho de dissertação. Ao Rodrigo Grumiche pela explicação e disponibilidade de seu trabalho e ao Igor Tromel pelo atual desenvolvimento colaborativo de uma extensão do trabalho.

Um agradecimento especial vai para a minha amiga e namorada Bruna, que esteve ao meu lado em todos os momentos desta jornada. Agradeço também a todos os meus amigos, pelas diferentes formas de contribuição ao longo do meu mestrado (churrascos, conversas, divisão de apartamento, etc), em especial ao Cristiano e ao professor Caio da UFSM, que trouxeram grandes contribuições para o meu trabalho.

Por fim, gostaria de agradecer imensamente a minha família: ao meu pai Zélio, minha mãe Linamares e minhas irmãs Maríndia e Marjara. Foram essas as pessoas que sempre me incentivaram e me apoiaram. Dedico essa dissertação e conquista especialmente a minha mãe que sempre me apoiou; e por ter me ensinado, novamente, através de sua luta e vitória contra um problema de saúde, que nunca devemos desistir, independentemente dos desafios que nos são impostos. Muito obrigado!

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

2PC	*Two-phase Commit*
3G	*3rd Generation Mobile Telecommunications*
API	*Application Programing Interface*
AR	*Advanced Reservation*
ASP	*Application Service Provisioning*
BoT	*Bag-of-Tasks*
CPU	*Central Processing Unit*
CSF	*Community Scheduling Framework*
DRM	*Distributed Resource Management*
FAFNER	*Factoring via Network-Enabled Recursion*
FIFO	*First-In First-Out*
GIS	*Grid Information Service*
GPS	*Global Positioning System*
GRAM	*Grid Resource Allocation and Management*
GridARM	*Askalon's Grid Resource Management System*
GridARS	*Grid Advance Reservation-based System Framework*
GRS	*Global Resource Scheduler*
GSI	*Grid Security Infrastructure*
GT	*Globus Toolkit*

GUI	*Graphical User Interface*
HTTP	*HyperText Transfer Protocol*
I-WAY	*The Information Wide Area Year*
LAN	*Local Area Network*
LRM	*Local Resource Manager*
MPI	*Message Passing Interface*
NOW	*Network of Workstations*
OGE	*Oracle Grid Engine*
OGSA	*Open Grid Services Architecture*
OGSI	*Open Grid Services Infrastructure*
PDA	*Personal Digital Assistant*
PDP	*Policy Decision Point*
PVM	*Parallel Virtual Machine*
QoE	*Quality of Experience*
QoS	*Quality of Service*
RMS	*Resource Management Systems*
SGE	*Sun Grid Engine*
SuMMIT	*Submission, Monitoring and Management of Interactions of Tasks*
UML	*Unified Modeling Language*
VO	*Virtual Organization*
WAN	*Wide Area Network*
WSRF	*Web Services Resource Framework*

Resumo

As grades computacionais são amplamente utilizadas para a resolução de problemas que demandam um grande poder computacional. Os dispositivos móveis, por apresentarem cada vez mais recursos e maior capacidade de processamento, têm sido utilizados em ambientes de grades. Com o crescente acesso a tais ambientes, a qualquer momento e localização, deseja-se que o usuário tenha o mínimo conhecimento sobre o ambiente, preocupando-se apenas com as características de sua aplicação.

Nesses ambientes, em caso de indisponibilidade de recursos que atendam às requisições, as aplicações são colocadas em filas, não ocorrendo garantias de execução. A reserva antecipada de recursos, nesse contexto, é um mecanismo importante, pois permite um melhor planejamento de uso da grade, garantindo um maior aproveitamento na utilização dos recursos. Através deste mecanismo, um usuário pode requisitar um futuro uso de recursos, a fim de garantir maiores níveis de QoS e de QoE.

Nesta dissertação é apresentada uma arquitetura para reserva antecipada de recursos, considerando as características da aplicação como fator determinante para a reserva. Especificamente, a abordagem proposta visa melhorar a qualidade das reservas, procurando o melhor nível de adequação dos recursos com base nas particularidades requisitadas pelo usuário. Assim, as reservas realizadas, além de garantirem uma QoS, buscam melhorar o desempenho durante a execução das tarefas. A arquitetura proposta ainda apresenta uma interface móvel de acesso para os usuários interagirem através de dispositivos móveis.

Nos experimentos realizados, a arquitetura, quando comparada com outras abordagens, mostrou ser eficiente. Realizou uma boa distribuição das reservas, alcançando uma maior eficiência computacional e garantindo um bom desempenho das aplicações executadas nos recursos previamente reservados.

Palavras-chave: grades computacionais, reserva antecipada de recursos, computação móvel e pervasiva

Abstract

Grid computing is widely used to solve problems that require high computing power. Mobile devices have been used in grid environments due to their increasing number of resources and growing processing power. The increasing accesses to these environments at anytime and anywhere requires the least knowledge from the user who only has to worry about the characteristics of his application in the grid environment.

In such environments, in case of unavailability of resources to meet the requests, applications are placed in queues without executions guarantees. In this context, advanced reservation is an important mechanism that enables better planning use of the grid by ensuring a better use of its resources. Through this mechanism, a user can request a future use of resources in order to ensure higher levels of QoS and QoE.

This dissertation presents an architecture for advanced resource reservation that considers the application characteristics as the major factor for the reservations. Specifically, the proposed approach aims to improve the reservation quality, seeking the highest levels of adequacy of resources based on specifics user requirements. Besides ensuring QoS, the performed reservations also aims to improve performance during a job execution. The proposed architecture also presents a mobile access interface for users to interact with the grid through mobile devices.

In the experiments, the architecture has shown to be efficient when compared with other approaches. Our approach performed a good distribution of reserves, achieving greater computational efficiency by ensuring a good performance of applications running on the resources reserved in advance.

Keywords: grid computing, advanced resource reservation, mobile and pervasive computing

Capítulo 1

Introdução

Grades computacionais são caracterizadas por ambientes geograficamente distribuídos, paralelos e colaborativos, com uma variedade de recursos (incluindo serviços, dispositivos e aplicações) sendo utilizados para solucionar problemas em organizações virtuais dinâmicas de diferentes instituições. Esses ambientes descrevem tecnologias que permitem que consumidores obtenham poder computacional sobre demanda [Foster et al., 2008]. Ainda, os recursos computacionais, que podem ser entendidos como processadores, discos, memórias, redes, largura de banda, *clusters* e outras facilidades, são disponibilizados aos usuários de forma transparente, através de um ambiente de rede [Bote-Lorenzo et al., 2004].

Nesse contexto, diferentes organizações virtuais (grupos de indivíduos provedores e/ou consumidores de recursos) possuem diferentes tipos de recursos disponíveis e diferentes políticas de utilização dos mesmos. Assim, como resultado de sua natureza, as grades provêm uma variedade de serviços que os usuários podem incorporar e combinar para alcançar suas tarefas e objetivos computacionais. Especificamente, os usuários podem acessar os recursos, aplicações e serviços, submeter *jobs* para execução, através de filas ou por reserva antecipada de recursos, criar combinações de processos em *workflows*, e verificar o *status* dos *jobs* ou do próprio sistema.

Nos últimos anos, percebe-se um movimento na direção de integração de ambientes de grades computacionais com ambientes de computação móvel ([Rossetto et al., 2007], [Chu and Humphrey, 2004] e [Gomes et al., 2007]). Consequentemente, os dispositivos móveis, dentro deste contexto, são considerados como interfaces de acesso a serviços da grade, ou como recursos desta. Apesar do poder computacional dos dispositivos móveis ter apresentado uma melhora significativa nos últimos anos, a atual capacidade de processamento e armazenamento, além da autonomia de bateria e conectividade, são fatores limitantes destes aparelhos. Desta forma, estes dispositivos ainda não são suficientes para a resolução de problemas complexos. Por isso, em diversos trabalhos, considera-se o uso de

dispositivos móveis como interfaces de acesso aos recursos e serviços de uma grade a partir de qualquer lugar e a qualquer momento.

Sendo os recursos distribuídos acessíveis por diferentes usuários, as grades computacionais devem gerenciá-los da melhor forma possível, buscando não prejudicar a experiência do usuário. Ainda, grades com configurações de *multi-clusters*, quando bem orquestradas, disponibilizam aos usuários um grande poder computacional para execução de aplicações paralelas. O escalonamento de tarefas, ou meta-escalonamento, para o caso de uma grade com diversas organizações virtuais e, consequentemente, diversos gerenciadores de recursos, torna-se um desafio encontrado nestes ambientes, dada a necessidade de selecionar de forma eficiente os recursos mais apropriados para executar determinada tarefa [Venugopal et al., 2004].

Por outro lado, diversos desses ambientes com meta-escalonadores, tendo uma carga de trabalho atual ocupando praticamente todos os recursos, quando utilizados, colocam as tarefas dos *jobs* nas filas dos gerenciadores locais por tempo indeterminado, conforme os mesmos vão chegando na grade para serem executados. *Jobs* podem ser definidos como um conjunto de tarefas de uma aplicação. Dependendo das aplicações, como por exemplo as de tempo crítico, podem resultar em problemas de qualidade de serviço (QoS) e de garantias de execução, se forem enfileiradas para execução posterior. Uma solução é o ambiente possuir reserva antecipada de recursos. Com isso, os usuários podem planejar o futuro uso dos recursos para a execução de suas aplicações, garantindo um conjunto mínimo de recursos reservados em um dado tempo. Com esse mecanismo evita-se que *jobs* fiquem em filas de espera por um dado recurso, pois eles são previamente reservados e alocados, melhorando as garantias de execução da aplicação e, por conseguinte, melhorando a QoS [Min and Maheswaran, 2001]. Alguns meta-escalonadores possuem suporte à reserva antecipada de recursos, como por exemplo, o CSF [Xiaohui et al., 2006], o GridARS [Takefusa et al., 2007] e o Viola [Eickermann et al., 2007].

As organizações virtuais pertencentes ao ambiente da grade, em adição, possuem diferentes políticas e formas de descrição dos seus recursos, dificultando o escalonamento e a reserva dos mesmos. Nesse sentido, o uso de ontologias tem sido considerado nesses ambientes, a fim de melhorar a qualidade de informação dos recursos heterogêneos pertencentes a diferentes organizações. Portanto, a utilização do ambiente é dada por uma requisição semântica, a nível de hardware e software, que satisfaça o usuário para executar sua aplicação. Consequentemente, isso acarreta um conhecimento sobre as características dos recursos computacionais por

parte do usuário.

O presente trabalho de pesquisa visa estudar os ambientes de grades computacionais a fim de propor uma abordagem de arquitetura para reserva antecipada de recursos baseada nas características da aplicação e utilizando como interface de acesso ao ambiente, os dispositivos móveis. Desta forma, o meta-escalonador proposto é encarregado de decidir o recurso a ser reservado com base nas características da aplicação fornecidas pelo usuário, fazendo com que este não necessite ter um conhecimento aprofundado sobre a grade, melhorando, consequentemente, sua qualidade de experiência (QoE). Esta dissertação objetiva, especificamente, estudar e comparar alguns ambientes de grades com suporte à reserva, considerando a forma que os recursos são alocados para as aplicações. Com isso, visa-se aperfeiçoar o aproveitamento do ambiente através de um melhor planejamento da reserva, considerando características estruturais e comportamentais da aplicação. Em adição, constatou-se uma falta de abordagens que realizam reserva antecipada de recursos através de dispositivos móveis e, portanto, buscou-se, através de uma interface móvel, uma interação com o ambiente (a partir de qualquer lugar e a qualquer momento) para a realização de reservas e interação com as mesmas. Por fim, propõe-se uma integração com outros trabalhos previamente realizados pelo grupo de pesquisa do laboratório LaPeSD/UFSC que utilizam, por exemplo, ontologias e lógica *fuzzy*.

Com a finalidade de verificar o comportamento da abordagem proposta, buscou-se conferir a qualidade e distribuição de reservas em um ambiente de grade com configuração de *multi-clusters*, considerando as particularidades da aplicação e avaliando o melhor nível de adequação, utilizando lógica *fuzzy*, de um recurso a ser reservado frente às necessidades requisitadas. Ainda, analisou-se a eficiência computacional dos recursos reservados e o quanto isso influencia no desempenho das aplicações.

A dissertação está subdividida em seis capítulos. No capítulo 2 é realizado um estudo geral sobre ambientes de grades computacionais, demonstrando uma comparação desses ambiente com outros ambientes de distribuídos. Ainda, mostra-se a evolução das grades e sua arquitetura, sendo apresentados alguns *middlewares* para esse tipo de ambiente. O tópico relacionado à reserva de recursos é abordado no capítulo 3, onde são levantados conceitos fundamentais sobre reservas e ambientes com meta-escalonadores. Também apresentam-se nesse capítulo os trabalhos correlatos da área. No capítulo 4 é apresentada a proposta da arquitetura de reserva de recursos através de dispositivos móveis, que considera as características da aplicação como requisito. No capítulo 5 são analisa-

dos resultados experimentais dos testes da arquitetura proposta, focando a qualidade das reservas, eficiência e mobilidade. Por fim, o capítulo 6 traz as considerações e conclusões obtidas com o trabalho, mostrando as indicações de trabalhos futuros a respeito desta dissertação.

Capítulo 2

Grades Computacionais

Pesquisas em diferentes áreas, atualmente, possuem uma grande necessidade de poder computacional. Isso se deve ao fato de muitas simulações ou aplicações, que tem como objetivo a resolução de problemas complexos, necessitarem de um grande número de recursos para que sejam executados. Dada a complexidade de um problema, a solução pode ser inviável fazendo-se uso de um único computador. Com isso, o processamento distribuído, em que vários computadores estão interligados através de uma rede e atuando como provedores de recursos, torna-se necessário para obter as respostas das resoluções desses problemas. Nesse contexto, a computação em grade demonstra-se uma alternativa promissora, pois se utiliza, em muitos casos, do poder computacional de diferentes *clusters* conectados às grades, ou de ciclos ociosos de computadores pessoais.

A computação em grade pode ser caracterizada por uma variedade de recursos distribuídos geograficamente, incluindo serviços, dispositivos e aplicações, disponíveis para um grande número de usuários [Foster, 2002]. Estes recursos são pertencentes a diferentes organizações virtuais, do inglês VO (*Virtual Organizations*, que podem ser indivíduos ou instituições, que possuem tipos variáveis de políticas quanto ao acesso e uso dos recursos e restrições quanto à disponibilidade dos mesmos, que podem, dinamicamente, tornarem-se inacessíveis [Foster et al., 2001]. Consequentemente, os recursos pertencentes a estas organizações podem ser acessados e combinados por diferentes usuários para alcançarem seus objetivos computacionais.

2.1 Características de Ambientes Distribuídos

Uma das utilizações básicas de um ambiente de grade é poder executar uma aplicação existente em outro computador. Muitos computadores, em diversas organizações têm menos de 5% de seus recursos ocupados em um dia de trabalho. Os recursos existentes nestes computadores podem ser disponibilizados em um ambiente de grade para colaborar com

a execução de tarefas que necessitem de mais poder computacional. Além disso, muitos recursos que não são utilizados nessas organizações podem ser vistos como recursos adicionais a uma grade, não só para serem usados para um aumento de capacidade deste ambiente, como também por serem recursos que não existiam no ambiente até então.

Como resultado de sua natureza, as grades provêm uma variedade de serviços que os usuários podem incorporar para atingir os seus objetivos computacionais. Especificamente, usuários podem acessar recursos, aplicações e serviços, submeter tarefas para execução, criar uma combinação de processos em *workflows* e verificação do *status* da tarefa ou do ambiente de grade.

Este ambiente distribuído também é frequentemente utilizado como potencial para computação paralela massiva. Existem, atualmente, muitas aplicações que são desenvolvidas para serem executadas em ambientes paralelos, aos quais, porém, os usuários não possuem acesso. Uma grade pode prover este ambiente para execução, desde que os algoritmos já estejam implementados para serem executados em sub-tarefas paralelas.

Grades computacionais, em muitos casos, utilizam-se de *clusters* e *multi-clusters* para o seu poder computacional. Faz-se importante, então, a diferenciação entre estes conceitos e outros bastante utilizados em sistemas distribuídos. Para tal propósito as subseções seguintes mostram as diferenças entre *cluster*, *multi-cluster*, grades e nuvens computacionais.

2.1.1 Clusters

Segundo Buyya [Buyya, 1999], *cluster*, que em português é conhecido como agregado computacional, é um sistema para processamento paralelo e distribuído que consiste em uma coleção de computadores interconectados e trabalhando juntos como um único recurso computacional integrado.

Para Dantas [Dantas, 2005], um *cluster* pode ser entendido como uma agregação de computadores de uma forma dedicada (ou não) para a execução de aplicações específicas de uma organização ou de uma instituição.

Assim, é um conjunto ou aglomerado de computadores que torna possível a resolução de problemas que, em muitos casos, só seria viável utilizando super computadores. Com isso, o objetivo é fazer com que todo o processamento da aplicação seja distribuído aos computadores, mas de forma que pareça com que eles sejam um único computador.

Em um *cluster* tem-se como unidade básica um único núcleo(*core*), também chamado de nó. Todos os computadores pertencentes ao *cluster* devem ser interconectados através de uma rede, de qualquer topologia.

Essa rede precisa ser criada de forma que permita trabalhar a questão de escalabilidade de forma rápida, ou seja, tratar do acréscimo ou da retirada de um nó sem interromper o funcionamento do *cluster*. Além disso, os computadores devem possuir o mesmo sistema operacional instalado, mantendo-se assim uma homogeneidade entre sistemas operacionais. Quanto ao hardware, o grau de complexidade do *cluster* é diretamente proporcional à heterogeneidade dos equipamentos.

Em um *cluster*, portanto, tem-se um conjunto centralizado de recursos, estando estes conectados através de uma rede em que os mesmos podem ou não serem dedicados. Essa dedicação depende de políticas de gerenciamento, escalonamento de processos, entre outros. Dessa forma, os computadores pertencentes a um *cluster* trabalham como uma única identidade física, em que seus recursos são centralizados e gerenciados por um nó central.

Entre os tipos de aplicações para *clusters*, destacam-se as que buscam uma alta disponibilidade do ambiente e as que procuram um alto desempenho em sua execução [Dantas, 2005]. Aplicativos que buscam alta disponibilidade são caracterizados por não tolerarem interrupções. Entretanto, os que tem como foco o alto desempenho preocupam-se com o número de processadores, quantidade de memória e espaço em disco. Ainda, em se tratando de alto desempenho, para aplicações paralelas e distribuídas, considera-se o desempenho da rede de interconexão. Estas particularidades são facilmente encontradas em ambientes de *cluster*.

A figura 2.1 representa um exemplo de arquitetura de *cluster*.

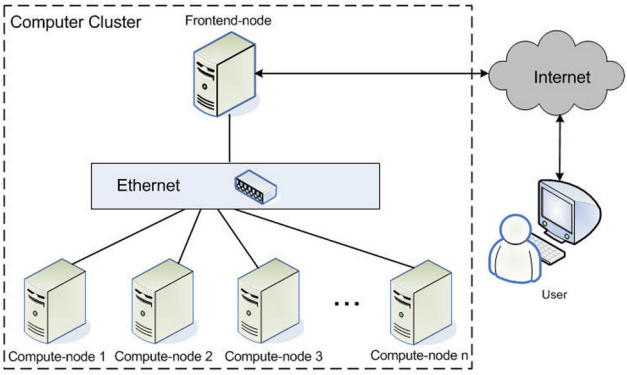


Figura 2.1: Exemplo de uma arquitetura de *cluster* [Sangket et al., 2010]

2.1.2 Multi-Clusters

A utilização de ambientes de *clusters* é bastante comum em diversas áreas. Entretanto, diversos problemas computacionais demandam utilização de mais poder computacional do que os encontrados em um ambiente único de *cluster*. Neste caso, um ambiente de múltiplos *clusters* pode contribuir para um aumento do poder computacional e auxiliar na resolução destes problemas de uma maneira colaborativa. Estes ambientes, portanto, são conhecidos como *multi-clusters* e diferem de grades computacionais por, usualmente, utilizarem redes dedicadas de interconexão entre os *clusters* com uma topologia conhecida e características previsíveis de desempenho [Javadi et al., 2006].

Diferentemente de *cluster*, que é um conjunto de estações de trabalho independentes interconectadas por uma rede local LAN (*Local Area Network*), *multi-clusters* são sistemas computacionais formados por conjuntos de *clusters* independentes conectados por uma rede WAN (*Wide Area Network*), permitindo o compartilhamento de recursos em diferentes domínios através da conexão de ambientes de *cluster* geograficamente dispersos.

Entretanto, existem diversos desafios que devem ser levados em consideração quanto ao grau de complexidade no escalonamento de tarefas nos ambientes de *multi-clusters*. Os recursos em *multi-clusters* são:

- distribuídos;
- heterogêneos;
- altamente compartilhados em tempo e espaço.

Dadas essas particularidades de recursos em *multi-clusters*, a constante submissão de tarefas e a dinamicidade de disponibilidade da capacidade de CPU, por exemplo, torna difícil a aplicação de algoritmos tradicionalmente utilizados em ambientes de *cluster* [Abawajy and Dandamudi, 2003]. Desse modo, os principais componentes deste ambiente são o gerenciador dos recursos compartilhados , responsável por fornecer informações sobre as filas do sistema, carga de processamento e disponibilidade de cada nó, e um escalonador de tarefas, responsável por decidir onde alocar as tarefas de acordo com sua estratégia.

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, independentemente de onde estão associados fisicamente, permitindo um melhor aproveitamento dos recursos do sistema como um todo [Yeo et al., 2006a].

A figura 2.2 representa um exemplo de arquitetura de *multi-cluster*.

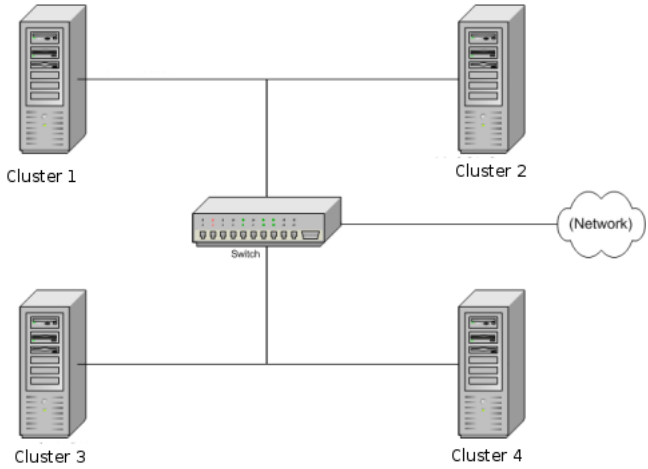


Figura 2.2: Exemplo de uma arquitetura de *multi-clusters* [MySQL, 2011]

2.1.3 Grades Computacionais

Grades computacionais podem ser entendidas como sendo uma plataforma heterogênea de computadores, que estão geograficamente espalhados, e na qual se tem um compartilhamento de tecnologias, serviços e recursos, de acesso comum através de uma interface única. Portanto, um dos principais objetivos é alcançar uma interoperabilidade entre organizações virtuais, através de um compartilhamento e cooperação entre recursos computacionais distribuídos.

Uma característica encontrada nestes ambientes trata da heterogeneidade de computadores e recursos tanto em termos de hardware quanto de software, sendo que, usualmente, os mesmos estão fora de uma rede local LAN. Com isso, não há uma centralização de recursos e gerenciamento destes, tendo-se uma maior escalabilidade do ambiente. O ambiente escalável demanda de melhores políticas de gerenciamento, a fim de evitar qualquer tipo de perda de performance ou degradação de desempenho no sistema [Bote-Lorenzo et al., 2004].

As organizações virtuais, no entanto, passam a exercer o papel de gerenciar seus computadores, fazendo uso de suas políticas especificas, não objetivando uma visão única do sistema [Dantas, 2005]. Por este

ambiente distribuído ser gerenciado através de regras determinadas pelas organizações, normalmente o poder computacional não é dedicado, utilizando-se da ociosidade de computadores conectados na grade. Mas a dedicação ou não ao ambiente, bem como disponibilidade, capacidade e desempenho, ficam a cargo das políticas gerenciais de cada organização.

Outra característica importante em grades computacionais é a de que as mesmas devem possuir mecanismos que lidem com balanceamento de carga, ou seja, fornecer uma distribuição equilibrada do processamento. Em casos em que um computador esteja utilizando todos os recursos disponíveis de forma intensa, pode-se escalonar tarefas para computadores que estão mais ociosos. Escalonamento (do inglês *scheduling*) em grades é selecionar o computador, ou o conjunto de recursos, mais apropriado para executar determinada tarefa [Venugopal et al., 2004], ou seja, um ajuste mais justo de processamento disponível por aplicações que requeiram processamento. Ainda, além deste ajuste, existem casos em que praticamente todos os recursos da grade estão sendo utilizados, podendo, portanto, suspendê-los temporariamente ou até mesmo cancelar tarefas de menor prioridade, para liberar recursos que possam ser balanceados para as tarefas de maior prioridade. Outra alternativa é colocar as aplicações que demandam recursos não disponíveis em uma fila para serem executados quando os recursos estiverem disponíveis.

Além dessas características, outras, também importantes, são: monitoramento de tarefas, que auxilia no acompanhamento da execução e num controle de erros; ambientes dinâmicos, já que os recursos disponíveis sofrem mudanças na mesma escala de tempo que a duração de uma aplicação [Foster et al., 2001]; segurança na utilização de recursos e no fluxo de dados; e confiabilidade através da replicação de tarefas em mais de um computador ou conjunto de computadores que possuam os recursos necessários para a execução das mesmas.

Assim, as grades computacionais buscam através de um ambiente distribuído, prover aos usuários serviços com os requisitos de qualidade corretos para o perfeito funcionamento de suas aplicações [Berman et al., 2003].

A figura 2.3 mostra um exemplo de um ambiente grade computacional.

Entre todas as características citadas, porém, deve haver um equilíbrio sobre a utilização das mesmas, para não ocorrer uma degradação de desempenho no sistema. A simples existência de uma grande quantidade de recursos computacionais homogêneos e/ou heterogêneos distribuídos, em qualquer organização, bem como a natureza distinta dos usuários nas organizações, leva à criação de diferentes políticas de utilização de recur-

Figura 2.3: Exemplo de um ambiente grade computacional [Adarsh, 2011]

sos, segurança de acesso e controle destes recursos computacionais. Desta forma, o processamento cooperativo das aplicações de diferentes usuários com um grau de desempenho desejável representa um grande obstáculo para estes ambientes distribuídos.

Para conseguir manter essa autonomia no ambiente, provendo e compartilhando recursos, sem uma perda de desempenho, tem-se utilizado os pacotes de software de gerenciamento de recursos [Krauter et al., 2002], conhecidos em inglês como RMS (*Resource Management Systems*). A grande motivação para o estudo dos pacotes RMS baseia-se no crescimento da complexidade necessária para prover às aplicações uma utilização justa dos recursos computacionais existentes nas organizações.

Assim, os RMS devem prover, entre outros:

- descoberta de recursos;
- mecanismos de seleção;
- mecanismos de verificação de capacidade;
- alocação de recursos com reserva;
- requisição de recursos;

- permissão de interação para negociação e mecanismos de notificação.

A depender do domínio das aplicações alvo e suas características, os sistemas de grades podem ser classificados nas seguintes categorias [Yeo et al., 2006b]:

- **Grades Computacionais:** responsáveis pela distribuição de facilidades computacionais para a execução de aplicações de computação intensiva, como por exemplo, aplicações do tipo BoT (*Bag-of-Tasks*), que são coleções de *jobs* independentes [Cirne et al., 2003]. Isto é, preocupam-se em agrupar o máximo de recursos ou compartilhar ciclos de processamento para aumentar o poder computacional Alguns exemplos de projeto que se encaixam nesta categoria são: Nimrod-G [Buyya et al., 2000] e SETI@home [Anderson et al., 2002];

- **Grades de Dados:** do inglês *Data Grid*, estes ambientes provêm uma infraestrutura de acesso, transferência e gerenciamento de uma grande quantidade de dados distribuídos entre diferentes repositórios. Os dados são armazenados e acessados de forma transparente quanto a sua localização. Estas grades ainda provêm uma análise destes dados e o compartilhamento de resultados. Tais ambientes são comumente encontrados nas áreas de astronomia e meteorologia. Um exemplo deste tipo de grade é o Virtual Observatory [Ohishi, 2006];

- **Grades Provedoras de Serviços para Aplicativos:** conhecidas em inglês como grades ASP (*Application Service Provisioning*), essas grades concentram-se em prover acesso remoto a aplicativos, módulos e bibliotecas hospedados em *Data Centers* ou em outros ambientes distribuídos, do tipos Grades Computacionais. Um exemplo é o NetSolve [Seymour et al., 2005];

- **Grades Interativas:** provêm serviços e plataformas que permitem que o usuário relacione-se com as aplicações, em tempo real, que estão executando na grade. São bastante usados com aplicativos multimídia do tipo vídeo-conferência. Como exemplo tem-se o AccessGrid [Childers et al., 2000];

- **Grades de Conhecimento:** estes ambientes trabalham com serviços de análises de negócios que são integrados com serviços do tipo *Data Mining* [Han and Kamber, 2006], o qual é um processo de

extração de padrões em grandes quantidades de dados. Um exemplo de grades de conhecimento é o KnowledgeGrid [Cannataro and Talia, 2003];

- **Grades Utilitárias:** focadas no fornecimento de serviços de grades, como utilitários de TI, para usuários finais baseando-se no princípio de acesso pago. Isso ocorre mediante um framework para negociação e estabelecimento de contratos, bem como alocação de recursos necessários aos usuários. Exemplos destas grades são o Utility Data Center [Graupner et al., 2003] e o Gridbus [Buyya and Venugopal, 2004].

Foster e Kesselman [Foster and Kesselman, 2004] definem, ainda uma classificação para os aplicativos que utilizam-se de grades computacionais para atingirem seus objetivos. Veja-se:

- **Supercomputação Distribuída:** o grid é usado para aplicativos que não conseguem resolver o seu problema em um único sistema computacional. Estes aplicativos demandam de muitos recursos como CPU e memória;

- **Computação de Alto Desempenho:** os aplicativos utilizam recursos da grade para escalonar um grande número de tarefas fracamente acopladas ou de tarefas independentes, buscando melhora de desempenho;

- **Computação Sob-demanda:** neste caso, as aplicações demandam de um uso intensivo de recursos computacionais que se encontram indisponíveis localmente. Diferencia-se da Supercomputação Distribuída, haja vista que trabalha colaborativamente com os recursos e não faz uso totalitário dos mesmos para ganho de desempenho;

- **Computação para Grandes Quantidades de Dados:** os aplicativos pertencentes a esta classe processam uma enorme quantidade de dados mantidos em diferentes repositórios ou bases de dados, espalhados geograficamente;

- **Computação Colaborativa:** aplicativos que possuem uma preocupação primordial em melhorar as relações humanas. Em outras palavras, preocupam-se em compartilhar recursos computacionais, como por exemplo, dados e simulações.

2.1.4 Nuvem Computacional

Computação em nuvem ou nuvem computacional, do inglês *cloud computing*, tem se tornado bastante utilizada como um modelo popular de computação para suportar o processamento de grandes volumes de dados. É um novo paradigma de computação que visa à integração de diversos conceitos e tecnologias para a geração de um ambiente tecnológico ubíquo.

O objetivo do modelo de computação em nuvem é realizar uma melhor utilização de recursos distribuídos, colocando-os juntos, visando uma maior vazão, bem como atacar uma grande escala de problemas computacionais [Rimal et al., 2009].

O conceito de computação tem como principal característica o fornecimento de recursos através de uma nuvem computacional, ou seja, um ambiente ubíquo no qual os usuários utilizam os serviços disponibilizados sem a necessidade de possuírem informações sobre os elementos que os compõem.

Segundo Foster et al.[Foster et al., 2008], computação em nuvem pode ser definida como um paradigma de computação distribuída de larga escala que é impulsionada pela economia em que, cada conjunto de recursos (abstraídos, virtualizados, dinamicamente escaláveis, armazenáveis, com poder computacional gerenciável), bem como plataformas e serviços são fornecidos sob demanda para clientes externos através da Internet. Ainda, segundo Foster, nesta definição existem alguns pontos chaves que demonstram que computação em nuvem é um paradigma de computação distribuída especializado, distinguindo-se das abordagens tradicionais nos seguintes aspectos:

1. altamente escalável;
2. pode ser encapsulado como uma entidade abstrata que fornece diferentes níveis de serviço para os clientes fora da nuvem;
3. impulsionada pelas economias de escala;
4. os serviços pode ser dinamicamente configurados, através de virtualização e outras abordagens, e entregues sob demanda.

Segundo Rimal, o conceito de computação em nuvem não é totalmente novo para o desenvolvimento e operação de aplicações Web. Ele permite um desenvolvimento com melhor custo-benefício de portais Web escaláveis com grande disponibilidade e infraestrutura tolerante a falhas [Rimal et al., 2009].

A figura 2.4 mostra a arquitetura em camadas da computação em nuvem. Tal arquitetura foi feita para aplicações de software que usam serviços sob demanda acessíveis pela internet.

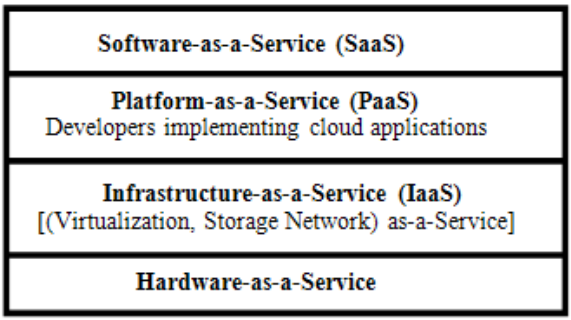


Figura 2.4: Arquitetura de camadas de uma nuvem computacional [Rimal et al., 2009]

2.1.5 Comparativo entre ambientes distribuídos

Todos os ambientes observados possuem diversas diferenças. Um *cluster* pode ser visto como uma configuração local de hardware e software, usualmente em uma LAN, que objetiva solucionar problemas de uma organização. Por outro lado, grades e nuvens computacionais possuem como característica a distribuição geográfica de recursos, portanto, em uma WAN (*Wide Area Network*).

Outra diferenciação destes ambientes reside em como ocorre o gerenciamento de recursos e serviços. Em ambientes de *clusters* os recursos são gerenciados por um nó central e o conjunto de computadores é visto como uma identidade única. Por outro lado, os recursos das grades e nuvens são gerenciados especificamente por cada organização virtual que os possuem.

Portanto, os ambientes distribuídos apresentados possuem características distintas que podem ser visualizadas na tabela 2.1.

2.2 Visão Geral de Grade Computacional

Grades computacionais podem possuir diferentes significados para diferentes indivíduos. Em uma visão mais ampla, normalmente uma grade é vista como uma analogia às redes elétricas, em que os usuários têm acesso à energia sem uma preocupação sobre quem a está gerando e for-

Tabela 2.1: Resumo das características dos *clusters* e grades computacionais

Características	Clusters	Multi-Clusters	Grades	Nuvem
Sistema Homogêneo	X			
Sistema Heterogêneo		X	X	X
Recursos Centralizados	X			
Recursos Decentralizados		X	X	X
Rede Local	X			
Rede de Longa Distância		X	X	X
RMS	X	X	X	X
Alta Disponibilidade	X	X		X
Ambiente Dinâmico e diverso			X	X
Compartilhamento de Recursos Distribuídos		X	X	X

necendo [Foster and Kesselman, 2004]. Nessa visão, a computação torna-se pervasiva e os usuários têm acesso aos recursos computacionais, como por exemplo, processadores, memória e dados, sem um necessário conhecimento sobre a localização desses recursos. Além disso, outras informações como hardware, sistemas operacionais e outras tecnologias, que estão entre o usuário e o recurso, são dispensáveis para o usuário. Ele possui uma visão única e transparente da grade, como sendo um grande ambiente de alto desempenho.

Entretanto, computação em grade também pode ser vista como uma integração de várias tecnologias e soluções que objetivam um fim em comum. Neste sentido, o ponto chave para a integração dessas tecnologias em ambientes distribuídos reside em sua infraestrutura, que vem evoluindo nos últimos tempos, a fim de oferecer um suporte inter-organizacional de aplicações e compartilhamento de recursos através de uma integração de plataformas, organizações e tecnologias.

Assim, o termo computação em grade foi gerado por Foster e Kesselman [Foster and Kesselman, 1999] como sendo uma infraestrutura de hardware e software que fornece um meio confiável, consistente, pervasivo e de acesso barato a recursos computacionais de alta qualidade.

Ademais, como grade computacional não possui uma única definição precisa, algumas tentativas de definição e de atribuição do seu

conceito a um ambiente distribuído, podem ser observadas na literatura. Segundo Foster [Foster, 2002] uma grade computacional pode ser definida como um sistema que coordena recursos não sujeitos a um controle centralizado; utiliza padrões e interfaces de propósito geral e abertos; e entrega uma qualidade de serviço não trivial. Foster ainda destaca o conceito de grade como sendo um compartilhamento coordenado de recursos para resolução de problemas, de forma dinâmica, entre organizações virtuais multi-institucionais. Esse compartilhamento é altamente controlado, sendo definido o que e com quem são compartilhados recursos, mediante regras definidas pelas organizações.

Por outro lado, Németh e Sunderam [Németh and Sunderam, 2005] apresentam o que uma grade computacional deve prover, focada na diferença semântica dessa para um ambiente tradicional de computação distribuída. Segundo os autores, uma grade computacional possui uma gama de recursos heterogêneos, incluindo, por exemplo, rede, dados, armazenamento, sensores e dispositivos de entrada e saída de áudio. Diferentemente de um ambiente distribuído, grades computacionais possuem um *pool* virtual de recursos e são geograficamente distribuídos entre diferentes tipos de domínios, ao contrário do ambiente tradicional em que os nós computacionais pertencem a um único domínio. Esse *pool* virtual de recursos é dinâmico, e não estático, uma vez que recursos podem ser adicionados e retirados a qualquer momento de acordo com critérios de seus donos. Além disso, o desempenho e a carga dos recursos são constantemente alterados. Com isso, os usuários não precisam possuir conhecimento sobre o estado atual da grade computacional e seus recursos, além de possuírem acesso a grade como um todo, podendo ser restrito o acesso a recursos específicos devido a politicas de acessos de organizações virtuais, e não a *sites* individuais com acesso a todos os recursos, como no caso dos ambientes tradicionalmente distribuídos.

Ainda, de uma forma mais geral, Buyya [Buyya, 2011] apresenta o conceito de grade computacional como sendo um tipo de sistema paralelo e distribuído que permite compartilhamento, seleção e agregação dinâmica em tempo de execução de recursos autônomos geograficamente distribuídos, dependendo de sua disponibilidade, capacidade, desempenho, custo e requisitos de qualidade de serviço necessários aos usuários.

Desse modo, é possível concluir que computação em grade pode ser definida como vários usuários separados geograficamente e podendo utilizar recursos computacionais provenientes de outros computadores que não sejam os seus [Foster and Kesselman, 2004]. Uma grade, deve ser confiável e transparente quando utilizada, não interessando onde serão processados ou armazenados os dados. Quando um usuário faz uma re-

quisição ele obtém a resposta, sendo o processamento realizado em algum computador independentemente de sua localização geográfica.

2.3 Evolução da Computação em Grade

A computação em grade surgiu como uma evolução dos ambientes distribuídos, com seu desenvolvimento nas comunidades acadêmicas. A criação do novo conceito deveu-se ao fato de pesquisas estarem sendo feitas de forma conjunta, porém com seus pesquisadores geograficamente separados uns dos outros. Com isso, notou-se a importância de criar um ambiente que pudesse prover poder computacional, além de disponibilizar recursos e resultados de forma dinâmica. Uma das primeiras ferramentas de destaque em ambientes distribuídos foi a biblioteca MPI (*Message Passage Interface*) [Snir, 1998], que fornece comunicação entre processos, em muitos casos utilizando-se de *clusters* como ambiente.

Com o passar do tempo, surgiram projetos de infraestrutura que visavam a facilitar a utilização de recursos disponíveis nesses ambientes através da distribuição e coordenação de processos, dando início às grades computacionais. Essas infraestruturas passaram a ser aperfeiçoadas com a agregação de novas funcionalidades. Dentre elas pode-se citar: escalonadores, que têm o objetivo de otimizar a distribuição de processos e realizar o monitoramento de recursos, tais como, memória, rede e processador.

Esta evolução da computação em grade foi apresentada, através de características técnicas e históricas, por Roure *et al* [De Roure et al., 2003] em uma divisão de três gerações. As mesmas são descritas a seguir.

2.3.1 Primeira geração

A primeira geração inclui os precursores da computação em grade, que eram projetos de supercomputação para conectar *sites*. Em outras palavras, esta geração era composta de sistemas que envolviam soluções proprietárias para o compartilhamento de recursos de computação de alto desempenho. Naquele tempo, esta abordagem ficou conhecida como metacomputação. Dois grandes projetos fizeram parte dessa geração: FAFNER (*Factoring via Network-Enabled Recursion*) [Fafner, 2011] e I-WAY (*The Information Wide Area Year*) [Foster et al., 1996]. Apesar das grandes diferenças, ambos projetos tiveram que lidar com questões de comunicação, gerência de recursos e manipulação remota de dados.

2.3.2 Segunda Geração

Os projetos dessa geração possuem um foco maior em *middlewares* para suportar uma larga escala de dados e processamento, além de ambientes bastante heterogêneos.

Middlewares geralmente são considerados como uma camada de

software entre o sistema operacional e a camada de aplicações, provendo uma variedade de serviços requisitados por uma aplicação para funcionar de forma correta [Bernstein, 1996]. Ainda, segundo Dantas [Dantas, 2005], *middleware* é um ambiente que provê para os usuários de *clusters* e grades computacionais uma transparência segura de acesso, facilidades de submissão das aplicações, gerenciamento de recursos e serviços da configuração distribuída geograficamente.

Nessa geração, as grades computacionais passaram a serem vistas como infraestruturas de escala global, suportando diversidades de aplicações, porém, confrontando três questões principais:

- *Heterogeneidade:* uma grade computacional envolve múltiplos recursos de natureza heterogênea, que podem estar localizados em diferentes domínios administrativos separados geograficamente.

- *Escalabilidade:* uma grade deve ser escalável, possuindo a habilidade de lidar com o crescimento de poucos recursos para diversos recursos. Isso pode trazer problemas de desempenho com aumento do tamanho do ambiente. Consequentemente, os aplicativos que exigem um grande número de recursos, geograficamente distribuídos, devem ser projetados com tolerância à latência e capazes de explorar a localização dos recursos acessados.

- *Adaptabilidade ou Dinamicidade:* em uma grade, uma falha de recurso é a regra e não a exceção. Na verdade, com tantos recursos em uma grade, a probabilidade de falha em algum recurso é muito alta. Os gerenciadores de recursos ou os próprios aplicativos devem adequar-se dinamicamente, de modo a extrair o máximo desempenho a partir dos recursos e serviços disponíveis.

Os dois projetos mais representativos dessa geração são o Legion [Chapin et al., 1999] e o Globus [Foster and Kesselman, 1999].

Legion é um meta-sistema orientado a objetos e modelado como um *middleware* para grades. Diferentemente do Globus (que pode ser caracterizado como uma soma de serviços), possui uma arquitetura integrada. É apropriado para uso em programas paralelos de granularidade grossa. Os recursos, usuários e entidades são objetos [Lewis and Grimshaw, 1996], os quais são endereçados com identificador único em um espaço no contexto.

O projeto Globus possui como um de seus principais resultados o *Globus Toolkit* (*GT*) [Foster and Kesselman, 1999] e é um software *open source*. O GT 2 pode ser classificado como sistema da segunda geração ao passo que é um conjunto de componentes que compõem um *middleware*.

2.3.3 Terceira geração

Nessa geração, que é a atual, os ambientes de grade focam em colaboração global distribuída e em uma abordagem orientada a serviços. Além desta nova abordagem, adota-se uma visão holística da infraestrutura de *e-Science* e começa-se a trabalhar com metadados bem como com habilidades de apresentação autonômicas de características. Entende-se por *e-Science* uma colaboração global em áreas chaves da ciência utilizando computação intensiva em ambientes altamente distribuídos [Hey and Trefethen, 2002].

Foi nessa geração que surgiram diversas tentativas de padronização de uma arquitetura para aplicações baseadas em ambientes de grades. A padronização tornou-se um elemento fundamental para esses ambientes, sendo necessário que esforços de desenvolvedores fossem unidos em busca de uma arquitetura aberta e padrão, trazendo a integração e a interação entre recursos e serviços de forma natural [Dantas, 2005] Entre essas propostas, destacam-se:

- **OGSA**: O padrão OGSA *(Open Grid Services Architecture)* [Butler et al., 2000] teve seu surgimento devido ao fato de diferentes tipos de *middlewares* para grade estarem sendo criados e, com isso, acarretando uma falta de interoperabilidade entre os mesmos. Esse padrão traz um suporte à criação e gerenciamento de aplicações que são mantidos pelas organizações virtuais. Esse padrão ainda define o conceito de serviços de grade, baseado em Web Services [Cerami and Laurent, 2002] que provê um conjunto de interfaces [Talia, 2002]:
  - descoberta;
  - serviços de criação dinâmica;
  - gerenciamento de vida útil;
  - notificações;
  - ambiente simples de execução; e
  - gerenciamento.
- **OGSI**: O padrão OGSI *(Open Grid Services Infrastructure)* [Tuecke et al., 2003] é uma especificação técnica e formal dos conceitos descritos pelo OGSA, expressadas em WSDL *(Web Services Description Language)* [Christensen et al., 2001], definindo padrões de interface, ações e esquemas para grades computacionais, incluindo os serviços de grade. Estas interfaces ajudam a definir como construir, gerenciar e expandir estes serviços de grade. OGSI também

introduz a idéia de *stateful Web service* [Foster et al., 2005]. A versão GT 3 do Globus possuía essa nova filosofia de serviços de grade implementando o OGSI.

- **WSRF**: O novo padrão, conhecido como WSRF *(Web Services Resource Framework)* [Czajkowski et al., 2005] foi apresentado para substituir o OGSI, visto que este possui algumas desvantagens, por exemplo, de não trabalhar com ferramentas Web Services atuais e por possuir uma especificação longa e densa. Estas e outras desvantagens impediram a convergência de serviços de grade em um padrão Web Services. O WSRF, portanto, evoluiu do OGSI para explorar novos padrões de serviços Web, especificamente WS-Addressing, e para responder a implementações anteriores. O WSRF mantém praticamente todos os recursos funcionais presentes no OGSI. A abordagem OGSA não é afetada pela mudança de padrão para WSRF. Todos os serviços de alto nível definidos em OGSA continuarão tendo as mesmas interfaces e especificações. A versão GT 4 do Globus possui este novo padrão.

O projeto GT 4 é uma coleção de componentes de software, implementações de serviços e bibliotecas, formando um ecossistema para a construção de infraestruturas de grades computacionais e serviços de grade [Foster, 2006]. O principal objetivo do projeto é proporcionar todos os serviços comuns e básicos de software para formar uma plataforma base que pode ser usada para construir diferentes tipos de grades computacionais. Como já mencionado, o desenvolvimento do GT 4 segue uma arquitetura orientada a serviços e, consequentemente, usa tecnologia Web Service extensível para integrar componentes de software e disponibilizar os serviços. Atualmente o Globus encontra-se na versão 5.

Na terceira geração de grades computacionais surge o contexto de *e-Science* e com ele a noção de Grade Semântica. Essa noção foi observada devido a uma necessidade de se atingir um alto nível de facilidade de uso e uma automatização simples com o objetivo de permitir computação e colaboração flexível em uma escala global [De Roure et al., 2005]. A Grade Semântica refere-se a uma abordagem para ambientes de grades em que informações, recursos computacionais e serviços são descritos através de um modelo semântico de dados. Neste modelo, dados e metadados são expressos através de ontologias, que consistem em descrições de conceitos e seus relacionamentos [Bettini et al., 2010]. O uso de ontologias possibilita um ambiente mais entendível para os usuários, além de tornar a descoberta de recursos facilitada e automática nas organizações virtuais.

A Grade Semântica, portanto, é uma extensão das grades compu-

tacionais em que as informações e serviços são bem definidos, permitindo que usuários e computadores trabalhem cooperativamente [De Roure et al., 2005]. Ela envolve as três camadas conceituais das grades: conhecimento, informação e computação de dados. Estas camadas provêm um acesso rico, sem custo e pervasivo aos recursos heterogêneos distribuídos globalmente [De Roure et al., 2005].

2.4 Arquitetura de Grade Computacional

Na definição de uma arquitetura de grade computacional, parte-se da perspectiva de que operações efetivas entre organizações virtuais requerem o estabelecimento de uma relação de compartilhamento entre quaisquer potenciais participantes, tornando a interoperabilidade a questão central a ser abordada [Foster et al., 2001]. Na arquitetura de grades computacionais identifica-se, portanto, a interoperabilidade nas aplicações de organizações virtuais como questão central, utilizando-se de protocolos comuns para negociação de recursos, assim como estabelecendo e gerenciando relações de compartilhamento [Dantas, 2005].

Uma arquitetura baseada em padrões de protocolos abertos facilita a extensibilidade, interoperabilidade, portabilidade e compartilhamento de códigos, tornando mais fácil definir um padrão de serviços que oferecem recursos.

Na figura 2.5, tem-se, portanto, a arquitetura de grades apresentada por Foster [Foster et al., 2001] e inicialmente organizada por camadas, semelhante a arquitetura TCP/IP, constituída por uma pilha de protocolos, sendo que as camadas inferiores dão suporte às camadas superiores.

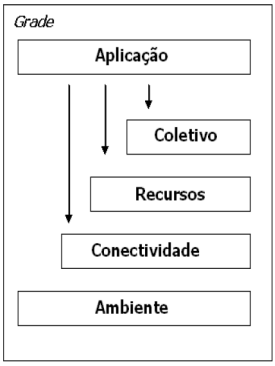


Figura 2.5: Arquitetura em cinco camadas em ambientes de grades computacionais

As camadas apresentadas na figura 2.5 são explicadas a seguir [Dantas, 2005] [Foster et al., 2001]:

- **Ambiente:** os componentes desta camada implementam operações específicas locais que ocorrem em cada recurso como resultado das operações de compartilhamento nos níveis superiores. Deve-se ter implementados mecanismos que lidem com a negociação, obtendo informações referentes à estrutura, o estado e as possibilidades de recursos. Também, deve-se atentar para os mecanismos de gerenciamento de recursos que são encarregados de fornecer formas de monitoramento da qualidade de serviço, *QoS (Quality of Service)*;

- **Conectividade:** esta camada define os protocolos básicos de comunicação e autenticação necessários para as transações de rede específicas do ambiente de grade. Os protocolos de comunicação têm o papel de permitir a troca de dados entre os níveis de Ambiente e Recursos. Por outro lado, os protocolos de autenticação ficam responsáveis por prover mecanismos seguros para a verificação da identidade de usuários e recursos;

- **Recursos:** neste nível, encontram-se os protocolos de autenticação e comunicação do nível conectividade para definir protocolos e *APIs (Application Programming Interface)* que forneçam segurança na negociação, iniciação, monitoramento, controle, geração de relatórios e outros detalhes envolvidos nas operações com recursos individuais. Os protocolos desta camada preocupam-se apenas com recursos individuais, ignorando questões a respeito de estado global e coleções distribuídas, sendo estas tratadas pela camada coletivo;

- **Coletivo:** enquanto no nível de recursos são tratadas as operações no âmbito de cada recurso individualmente, no nível coletivo os componentes atuam nas interações entre coleções de recursos. Os protocolos implementados nesta camada baseiam-se nas camadas de recursos e aplicação e implementam serviços tais como:

    1. Serviços de diretório que permitem aos membros de uma organização virtual descobrirem a existência das propriedades de recursos;
    2. Serviços de autorização comunitários que reforçam a política de acesso aos recursos.

- **Aplicação:** esta camada compreende as aplicações dos usuários que operam no ambiente da organização virtual. Os níveis anteriores provêm serviços úteis às aplicações desenvolvidas que as invocam.

A computação em grade atualmente adota como padrão de arquitetura, a OGSA, mencionado na seção da evolução dos ambientes, com seu componente OGSI.

Entretanto, Bernam [Berman et al., 2003] apresenta uma outra proposta de arquitetura de grade contando com quatro camadas. A figura 2.6 apresenta esta arquitetura tendo o esclarecimento de suas camadas a seguir [Dantas, 2005]:

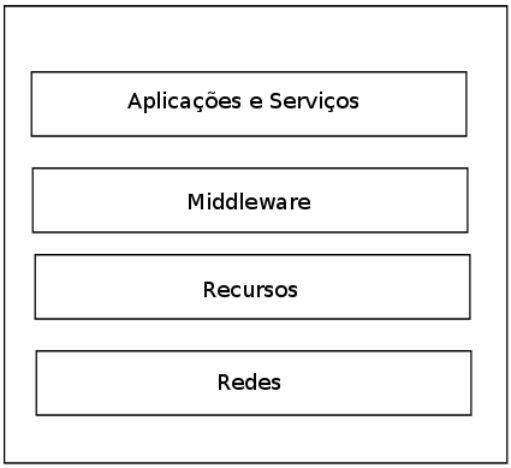


Figura 2.6: Arquitetura em quatro camadas em ambientes de grades computacionais

- **Redes:** tem-se neste nível a base de toda a conectividade, *switches*, roteadores, entre outros, dos recursos das grades;
- **Recursos:** esta camada possui o conjunto de recursos que são encontrados nos ambientes de grades. Um exemplo desses recursos são os *clusters* computacionais que podem estar inseridos para trabalhar de forma dedicada ou colaborativa nestes ambientes;
- **Middleware:** esta camada é responsável por fornecer todos os protocolos que permitam que múltiplos elementos participem de um ambiente de grade. Portanto, é responsável por fornecer as ferramentas que possibilitam aos vários elementos pertencentes ao ambiente distribuído formarem um ambiente único. A camada pode ser caracterizada pela sua inteligência, trazida aos vários elementos

unidos através de software e domínio. O *Middleware* permite que os utilizadores interajam com os recursos computacionais através desta camada, escondendo a complexidade e diversidade da infraestrutura e oferecendo uma interface uniforme para acesso aos recursos;

- **Aplicações e Serviços:** este nível define a camada mais alta de um modelo de grade computacional, incluindo o conjunto de ferramentas de desenvolvimento. Nele encontram-se, também, diversos tipos de aplicações buscando a solução de problemas em diversas áreas. Ainda, a porção de serviços tem a função de prover funções de gerenciamento do uso virtual dos recursos compartilhados entre diferentes usuários, departamentos e organizações virtuais.

2.5 Middlewares de Grades Computacionais

A simples existência de uma grande quantidade de recursos computacionais homogêneos e/ou heterogêneos distribuídos, em qualquer organização, não significa que os mesmos deverão trabalhar cooperativamente para responder às diferentes requisições de diferentes aplicações dos usuários do ambiente.

A natureza distinta dos usuários nas organizações leva à criação de diferentes políticas de utilização de recursos, segurança de acesso e controle dos recursos computacionais, podendo representar, desta maneira, um grande obstáculo para que esses ambientes possam efetivamente processar as aplicações de diferentes usuários cooperativamente e com certo grau de desempenho. Para isso, faz-se necessária a utilização de RMS, seja para gerenciamento de grades computacionais ou de ambientes de *clusters* e *multi-clusters*.

A seguir são apresentados alguns softwares de gerenciamento de recursos computacionais.

2.5.1 Condor

O Condor, desenvolvido pela Universidade de Wisconsin, é um pacote de público acesso e de código aberto, que tem por finalidade o gerenciamento de recursos especializado em *jobs* que requeiram intenso uso computacional [Condor, 2011]. Semelhante a outros pacotes de gerenciamento de recursos, o sistema possui um mecanismo de formação de filas de execução de *jobs*, planejamento de política, esquema de prioridade, monitoramento e gerência de recurso. Algumas características intrínsecas do Condor são:

- Tolerância a falhas: faz *checkpoints* (processo de preservação de um estado da aplicação, que permitirá que a mesma continue sendo

computado do ponto em que o *checkpoint* foi realizado) [Jankowski et al., 2006] periódicos, permitindo que *jobs* migrem e evitando que se percam computações acumuladas caso haja falhas no sistema, no qual estava sendo executado o *job*;

- Os *jobs* são ordenados de acordo com suas dependências.

2.5.2 Oracle Grid Engine

O OGE (*Oracle Grid Engine*), antigo SGE (*Sun Grid Engine*), é um gerenciador de recursos distribuídos - do inglês DMR (*Distributed Resource Management*) - desenvolvido para maximizar a utilização dos recursos em ambientes distribuídos [OGE, 2011]. O sistema, portanto, tem por objetivo gerenciar as requisições dos usuários para recursos computacionais disponíveis. Quando um usuário submete sua aplicação para o OGE o software monitora o estado atual de todos os recursos do ambiente e é capaz de escalonar esse *job* para os recursos mais adequados [OGE, 2011]. O OGE possui quatro tipos de *jobs* que podem ser submetidos. São eles:

- Batch: possui uma sequência única de execução;
- Paramétrico: diferentes tarefas independentes entre si e que podem ser executadas em paralelo. As tarefas são idênticas, o que muda é o conjunto de dados em que operam;
- Paralelo: conjunto de tarefas que podem ser dependentes e cooperarem entre si. Fazem uso de ambientes paralelos como MPI e PVM (*Parallel Virtual Machine* - conjunto integrado de ferramentas de softwares e bibliotecas, os quais emulam um *framework* concorrente em computadores heterogêneos interconectados) [PVM Projects, 2011];
- Interativos: permite que o usuário interaja diretamente com algum recurso disponível no *cluster*.

O OGE atualmente é um software de versão proprietária, entretanto, existe uma versão livre e de código aberto: o *Open Grid Scheduler* ou simplesmente *Grid Scheduler* [Grid Engine, 2011], que é uma continuação do SGE em sua versao 6.2 update 5.

2.5.3 XtremWeb

O XtremWeb é uma ferramenta de código aberto desenvolvida pelo *Laboratoire de Recherche en Informatique* da *Université Paris-Sud*, na

França [XtremWeb, 2011]. Esta ferramenta foi desenvolvida para pesquisa em torno de plataformas de computação distribuída, incluindo grades computacionais, computação global e sistemas *peer-to-peer*.

A ferramenta busca reunir recursos computacionais (CPU, armazenamento e rede) não utilizados em computadores. Este sistema permite, bem como outros sistemas similares, que computadores remotos cooperem provendo os ciclos ociosos de seus processadores e transformando um conjunto de recursos voláteis, distribuídos pela LAN ou Internet, em um ambiente de tempo de execução de aplicativos paralelos. Dentre suas características pode-se citar: permite múltiplos usuários, múltiplas aplicações e implementações através de múltiplos domínios.

2.5.4 Ourgrid

O OurGrid é uma ferramenta de código aberto, desenvolvida na Universidade Federal de Campina Grande, para execução de aplicações do tipo *bag-of-tasks* (onde não existe comunicação entre as tarefas) em grades computacionais [Ourgrid, 2011]. Este ambiente distribuído trabalha com um conceito de grade colaborativa em que cada organização virtual, quando necessário, doa seu poder computacional ocioso em troca de acesso a outras organizações virtuais que tenham recursos disponíveis para uso. Este mecanismo de incentivo torna possível um melhor interesse dos participantes em colaborar com o sistema. As aplicações que são submetidas para o OurGrid podem ser caracterizadas pela forma de interação com o *middleware* [Ourgrid, 2011]. Sendo assim:

- aplicações baseadas em *scripts*;
- aplicações podem incorporar chamadas para a API do Ourgrid;
- aplicações baseadas em *framework*;
- aplicações baseadas em portais.

Capítulo 3

Reserva de Recursos

3.1 Reserva de Recursos em Grades Computacionais

Na maioria dos ambientes de grades computacionais, a submissão de aplicações é feita colocando-se as tarefas em uma fila, caso não exista recursos disponíveis no gerenciador local ou no escalonador. Para resolver este problema e garantir que recursos específicos estejam disponíveis para a aplicação, diversas pesquisas têm proposto uma necessidade de reserva de recursos. Neste sentido, algumas grades computacionais, atualmente , além de prover recursos e um gerenciamento bastante refinado de seu ambiente, estão trabalhando com o conceito de reserva de recursos.

Reserva de recursos permite que usuários solicitem recursos, durante um espaço de tempo determinado e de múltiplos sistemas de escalonamento, necessários para a execução de suas aplicações [Smith et al., 2000]. Em outras palavras, isso significa que um usuário pode solicitar recursos para utilizá-los após um determinado período de tempo, através de uma alocação prévia destes recursos. Como mencionado em [TeraGrid, 2002], uma reserva antecipada dedica um conjunto de recursos, por exemplo, processadores ou memória, para um usuário ou grupo de usuários, durante um espaço de tempo específico no futuro.

Assim, utilizando reserva de recursos em grades computacionais, usuários destes ambientes podem planejar de forma prévia a utilização de recursos baseados nos requisitos de suas aplicações buscando garantir uma qualidade de serviço esperada para cada aplicação.

A reserva de recursos em ambientes de grades computacionais pode ser caracterizada em dois tipos:

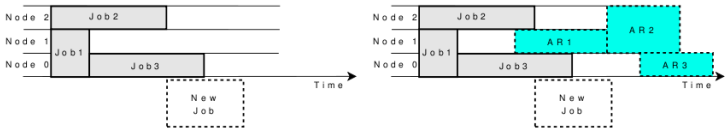
- *imediatas:* é realizada quando a utilização dos recursos é feita no momento atual da reserva;

- *antecipadas:* a reserva antecipada de recursos, também conhecidas pelo acrônimo AR (do inglês: *Advanced Reservation*), é caracterizada por um planejamento do futuro uso dos recursos e, portanto,

seu tempo inicial é diferente do tempo corrente.

Uma reserva de recursos, portanto, é baseada no tempo que os recursos requisitados pelo usuário ficam disponibilizados para serem utilizados. Assim, trabalha-se com dois tempos. Primeiramente, o tempo inicial da reserva, que é o momento em que o recurso reservado previamente é alocado para que possa ser utilizado, sendo este tempo inicial o momento atual, no caso de reserva imediata. O segundo tempo é o de duração ou tempo final (alguns RMS utilizam tempo final calculando a duração a partir do tempo inicial), que é o tempo em que o recurso pode ser utilizado até que o mesmo seja liberado pelo gerenciador.

A figura 3.1 mostra um exemplo de um escalonamento de um RMS com suporte à reserva de recursos e outro sem suporte. Observa-se na figura 3.1(b) que ocorreu uma reserva sem a necessidade de esperar o término dos *jobs*, contribuindo para uma melhor utilização dos recursos.



(a) without advance reservation. (b) with advance reservation.

Figura 3.1: Comparação de escalonamento com e sem reserva antecipada [Sulistio, 2008]

A reserva antecipada pode ser útil para diversos tipos de aplicativos como é descrito a seguir:

1. aplicativos utilizam-se de programação paralela em que cada tarefa requer múltiplos nós para execução simultânea;

2. execução de *workflows*, em que cada tarefa pode depender da execução de outra tarefa e, com isso, o aplicativo deve esperar até que todas as dependências sejam satisfeitas;

3. aplicações multimídia ou de tempo-real que necessitam de uma certa quantidade de largura de banda para evitar o travamento destes tipos de aplicações, fazendo-se necessária a reserva antecipada desta largura de banda a fim de não comprometer a qualidade.

Ao mesmo tempo em que a possibilidade de reservar os recursos e planejar um uso imediato, ou futuro, do ambiente distribuído traz benefícios para o usuário, também implica em uma série de desafios de gerenciamento de recursos para os RMS que possuem suporte à funcionalidade de reserva. Um dos desafios visualizados em mecanismos de *AR* de grades computacionais, como já mencionando, é o de que as grades são ambientes heterogêneos com diferentes tipos de recursos localizados em diferentes organizações virtuais, cada uma com política própria de controle e utilização, o que pode dificultar a reserva. Lidar com a complexidade de requisições dos usuários para reservar um determinado conjunto de recursos é um novo desafio, pois deve-se buscar um mecanismo que evite que muitas requisições sejam realizadas e negadas, ocasionando muita comunicação com o RMS. Ainda, o RMS deve possuir políticas eficientes de alocação de recursos para realizar uma melhor reserva, a fim de evitar a degradação do desempenho do sistema.

Outro desafio para ambientes com suporte à reserva antecipada de recursos é um usuário poder realizar uma reserva com o mínimo conhecimento da grade e de seu estado. Muitos dos usuários de grades computacionais não têm interesse em saber sobre o ambiente e querem somente informar seus requisitos a serem reservados. Além disso, é interessante que os usuários que possuam conhecimento sobre as características de sua aplicação, e se essas influenciam diretamente no desempenho da execução, possam informar as mesmas deixando a cargo do RMS selecionar o melhor recurso a ser reservado.

Além do gerenciador de recursos, ou RMS, lidar com desafios de heterogeneidade, múltiplas organizações virtuais, complexidade de requisição e políticas eficientes de alocação, muitos RMS que possuem suporte à reserva de recursos são classificados como meta-escalonadores e os mesmos são explicados na seção seguinte.

3.2 Meta-Escalonamento

A abordagem de meta-escalonamento, do inglês *meta-scheduling* provê interface de acesso a recursos virtuais para os usuários finais, adotando políticas globais tanto para os provedores de recursos quanto para os consumidores [Xiaohui et al., 2006].

Meta-escalonamento tem como objetivo controlar, facilitar o acesso e manter um gerenciamento integrado sobre as informações entre diversos gerenciadores de recursos. De outra forma, pode-se entender a abordagem de *meta-scheduling* como um ambiente de *middleware* que permite ao usuário usufruir, de uma forma global, dos recursos computacionais existentes em sua organização, sem a preocupação com os gerenciadores

de recursos (RMS) existentes.

O diferencial da abordagem é que nodos computacionais agregados por diferentes gerenciadores de recursos podem se comunicar através do meta-escalonador. Esta é uma camada de mais alto nível que procura atender às requisições das aplicações sem os limites locais de recursos. A figura 3.2 ilustra a idéia de uma estrutura computacional que se utiliza do meta-escalonamento.

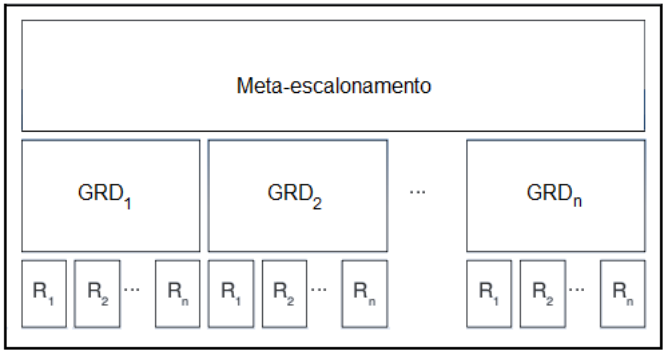


Figura 3.2: Estrutura de um ambiente com um meta-escalonador

Meta-escalonadores, portanto, são caracterizados por receberem requisições de usuários e então, escalonarem a aplicação para um determinado RMS ou *cluster* local, possuindo também a função de orquestrar configurações de *multi-cluster*. A funcionalidade de reserva de recurso de um meta-escalonador necessita que os módulos de co-escalonamento, co-reserva e co-alocação sejam eficientes. Faz-se necessária a explicação de alguns conceitos encontrados nestes ambientes, descritos a seguir.

3.2.1 Co-escalonamento

Como mencionado anteriormente, escalonamento consiste em quando e onde os *jobs* serão executadas e quantos recursos serão alocados. Ou seja, os escalonadores decidem quando e onde um *job* vai executar. O co-escalonamento (do inglês *Co-scheduling*) pode ser entendido como uma habilidade de escalonar tarefas para as executar ao mesmo tempo, mas em diferentes nós de processamento [Sodan, 2005], isto é, a utilização simultânea de múltiplos recursos. É o escalonamento de processos de uma aplicação realizados ao mesmo tempo e de forma coordenada. Existem

dois tipos de co-escalonamento que são descritos a seguir [Sodan, 2005]:

- **Gang:** o escalonamento do tipo *gang* significa escalonar toas as tarefas de um *job* simultaneamente em diferentes nós ou processadores alocados para este *job*. As fatias de tempo são coordenadas de forma global e todas as tarefas de um *job* são preemptivos caso outro *job* tente executar. Não é muito flexível, sendo as decisões de escalonamento estritamente executadas. Este tipo de escalonamento pode ser suportado por hardware (em caso de redes de *broadcast*), podendo também ser totalmente executado por software;

- **Loosely:** essa abordagem originária dos ambientes NOW (*Network of Workstations*) e busca resolver o problema de escalonamento serial e paralelo de cargas de trabalho. Este tipo de escalonamento é baseado no escalonamento local e na comunicação organizada dos nós envolvidos. O objetivo é manter a comunicação de tarefas de um mesmo *job* no mesmo espaço de tempo. É uma abordagem mais flexível para execução das tarefas pois permite alterações dinâmicas de acordo com as necessidades no momento da alocação.

3.2.2 Co-reserva

A função de co-reserva é caracterizada por uma coordenação na utilização de recursos de *clusters* como uma sequência de configuração da grade [Roblitz and Reinefeld, 2005]. Em outras palavras, pode significar, por exemplo, a reserva de computadores disponíveis bem como de recursos de redes de um ambiente distribuídos para a execução de uma aplicação global. No contexto de co-reserva, duas relações são utilizadas:

- **Relação temporal:** pode ser definida como um relacionamento entre tarefas se, e somente se, expressarem uma ordem de execução de tarefas [Bouziane et al., 2008]. Em outras palavras, significa que o horário de uso de algum recurso tem relação com a utilização de outro recurso.

- **Relação espacial:** pode ser definida como um relacionamento entre os componentes se, e somente se, os componentes envolvidos na relação estiverem simultaneamente ativos durante o tempo em que esta relação é válida [Bouziane et al., 2008]. Refere-se, também, à localização desses recursos reservados, como, por exemplo, reservar uma rede de interconexão entre CPUs e um pipeline, no mesmo horário em que o pipeline for reservado.

Para a representação da co-reserva são utilizadas, geralmente, tabelas de *timeslot* para demonstrar o percentual de recursos disponíveis e

alocados em um determinado tempo [Foster et al., 2002]. Essas tabelas fornecem um interessante monitoramento da disponibilidade e capacidade de reservas de inúmeros recursos. A figura 3.3 mostra uma ilustração hipotética de uma tabela *timeslot* com alocações e reservas antecipadas.

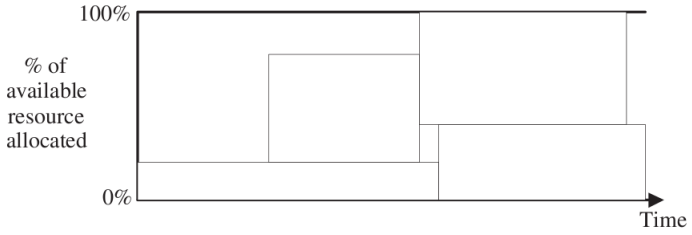


Figura 3.3: Exemplo de tabela *timeslot* [Foster et al., 2002]

3.2.3 Co-alocação

Em um ambiente com meta-escalonador, após a submissão de um *job* para um RMS, o escalonador, baseado em suas políticas de escalonamento, decide a ordem de execução das tarefas desse *job*. A alocação, quando as tarefas serão executadas, nada mais é do que alocar recursos para as tarefas de acordo com os requisitos da mesma e com o estado do sistema. Co-alocação pode ser definida, portanto, como o uso simultâneo de recursos da grade entre *clusters* e *multi-clusters* [Foster et al., 2002], ou seja, é a alocação de processos em recursos utilizando algum critério de carga. A co-alocação também pode ser vista como uma forma de reduzir o tempo de resposta geral de aplicações pois permite que os processos sejam executados globalmente, não se restringindo a alocações em um único *cluster* [Qin and Bauer, 2009].

Se diversos recursos estão envolvidos na resposta a uma requisição do usuário, o RMS pode, antecipadamente, utilizar-se do mecanismo de co-alocação para reservar esses recursos buscando alcançar uma melhor QoS. Com isso uma demanda de recursos pode ser reservada, antecipadamente, por um dado intervalo de tempo para garantir uma QoS para o usuário sobre algum critério pré-definido [Foster et al., 2002]. A co-alocação, portanto, faz-se necessária em ambientes com suporte à reserva de recursos.

3.2.4 Estudo de Caso do Meta-Escalonador CSF

A abordagem conhecida como CSF *(Community Scheduling Framework)* [Ding et al., 2008] é um meta-escalonador do tipo WSRF, de código aberto, capaz de realizar submissão e gerenciamento de tarefas, reservas antecipadas de recursos e cumprimento de políticas requeridas pelas aplicações. Este projeto já foi parte do Globus Toolkit. A versão atual desse ambiente é conhecida como CSF4.

O CSF disponibiliza ao usuário final uma interface de acesso a recursos virtualizados e aplica políticas globais tanto para os provedores quanto para os consumidores destes recursos.

Algumas características do CSF são:

- Permite integração com diversos escalonadores locais (SGE [Grid Engine, 2011], LSF [Platform Computing, 2011], Condor [Condor, 2011], PBS [PBS, 2011]);
- Mantém independência dos escalonadores locais;
- Possui visão global dos recursos;
- Possui um serviço de reserva de recursos;
- Permite controle de *jobs* em ambientes heterogêneos e escalonadores diferentes.

A figura 3.4 mostra uma visão do meta-escalonador CSF4 interagindo com outros gerenciadores de recursos.

O CSF4 foi utilizado em um projeto desenvolvido pelo Laboratório de Pesquisa em Sistemas Distribuídos da própria instituição (LaPeSD/UFSC), no qual montou-se um ambiente de gestão de submissão, monitoramento e coleta de resultados das aplicações.

O ambiente em questão encarregava-se, utilizando o CSF4, de conectar com as melhores opções de gerenciadores RMS (no caso foram utilizados o SGE e Condor) e fazer com que a aplicação do usuário fosse alocada no melhor ambiente disponível e ocioso possível. Através do meta-escalonador foi possível integrar diferentes tipos de *clusters* heterogêneos, com diferentes políticas de acesso. Ainda, as submissões de aplicações podiam ser feitas tanto local quanto globalmente.

Na figura 3.5 é possível observar um exemplo do ambiente experimental onde foram realizados os testes empíricos.

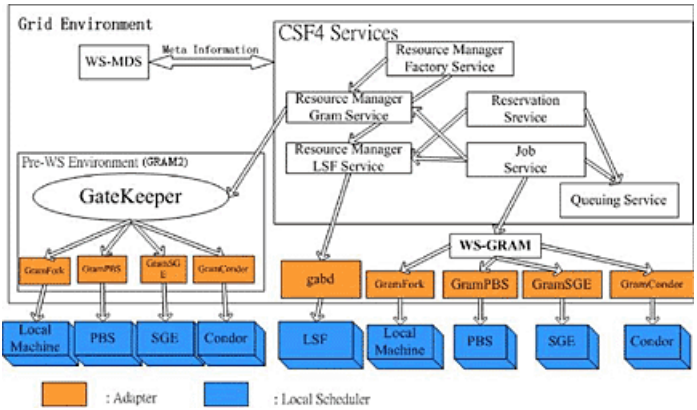


Figura 3.4: Visão do CSF4 e sua interação com gerenciadores de recursos [CSF, 2011]

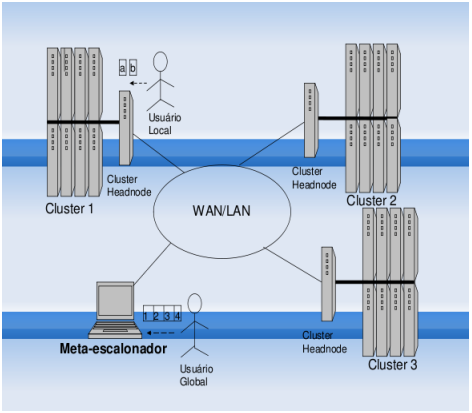


Figura 3.5: Exemplo de estudo de caso da proposta desenvolvida no LaPeSD utilizando o CSF4

3.3 Grades Móveis

Dispositivos móveis evoluíram de uma simples utilização de realização e recebimento de ligações, para equipamentos indispensáveis, dada a sua oferta de recursos cada vez mais sofisticados. Os antigos celula-

res evoluíram para PDAs *(Personal Digital Assistant)*, conhecidos como *palmtops* e Smartphones, com uma integração de diversas funcionalidades como internet 3G *(3rd Generation Mobile Telecommunications)* e GPS *(Global Positioning System)*. Contudo, o expressivo aumento no poder de processamento e capacidade de armazenamento tornou possível que estes dispositivos, através da computação móvel, passassem a ser utilizados em diversos ambientes de computação distribuída, compartilhando dados, recursos e serviços.

Neste contexto evolutivo dos dispositivos móveis, surgiram as grades móveis. Algumas vezes chamadas de Grade Ubíqua (Grade Pervasiva) [Parashar and Pierson, 2007], as grades móveis significam que dispositivos móveis são integrados em grades computacionais tradicionais com o objetivo de compartilhar recursos da grade (CPU, armazenamento, dados, etc.), enquanto os dispositivos móveis propriamente ditos fornecem recursos e serviços para os usuários da grade [Ahuja and Myers, 2006]. Essa combinação de computação móvel com tecnologias de grade consiste em uma habilidade de recriar o poder dos supercomputadores com um acesso através de dispositivos móveis a qualquer lugar e em qualquer momento [Rajachandrasekar et al., 2009].

Atualmente, dispositivos móveis já possuem uma melhora em termos de hardware [Black and Edgar, 2008], entretanto, algumas limitações ainda são encontradas, destacando-se:

- baixa duração de bateria, causada por desconexões e pelo aumento do poder de processamento;
- instabilidade e qualidade da conexão, tornando difícil, por exemplo, obter dados a serem processados no dispositivo;
- heterogeneidade de dispositivos (tipo de hardware, diferentes tamanhos de telas) e seus diferentes tipos de sistemas operacionais; e
- interfaces pouco amigáveis, apesar da evolução das mesmas encontradas em *smartphones*, para a entrada explícita de dados.

A arquitetura das grades móveis, usualmente, inclui três partes [Zeng et al., 2008]: grade *wireless* (grade formada pelos dispositivos móveis), *gateway* (ou um *proxy*) e grade *wired* ou estática (ou seja, a grade computacional propriamente dita). Essencialmente, os recursos em uma grade são dinâmicos, podendo deixar de fazerem parte da grade a qualquer momento. Dado o fator da mobilidade dos dispositivos móveis, grade *wired* e grade *wireless* são diferentes, uma vez que a segunda não é tão confiável dadas as limitações dos dispositivos. Estes dois tipos de grades

são conectados como uma grade lógica única, através de um *gateway* que comunica-se com diferentes servidores a fim de gerenciar a comunicação e computação das grades móveis.

Apesar da arquitetura da grade móvel, existem duas formas de interação da mesma com os ambientes de grades computacionais [Park et al., 2003]:

- **como recursos**: dado um aumento do poder de processamento, memória, entre outros, encontrados nos atuais *smartphones*, os dispositivos móveis podem ser utilizados como provedores de recursos quando não estão sendo utilizados pelos usuários. Entretanto, quando utilizados dessa forma, algumas limitações podem gerar conflitos de dados, por exemplo, após executar uma tarefa, o dispositivo não estar apto a retornar os resultados a tempo devido a uma falha de conexão;

- **como interface de acesso**: apesar das melhoras de hardware dos dispositivos, ainda assim, é uma limitação computacional quando comparados com computadores pessoais. Os dispositivos móveis podem ser utilizados como interface de acesso ao ambiente de grades computacionais. Neste caso, não disponibiliza seus recursos utilização, servindo como um meio móvel para inicializar o uso da grade, monitorar a mesma e coletar resultados, de qualquer lugar e a qualquer momento. Mesmo tratando-se somente de interface de acesso ao meio, as limitações dos dispositivos, ainda não tão visíveis nesta forma de interação, também são fatores a serem contornados. A figura 3.6 mostra um exemplo de como um dispositivo pode ser utilizado como interface de acesso a um ambiente de grade computacional.

Assim, o ambiente de grade pode oferecer o compartilhamento de recursos em larga escala aos usuários móveis para processar as aplicações, fornecendo assim maior flexibilidade, desempenho e confiabilidade para o usuário de dispositivos móveis [Black and Edgar, 2008]. Desta forma, os usuários destes aparelhos conseguem contornar algumas limitações dos dispositivos, visto que podem utilizar os recursos compartilhados na grade.

3.4 Semântica e Ontologias

Os escalonadores utilizam-se de vários mecanismos estratégicos baseados em algoritmos para escolherem o melhor conjunto para executarem uma aplicação. Estes mecanismos são baseados nas descrições dos atributos de uma dada aplicação e nos recursos da grade. Por isso,

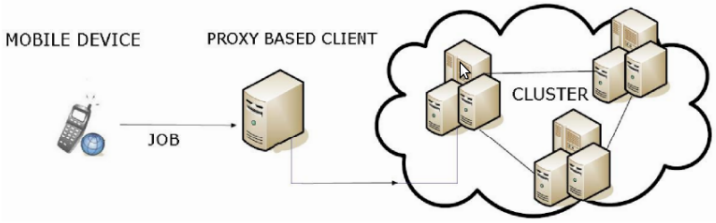


Figura 3.6: Dispositivo móvel sendo utilizado como interface de acesso a um ambiente de grade

a forma como a informação é capturada, armazenada, organizada e disponibilizada é de extrema importância, principalmente em se tratando da heterogeneidade de recursos de uma grade computacional.

Neste sentido, o sucesso da alocação de recursos em uma grade depende, dentre outros fatores, da qualidade das informações disponíveis sobre os artefatos de software e os recursos em um grade [Vidal et al., 2007]. Muitas grades computacionais têm se preocupado com esta qualidade de informação e utilizado ontologias para descrever recursos de forma mais clara. Têm-se, assim, as grades semânticas nas quais informações, recursos e serviços possuem significados descritos em um modelo de dados semântico.

Uma ontologia define um conjunto de primitivas de representação, com as quais se modela um domínio de conhecimento ou discurso [Liu and Zsu, 2009]. O uso de ontologias em grades computacionais além de melhorar a qualidade de informação, tanto dos atributos de um software que será executado na grade quanto dos recursos, também habilita a reutilização de domínios de conhecimento, o compartilhamento da compreensão comum de conceitos de domínios e melhora a interoperabilidade entre recursos de diferentes domínios [Vidal et al., 2007].

O trabalho de [Vidal et al., 2007], que é uma abordagem semântica para integrar a seleção de recursos com os equivalentes requisitos, mostra que o uso de ontologias aumenta o conjunto de possibilidades para execução de uma determinada tarefa, desde que uma dada aplicação possa ser relacionada com diferentes conjuntos de recursos não relacionados, sem o uso da abordagem semântica.

3.5 Trabalhos Correlatos

Esta seção tem como objetivo analisar algumas das abordagens relacionadas ao trabalho proposto, sendo algumas com suporte à reserva antecipada de recursos em ambientes de grades computacionais. Por fim, é apresentado um comparativo entre as abordagens analisadas.

3.5.1 Abordagem de Siddiqui et al

A pesquisa de [Siddiqui et al., 2005], continuação de um trabalho anterior [Siddiqui and Fahringer, 2005], traz um foco detalhado de como é feita a reserva de recursos em ambientes de grades computacionais. No trabalho anterior, propõe-se uma grade computacional, chamada de GridARM *(Askalon's Grid Resource Management System)*, em que o GRM possui funcionalidades de autorização entre diferentes VO's e reserva antecipada de recursos. Além disso, o sistema de gerenciamento de recursos faz descoberta, seleção e alocação através de solicitações de recursos e mecanismos de interação. Para esse fim, utiliza a ontologia para descrição de recursos. A arquitetura foi modularizada em serviços separados para lidar com determinadas funcionalidades da grade, contribuindo para que a mesma seja extensível e escalável, em caso de adição de novos protocolos, necessitando somente da modificação de módulos separados e não do *broker* (que funciona como um *gateway* para o sistema, interagindo com todos os módulos e serviços da grade) como um todo.

Na pesquisa seguinte, propõe-se uma extensão do GridARM para trabalhar com WS-GRAM, que é um WSRF baseado em GRAM (*Grid Resource Allocation and Management*) [Czajkowski et al., 1998], e com o *framework* de autorização do Globus Toolkit 4 (GT4) [Foster, 2006]. Foi implementado um mecanismo adicional ao de autorização do GT4, o PDP (*Policy Decision Point*), possibilitando que a autorização seja feita independentemente do tipo de gerenciador local de recursos, ou LRM (*Local Resource Manager*).

O sistema de reserva de recursos possui similaridades com o trabalho anterior, podendo ser destacadas:

- *timeframe*: composto por tempos de início, duração e final da reserva. Quando uma reserva é realizada, um tempo de duração é atribuído a ela. Se não for realizado um *commit* durante esse intervalo de tempo, a reserva é automaticamente cancelada. Quando aceita, a reserva pode começar a qualquer instante, após um determinado tempo de início, e terminar a qualquer momento, antes de um determinado tempo final;

- *tipo de reserva*: além das reservas instantâneas e reservas antecipa-

das de recursos, os tipos de reserva são relacionados ao número de recursos. Caso a reserva seja de um tipo de recurso apenas, ela é chamada de simples. Sendo de múltiplos recursos, quando o gerenciador de co-alocação é acionado, a reserva será composta;

- *ticket*: o serviço de autorização define regras de acesso ou credenciais para o usuário. Quando uma reserva de recursos é feita, o usuário recebe um *ticket* válido que garante todas as intervenções futuras, tais como, monitoramento, modificação, cancelamento, entre outras.

Para a nova versão, algumas funcionalidades foram adicionadas, destacando-se a *endpoint*. Trata-se de uma referência para o serviço de reserva ou gerenciador de co-alocação, que permite uma interação futura com o sistema. Uma desvantagem dessa abordagem é a possibilidade de escolha somente de número de processadores na requisição de uma reserva.

A figura 3.7 mostra os componentes do sistema de reserva. Uma reserva realizada é independente do gerenciador de recurso local, podendo ser feita em qualquer gerenciador de recurso local.

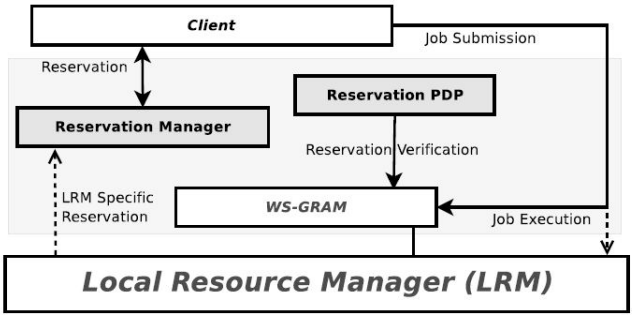


Figura 3.7: Componentes do sistema de reserva [Siddiqui et al., 2005]

Quando um cliente deseja realizar uma reserva, ele envia uma requisição para o gerenciador de reserva com suas restrições. O gerenciador, então, calcula as melhores possibilidades e as retorna para o usuário selecionar ou renegociar a reserva. Quando a reserva é confirmada, o usuário recebe um *ticket* que permite a interação com o sistema.

Na figura 3.8, a reserva antecipada de recursos possui um ciclo de vida que pode ser observado como uma sequência de estados. No referido trabalho, quando ocorre uma troca de estado, o cliente recebe uma notificação, podendo cancelar a sua reserva a qualquer momento ou, ainda, ser finalizada no caso de alguma violação ocorrer ou o cliente não utilizar os recursos em um determinado tempo.

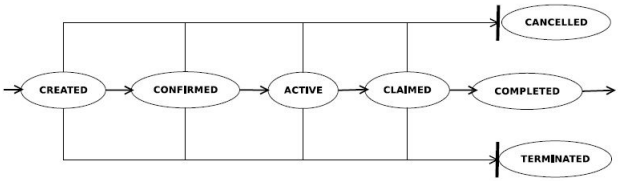


Figura 3.8: Possíveis estados de uma reserva [Siddiqui et al., 2005]

O mecanismo de negociação para reserva de recursos proposto, traz a possibilidade de os usuários ajustarem suas restrições e QoS para a execução de suas aplicações.

O trabalho propõe um mecanismo de reserva antecipada de recursos e co-alocação em que usuários podem negociar com provedores de recursos. O mecanismo ainda permite a reserva de múltiplos recursos em organizações virtuais, independente dos gerenciadores locais de recursos.

3.5.2 Abordagem Takefusa et al

Em um trabalho similar [Takefusa et al., 2007], propõe-se o GridARS (*Grid Advance Reservation-based System Framework*), que é um grid com *framework* de co-alocação para recursos distribuídos e realiza reservas antecipadas de recursos sobre o protocolo WSRF.

A arquitetura do GridARS consiste em um escalonador global de recursos - GRS (*Global Resource Scheduler*) - e gerenciadores de recursos - RM (escalonadores locais) - que, automaticamente, co-alocam os recursos através de WSRF. Isto habilita processos simultâneos de reserva de recursos utilizando um protocolo hierárquico de *commit* de duas fases entre os usuários-GRS e GRS-RMs.

Igualmente à pesquisa de [Siddiqui et al., 2005], o GridARS também foi desenvolvido utilizando o GT4. O objetivo buscado com a utilização do GT4 foi utilizar a estrutura GSI (*Grid Security Infrastructure*) que é encarregada da autenticação e autorização dos usuários. Por utilizar o GT4, o GridARS fornece uma interface para diferentes tipos de

escalonadores.

Os principais componentes são: o *GridARS-co-escalonador*, que seleciona os recursos adequados e os co-aloca, com base em transações distribuídas; e o *GridARS-WSRF*, que é um módulo de interface que realiza o *commit* de duas fases, chamado de 2PC (*Two-phase commit*) para reserva antecipada de recursos sobre o protocolo WSRF. O 2PC permite que escalonadores globais possam alocar recursos distribuídos simultaneamente, baseando-se em transações distribuídas.

A arquitetura do *framework* GridARS pode ser visualizada na figura 3.9.

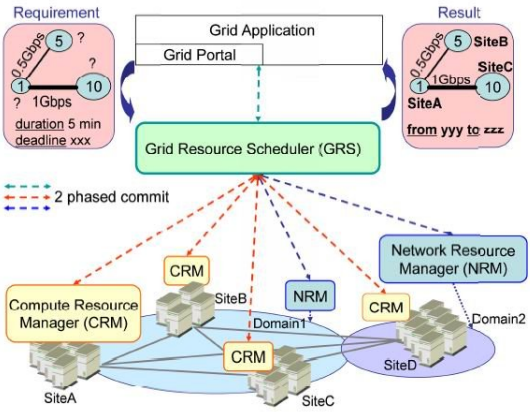


Figura 3.9: Visão geral da arquitetura do GridARS [Takefusa et al., 2007]

Um usuário envia um pedido de recurso e um tempo de reserva para o GRS, e o mesmo co-aloca os recursos adequados em coordenação com os RMs relacionados. Um módulo principal, então, retorna uma resposta para o cliente de forma não-bloqueante e também envia a requisição para um escalonador de recursos ou para o GridARS-co-escalonador. A importância de utilizar-se o mecanismo não-bloqueante é que este evita problemas de desconexões e também habilita, de uma forma simples, a recuperação de processos que possam ter falhado. O módulo principal, também, utiliza o método de *polling* para checar o estado do gerenciador de reserva de recursos.

A proposta apresentada também trabalha com um tempo de duração para que os recursos sejam utilizados na grade. Com o tempo ultra-

passado, a reserva é cancelada e os recursos liberados.

A figura 3.10 mostra a sequência de protocolos do processo de reserva antecipada. Nela pode-se visualizar o mecanismo de *polling* utilizado e as requisições sendo passadas do usuário para o GRS e, após, para os RMs.

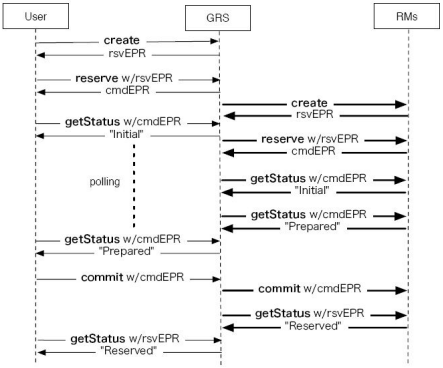


Figura 3.10: Sequência de protocolos de um processo de Reserva Antecipada [Takefusa et al., 2007]

3.5.3 Abordagem de Rossetto et al

Na pesquisa de [Rossetto et al., 2007] apresenta-se um *framework*, o SuMMIT (*Submission, Monitoring and Management of Interactions of Tasks*), para a submissão, monitoramento e coordenação da execução de *workflows* em ambientes de grades computacionais. Toda a interação com o ambiente é feita através de dispositivos móveis, sendo esses utilizados como interface de acesso ao ambiente. A proposta apresenta uma solução genérica e de granularidade fina para definição de recursos da grade computacional usados em cada estágio da execução de aplicação em um modo coordenado e automatizado.

O trabalho traz uma particularidade que é a possibilidade de adaptação do fluxo de execução buscando garantir a consistência da aplicação. Esse fluxo de execução é tratado de forma personalizada em caso de uma desconexão do dispositivo, seja por falha na rede ou término da bateria.

A arquitetura da proposta é apresentada na figura 3.11 em que é possível visualizar a interação dos componentes. A arquitetura é composta de três componentes principais: *Workflow Manager*, *Agent* e *Mobile GUI*.

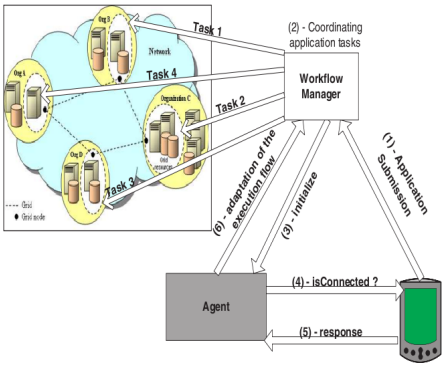


Figura 3.11: Interação dos componentes da arquitetura SuMMIT [Rossetto et al., 2007]

O *Workflow Manager* é o módulo que gerencia e processa as requisições vindas dos dispositivos móveis de forma transparente ao usuário. Ele provê um meio automatizado de submissão de tarefas para a grade e realiza uma coleta de informações a respeito da execução dessas tarefas no ambiente distribuído, sem a interação do usuário, contribuindo, assim, para uma redução no consumo de bateria dos dispositivos móveis.

Tal arquitetura ainda provê um mecanismo de tolerância a falhas que trata, especialmente, da ocorrência de desconexões dos dispositivos móveis. A verificação de uma possível desconexão é feita pelo módulo *Agent* que em um determinado intervalo de tempo, pré-definido, verifica se o dispositivo está conectado. A manipulação de falhas consiste na detecção da falha e adaptação da execução da aplicação no ambiente e também é realizada pelo módulo *Agent*. Através de opções definidas pelo usuário móvel, o *Agent*, em caso de uma desconexão, pode adaptar a execução e continuar, abortar, ou pará-la, bem como dar continuidade quando voltar a receber resposta do dispositivo. Estas ações são realizadas conforme a existência de dependências da aplicação do usuário. Através deste módulo garante-se a consistência de aplicação.

Todas as interações com a grade computacional ocorrem por intermédio de uma interface, *Mobile GUI*. Esta interface, além de possibilitar a submissão de *workflows*, permite a visualização de resultados finais e parciais da aplicação de uma forma otimizada. Tal otimização é obtida, uma vez que, somente as partes relevantes do arquivo que contém os re-

sultados são carregados para o dispositivo móvel. Desta forma, o usuário pode monitorar a execução da aplicação de forma adaptada para cada tipo de dispositivo móvel e seus diferentes tamanhos de tela.

3.5.4 Abordagem de Silva e Dantas

A bordagem de [Silva and Dantas, 2007] propõe um mecanismo de *matching* semântico de recursos de grades computacionais, baseado na integração semântica de múltiplas ontologias. Esta integração foi alcançada desenvolvendo-se uma ontologia de referência que tem como objetivo servir de base para que os desenvolvedores das diferentes ontologias de recursos das organizações virtuais possam estabelecer relações de equivalência semântica.

O trabalho também considera interações humanas com o objetivo de construir um conhecimento mais lógico na criação de requisições, visto que sistemas inteiramente automáticos não são capazes de reconhecer todas as possíveis relações entre as diferentes ontologias. Para integrar estas últimas foi desenvolvida uma ontologia de referência (OR), que pode ser visualizada na figura 3.12. Também foi criada uma ontologia de requisição (QO), que auxilia o integrador de ontologias para que o usuário final possa fazer consultas de recursos no ambiente.

O sistema considera somente as ontologias previamente inscritas pelos provedores, ou seja, de forma estática, sem considerar que os recursos possam estar ocupados no momento.

A figura 3.13 mostra a arquitetura de *matching* de recursos em ambientes de grade.

Na arquitetura do sistema é possível visualizar os três principais componentes:

- *Ontologies Integration Portal*: este componente representa a interface usada pela organização virtual para realizar a integração semântica de sua própria ontologia com o sistema de *matching*;
- *Information Providers*: é o componente responsável por coletar as informações dos recursos de cada organização virtual;
- *Matchmaker*: a operação de *matching* semântico é a função principal deste componente. O serviço interage com duas bases de informação: a primeira contém as relações de equivalência e as informações dos recursos (descritas por ontologias) publicadas pelas VOs. A outra base contém a OR: a ontologia de pedidos e dois tipos de regras, quais sejam: verificação de consistência e ampliação semântica da consulta, baseado no conhecimento de domínio modelado e nas informações definidas na OR.

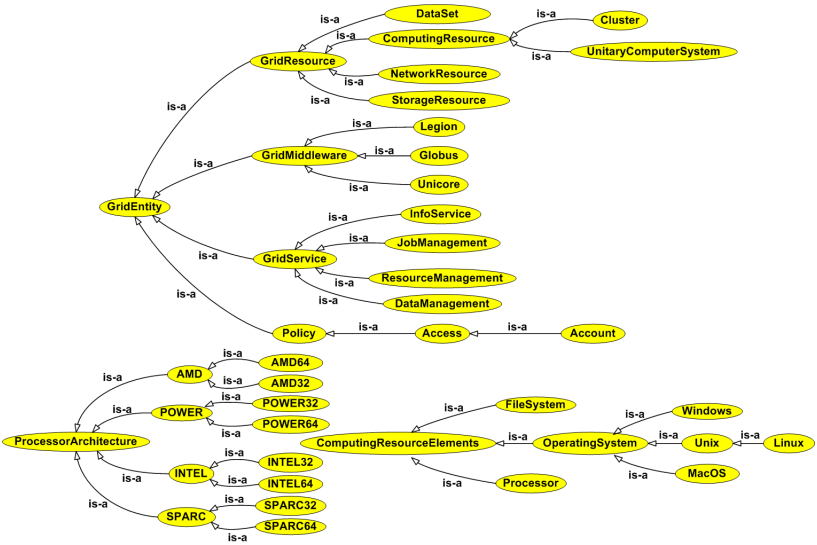


Figura 3.12: Ontologia de referencia da abordagem Silva e Dantas [Silva and Dantas, 2007]

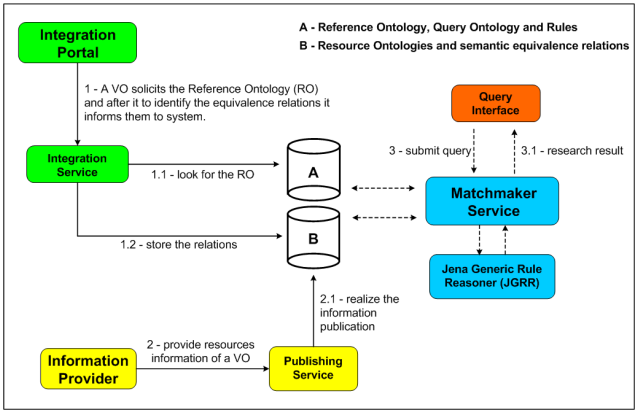


Figura 3.13: Arquitetura de *matching* de recursos [Silva and Dantas, 2007]

Por meio da utilização deste sistema, o usuário consegue refinar melhor sua consulta, como por exemplo, selecionando o tipo de sistema operacional a ser utilizado, versão do mesmo, arquitetura, entre outros parâmetros. Após uma consulta para determinada tarefa, o usuário recebe um retorno, com os IPs dos possíveis recursos para executar a tarefa requisitada.

3.5.5 Abordagem de Vahdat-Nejad et al

A pesquisa de [Vahdat-Nejad et al., 2007] propõe um algoritmo *fuzzy* para o escalonamento global de *jobs* em ambientes de grades com configurações de *multi-clusters*. Através de um escalonador global, *clusters* são selecionados para serem utilizados, baseados em suas características e também nas características dos *jobs*. Esta pesquisa utiliza aplicações paralelas rígidas, com número constante de tarefas executadas em paralelo. O selecionador de recurso leva em consideração as seguintes características da aplicação:

- número de processos paralelos a serem executados;
- taxa de comunicação da aplicação.

A seleção global do *cluster* passa a ser feita levando em conta estas características da aplicação e considerando as cargas atuais de comunicação dos *clusters*, a fim de buscar o mais adequado para executar estes *jobs*. Para tal funcionalidade, foi necessário priorizar os *clusters* disponíveis. A lógica *fuzzy* foi utilizada, portanto, como um controle que contém as regras para realizar o *match* entre as especificações requeridas de recursos por parte dos *jobs* com os recursos disponíveis no *cluster*. O objetivo foi incluir a carga de rede de um *cluster* como um recurso do mesmo, e os requisitos de comunicação de um *job* como especificações de recurso no processo de priorização. Utilizando a lógica *fuzzy* é possível raciocinar sobre estes parâmetros de forma qualitativa e, ao mesmo tempo, melhorar as decisões de escalonamento.

Após o escalonamento global, através de seu mecanismo de seleção baseado nas características da aplicação e, também, da carga atual de comunicação de cada recurso, o escalonador local do *cluster* decide em qual máquina deve ser executado o *job* utilizando a política de FIFO (*first-in first-out*).

A caracterização do estado de um dado *cluster* considera o número de CPUs, além da carga da rede. Quando um *job* chega ao escalonador o mesmo é acionado para destinar este *job* a um *cluster*. Neste momento são criados dois pesos para cada *cluster*. O primeiro peso, W1, determina

a quantidade de máquinas disponíveis no *cluster* em relação ao número de tarefas do *job*. O segundo peso, W2, considera o requisito de comunicação do *job* em relação à carga de rede do *cluster*. Através desses pesos e da definição de coeficientes para os mesmos, chegou-se à formula de cálculo de prioridade:

$$Prioridade = 0.7W1 + 0.3W2 \tag{3.1}$$

A lógica *fuzzy* é utilizada para calcular o peso W2. São utilizados dois parâmetros de entrada para as regras da lógica: frequência de comunicação das tarefas paralelas e disponibilidade da rede de interconexão do *cluster*. A saída gerada é o peso W2, conforme pode-se observar na tabela 3.1

Tabela 3.1: Regras para mapeamento de parâmetros em de [Vahdat-Nejad et al., 2007]

Ratio/Available BW	Low	Medium	High
Low	Moderate	Large	Large
Medium	Small	Moderate	Large
High	Very small	Small	Large

Através de testes foi possível comprovar uma melhora significativa na redução do tempo de resposta utilizando-se a abordagem proposta, quando comparada a um algoritmo de *best-fit*.

3.5.6 Considerações sobre os trabalhos correlatos

É possível observar na tabela 3.2 uma análise comparativa entre cada um dos trabalhos relacionados que foram citados em cada subseção.

A primeira pesquisa, [Siddiqui et al., 2005], mostra um *middleware* de grade distribuído que provê um mecanismo de co-alocação e reserva antecipada de recursos. Este trabalho foi uma continuação de um trabalho anterior que já provia um mecanismo de reserva de recursos com auxílio de uma ontologia para descrição dos mesmos. Uma vantagem observada é a possibilidade de reserva antecipada de recursos em diferentes VOs, independentemente dos tipos de gerenciadores locais (LRM). Isso foi conquistado através do mecanismo PDP que é integrado ao WS-GRAM. Entretanto, dentre os recursos a serem reservados, cita-se: número de processadores, restrições quanto à arquitetura e sistema operacional como

parâmetros de reserva. Largura de banda e memória não são consideradas. Além disso, características da aplicação não são consideradas nesta abordagem, levando-se em conta somente a descrição dos recursos.

A proposta de [Takefusa et al., 2007] apresenta um *framework*, intitulado GridARS, de co-alocação e de reserva antecipada de recursos. O trabalho foi desenvolvido utilizando o GT4 e, com isso, tem como ponto positivo a interface de comunicação com outros escalonadores. O GridARS também utiliza o WSRF/GSI e o protocolo 2PC que permite um processo genérico e seguro de reserva antecipada de recursos baseado em transações distribuídas. Por outro lado, a descrição de recursos não é feita por meio de uma abordagem semântica ou ontologias, sendo este um dos pontos negativos da proposta, tendo em vista que o uso dessa descrição melhora o conjunto de possibilidades de uma seleção, conforme já foi demonstrado por [Vidal et al., 2007]. Além disso, a proposta não considera as características da aplicação. E os recursos que podem ser reservados são: número de processadores e largura de banda.

O trabalho explicado em [Rossetto et al., 2007] não apresenta o mecanismo de reserva antecipada de recursos, mas possui a vantagem (para os usuários de ambientes de grades computacionais) de ter uma interface de acesso, através de dispositivos móveis, que torna possível a submissão, monitoramento e coordenação da execução de *workflows* nestes ambientes distribuídos. A utilização dos dispositivos torna possível o acesso à grade *anytime and anywhere* permitindo que os usuários utilizem melhor o ambiente sem que estejam necessariamente próximos de um computador pessoal. Ainda, este trabalho apresenta um mecanismo de tratamento a desconexões, usualmente encontradas neste tipo de ambiente.

O trabalho já citado de [Silva and Dantas, 2007] mostra um mecanismo de seleção de recursos computacionais que faz integração semântica realizando um *matching* de diferentes descrições das ontologias de diferentes VOs. O trabalho tem por objetivo facilitar o processo de seleção de recursos, fornecendo ao usuário uma visão homogênea do ambiente. Assim, é possível selecionar uma grande variedade de recursos, tal como tipo de processador, tipo de sistema operacional, quantidade mínima de processadores, memória, entre outros. Um ponto negativo da abordagem é que a seleção é feita de forma estática, não considerando a situação dos recursos no momento. Também não considera as características da aplicação.

O trabalho apresentado na última subseção, de [Vahdat-Nejad et al., 2007], traz uma abordagem que visa otimizar a seleção de recursos computacionais considerando características da aplicação e o estado dinâmico do *cluster*, em relação ao uso de CPUs e interconexão de rede. Caso o

usuário tenha um conhecimento mais aprofundado sobre sua aplicação pode conseguir tirar proveito desta abordagem e ter uma seleção que permita tirar maior proveito dos recursos. Entretanto, o trabalho considera somente a carga da rede e não a tecnologia de rede e o seu desempenho.

Tabela 3.2: Comparação entre trabalhos correlatos

	[Siddiqui et al., 2005]	[Takefusa et al., 2007]	[Rossetto et al., 2007]	[Silva and Dantas, 2007]	[Vahdat-Nejad et al., 2007]
Reserva Recursos	Sim	Sim	Não	Não	Não
Descrição Características Aplicação	Não	Não	Não	Não	Sim
Interface Móvel	Não	Não	Sim	Não	Não
Múltiplas VOs	Sim	Sim	Não	Sim	Sim
Ontologia	Sim	Não	Não	Sim	Não
Tipo de Recurso para Reserva	nº de processadores, SO	nº de processadores, banda	Não	Não	Não

Capítulo 4

Arquitetura para Reserva de Recursos através de Dispositivos Móveis

Conforme apresentado nos capítulos anteriores, alguns ambientes de grades computacionais já estão se preocupando em fornecer um suporte à reserva antecipada de recursos com a finalidade de melhorar a QoS. Entretanto, não se visualizam trabalhos que permitam que os usuários utilizem dispositivos móveis para interagirem com esses ambientes, no que tange ao planejamento de um futuro uso de recursos. Ainda, os modelos que permitem reserva de recursos, impõem ao usuário, ao requisitarem uma reserva, o fornecimento de informações a nível de hardware e software. Desta forma, o usuário necessita conhecer as características dos recursos, pois estas influenciam diretamente no desempenho da aplicação.

Portanto, propõe-se um mecanismo que visa contornar esses problemas, permitindo que o usuário, através de uma interface móvel, descreva somente as características da aplicação, deixando o meta-escalonador encarregado da tarefa de realizar a reserva do recurso mais adequado às características, no caso, o nível de granularidade da aplicação. Portanto, o meta-escalonador fica encarregado de achar o *cluster* com a rede de interconexão adequada à aplicação

Este capítulo tem por objetivo apresentar a solução desenvolvida. Inicialmente, é mostrada uma breve introdução que descreve um pouco mais sobre o problema, bem como é apresentada a proposta. A seguir, são expostos os aspectos relacionados à Qualidade de Experiência do usuário, na seção 4.2. Na seção 4.3 são apresentadas as definições a respeito das características da aplicação. Na seção 4.4 é apresentada a arquitetura desenvolvida. Por fim, são apresentadas algumas considerações sobre a abordagem proposta.

4.1 Introdução

Como mencionado anteriormente, ambientes de grades utilizam diferentes recursos computacionais distribuídos geograficamente, porém, coordenados e disponíveis, através de políticas de gerenciamento, para

os usuários. Além disso, o ambiente que utiliza um meta-escalonador torna possível, através de um escalonamento global, a integração de diversos nodos computacionais ou *clusters* gerenciados por seus RMS. O meta-escalonador é uma camada de mais alto nível que procura atender às requisições das aplicações sem os limites locais de recursos e, portanto, permitindo que recursos sejam acessados e combinados afim de alcançar um objetivo computacional.

Por outro lado, diversos desses ambientes, quando utilizados e, não havendo recursos disponíveis no tempo de submissão de uma aplicação, colocam as tarefas em uma fila do gerenciador local. Desta forma, não existem garantias sobre quando este conjunto de tarefas de um determinado *job* será executado, podendo causar problemas, por exemplo, com aplicações de tempo crítico. Faz-se necessária a utilização de reserva antecipada de recursos, que busca resolver este problema permitindo que usuários planejem a execução de suas aplicações, garantindo um conjunto mínimo de recursos reservados em um dado tempo.

A utilização de ontologias para descrição de recursos em ambientes de grades computacionais, como apresentado em [Vidal et al., 2007], aumenta o conjunto de possibilidades para execução de uma determinada tarefa. Entretanto, também é importante observar as características da aplicação que será executada neste ambiente, a fim de conseguir um melhor *matching* entre recurso e aplicação.

Neste contexto, o trabalho apresenta uma arquitetura para reserva antecipada de recursos baseada nas características da aplicação e utilizando como interface de acesso ao ambiente, os dispositivos móveis. O sistema proposto visa melhorar o aproveitamento do ambiente através de um melhor planejamento do uso de recursos por meio da reserva, considerando-se particularidades da aplicação, neste caso, o grau de granularidade da mesma na hora da seleção do recurso a ser reservado. Ontologias são utilizadas para uma melhor descrição da aplicação e recursos do sistema. Objetiva-se, assim, melhorar o nível de utilização dos recursos reservados e o tempo de resposta total do sistema. Ainda, através de uma interface móvel, busca-se explorar a mobilidade oferecida pelos dispositivos móveis e prover uma interação com a grade *anytime and anywhere*. Por fim, busca-se a integração com alguns trabalhos desenvolvidos previamente pelo grupo de pesquisa do laboratório LaPeSD/UFSC, realizando uma adaptação de alguns mecanismos.

O desenvolvimento da abordagem passou por uma série de etapas. Inicialmente, como mencionando no estudo de caso, foi utilizado o meta-escalonador CSF, através de um projeto do laboratório LaPeSD, para uma gestão de submissão, monitoramento e coleta de resultados de aplicações.

O meta-escalonador foi utilizado para conectar diferentes gerenciadores de recursos locais (no projeto foram utilizados o Condor e SGE) fazendo com que a aplicação fosse alocada ao ambiente mais ocioso possível, incluindo ambientes Windows. Ainda, tinha-se um interesse em um suporte a reserva de recursos para que os mesmos fossem utilizados para executar as aplicações fora do horário comercial.

O projeto em questão tratava como aplicação alguns simuladores, no entanto, o CSF não tornava possível uma melhor seleção de recursos considerando as características destes simuladores. A seleção era baseada na disponibilidade de recursos. Outro ponto restritivo do CSF é o fato de possuir um conjunto limitado de funcionalidades de co-escalonamento. O CSF, conecta-se com os gerenciadores de recursos locais através do GRAM. Como também pôde-se observar, apesar de alguns destes gerenciadores possuírem suporte a MPI e PVM, somente com alguns gerenciadores foi possível submeter aplicações MPI, e não PVM, através do CSF. Mesmo os gerenciadores locais possuindo suporte a estas aplicações paralelas, o CSF ainda não possuía tal suporte implementado.

Ainda, o mecanismo de reserva de recursos observado neste ambiente, tratava somente de reservas com o escalonador LSF. O mecanismo não pôde ser efetivamente testado por falta da licença do LSF, porém, observando as configurações, aparentemente o CSF não trata o estado atual de um recurso sob a gerência do LSF. Por último, observou-se que o CSF não possui uma interface de acesso amigável. Para fins de utilização do meta-escalonador empregou-se um *portlet* do GridSphere [Novotny et al., 2004] já desenvolvido para o CSF. Por mais que se tenha deixado de usar o ambiente através de uma interface de linha de comando, o portal Web ainda deixou muito a desejar, não sendo trivial a utilização da grade computacional. Por todos esses motivos optou-se por não utilizar o CSF no trabalho proposto.

Em uma segunda etapa foi desenvolvido o mecanismo de reserva antecipada de recursos através de dispositivos móveis. Com isso, permitiu-se que um usuário pudesse planejar um futuro uso de recursos a qualquer momento e em qualquer lugar. Ainda, nesta etapa, buscou-se uma interface amigável de acesso ao ambiente e que buscasse explorar o conhecimento do usuário sobre os recursos a serem utilizados pela aplicação, a fim de realizar uma reserva com os recursos mais apropriados. Esse mecanismo foi desenvolvido e validado em um simulador de ambientes de grades computacionais.

Em uma terceira etapa, trabalhou-se em uma adaptação do mecanismo de seleção utilizado em [Grumiche et al., 2010], o qual seleciona recursos baseados nas características da aplicação, deixando o meta-

escalonador tomar decisões baseadas nas características da aplicação. Desta forma, os usuários deixam de se preocupar com características específicas de recursos. A fim de descrever as características da aplicação, neste trabalho foi criada uma ontologia que estendeu a ontologia de referência de descrição de recursos proposta por [Silva and Dantas, 2007].

Por fim, foi desenvolvido um ambiente de execução, utilizando um simulador como ambiente de teste para o protótipo.

4.2 Qualidade de Experiência

Provedores de serviços frequentemente usam parâmetros (vazão, taxa de perda e atraso) para especificar QoS em redes, como medida de performance do serviço. Em contraste, usuários enxergam a performance de um serviço mais subjetivamente e não em termos técnicos, não levando em consideração esses parâmetros da rede [Shaikh et al., 2010]. Essa percepção do usuário pode ser considerada a Qualidade de Experiência em relação às perspectivas da rede.

Qualidade de Experiência, do inglês QoE (*Quality of Experience*), pode ser definida com as características de sensação, percepção e opinião de pessoas que interagem com seus ambientes [Corrie et al., 2003]. Essas características podem ser agradáveis e divertidas, como também podem ser desagradáveis e frustrantes. É, portanto, o resultado final da interação do usuário com a tecnologia.

QoE, desse modo, é a forma como o usuário se sente sobre uma aplicação ou serviço que foi entregue, em relação às suas expectativas e requerimentos [Corrie et al., 2003].

Existem tentativas sobre como definir melhor uma QoE de um usuário. O assunto ainda é bastante debatido, pois esta qualidade de experiencia é muito subjetiva e difícil de ser medida.

Nesse contexto, QoE pode ter diferentes significados para diferentes tipos de aplicação. Portanto, o trabalho proposto não objetiva entrar em detalhes e métricas de QoE. Procura, por outro lado, melhorar a experiência do usuário através de um ambiente que leva em consideração as características da aplicação como critério de seleção para a reserva de recursos.

4.3 Características da Aplicação

Usualmente, grades computacionais possuem uma série de recursos que podem ser descritos através da utilização de ontologias. Na maioria desses ambientes é o usuário que deve possuir o conhecimento sobre as características de hardware e software de um recurso e, portanto, procurar o recurso mais indicado para a sua aplicação. Nestes casos, o conhecimento sobre o ambiente distribuído é primordial. Caso o usuário

tenha esse conhecimento e ainda possua uma experiência a respeito da aplicação, isso pode ser um ponto positivo na utilização destes ambientes. Para que este cenário seja possível, porém, o ambiente deve permitir que o usuário forneça em sua requisição as informações referentes à sua aplicação.

4.3.1 Tipos de Características

O trabalho de [Grumiche et al., 2010] propôs uma classificação das características da aplicação. Essa classificação foi baseada no conceito de padrões de projetos, em que cada um dos padrões é baseado em três grupos: criacionais, estruturais e comportamentais. Além de utilizar o conceito de padrões de projetos, a classificação também foi baseada em diagramas UML (*Unified Modeling Language*), que são classificados com base no que permitem representar: estrutural (representa características estáticas) e comportamental (representa características dinâmicas).

A classificação proposta em [Grumiche et al., 2010] classifica as características das aplicações em:

- *Características estruturais*: emergem da composição do corpo da aplicação. Essas características normalmente restringem quais recursos podem executá-la. São exemplos: tipo de sistema operacional ou tipo de arquitetura necessária para executar essa aplicação;

- *Características comportamentais*: são aquelas observáveis durante a execução da aplicação. Tais características estão relacionadas ao modo em que os recursos de processamento e comunicação são utilizados para a resolução do problema. A restrição de utilização de recursos por essas características também pode ser observada, por exemplo, na quantidade de memória RAM ou no número de processadores disponíveis para executar essa aplicação. Ainda, o nível de granularidade é considerado uma característica comportamental da aplicação.

O controle de granularidade de uma aplicação pode levar diretamente a um aumento de desempenho quando realizada pelo programador. Porém adiciona uma carga extra ao programador, exigindo conhecimento, tanto da arquitetura quanto do comportamento do algoritmo, assim como, reduz a clareza, reusabilidade e portabilidade do código. Para aplicações específicas, com conhecimento prévio da arquitetura, no entanto, o ganho de desempenho pode compensar os pontos negativos.

Existem muitos esforços que tem por objetivo fornecer ferramentas para auxiliar os usuários a melhorar o nível de granularidade em uma

aplicação. Um destes esforços é o FORGE 90 [Levesque, 1990], que é um ambiente de programação paralela que oferece uma abordagem de desenvolvimento portável e programas bem estruturados em Fortran [Press et al., 1996], os quais podem ser convertidos eficientemente para qualquer tipo de arquitetura paralela disponível. O FORGE 90 cria uma base de dados para o programa e, com isso, consegue realizar análises globais e coletas de estatísticas em tempo de execução sequencial, com o fim de determinar pontos de paralelização, alcançando um bom nível de granularidade para a aplicação.

Um outro trabalho semelhante é o SCOOPP [Sobral and Proença, 2001] que tem por escopo prover uma escalabilidade dinâmica e eficiente de aplicações paralelas orientadas a objetos. Tal escalabilidade se dá através de diferentes plataformas, empacotando a granularidade e mensagens, sem modificação de código. O controle de granularidade é feito em duas etapas: em tempo de compilação, o compilador e/ou o programador especificam o número de grãos finos dos objetos paralelos; em tempo de execução, os objetos são empacotados em grãos grossos, de acordo com o comportamento da aplicação ou plataforma, baseado em questões de segurança e desempenho. O SCOOPP utiliza, portanto, uma adaptação estática e dinâmica. Através destas metodologias é possível aumentar o tamanho do grão, obtendo uma melhora de *speed up* na mesma ordem de magnitude de abordagem. Estas metodologias baseiam-se unicamente em nível de programação, melhorando a reutilização de código para diferentes plataformas.

O nível de granularidade de uma aplicação pode ser definido como a razão entre a quantidade de etapas computacionais pela quantidade de etapas de comunicação durante o tempo de execução da aplicação. Pode ser classificada da seguinte forma [Pinto et al., 2008]:

- *fina*: quando a aplicação realiza mais comunicação do que computação, ou seja, o desempenho da interconexão do *cluster* é fator determinante;
- *grossa*: ao contrário da granularidade *fina*, é caracterizada pela grande quantidade de computação quando comparada com a comunicação. Neste caso, a maior capacidade de processamento de um *cluster* deve ser considerada;
- *embaraçosamente paralela*: são as aplicações distribuídas que fazem pouca ou nenhuma comunicação.

O trabalho proposto, entretanto, não possui seu foco em determinar o nível de granularidade de uma aplicação, considerando que as aplica-

ções já possuam este nível otimizado, em caso de utilização dos recursos disponíveis.

Considerando a granularidade da aplicação como característica, uma ontologia (*AppFeature*) foi criada, baseando-se no trabalho de [Silva and Dantas, 2007]. A mesma pode ser visualizada na figura 4.1.

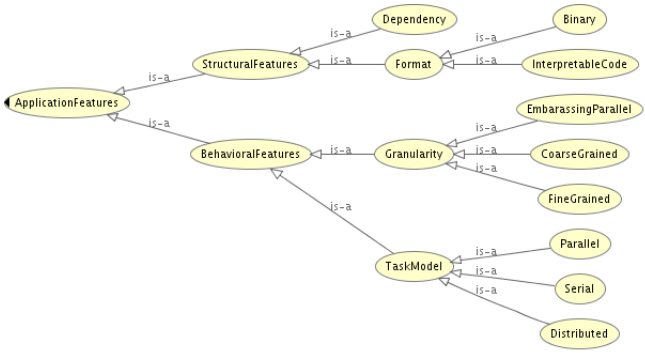


Figura 4.1: Ontologia que descreve as características de uma aplicação [Grumiche et al., 2010]

4.4 Arquitetura Proposta

De uma forma geral, pode-se visualizar os principais componentes da arquitetura proposta na figura 4.2, na qual, o usuário do sistema interage com o mesmo através de um dispositivo móvel. Através desse dispositivo - que, neste caso, é considerado como parte da grade, fornecendo uma interface de acesso interativa ao ambiente -, o usuário insere informações relacionadas à reserva de recursos que deseja realizar. Essa reserva pode ser antecipada ou imediata, neste segundo caso, podendo submeter o *job* no tempo atual.

A arquitetura proposta considera um ambiente de grade computacional com configuração de *multi-clusters*. Considera-se, portanto, que os *clusters* pertencentes a este ambiente são dedicados e, diferentemente de um ambiente de grade formado por computadores pessoais inseridos como nós de execução, possuem uma maior disponibilidade. O usuário tem, então, a possibilidade de reservar em um determinado tempo, um conjunto de recursos que satisfaça as suas necessidades.

Considera-se, ainda, a reserva antecipada de recursos baseada no número de processadores necessários para executar o *job* a ser submetido

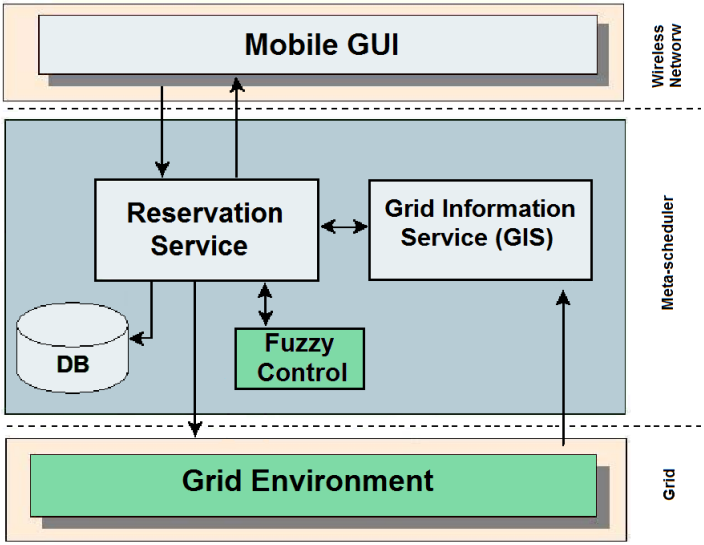


Figura 4.2: Visão Geral dos Componentes do Sistema

durante o período da reserva levando-se em consideração, no entanto, características da aplicação como critério de seleção dos recursos. Com o fim de atingir este objetivo a arquitetura é composta por uma interface de co-reserva de recursos globais, ou seja, de um meta-escalonador, responsável pela negociação da reserva de recursos com os gerenciadores locais dos *clusters* que compõem o ambiente da grade. Porém, a co-reserva considera somente recursos pertencentes a um mesmo *cluster*, ou seja, não é possível em uma mesma reserva alocar recursos de diferentes *clusters* a fim de satisfazer a requisição. Optou-se por não permitir esse tipo de reserva pois a qualidade de uma reserva é diretamente relacionada com o desempenho da rede de interconexão do *cluster*, podendo, portanto, comprometer o tempo de comunicação da aplicação.

A interface móvel de acesso (*Mobile GUI*), como pode-se visualizar na figura 4.2, comunica-se com meta-escalonador que possui o módulo que trata as reservas (*Reservation Service*), o módulo que interage com o ambiente, fornecendo informações (o GIS - *Grid Information Service*) e o módulo que calcula os níveis de adequação de cada *cluster* utilizando a lógica *fuzzy* (*Fuzzy Control*) proposta por [Grumiche et al., 2010].

De posse das informações referentes a reserva, o módulo de *Reservation Service* contata o *Fuzzy Control* e recebe de volta os níveis de adequação de cada *cluster* com relação as características da aplicação. Assim, com os níveis de cada recurso, o *Reservation Service* solicita informações do ambiente através do *GIS* para saber a disponibilidade e o *status* dos recursos do ambiente. Se há recursos disponíveis que atendam às requisições do usuário (tempo solicitado, número de processadores, etc.), o módulo de reserva aloca os recursos e salva essas informações (*DB*) para futuras interações entre o usuário e o sistema, possibilitando ao usuário a verificação do *status* de uma reserva, o cancelamento da mesma ou o monitoramento da execução do *job*. Portanto, observa-se nessa figura que o meta-escalonador é formado pelos componentes *Reservation Service*, *GIS*, *DB* e *Fuzzy Control*, sendo os três primeiros desenvolvidos pelo trabalho proposto e os último adaptado do trabalho de [Grumiche et al., 2010].

4.4.1 Interface Móvel de Acesso

A GUI (*Graphical User Interface*) móvel é responsável por fornecer a interface de interação com o ambiente. Através dela é possível realizar reserva antecipada de recursos no ambiente distribuído permitindo que o usuário utilize ou não o seu conhecimento a respeito da sua aplicação.

Caso o usuário seja experiente e saiba como sua aplicação se comporta e o que é necessário para se conseguir um melhor proveito dos recursos reservados, o mesmo pode fornecer essas informações na hora em que realizar a reserva. Por outro lado, se o usuário não tem um conhecimento apurado ou não quer se preocupar com estas informações, ou até mesmo, se o tempo de resposta não é tão importante, o mesmo fornece somente informações básicas sobre a reserva.

4.4.1.1 Funcionalidades

As funcionalidades que podem ser encontradas na GUI móvel são as seguintes:

- Solicitar reserva antecipada ou imediata de recursos;
- Fornecer informações sobre a aplicação a ser submetida para uma reserva, a fim de conseguir que a reserva de um recurso seja a mais adequada para aquela aplicação;
- Verificar o *status* de uma reserva;
- Interagir com uma reserva. É possível cancelar, modificar ou fazer um *commit* de uma reserva.

Na interação com a reserva, em caso de modificação, pode-se solicitar mais recursos, mantendo o tempo inicial da reserva, sendo, portanto, tratada como uma reserva imediata. Outra alternativa é modificar a reserva incluindo o tempo da mesma, desde que seja para um tempo futuro em comparação ao tempo corrente. Ainda, é possível realizar um *commit* da reserva (ou seja: quando ultrapassar o tempo de inicio da reserva) e submeter a aplicação. Uma outra alternativa ao *commit*, que busca evitar a comunicação com o servidor é dizer, já no momento da reserva, qual aplicação será executada no período da reserva. Dessa forma, não é necessário realizar o *commit*, uma vez que o mesmo torna-se automático.

4.4.1.2 Requisitos de uma Reserva

Como requisitos mínimos exigidos para a realização de uma reserva única ou co-reserva de recursos, tem-se os seguintes:

- *start-time*: tempo inicial da reserva, podendo ser o tempo atual em caso de uma reserva imediata;
- *end-time* ou *duration*: tempo final ou tempo de duração da reserva;
- *number of processors*: quantidade de processadores mínima necessária levando-se em consideração o tamanho do *job*.
- *application*: o usuário pode definir previamente a aplicação que vai executar no tempo de reserva, realizando, portanto, automaticamente, o *commit* da reserva.

Pelo fato das características da aplicação poderem ser fator determinante para a reserva única ou co-reserva de recursos e podendo, portanto, melhorar a experiência do usuário, o mesmo possui a opção de dizer o nível de granularidade da aplicação. Com isso, o mecanismo de seleção do recurso levará em conta essa característica tentando alocar/reservar os recursos que melhor se adaptarem a esta aplicação. Caso o usuário não possua conhecimento dessa característica, a mesma não precisa ser informada e o mecanismo de seleção utiliza outros algoritmos para a reserva de recursos.

4.4.2 Arquitetura do Meta-escalonador

O usuário, através da interface móvel, realiza suas requisições de reserva de recursos que são repassadas para o meta-escalonador. A figura 4.3 mostra os módulos que o meta-escalonador possui para tratar as requisições do usuário. Os módulos destacados em amarelo são originalmente do trabalho de *silva* e em verde do trabalho de *grumiche*, sendo

esses adaptados para a arquitetura proposta. A seguir são explicadas as funcionalidades dos módulos.

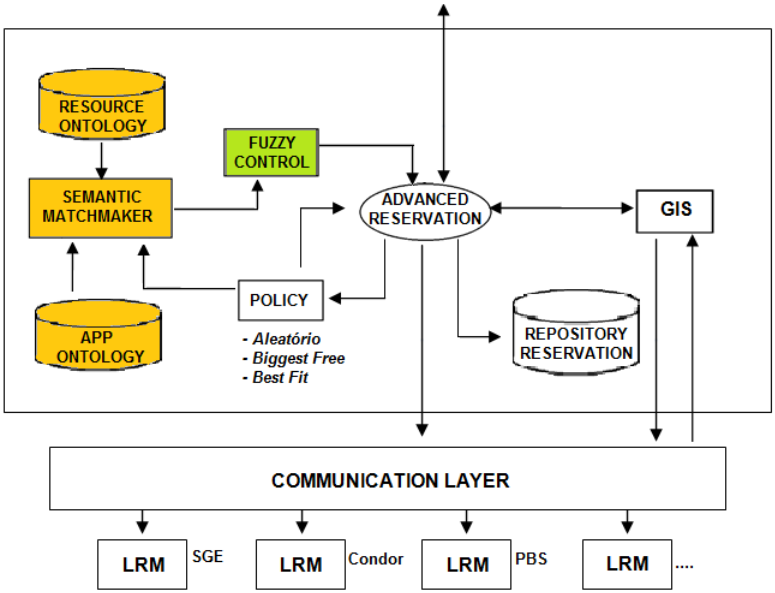


Figura 4.3: Arquitetura do Meta-escalonador

- *Advanced Reservation:* Esse módulo é responsável por coletar as informações de reserva do usuário e contatar os demais módulos a fim de atender às especificações requisitadas. Funciona como um mecanismo de *broker*, tomando as decisões e integrando os módulos;

- *Reserve Repository:* Mantém uma base de informações de todas as reservas que foram realizadas. Isso possibilita interações futuras entre o usuário e o sistema, permitindo que seja verificado o *status* de uma reserva, o cancelamento e a alteração da mesma e, ainda, o monitoramento sobre o andamento dos *jobs*. Esse módulo já foi implementado a fim de facilitar o futuro desenvolvimento de um mecanismo de *checkpoint*, que permitirá o acesso às execuções

prévias e possibilitará a criação de um mecanismo de tolerância a falhas;

- *Selection Policy Manager:* A seleção dos recursos a serem reservados é de extrema importância, pois pode impactar diretamente em uma queda de desempenho de uma aplicação e, consequentemente, gerar um aumento no tempo de resposta. Esse módulo possui algoritmos que são responsáveis por buscar o conjunto de recursos disponíveis mais ideal a ser reservados para o usuário. A escolha de um destes algoritmos é feita, dentre outros fatores, mediante o conhecimento das características da aplicação. No caso do usuário não possuir esse conhecimento, alguns algoritmos de seleção de recursos foram adaptados para considerar as peculiaridades de uma reserva, os quais serão mostrados na seção 5.3 do próximo capítulo;

- *Matchmaker Semântico:* Este módulo, implementado por [Silva and Dantas, 2007], tem acesso a informações estáticas armazenadas sobre os recursos que estão compartilhados no ambiente. Com este módulo é possível obter informações sobre estes recursos de forma transparente com relação à heterogeneidade das descrições. Ainda, tem por objetivo realizar a integração da descrição dos parâmetros pedidos, ou seja, da requisição do usuário com as descrições existentes no sistema e verificar os recursos compartilhados existentes;

- *Application Ontology:* Possui as ontologias de descrição das características da aplicação, *AppFeature*, baseadas no trabalho de [Grumiche et al., 2010] que considera as características comportamentais da aplicação, baseando-se, também, na ontologia de pedidos e de referências para descrição de recursos computacionais proposta em [Silva and Dantas, 2007]. A descrição de uma aplicação por meio do uso desta ontologia se dá pela criação de um indivíduo que representa a aplicação a ser executada e sua respectiva inclusão nos conceitos desta ontologia;

- *Resource Ontology:* Dada a heterogeneidade dos ambientes da grades computacionais, as ontologias de descrição de recursos são necessárias para uma melhor seleção. Este módulo contém ontologias também estendidas de [Silva and Dantas, 2007], buscando expressar com mais detalhes os recursos computacionais presentes no ambiente. Com este módulo, portanto, é possível obter informações sobre estes recursos de forma transparente com relação à heterogeneidade das descrições. Entre outras informações, essas ontologias

descrevem a *latência* e a *taxa de transmissão máxima* da rede de interconexão;

- *Fuzzy Control:* Esse módulo, utilizado no trabalho de [Grumiche et al., 2010] e adaptado para o sistema proposto, é responsável por calcular os níveis de adequação de cada *cluster*, baseado em suas ontologias de descrições de recursos em relação a uma aplicação, segundo suas características;

- *GIS:* Esse módulo é responsável pelas informações a respeito do ambiente. É utilizado para registro de novos *clusters*, bem como para fornecer informações sobre o ambiente e o estado dos recursos. Utiliza-se este módulo, por exemplo, para saber se, em um determinado tempo futuro, haverá recursos disponíveis;

- *Communication Layer:* Componente que permite a comunicação do servidor global, ou seja, do meta-escalonador com os LRMs de cada *cluster* que compõe o ambiente;

- *Local Resource Manager*: Os LRMs, ou seja, os Gerenciadores de Recursos Locais são responsáveis por tratar localmente os recursos de cada *cluster* e interagir com o meta-escalonador. No ambiente em questão, desconsiderou-se a interação direta com os LRMs. Portanto, o usuário só pode interagir com os *clusters* através do meta-escalonador.

O módulo de *Advanced Reservation* possui o algoritmo que trata a reserva antecipada dos recursos de uma forma global entre os LRMs. É o módulo do sistema que possui interface de comunicação com o dispositivo móvel e trata todas as requisições do usuário. Portanto, possui toda a lógica do sistema, do lado do servidor.

No caso de uma interação com alguma reserva, esse módulo se conecta com repositório de reservas (*Reserve Repository*) que contém todas as reservas previamente realizadas. Se a requisição for de modificação ou cancelamento, através do *bookId* a reserva é localizada no repositório, podendo ser alterada ou cancelada.

Por outro lado, no caso de uma nova reserva, considera-se o conhecimento do usuário a respeito de sua aplicação. Caso ele saiba o nível de granularidade da mesma e a informe como requisito da reserva na interface de acesso, o gerenciador de políticas de seleção de recursos, *Selection Policy Manager*, retorna o algoritmo *AR Application* que seleciona o melhor recurso, com base nas características da aplicação. Entre as políticas de seleção de recursos, foram implementados também os algoritmos *Ar*

Aleatório, *AR Biggest Free* e *AR Best Fit* e os mesmos são detalhados no em 5.3.

Neste caso, as características da aplicação, previamente representadas por *AppFeature*, são traduzidas para a ontologia de pedidos proposta por [Silva and Dantas, 2007], que se faz necessária à utilização com o serviço de *matchmaking semântico* de recursos, também implementado por [Silva and Dantas, 2007]. Este serviço retorna uma lista de descrições ontológicas de recursos (clusters) que atendem às características restritivas da aplicação. Em seguida o sistema de controle *Fuzzy* é acionado para calcular o nível de adequação de cada recurso. Esta etapa inicial não considera a disponibilidade dos recursos e seu funcionamento pode ser visualizado no Algoritmo 4.1.

Algoritmo 4.1 Calculo de Nível de Adequação - $fuzzy()$

Input: Características da aplicação.
Output: Lista com os níveis de adequação de cada para a aplicação em questão.

```
Início
Gera ontologia de pedidos
Matchmakingk Semantico para localização de recursos
for all recurso retornado do Matchmaking do
    ListAdequacao ⇐ NivelAdequacaoRecurso(recurso, pedidos)
end for
Fim
```

O sistema de controle *Fuzzy*, proposto por [Grumiche et al., 2010], possui as seguintes entradas:

- *resourceInterconnectionLatency*: refere-se à latência da interconexão;
- *resourceInterconnectionBitRate*: refere-se à taxa de transmissão máxima da rede de interconexão do *cluster*;
- *applicationGranularity*: refere-se ao nível de granularidade, que pode ser: embaraçosamente paralela, granularidade grossa e granularidade fina.

O sistema aplica as regras de inferência e, para a *defuzificação* do resultado, utiliza a função CoG (*center of gravity*) [Bai et al., 2006].

O funcionamento do *AR Application*, que pode ser visualizado no Algoritmo 4.2, utiliza inicialmente a lógica *fuzzy* para saber quais *clusters* são os mais indicados para comportarem a reserva. A partir deste

momento, considera-se o estado de cada *cluster*, verificando a disponibilidade de recursos, através do módulo GIS, para o tempo solicitado de reserva. O GIS verifica em cada *cluster* essa disponibilidade no tempo requisitado, sendo que os *clusters* gerenciam suas tabelas de alocação temporal. Em uma abordagem sem reserva antecipada de recursos, os *jobs* poderiam ser colocados na fila, caso os recursos estivessem ocupados no momento. Como a reserva busca garantir uma QoS, esta verificação de disponibilidade se faz necessária. O funcionamento desta verificação pode ser visualizado no Algoritmo 4.3.

Algoritmo 4.2 Funcionamento do AR Application

Input: Características da aplicação ($numeroTasks, granularidade, sistemaOperacional, ...$); e requisição da reserva R ($startime, duracao$)
Output: Id do recurso selecionado.

```
Início
ListAdequacao ⇐ fuzzy(caractAplicacao)
for all recurso em ListAdequacao do
    // Utiliza o GSI para obter informações do ambiente verificando o
    número de processadores livres no tempo solicitado
    nFree ⇐ numProcLivresFut(startime, duracao, recurso)
    if nFree >= tasks then
        ListPossivelRecurso ⇐ recurso
    end if
end for
if Lista de Possíveis Recursos estiver vazia then
    return Não há recursos disponíveis para o tempo solicitado
end if
for all recurso em ListPossivelRecurso do
    Percorre a lista comparando nível de adequação dos recursos
    if Caso dois recursos possuam mesmo nível de adequação then
        Seleciona recurso com mais processadores disponíveis
    end if
end for
Retorna id do recurso com maior nível de adequação
Fim
```

O módulo de *Advanced Reservation*, utilizando o algoritmo *AR Application* seleciona o melhor *cluster*, de acordo com as características da aplicação, para reservar recursos, no caso, reservar o número de processadores requisitados. De posse desta informação, a reserva é enviada para

Algoritmo 4.3 Verificação de disponibilidade de um recurso em um determinado tempo - $numProcLivresFut()$

Input: Tempo da reserva ($startime, duracao$); Recurso.
Output: Número de processadores livres.

Início
Realiza uma consulta ao recurso retornando sua $timeslot\ table$ em uma lista
Localiza o tempo de reserva através do $startime$ e $duracao$
Para este tempo, retorna o número dos processadores livres:
$return\ procLivres \Leftarrow totalProcRecurso - procReservados$
Fim

o recurso com o tempo de reserva e número de processadores a serem reservados. A reserva é realizada e colocada na lista de *timestlot* e é retornado então o *id* da reserva, (*bookId*). De posse da requisição do usuário e do *bookId*, o módulo de *Advanced Reservation* conecta-se ao repositório de reservas (*Reserve Repository*), salvando todas estas informações para interações posteriores.

As reservas de recursos possuem quatro estados durante o seu ciclo, podendo a reserva estar em qualquer uma delas. A figura 4.4 mostra o diagrama de transição de estados de uma reserva antecipada e as possíveis transições entre os mesmos. As definições de cada estado são descritas a seguir:

- **Request**: é o primeiro estado, ocorrendo quando uma requisição de reserva, com seus parâmetros, é feita pelo usuário;

- **Reserved**: este estado ocorre quando uma requisição de uma reserva é aprovada. A partir deste estado, também, pode-se modificar uma reserva ou cancelar a mesma;

- **Activated**: este estado começa quanto o tempo de inicialização de uma reserva é atingido e, portanto, pode-se começar a utilizar a reserva a qualquer momento, antes que o tempo de duração ou tempo final da reserva termine. As opções neste estado são: requisitar um outro tempo de reserva; requisitar mais recursos para esta reserva, através de uma reserva imediata; cancelar a reserva. Como mencionado anteriormente, uma reserva imediata significa utilizar o tempo corrente como tempo inicial da reserva;

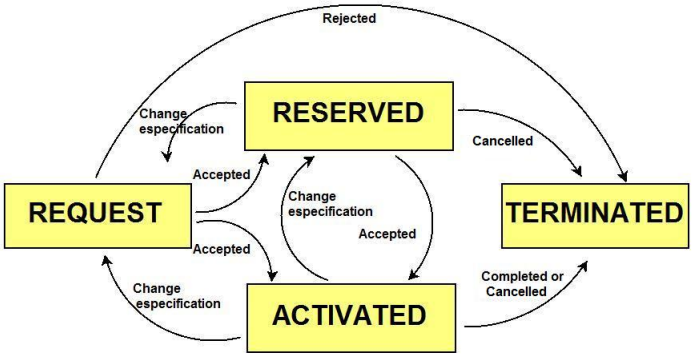


Figura 4.4: Diagrama de transição de estados de uma reserva antecipada

- **Terminated**: este estado representa o final do ciclo de uma reserva. Uma reserva pode ser completada, cancelada ou rejeitada.

4.5 Considerações sobre a Abordagem Proposta

A arquitetura proposta para reserva de recursos em ambientes de grades computacionais com configuração de *multi-clusters* visa explorar o conhecimento do usuário sobre as características da aplicação que serão submetidas em um futuro intervalo de tempo. Através da abordagem proposta é possível selecionar o *cluster* que melhor se adéque às características comportamentais de um aplicação, sendo estas as que influenciam diretamente no tempo de resposta da execução. Por tal motivo, a abordagem não realiza co-reserva entre diferentes *clusters*, pois o tempo de comunicação da aplicação é um fator determinante nesta arquitetura.

Tendo em vista que ambientes de grades computacionais possuem recursos distribuídos entre diferentes VOs e cada uma utiliza políticas e formas de descrição de características próprias, não cabe aos usuários preocupações com as particularidades do ambiente, deixando o meta-escalonador tomar a decisão de escolha dos recursos a serem reservados. Ainda, busca-se através de uma interface móvel melhorar a QoE do usuário, permitindo uma maior flexibilidade na descrição de sua requisição e disponibilizando acesso ao ambiente a qualquer lugar e a qualquer momento.

Capítulo 5

Ambiente e Resultados Experimentais

Este capítulo apresenta o ambiente experimental desenvolvido para validar a abordagem proposta. O objetivo deste ambiente é possibilitar uma reserva de recursos em ambientes de grades, através de dispositivos móveis, que satisfaça os requisitos do usuário considerando a aplicação a ser executada no ambiente. Ambientes de grades computacionais com configuração de *multi-clusters* nem sempre são factíveis de serem utilizados como ambientes de testes de propostas. Desta forma, foi necessário utilizar um simulador para que os testes e a validação da proposta fossem possíveis. Neste ambiente simulado, considerou-se que todos os *clusters* oferecem suporte à reserva antecipada de recursos, ou seja, cada *cluster* gerencia sua própria tabela de alocação temporal.

A descrição da simulação e seus aspectos, assim como a escolha do simulador a ser utilizado como ambiente de testes, são discutidas na seção 5.1. Na seção 5.2 é descrito o ambiente e algumas opções de implementação são justificadas. Na seção 5.3 são apresentados os algoritmos que foram utilizados a nível de comparação e validação da abordagem proposta. A seção 5.4 mostra uma análise comparativa entre a abordagem proposta nesta dissertação e outras abordagens utilizadas em ambientes distribuídos. Por último, a seção 5.5 traz as considerações finais e uma tabela comparativa entre as principais características dos trabalhos relacionados com as características da abordagem proposta.

5.1 Simulação

A simulação tem sido amplamente utilizada para modelagem e avaliação de sistemas no mundo real, desde processos de negócios e linhas de montagens de fábrica a projetos de sistemas computacionais. Consequentemente, a modelagem e a simulação surgiram como uma disciplina importante e muitas ferramentas padronizadas para aplicações específicas foram construídos. Para ambientes de grades computacionais há um número ainda pequeno de ferramentas de simulação.

Neste sentido, considerando a utilização de um simulador para a

criação do ambiente de grade formado por *multi-clusters*, inicialmente, realizou-se um estudo sobre estas ferramentas disponíveis, a fim de utilizar aquele que melhor servisse para o objetivo da proposta. Assim, a subseção seguinte apresenta estas ferramentas.

5.1.1 Simuladores

Buscou-se, para este trabalho, simuladores que possibilitassem de forma fácil, criar um ambiente de *multi-clusters* e, se possível, ter algum suporte à reserva antecipada de recursos nestes ambientes simulados. As ferramentas analisadas foram: OptorSim [Bell et al., 2003], SimGrid [Casanova et al., 2008], MicroGrid [Song et al., 2000] e GridSim [Sulistio and Buyya, 2004].

O OptorSim tem como principal objetivo estudar a eficiência de diversas estratégias de replicação em ambientes de grades. É um pacote bastante completo e incorpora alguns protocolos de otimização destas réplicas de dados, sendo este o seu maior foco.

O SimGrid é um *toolkit*, desenvolvido em linguagem C, para simulação de escalonamento de aplicações. Possui suporte à modelagem de recursos com tempo compartilhado e a carga pode ser inserida através de constantes ou através de *traces* reais. Permite, ainda, a criação de tarefas quanto aos seus tempos de execução e recursos com seus respectivos padrões e capacidades.

O emulador MicroGrid é modelado com base no Globus. O emulador permite a execução de aplicações construídas, utilizando o Globus como ambiente virtual controlado de recursos. A diferença do MicroGrid para os outros simuladores é que, por se tratar de um emulador, possibilita que o código do aplicativo seja executado na grade virtual. Assim, os resultados produzidos pela execução de uma aplicação no MicroGrid são muito próximos do mundo real. No entanto, tal emulador requer o conhecimento do Globus e a implementação de um sistema/aplicação para o estudo.

O GridSim, desenvolvido em Java, suporta a simulação de vários tipos de grades computacionais e modelos de escalonamento de aplicações. O simulador permite a modelagem de diferentes recursos com suas características particulares e propriedades de falhas. Similar ao SimGrid, permite a simulação de *traces* de cargas de tarefas de supercomputadores reais. O pacote permite a implementação de escalonadores específicos para compartilhamento de recursos, mas oferece dois escalonadores padrões para uso: *TimeShared*, usando a política de *Round Robin*; ou *SpaceShared*, com o *First Come First Serve* (*FCFS*). Essa ferramenta oferece, portanto, um *framework* de eventos discretos para simulação de diversas

classes de entidades de todo um sistema de grade, tais como, recursos heterogêneos, usuários, tarefas, diferentes VOs, escalonadores. Ainda, possui suporte a mecanismo de reserva para alocação de recursos, o que permite que sejam desenvolvidos mecanismos de reserva antecipada de recursos em grades computacionais.

O simulador escolhido para validação da abordagem proposta foi o GridSim por possuir o suporte à reserva de recursos.

5.1.2 Aspectos da Simulação

Como mencionado no capítulo 4, o trabalho proposto baseia-se em uma arquitetura na qual os usuários interagem com o ambiente através de uma interface de acesso móvel que conecta-se a um servidor, funcionando como um meta-escalonador para o ambiente de grade computacional com configuração de *multi-clusters*. O meta-escalonador possui o GSI, que conecta-se com todos os *clusters* presentes no ambiente e, de uma forma global, reserva recursos nestes *clusters*. Para tal fim, todos os *clusters* locais simulados pelo GridSim oferecem suporte à reserva de recursos.

No GridSim cada recurso possui uma política de escalonamento que faz o papel de um gerenciador de recurso local (LRM). Como mencionado anteriormente, o GridSim oferece alguns padrões de escalonadores para o uso. Entretanto para o trabalho proposto, estendeu-se a classe *AR-ConservativeBackfill* que implementa o algoritmo de *Conservative Backfilling* [Mu'alem and Feitelson, 2001], mas com suporte à reserva antecipada de recursos e simulação de recursos computacionais paralelos. A extensão desta classe fez-se necessária pois o simulador não leva em consideração os custos com comunicação intra-recurso de uma aplicação paralela distribuída. O GridSim considera o custo de comunicação em termos de tempo considerando o total de dados transmitidos pela taxa máxima de transmissão do Link.

Portanto, nesta extensão implementou-se o tempo de comunicação, através de um cálculo do volume de dados transmitidos pelo *job*, baseado na equação utilizada no trabalho de [Grumiche et al., 2010], de modo a considerar o tempo de execução da aplicação com diferentes tipos de granularidade. A equação ainda trata dos ajustes de queda de crescimento de *speed up* comumente vistos em aplicações de granularidade fina e grossa, dado o aumento da utilização de processadores, tornando a execução mais próxima da realidade. Isso se fez necessário para testar a execução do *job* no recurso reservado, porém, para a reserva, considera-se somente a granularidade da aplicação e não seu tempo de comunicação. A equação 5.1 mostra o cálculo do volume total de tráfego de dados em bytes.

$$V = \frac{T_{cpu} \cdot P_{com}}{100 - P_{com}} \cdot \frac{N}{8} \cdot 100000000 \quad (5.1)$$

O T_{cpu} é o tempo de CPU em segundos previsto por tarefa da aplicação. No caso, utilizou-se 3600 segundos, ou seja, 1 hora de tempo de CPU, podendo A assumir os valores 4, 6, ou 8 e, portanto, representando 4 horas, 6 horas ou 8 horas. O calculo é mostrado na equação 5.2:

$$T_{cpu} = 3600 \cdot aleatorio(A) \quad (5.2)$$

O P_{com} é a porcentagem do tempo de execução gasto com comunicação. Esta porcentagem, segundo [Grumiche et al., 2010], foi definida com referência de uma interconexão *Gigabit Ethernet* e teve a seguinte parametrização: aplicação com granularidade fina terá um P_{com} de 30%; granularidade grossa o P_{com} será 10%; embaraçosamente paralela o P_{com} será de 1%.

A segunda parte da fórmula (coeficiente multiplicador) é a encarregada de causar a redução no *speed up*, forçando um aumento no tempo de execução das aplicações conforme maior for o número de tarefas paralelas e distribuídas representadas por N, de forma a garantir um crescimento não linear. A divisão por 8 é utilizada pois é o valor mínimo de tarefas que são simuladas no ambiente de testes.

Por fim, o valor 100000000 é a taxa de transmissão da interconexão de rede *Gigabit Ethernet*, para transformar o tempo em bytes.

Outra particularidade do GridSim é que ele trabalha com eventos discretos para a simulação de suas entidades. Para a arquitetura proposta, este fator este fator trouxe um pouco de limitação. As entidades da simulação são criadas antes da mesma ser iniciada. Quando a simulação inicia, não há possibilidade de interação com o ambiente em tempo real. Portanto, cria-se as cargas de trabalho, no caso, reservas e seus respectivos *jobs*, que são carregadas e registradas no GridSim como um evento discreto futuro. Neste caso, cada *job* é representado por uma entidade *gridlet* do GridSim. Esse pacote contém todas as informações relacionadas ao *job* e seus detalhes de execução, como por exemplo, seu tamanho expresso em milhões de instruções, tamanho de arquivos de entrada e saída, entre outros. Portanto, inicializada as entidades e registrados os eventos, a simulação ocorre sem intervenção do usuário. Apesar desta limitação, o simulador serviu para o propósito esperado.

Para validação da proposta, outros algoritmos que selecionam os recursos a serem reservados foram implementados e serão mostrados na seção 5.3. Os algoritmos não consideram as características da aplicação

e, portanto, atuam em um cenário no qual o usuário não possui este tipo de conhecimento na hora de criar uma requisição de reserva.

5.2 Descrição do Ambiente

A configuração para o desenvolvimento da abordagem consiste, do lado da grade computacional, em um cenário simulado de um ambiente de configuração de *multi-cluster*. Por outro lado, a interface móvel, também chamada de grade computacional móvel, permite que o usuário utilize os recursos compartilhados no ambiente a fim de resolver seus problemas computacionais. Assim, a utilização de dispositivos permite o acesso *anytime and anywhere* ao ambiente, contornando algumas limitações de recursos encontradas nestes tipos de aparelhos. A figura 5.1 mostra o ambiente desenvolvido para a abordagem. Pode-se observar nesta figura que foi utilizado o modelo de comunicação cliente-servidor para a integração do ambiente móvel, através dos dispositivos, com o ambiente de grade computacional, tendo o meta-escalonador o papel de servidor.

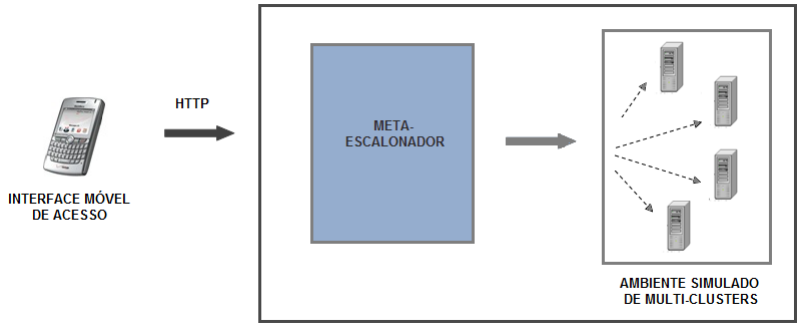


Figura 5.1: Ambiente Experimental

Ainda, em se tratando da interface móvel de acesso (grade computacional), esse ambiente consiste em um cenário típico de computação móvel, no qual os usuários acessam o ambiente dentro de uma rede estruturada por meio de dispositivos móveis, através de redes sem fio. Assim, para prover a comunicação entre os usuários e o servidor, foi utilizado o protocolo HTTP (*HyperText Transfer Protocol*). O principal motivo que levou a adotar este protocolo de comunicação foi o fato do mesmo ter suporte garantido em diversos dispositivos móveis.

Optou-se, nesta dissertação, pela adoção de uma das tecnologias

mais recentes para *smartphones*. Desta forma, a interface móvel foi desenvolvida utilizando a plataforma Android [Google, 2011]. Somado ao fato de o Android ser um sistema operacional *OpenSource*, a sua API de desenvolvimento permite a interação com diversos recursos disponíveis nos *smartphones* que podem vir a acrescentar novas funcionalidades ao sistema proposto. Ainda, essa plataforma fornece um emulador para testes e uma boa documentação.

A escolha desta plataforma foi feita a partir da decisão de passar parte das ontologias de descrição da aplicação e ontologias de descrição dos recursos computacionais para o dispositivo móvel, realizando um pré-processamento de *matching* no aparelho, visto que os atuais *smartphones* contam com um poder computacional expressivo. Este trabalho está sendo desenvolvido em conjunto com um aluno de graduação do curso de Sistemas de Informação da própria Instituição, que é membro do laboratório LaPeSD. Além de permitir que o poder de processamento crescente seja utilizado no ambiente de grade, busca-se neste trabalho colaborativo, suprir a limitação do GridSim para execução da simulação em tempo real, sendo criado um ambiente interativo para os testes com os dispositivos móveis.

Conforme mostrado na arquitetura proposta, no capítulo 4, o meta-escalonador é o responsável pelas decisões globais de reserva de recursos. Estas reservas, então, são realizadas no ambiente da grade computacional composta de *multi-clusters*. Este ambiente, que pode ser visto na figura 5.1, foi simulado no GridSim e possui as características listadas na tabela 5.1. As interconexões de rede que foram utilizadas seguiram os parâmetros do trabalho de [Yeo et al., 2006a] e podem ser vistas a seguir:

- **Gigabit Ethernet:**
  - *latência mínima* = 100 microssegundos
  - *taxa de transmissão máxima* = 100MBytes/s
- **Infiniband:**
  - *latência mínima* = 7 microssegundos
  - *taxa de transmissão máxima* = 850MBytes/s

Assim, cada *cluster* que faz parte do ambiente possui um total de 64 núcleos de processamento. Optou-se por uma configuração similar, porém alterando a rede de interconexão por ser fator diretamente ligado à granularidade da aplicação, portanto, buscando verificar efetivamente o mecanismo de reserva priorizando as características do *cluster* que mais se adeque a aplicação.

Tabela 5.1: Ambiente de *multi-cluster* simulado

Recurso	**Cluster1**	**Cluster2**	**Cluster3**	**Cluster4**
Nº de máquinas	8	8	8	8
Nº de núcleos por máquina	8	8	8	8
Tipo Processador	Xeon 5506	Opteron 2350	Xeon 5506	Opteron 2350
MIPS por Processador	8,52	8	8,52	8
MIPS Total	545,28	512	545,28	512
Interconexão de rede	Infiniband	Infiniband	Gigabit Ethernet	Gigabit Ethernet

A tabela 5.2 apresenta as características de hardware e pacotes de software do servidor e dos clientes utilizados no protótipo implementado.

Tabela 5.2: Características de hardware e software do ambiente experimental

Nodos	**Modelo**	**Clock**	**Memória**	**SO**	**SW**
Celular	HTC Desire	1GHz	512 MB	Android 2.2	-
Servidor	Intel Core 2 Duo	2,1GHz	4 GB	Ubuntu 9.04	GridSim 5.2

5.3 Algoritmos de Seleção

Esta seção tem por objetivo descrever outras estratégias de seleção de recursos computacionais que não consideram características comportamentais da aplicação como critério para reserva.

A abordagem proposta visa melhorar a qualidade de experiência do usuário. Para tanto, busca prover um mecanismo que torne possível fornecer o nível de granularidade de sua aplicação, como um dos requisitos de uma reserva. No entanto, a informação sobre este nível depende de um conhecimento mais detalhado da aplicação no caso de o usuário não ser o próprio criador da mesma. Neste contexto e, a título de testes da abordagem proposta, foram implementados alguns algoritmos comumente utilizados em ambientes de grades computacionais. Ambientes,

porém, adaptados para suportar reserva antecipada de recursos já que não consideram fila de espera de execução. Dessa maneira, diferentemente de um ambiente sem suporte à reserva, no qual os *jobs* são enfileirados para serem executados posteriormente, na abordagem atual, seleciona-se outro recurso disponível no tempo solicitado ou retorna-se uma informação de indisponibilidade de recurso para o instante solicitado. Os algoritmos são descritos a seguir, sendo os três apresentados no trabalho [Hamscher et al., 2000]:

- **AR Aleatório:** esta estratégia seleciona, de forma aleatória, um dos recursos pertencentes ao ambiente, desde que o recurso tenha processadores livres para o número de tarefas requisitadas. Na média, provê uma distribuição justa das reservas nos recursos disponíveis, mantendo uma boa distribuição de carga. Caso o recurso não esteja disponível no tempo solicitado é escolhido aleatoriamente um novo recurso;

- **AR Biggest Free:** este algoritmo seleciona o recurso com o maior número de processadores livres. Para o atual uso, considera-se o tempo de reserva. Uma das desvantagens desta estratégia é que pode ocasionar o atraso de tarefas que exigem muitos processadores, pois, a cada nova seleção (em caso de tarefas pequenas), recursos que seriam necessários para uma próxima tarefa maior, podem acabar sendo utilizados;

- **AR Best Fit:** este mecanismo seleciona o recurso que tenha o número de máquinas livres maior do que o número de tarefas do *job* a ser executado, mas cujo saldo de computadores livres seja o menor dentre todos os recursos. No entanto, o algoritmo foi adaptado para considerar o tempo de reserva. Quando comparado à estratégia de *Biggest Free* não necessariamente preenche com tarefas menores recursos com mais processadores.

5.4 Análise Comparativa da Abordagem Proposta

Esta seção tem como objetivo apresentar os experimentos realizados, a fim de validar a abordagem proposta para reserva de recursos em ambientes de grades computacionais, conforme apresentado na seção 4.3 do capítulo 4. Na seção 5.4.1 é apresentada a forma como uma requisição de reserva de recursos é feita para a simulação. Após, são apresentados os estudos de caso que visam validar a proposta, sendo o primeiro relacionado ao comparativo de reserva considerando, e não às características da aplicação. Na seção 5.4.3 é apresentado o segundo estudo de caso,

que mostra a execução das tarefas no recurso buscando avaliar o nível de adequação da reserva. Por fim, na seção 5.4.4 são apresentados os experimentos, utilizando a interface móvel de acesso ao ambiente.

5.4.1 Requisição

Dada a limitação do GridSim para trabalhar com eventos discretos e com suas entidades criadas antes do início da simulação, para tornar possível o experimento e validação da abordagem proposta, criou-se um *Workload* que contém as informações que irão gerar diversas reservas para os seus respectivos *jobs*. As informações são as seguintes:

- identificador da aplicação;
- tipo de granularidade da aplicação: *fina* (*Fine*), *grossa* (*Coarse*) e *embaraçosamente paralela* (*Embarassing Paralell*);
- *startime* da reserva;
- *duração* ou *endtime* da reserva. O módulo que trata da reserva trabalha com a duração da mesma, portanto, caso o usuário passe o tempo final da reserva, a duração é:

$$duracao = endtime - startime \tag{5.3}$$

- Número de tarefas paralelas;
- Milhões de instruções que vão ser executadas por esta aplicação;
- Volume total de bytes que serão transferidos na comunicação da aplicação. Este volume é calculado pela equação 5.1 que considera a granularidade da aplicação.

Estes dois últimos itens do *Workload*, referentes ao número de instruções da aplicação e ao volume de bytes transferidos, serão utilizados no segundo estudo de caso, a fim de avaliar o tempo de execução da aplicação no recurso reservado. Para critério de reserva do recurso, a granularidade da aplicação é descrita a nível de ontologia. Tais informações são necessárias para a simulação utilizando o GridSim.

Por se tratar de uma arquitetura que visa reservar os recursos que melhor se adéquem ao nível de granularidade da aplicação, outras características, como por exemplo, sistema operacional ou mínimo de memória, não foram utilizadas na simulação.

5.4.2 Qualidade da Reserva

Com o objetivo de avaliar o impacto obtido com a abordagem proposta, inicialmente, busca-se verificar a reserva de recursos considerando as características comportamentais da aplicação, ou seja, utilizando o mecanismo de reserva proposto. A seguir, são consideradas as estratégias implementadas mostradas na seção 5.3, a fim de comparar os resultados com a abordagem proposta. Neste primeiro momento considera-se o nível de adequação da reserva, portanto, não se utilizou a interface de acesso móvel.

5.4.2.1 Validação da Reserva

Inicialmente, foi considerado um usuário (*User_AR*) que realiza 3 solicitações de reservas para 3 aplicações diferentes. Cada aplicação possui um nível diferente de granularidade. Para este primeiro teste, reservou-se recursos com *startime* sendo o tempo corrente, portanto, caracterizando uma reserva imediata. A duração da reserva foi estipulada em 20 horas e número de tarefas foi fixado em 16 para as 3 reservas. Podem-se observar na figura 5.2 os níveis de adequação para cada granularidade e o recurso que foi reservado.

```
Adequacy level for fine grained application:
Clusters:      Cluster1          Cluster2          Cluster3          Cluster4
Results:   84.41853088480731  84.41853088480731  50.00000000000019  50.00000000000019

Resource select is Cluster1

User_AR: reservation has been accepted by Cluster1 at time = 10.0000018912

Adequacy level for coarse grained application:
Clusters:      Cluster1          Cluster2          Cluster3          Cluster4
Results:   35.2129193433263  35.2129193433263  84.41853088480731  84.41853088480731

Resource select is Cluster3

User_AR: reservation has been accepted by Cluster3 at time = 10.0000018912

Adequacy level for embarassing paralell application:
Clusters:      Cluster1          Cluster2          Cluster3          Cluster4
Results:   15.52961730449245  15.52961730449245  50.00000000000019  50.00000000000019

Resource select is Cluster4

User_AR: reservation has been accepted by Cluster4 at time = 10.0000018912
```

Figura 5.2: Reserva imediata de aplicações com diferentes níveis de granularidade.

Como se pode observar na figura 5.2, uma aplicação de granularidade fina que necessita de uma maior velocidade na rede de interconexão, tem no *Cluster 1* o seus recursos alocados, a serem utilizados imediatamente. Esta escolha foi realizada em razão do *Cluster 1* possuir uma

rede interconexão rápida *Infiniband*, o que implicou em um nível de adequação elevado. As aplicações de granularidade grossa e embaraçosamente paralelas, tiveram recursos reservados nos *Clusters 3* e *4*, pelo fato de possuírem uma rede de interconexão *Gigabit Ethernet* com latência e taxa de transmissão médias. Apesar do nível de adequação dos *clusters 3* e *4* serem o mesmo, para a aplicação de embaraçosamente paralela foi reservado o *cluster 4* uma vez que o algoritmo da abordagem proposta considera, nestes casos, o recurso com mais processadores livres.

A seguir foram realizados os testes com as mesmas aplicações, entretanto, utilizando as demais estratégias para reservar os recursos requisitados. A figura 5.3 mostra a *time slot* das reservas sendo utilizados cada um dos quatro algoritmos. Nessa figura são apresentadas as reservas das três aplicações para cada algoritmo, sendo que a duração é de 20 horas e cada *startime* é, a partir do tempo corrente, diferenciado em 20 horas. Em outras palavras, as reservas utilizando o algoritmo *AR Application* são realizadas no intervalo de 0 a 20 horas. Para o algoritmo *AR Aleatório* é utilizado o intervalo de 20 a 40 horas e assim sucessivamente para os algoritmos *AR Biggest Free* e *AR Best Fit*. Para o caso da abordagem proposta, o comportamento é o descrito pela figura 5.2.

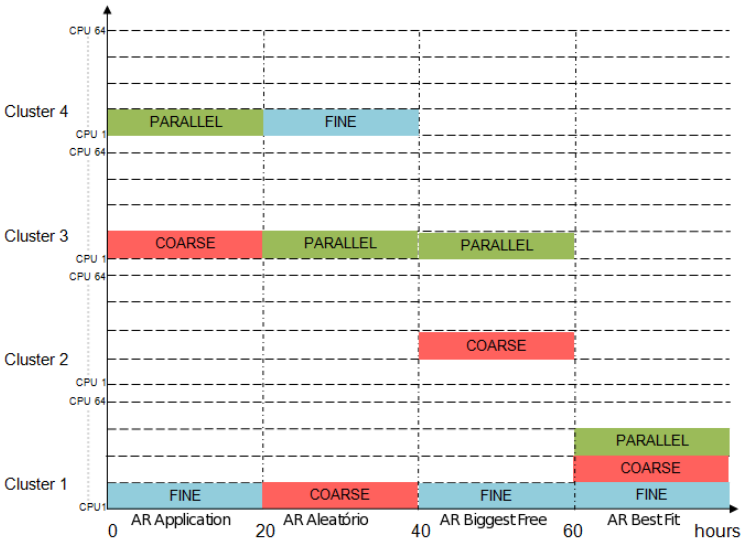


Figura 5.3: *Time slot* das 3 aplicações utilizando cada uma das estratégias de reserva.

O algoritmo *AR Aleatório* tem comportamento não determinístico e, por isso, cada reserva pode ser realizada em diferentes recursos para cada vez que a simulação é executada. Visando a não descaracterização da lógica aleatória, não foi realizado nenhum tipo de medição para os testes. No intervalo de 20 a 40 horas da figura 5.3 pode-se observar o seu comportamento. A aplicação de granularidade fina foi reservada para o *Cluster 4*. Este tipo de aplicação é dependente do desempenho da interconexão de rede e, neste caso, pode causar uma sub-utilização da capacidade de processamento, vindo a ocasionar um aumento no tempo de execução. A aplicação de granularidade grossa poderia ser reservada em *clusters* com interconexão *Gigabit Ethernet*. No entanto, a aplicação embaraçosamente paralela foi adequadamente reservada.

O intervalo de 40 a 60 horas, na figura 5.3, mostra a utilização da estratégia *AR Biggest Free*. Tal algoritmo, como descrito anteriormente, considera como candidatos à reserva, os recursos que têm mais processadores livres. Esta estratégia bem como a de *Best Fit*, possui comportamento determinístico. Então, se as simulações possuírem as mesmas requisições de reservas, o resultado será o mesmo. Como se pode observar na figura 5.3, a reserva dos recursos para as aplicações testadas se deu de forma sequencial, dada a particularidade do algoritmo e o fato da simulação possuir somente 3 reservas. As aplicações de granularidade fina e embaraçosamente paralela tiveram seus recursos devidamente alocados, porém a adequação da aplicação de granularidade grossa seria mais alta em um *cluster* com desempenho de interconexão médio.

O comportamento observado pelo estratégia *AR Best Fit* pode ser observado no último intervalo da figura 5.3. Todas as reservas foram realizadas no *Cluster 1*. As aplicações de granularidade grossa e embaraçosamente paralela poderiam ser alocadas nos *Clusters 3* ou *4* dada sua menor dependência da rede de interconexão.

O desempenho dos algoritmos *AR Biggest Free* e *AR Best Fit* são diretamente relacionados com a ordem das reservas no *Workload*, pois consideram somente o número de processadores requisitados e o número de processadores, com seus estados, do ambiente. Para este teste inicial a ordenação foi: *fine*, *coarse* e *parallel*; e o número de processadores requisitados favoreceram uma boa reserva para estes algoritmos.

5.4.2.2 Testes em Maior Escala

Os próximos testes, que têm seus resultados apresentados nas figuras 5.4, 5.5, 5.6 e 5.7, buscam simular um cenário real em que várias requisições de reservas são realizadas com diferentes tempos iniciais e tipos de granularidade de aplicações. Para tais testes, gerou-se de forma

aleatória um *Workload* com 15 aplicações, sendo 5 para cada tipo de granularidade. A duração de cada reserva foi definida com valor fixo em 20 horas, conseguindo assim, suprir todos os tempos de execução das aplicações, inclusive nos piores casos. O tempo inicial de cada reserva é um valor aleatório que pode ser o tempo corrente, 20, 40 ou 60 horas. O número de tarefas de cada aplicação é gerado aleatoriamente, podendo ser *8*, *16*, *24* ou *32* tarefas. A representação dessas tarefas é relativa ao tamanho do retângulo representado na figuras, sendo o menor retângulo referente à *8* tarefas e o maior representando *32* tarefas. O número de instruções (MIPS) que serão executadas por cada aplicação é um valor calculado de forma aleatória. O volume total de bytes que será transferido na comunicação da aplicação é calculado pela equação 5.1.

O *Workload* gerado pode ser visualizado na tabela 5.3.

Tabela 5.3: Workload utilizado para testes em maior escala

ID	Tipo	Startime	Tarefas	MIPS	Vol. Bytes
1	Paralela	216000s	32	230400000	3702857142857
2	Fina	72000s	16	115200000	1234285714285
3	Fina	216000s	16	172800000	1851428571428
4	Paralela	72000s	32	172800000	87272727272
5	Fina	0	24	115200000	1851428571428
6	Grossa	72000s	24	172800000	720000000000
7	Fina	144000s	32	115200000	2468571428571
8	Paralela	216000s	8	230400000	29090909090
9	Grossa	72000s	32	172800000	960000000000
10	Paralela	216000s	8	230400000	29090909090
11	Paralela	72000s	8	230400000	29090909090
12	Grossa	216000s	24	115200000	480000000000
13	Grossa	216000s	24	172800000	720000000000
14	Fina	216000s	24	230400000	3702857142857
15	Grossa	216000s	32	115200000	640000000000

Com a abordagem proposta é possível notar na figura 5.4 que as reservas foram realizadas priorizando as características comportamentais. As aplicações de granularidade fina foram alocadas nos *clusters 1* e *2*. As demais aplicações foram reservadas nos *clusters 3* e *4*. Somente uma aplicação de granularidade grossa foi reservada em um dos *clusters* com rede de interconexão *Infiniband*. Isso ocorreu, pois no tempo requisitado não havia processadores disponíveis nos demais *clusters* que pudessem

atender a solicitação. Com a abordagem proposta foi possível manter, no intervalo de 0 a 60 horas, um bom número de processadores disponíveis nos recursos com rede de interconexão *Infiniband*.

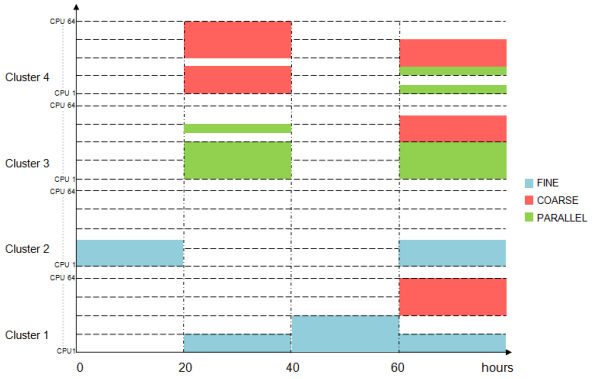


Figura 5.4: *Time slot* das reservas do *Workload* utilizando *AR Application*.

O algoritmo *AR Aleatório* tem seu *Workload* de reservas apresentado em 5.5. Dada a particularidade do critério de reserva desse algoritmo, duas aplicações de granularidade fina foram reservadas em *clusters* com rede de interconexão *Gigabit Ethernet*, portanto, sendo prejudicadas devido ao desempenho dessa rede. Uma aplicação com granularidade embaraçosamente paralela foi reservada no *Cluster 1*, sendo que a influência do desempenho de interconexão é insignificante para esse tipo de aplicação por ocupar somente 1% do tempo de execução. O mesmo ocorreu com uma aplicação de granularidade grossa, conforme observa-se no intervalo de 60 a 80, porém, neste caso a aplicação foi favorecida por ter 10% de seu tempo de execução gasto com comunicação.

Com o algoritmo *AR Biggest Free* notou-se um comportamento mais próximo da abordagem proposta, como pode ser visualizado na figura 5.6. Somente uma reserva de aplicação com granularidade fina foi feita em um recurso com desempenho médio da rede de interconexão. As outras quatro reservas desse tipo foram devidamente realizadas. No entanto, ocupou-se os *clusters* com rede *Infiniband* para reservas de aplicações com granularidade grossa e embaraçosamente paralela. Conforme já mencionado, as aplicações de granularidade grossa são favorecidas. Porém, no caso de novas requisições de reserva para aplicações de granularidade fina, isto torna-se um problema, pois o número de processadores

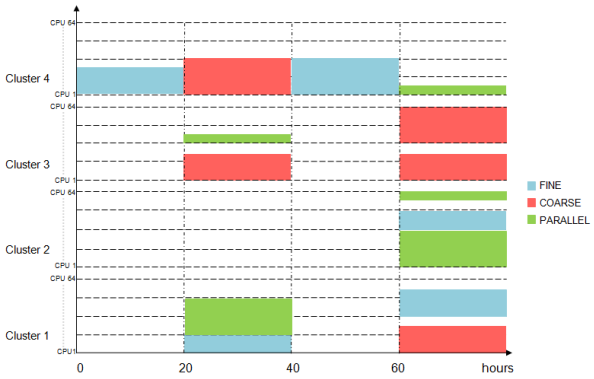


Figura 5.5: *Time slot* das reservas do *Workload* utilizando *AR Aleatório*.

disponíveis nestes recursos, *clusters 1* e *2*, estarão reduzidos.

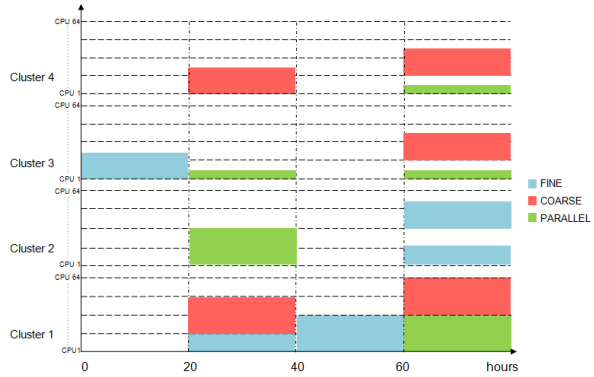


Figura 5.6: *Time slot* das reservas do *Workload* utilizando *AR Biggest Free*.

Para o *Workload* gerado para estes testes, o algoritmo *AR Best Fit* obteve o comportamento mais controverso dentre todos. Esse comportamento pode ser visualizado na figura 5.7. Das reservas requisitadas que tinham como característica da aplicação a granularidade fina, somente uma não foi devidamente alocada nos recursos com interconexão rápida. Esse

foi o ponto negativo neste cenário. Considerando, no entanto, que o cenário tivesse somente essas 15 reservas, quase todas as aplicações de granularidade grossa seriam favorecidas por serem reservadas no *Cluster 2*. No entanto, para novas requisições, os *clusters* com interconexão rápida já possuem várias reservas.

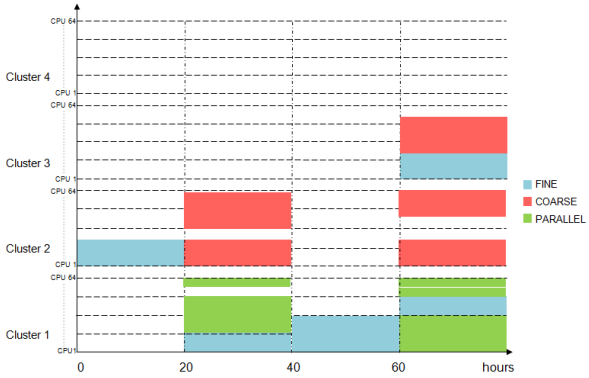


Figura 5.7: *Time slot* das reservas do *Workload* utilizando *AR Best Fit*.

Portanto, para o *Workload* gerado, observa-se que a abordagem proposta obteve a melhor distribuição de suas reservas, considerando a necessidade de comunicação de cada uma das aplicações. Isso contribuiu para manter disponíveis os recursos com rede de interconexão rápida para as próximas reservas. Por outro lado, os demais algoritmos não conseguiram reservar eficientemente todas as requisições que tinham aplicações com granularidade fina. No mínimo uma requisição era reservada em um recurso com desempenho médio de interconexão. Também, para o caso de próximas requisições de reservas com granularidade fina, a disponibilidade dos *clusters 1* e *2* ficou comprometida, com exceção do *AR Aleatório*.

5.4.3 Eficiência Computacional

Como mencionado anteriormente, o desempenho da rede de interconexão é diretamente ligado ao tempo de execução de aplicações de granularidade fina. As aplicações de granularidade diferente da fina não possuem tantos passos de comunicação e, portanto, podem ser alocadas em recursos que não possuem uma rede de interconexão rápida. Este fato pode ser observado nos estudos apresentados por [Pinto et al., 2008] e

[Pourreza et al., 2006], que mostram o impacto do desempenho de tecnologias de interconexão em aplicações com diferentes demandas de comunicação, comprovando que recursos precisam ser devidamente alocados.

Portanto, nesta subseção são apresentados os testes com a execução das tarefas nos recursos reservados, buscando avaliar o nível de adequação da reserva. Os cenários utilizados para estes testes foram os mesmos utilizados na subseção anterior, 5.4.2.

A tabela 5.4 mostra os tempos de execução referentes as execuções das aplicações que tiveram seus recursos reservados, conforme apresentado na subseção 5.4.2.1. O número de tarefas para cada uma das 3 reservas foi fixado em 16, possuindo mesmo tempo inicial e de duração, bem como o número de instruções a serem executadas. O que difere é o MIPS e o volume de bytes da comunicação. Nesta tabela é possível verificar o tempo de CPU, tempo de comunicação e o tempo de execução de cada aplicação.

Tabela 5.4: Resultados das execuções com base nos algoritmos de reserva

Tipo	**Recurso**	**Tempo CPU**	**Tempo Com.**	**Tempo Exec.**
AR Application				
Fina	Cluster 1	27043s	2770s	29813s
Grossa	Cluster 3	27043s	6104s	33147s
Paralela	Cluster 4	28801s	555s	29356s
AR Aleatório				
Fina	Cluster 4	28801s	23543s	52344s
Grossa	Cluster 1	27043s	719s	27762s
Paralela	Cluster 3	27043s	555s	27598s
AR Biggest Free				
Fina	Cluster 1	27043s	2770s	29813s
Grossa	Cluster 2	28801s	719s	29520s
Paralela	Cluster 3	27043s	555s	27598s
AR Best Fit				
Fina	Cluster 1	27043s	2770s	29813s
Grossa	Cluster 1	27043s	719s	27762s
Paralela	Cluster 1	27043s	66s	27109s

Observar-se nessa tabela que, apesar das outras estratégias não considerarem a granularidade da aplicação, ainda assim, obtiveram bons resultados. A aplicação de granularidade fina foi devidamente alocada pela

estratégia proposta e também nas abordagens *AR Biggest Free* e *AR Best Fit*. Isso deve-se ao fato das estratégias realizarem uma análise sequencial do ambiente, conforme o cadastro dos recursos, e avaliarem primeiramente o *Cluster 1*. Ainda, a abordagem proposta, de acordo com o nível de adequação, alocou as aplicações com granularidade grossa e embaraçosamente paralela nos *Clusters 3* e *4* respectivamente. Entretanto, as estratégias que alocaram essas aplicações nos *clusters* com rede de interconexão do tipo *Infiniband*, obtiveram um menor tempo de execução. Por mais que estas aplicações possuam pouca comunicação, a interconexão contribuiu, nesses casos, para uma redução no tempo de comunicação.

Dando continuidade a avaliação de eficiência computacional, foram realizados testes com a execução das tarefas previamente reservadas pelo *Workload* que foi gerado para o cenário da subseção 5.4.2.2, que pode ser visualizado na tabela 5.3. Os resultados das execuções podem ser observados nas tabelas 5.5, 5.6, 5.7 e 5.8.

Tabela 5.5: Workload executado com a estratégia proposta *AR Application*

ID	**Tipo**	**Recurso**	**Tempo CPU**	**Tempo Com.**	**Tempo Exec.**
1	Paralela	Cluster 3	27043s	1110s	28153s
2	Fina	Cluster 1	13522s	1385s	14907s
3	Fina	Cluster 1	20282s	2078s	22360s
4	Paralela	Cluster 3	20282s	833s	21115s
5	Fina	Cluster 2	14401s	2078s	16479s
6	Grossa	Cluster 4	21601s	6867s	28468s
7	Fina	Cluster 1	13522s	2770s	16292s
8	Paralela	Cluster 4	28801s	278s	29079s
9	Grossa	Cluster 4	21601s	9156s	30757s
10	Paralela	Cluster 4	28801s	278s	29079s
11	Paralela	Cluster 3	27043s	278s	27321s
12	Grossa	Cluster 4	14401s	4578s	18979s
13	Grossa	Cluster 3	20282s	6867s	27149s
14	Fina	Cluster 2	28801s	4155s	32956s
15	Grossa	Cluster 1	13522s	719s	14241s

A abordagem proposta neste trabalho obteve uma boa distribuição de suas reservas e, portanto, conseguiu aproveitar os recursos de acordo com o nível de adequação da aplicação. Na tabela 5.5 é possível verificar que, em se tratando de tempo de comunicação, nenhuma aplicação ultra-

passou um valor superior à *10000 segundos*. O algoritmo proposto obteve os seguintes resultados:

- **tempo total de execução:** 357335 segundos (aproximadamente 99,26 horas);
- **tempo médio de execução:** 22480,73 segundos (aproximadamente 6,24 horas).

Tabela 5.6: Workload executado com a estratégia *AR Aleatório*

ID	Tipo	Recurso	Tempo CPU	Tempo Com.	Tempo Exec.
1	Paralela	Cluster 2	28801s	131s	28932s
2	Fina	Cluster 1	13522s	1385s	14907s
3	Fina	Cluster 2	21601s	2078s	23679s
4	Paralela	Cluster 1	20282s	98s	20380s
5	Fina	Cluster 4	14401s	17657s	32058s
6	Grossa	Cluster 3	20282s	6867s	27149s
7	Fina	Cluster 4	14401s	23543s	37944s
8	Paralela	Cluster 4	28801s	278s	29079s
9	Grossa	Cluster 4	21601s	9156s	30757s
10	Paralela	Cluster 2	28801s	33s	28834s
11	Paralela	Cluster 3	27043s	278s	27321s
12	Grossa	Cluster 1	13522s	539s	14061s
13	Grossa	Cluster 3	20282s	6867s	27149s
14	Fina	Cluster 1	27043s	4155s	31198s
15	Grossa	Cluster 3	13522s	6104s	19626s

A estratégia de carácter aleatório foi a que obteve o pior conjunto de recursos reservados. O fato de aplicações com granularidade fina terem sidos reservadas em *clusters* com rede de interconexão média, fez com que seus tempos de comunicação, no momento da execução, fossem elevados. Esse comportamento pode ser observado na tabela 5.6. O tempo médio de execução de todas as aplicações reservadas com este algoritmo (*AR Aleatório*) foi de *26204,93 segundos*. Esse valor é superior a 1 hora em comparação ao tempo médio de execução da abordagem proposta no presente trabalho. Essa estratégia obteve os seguintes resultados:

- **tempo total de execução:** 393074 segundos (aproximadamente 109,19 horas);

- **tempo médio de execução:** 26204,93 segundos (aproximadamente 7,28 horas).

Tabela 5.7: Workload executado com a estratégia *AR Biggest Free*

ID	Tipo	Recurso	Tempo CPU	Tempo Com.	Tempo Exec.
1	Paralela	Cluster 1	27043s	131s	27174s
2	Fina	Cluster 1	13522s	1385s	14907s
3	Fina	Cluster 2	21601s	2078s	23679s
4	Paralela	Cluster 2	21601s	98s	21699s
5	Fina	Cluster 3	13522s	17657s	31179s
6	Grossa	Cluster 4	21601s	6867s	28468s
7	Fina	Cluster 1	13522s	2770s	16292s
8	Paralela	Cluster 3	27043s	278s	27321s
9	Grossa	Cluster 1	20282s	1078s	21360s
10	Paralela	Cluster 4	28801s	278s	29079s
11	Paralela	Cluster 3	27043s	278s	27321s
12	Grossa	Cluster 3	13522s	4578s	18100s
13	Grossa	Cluster 4	21601s	6867s	28468s
14	Fina	Cluster 2	28801s	4155s	32956s
15	Grossa	Cluster 1	13522s	719s	14241s

O algoritmo de *AR Biggest Free* teve o segundo melhor tempo médio de execução das aplicações do *Workload*, sendo esse tempo de *24149,60 segundos*. O resultado das execuções podem ser observado na tabela 5.7. Apesar de ter um programa de granularidade fina (id 5) sendo executado no *Cluster 3*, as aplicações de granularidade grossa (ids 9 e 15) foram favorecidas ao serem reservadas no *Cluster 1*. Isso reduziu o tempo de comunicação destas aplicações. O tempo total de execução e tempo médio para o *AR Biggest Free* foram os seguintes:

- **tempo total de execução:** 362244 segundos (aproximadamente 100,62 horas);
- **tempo médio de execução:** 24149,60 segundos (aproximadamente 6,71 horas).

Por último, pode-se observar na tabela 5.8 o resultado das execuções utilizando a estratégia *AR Best Fit*. O tempo médio de execução foi

Tabela 5.8: Workload executado com a estratégia *AR Best Fit*

ID	Tipo	Recurso	Tempo CPU	Tempo Com.	Tempo Exec.
1	Paralela	Cluster 1	27043s	131s	27174s
2	Fina	Cluster 1	13522s	1385s	14907s
3	Fina	Cluster 1	20282s	2078s	22360s
4	Paralela	Cluster 1	20282s	98s	20380s
5	Fina	Cluster 2	14401s	2078s	16479s
6	Grossa	Cluster 2	21601s	808s	22409s
7	Fina	Cluster 1	13522s	2770s	16292s
8	Paralela	Cluster 1	27043s	33s	27076s
9	Grossa	Cluster 3	20282s	9156s	29438s
10	Paralela	Cluster 1	27043s	33s	27076s
11	Paralela	Cluster 1	27043s	33s	27076s
12	Grossa	Cluster 2	14401s	539s	14940s
13	Grossa	Cluster 2	21601s	808s	22409s
14	Fina	Cluster 3	27043s	35314s	62357s
15	Grossa	Cluster 3	13522s	6104s	19626s

de *24666,60 segundos*. Novamente, uma das aplicações de granularidade fina (id 14) não foi devidamente reservada. Entretanto, quatro aplicações de granularidade grossa foram reservas nos *clusters* com redes de interconexão rápida, contribuindo para uma redução no tempo médio de execução. Os resultados do *AR Best Fit* foram:

- **tempo total de execução:** 369999 segundos (aproximadamente 102,72 horas);
- **tempo médio de execução:** 24666,60 segundos (aproximadamente 6,85 horas).

Através destes testes pode-se comprovar quantitativamente a eficiência da abordagem proposta, que obteve a melhor média do tempo de execução entre todos os algoritmos. No tempo total de execução das 15 aplicações previamente reservadas, obteve-se, com o mecanismo deste trabalho, um ganho aproximado de 7 horas em relação a abordagem *AR Biggest Free*, que apresentou o segundo melhor resultado. A tabela 5.9 apresenta mais dados referentes as abordagens analisadas.

Tabela 5.9: Resultados das execuções com base nos algoritmos de reserva

Tipo	**Tempo mínimo de execução**	**Tempo máximo de execução**	**Tempo médio de execução**	**Tempo mínimo de comunicação**	**Tempo máximo de comunicação**	**Tempo médio de comunicação**
AR Application						
Fina	14907s	32956s	20598,8s	1385s	4155s	2493,2s
Grossa	14241s	30757s	23918,8s	719s	9156s	5637,4s
Paralela	21115s	29079s	26949,4s	278s	1110s	555,4s
AR Aleatório						
Fina	14907s	37944s	27957,2s	1385s	17657s	9763,6s
Grossa	14061s	30757s	23748,4s	539s	9156s	5906,6s
Paralela	20380s	29079s	26909,2s	33s	278s	163,6s
AR Biggest Free						
Fina	14907s	32956s	23802,6s	1385s	17657s	5609s
Grossa	14241s	28468s	22127,4s	719s	6867s	4021,8s
Paralela	21699s	29079s	26518,8s	98s	278s	212,6s
AR Best Fit						
Fina	14907s	62357s	26479,8s	1385s	35314s	8725s
Grossa	14940s	29439s	21764,4s	539s	9156s	3483s
Paralela	20380s	27174s	25756,4s	33s	131s	65,4s

5.4.4 Mobilidade

A interface de acesso de uma grade computacional é um dos pontos chave para a interação com esses ambientes. A utilização de dispositivos móveis acrescenta a estes ambiente a mobilidade imposta neste meio. Com isso, interage-se com a grade sem a necessidade de estar utilizando um computador pessoal conectado a uma rede. Ainda, com o advento da tecnologia 3G é possível essa interação a qualquer lugar e momento.

A figura 5.8 mostra a tela inicial de interação com o ambiente através de um *smartphone* Android. Neste primeiro momento é possível selecionar a ação a ser realizada no ambiente.

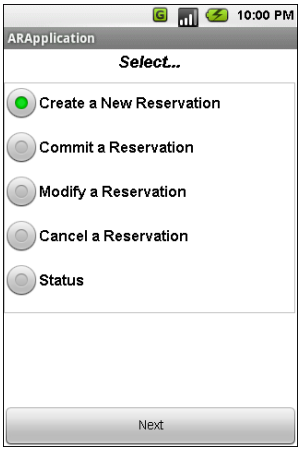


Figura 5.8: Tela Inicial

Para criar uma reserva, o usuário deve selecionar a opção *Create a New Reservation*. Depois de selecionar essa opção, o usuário deve especificar os requisitos desta reserva. Neste momento, informa-se o tempo inicial da reserva, sua duração e o número de processadores necessários. O tipo de granularidade da aplicação depende do conhecimento do usuário. Portanto, se o mesmo possui conhecimento, marca o *checkbox* e seleciona um dos níveis de granularidade. Caso não possua esse conhecimento, deixa o meta-escalonador reservar os recursos desconsiderando essa informação. A figura 5.9 mostra a interação de um usuário que possui conhecimento sobre sua aplicação.

Após a criação da reserva o usuário recebe a confirmação que pode

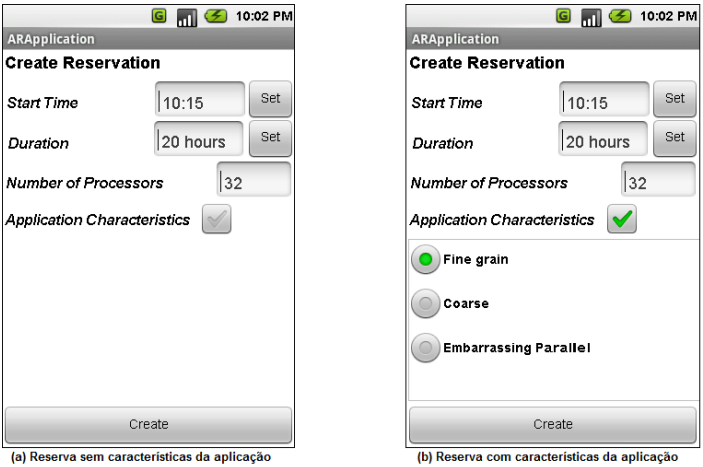


Figura 5.9: Interface de criação de uma nova reserva

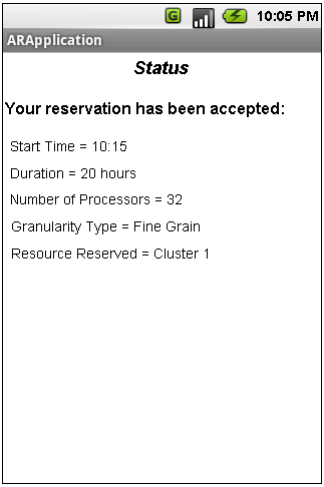


Figura 5.10: Confirmação de uma nova reserva

ser visualizada na figura 5.10.

Dada a limitação do GridSim de não permitir uma interação em tempo real, os testes realizados através da interface geraram um *Workload*, que é validado no ambiente simulado. Portanto, alguns parâmetros que deveriam ser necessários em um ambiente real, como por exemplo, código do executável, binário, ou até mesmo as datas das reservas, não foram considerados nessa interface.

5.5 Considerações Finais

Através dos testes realizados pôde-se observar que, com a abordagem proposta, é possível realizar uma reserva com base nas características das aplicações, considerando o nível de granularidade dessas e, consequentemente, reservando os recursos que melhor se adéquem às aplicações. Com isso, obteve-se uma melhor distribuição das reservas nos cenários testados, contribuindo para um melhor balanceamento de reservas futuras.

Também se comprovou que a qualidade das reservas influencia diretamente no desempenho das aplicações executadas. Com a abordagem proposta obteve-se uma redução no tempo de execução das aplicações quando comparadas a outros algoritmos. Essa redução, considerando o somatório de todos os tempos de execução das aplicações previamente reservadas, foi de 6,91% para o segundo melhor caso (*AR Biggest Free* e de 14,21% para o pior caso (*AR Aleatório*).

Por fim, mostrou-se como é feita uma reserva de recursos através de um dispositivo móvel. Esses aparelhos permitem, através de recursos, como por exemplo, internet 3G, uma interação com o ambiente a partir de qualquer lugar. Para tal propósito, foi criada uma interface amigável que permitisse que o usuário informasse os requisitos da sua reserva, bem como o nível de granularidade da aplicação, caso o usuário possua esse conhecimento.

A tabela 5.10 mostra a comparação entre as principais características dos trabalhos relacionados com as características da abordagem apresentada nesta dissertação.

Tabela 5.10: Quadro comparativo

	[Siddiqui et al., 2005]	[Takefusa et al., 2007]	[Rossetto et al., 2007]	[Silva and Dantas, 2007]	[Vahdat-Nejad et al., 2007]	Este trabalho
Reserva Recursos	Sim	Sim	Não	Não	Não	Sim
Descrição Características Aplicação	Não	Não	Não	Não	Sim	Sim
Interface Móvel	Não	Não	Sim	Não	Não	Sim
Múltiplas VOs	Sim	Sim	Não	Sim	Sim	Não
Ontologia	Sim	Não	Não	Sim	Não	Sim
Tipo de Recurso para Reserva	nº de processadores, SO	nº de processadores, banda	Não	Não	Não	nº de processadores

Capítulo 6

Conclusões e Trabalhos Futuros

Neste trabalho foram abordados problemas encontrados em ambientes de grades computacionais com suporte à reserva antecipa de recursos, isto é, maior qualidade na reserva de recursos e interface móvel de acesso para a realização de tais reservas. Observou-se que os ambientes de grades computacionais que permitem que recursos sejam reservados para um futuro uso não consideram as características comportamentais das aplicações. Ainda, a eficiência de uma reserva depende do conhecimento do usuário sobre o ambiente e de suas requisições em termos de hardware e software. Outra carência encontrada nestes ambientes é a falta de uma interface móvel que possibilite a reserva de recursos e não somente a submissão e monitoramento de aplicações. Desta forma, a abordagem desta dissertação visou suprir esses problemas, melhorando o nível de adequação de uma reserva e explorando os dispositivos móveis como interface de interação para realização dessas reservas.

Neste contexto, a dissertação apresentou uma proposta de arquitetura para reserva antecipada de recursos em grades computacionais com configurações de *multi-clusters*. Essa arquitetura tem por objetivo melhorar a qualidade dessas reservas com base nas características das aplicações, e fornecer uma interface para a realização dessas reservas através de dispositivos móveis.

Através de um estudo realizado em ambientes de grades computacionais e da utilização de um ambiente com um meta-escalonador (o CSF) foi possível detalhar conceitos fundamentais sobre meta-escalonamento, co-reserva e co-alocação de recursos. Além disso, dois aspectos importantes foram identificados nestes ambientes. O primeiro corresponde à importância do suporte à reserva antecipada, que permite que recursos sejam alocados para serem utilizados, por exemplo, durante o período noturno. Ainda, a reserva garante uma QoS, pois, diferentemente de abordagens sem esse suporte, não mantém as aplicações em filas. O segundo aspecto identificado está relacionado às características comportamentais de uma aplicação, considerando, nesse caso, o nível de granularidade. Essa pro-

priedade é importante, pois no momento da realização de uma reserva o desempenho da interconexão da rede deve ser considerado.

Desta forma, nesta dissertação as reservas são realizadas através das requisições dos usuários e são consideradas as características comportamentais da aplicação, possibilitando explorar a alocação de recursos de forma mais eficiente e, por conseguinte, maximizando a execução da aplicação no ambiente. Portanto, fica a cargo do meta-escalonador a decisão de qual recurso é o mais adequado a ser alocado. Para este fim, foi utilizada a lógica *fuzzy* proposta por [Grumiche et al., 2010], que adaptou ontologias de descrição de recursos e aplicação propostas por [Silva and Dantas, 2007].

O trabalho desenvolvido, além de fornecer uma revisão bibliográfica sobre ambientes de grades computacionais, introduziu um novo conceito nestes ambientes distribuídos: a qualidade de experiência (QoE). Esse conceito, bastante recente, trata da forma que o usuário se sente em relação às suas expectativas e requerimentos. A arquitetura desenvolvida visa, dessa maneira, melhorar tal experiência considerando as características da aplicação na requisição de uma reserva. Porém fica como trabalho futuro medir essa qualidade de experiência.

Para validação da proposta, utilizou-se o simulador GridSim, que apesar de algumas limitações, atendeu às necessidades esperadas. Os resultados experimentais compararam o trabalho desenvolvido com outros algoritmos para a seleção de recursos a serem reservados. Os resultados mostraram-se satisfatórios, pois se conseguiu uma melhor distribuição das reservas com as mesmas adequadas conforme as particularidades das aplicações e, consequentemente, obteve-se um bom desempenho na execução das aplicações reservadas.

Em termos de trabalhos futuros, espera-se realizar os seguintes experimentos e extensões da abordagem proposta, contribuindo ainda mais para uma melhor QoE e gerenciamento deste ambiente:

- Permitir que ocorra co-reserva entre diferentes *clusters*, porém, garantindo que aplicações de granularidade fina não sejam prejudicadas;

- Aumentar o número de características a serem requisitadas para a reserva, adaptando a adequação ao recurso;

- Integrar ao módulo de autenticação, já implementado em um trabalho conjunto, como se pode observar em [Viera et al., 2010] e [Rocha et al., 2010], acrescentando um sistema de autenticação ao trabalho desenvolvido nessa dissertação;

- Implementar as ontologias de descrição da aplicação e ontologias de descrição dos recursos computacionais no dispositivo móvel, realizando um pré-processamento de *matching* no aparelho, ou seja, fazendo uma pré-seleção de recursos baseados nas características da aplicação. Esta etapa encontra-se em desenvolvimento;
- Trabalhar com métricas para medir a QoE;
- Realizar testes em um ambiente real, criando interfaces de comunicação com gerenciadores de recursos locais.

Referências Bibliográficas

[Abawajy and Dandamudi, 2003] Abawajy, J. and Dandamudi, S. (2003). Parallel job scheduling on multicluster computing systems. pages J. H. Abawajy, S. P. Dandamudi. "Parallel Job Scheduling on Multicluster Computing Systems,"cluster, p. 11, IEEE International Conference on Cluster Computing (CLUSTER'03), 2003.

[Adarsh, 2011] Adarsh (2011). Grid-Distributed-Cluster Computing. http://www.adarshpatil.com/grid.htm.

[Ahuja and Myers, 2006] Ahuja, S. and Myers, J. (2006). A survey on wireless grid computing. *The Journal of Supercomputing*, 37(1):3–21.

[Anderson et al., 2002] Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. (2002). SETI@ home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61.

[Bai et al., 2006] Bai, Y., Zhuang, H., Wang, D., and NetLibrary, I. (2006). *Advanced fuzzy logic technologies in industrial applications*. Springer.

[Bell et al., 2003] Bell, W., Cameron, D., Millar, A., Capozza, L., Stockinger, K., and Zini, F. (2003). Optorsim: A grid simulator for studying dynamic data replication strategies. *International Journal of High Performance Computing Applications*, 17(4):403.

[Berman et al., 2003] Berman, F., Fox, G., and Hey, A. (2003). *Grid computing: making the global infrastructure a reality*. John Wiley & Sons Inc.

[Bernstein, 1996] Bernstein, P. (1996). Middleware: a model for distributed system services. *Communications of the ACM*, 39(2):86–98.

[Bettini et al., 2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180.

[Black and Edgar, 2008] Black, M. and Edgar, W. (2008). Exploring mobile devices as Grid resources: Using an x86 virtual machine to run BOINC on an iPhone. In *Grid Computing, 2008 10th IEEE/ACM International Conference on*, pages 9–16. IEEE.

[Bote-Lorenzo et al., 2004] Bote-Lorenzo, M., Dimitriadis, Y., and Gómez-Sánchez, E. (2004). Grid characteristics and uses: a grid definition. In *Grid Computing*, pages 291–298. Springer.

[Bouziane et al., 2008] Bouziane, H., Pérez, C., and Priol, T. (2008). A software component model with spatial and temporal compositions for grid infrastructures. *Euro-Par 2008–Parallel Processing*, pages 698–708.

[Butler et al., 2000] Butler, R., Welch, V., Engert, D., Foster, I., Tuecke, S., Volmer, J., and Kesselman, C. (2000). A national-scale authentication infrastructure. *Computer*, 33(12):60–66.

[Buyya, 1999] Buyya, R. (1999). High performance cluster computing: Architectures and systems. 1, 1/e.

[Buyya, 2011] Buyya, R. (2011). Grid Computing Info Centre (GRID Infoware). http://gridcomputing.com/.

[Buyya et al., 2000] Buyya, R., Abramson, D., and Giddy, J. (2000). Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid. In *hpc*, page 283. Published by the IEEE Computer Society.

[Buyya and Venugopal, 2004] Buyya, R. and Venugopal, S. (2004). The gridbus toolkit for service oriented grid and utility computing: An overview and status report. In *Grid Economics and Business Models, 2004. GECON 2004. 1st IEEE International Workshop on*, pages 19–66. IEEE.

[Cannataro and Talia, 2003] Cannataro, M. and Talia, D. (2003). The knowledge grid. *Communications of the ACM*, 46(1):89–93.

[Casanova et al., 2008] Casanova, H., Legrand, A., and Quinson, M. (2008). Simgrid: a generic framework for large-scale distributed experiments. In *Tenth International Conference on Computer Modeling and Simulation*, pages 126–131. IEEE.

[Cerami and Laurent, 2002] Cerami, E. and Laurent, S. (2002). *Web services essentials*. O'Reilly & Associates, Inc. Sebastopol, CA, USA.

[Chapin et al., 1999] Chapin, S., Katramatos, D., Karpovich, J., and Grimshaw, A. (1999). The legion resource management system. In *Job Scheduling Strategies for Parallel Processing*, pages 162–178. Springer.

[Childers et al., 2000] Childers, L., Disz, T., Olson, R., Papka, M., Stevens, R., and Udeshi, T. (2000). Access grid: Immersive group-to-group collaborative visualization. In *Proc. 4th International Immersive Projection Technology Workshop*. Citeseer.

[Christensen et al., 2001] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001). Web services description language (WSDL) 1.1.

[Chu and Humphrey, 2004] Chu, D. and Humphrey, M. (2004). Mobile ogsi. net: Grid computing on mobile devices.

[Cirne et al., 2003] Cirne, W., Brasileiro, F., Sauvé, J., Andrade, N., Paranhos, D., Santos-neto, E., and Medeiros, R. (2003). Grid computing for bag of tasks applications. In *In Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment*. Citeseer.

[Condor, 2011] Condor (2011). Condor High Throughput Computing. http://www.cs.wisc.edu/condor/.

[Corrie et al., 2003] Corrie, B., Wong, H., Zimmerman, T., Marsh, S., Patrick, A., Singer, J., Emond, B., and No
"el, S. (2003). Towards quality of experience in advanced collaborative environments. In *Third Annual Workshop on Advanced Collaborative Environments*, pages 1–8. Citeseer.

[CSF, 2011] CSF (2011). Community Scheduler Framework. http://www.globus.org/grid_software/computation/csf.php.

[Czajkowski et al., 2005] Czajkowski, K., Ferguson, D., Foster, I., Frey, J., Graham, S., Sedukhin, I., Snelling, D., Tuecke, S., and Vambenepe, W. (2005). Web services resource framework (wsrf). *Globus Alliance and IBM*.

[Czajkowski et al., 1998] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., and Tuecke, S. (1998). A resource management architecture for metacomputing systems. In *Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer.

[Dantas, 2005] Dantas, M. (2005). *Computação distribuída de alto desempenho: redes, clusters e grids computacionais*. Axcel Books.

[De Roure et al., 2003] De Roure, D., Baker, M., Jennings, N., and Shadbolt, N. (2003). The evolution of the Grid. *Grid computing: making the global infrastructure a reality*, 13:14–15.

[De Roure et al., 2005] De Roure, D., Jennings, N., and Shadbolt, N. (2005). The semantic grid: past, present, and future. *Proceedings of the IEEE*, 93(3):669–681.

[Ding et al., 2008] Ding, Z., Wei, X., Zhu, Y., Yuan, Y., Li, W., and Tatebe, O. (2008). Implement the Grid Workflow Scheduling for Data Intensive Applications with CSF4. In *Fourth IEEE International Conference on eScience*, pages 563–569. IEEE.

[Eickermann et al., 2007] Eickermann, T., Frings, W., W "aldrich, O., Wieder, P., and Ziegler, W. (2007). Co-allocation of mpi jobs with the viola grid metascheduling framework. In *Proc. German e-Sci. Conf*, pages 1–10. Citeseer.

[Fafner, 2011] Fafner (2011). Fafner overview. http://www.npac.syr.edu/factoring.html.

[Foster, 2002] Foster, I. (2002). What is the grid? a three point checklist. *GRID today*, 1(6):32–36.

[Foster, 2006] Foster, I. (2006). Globus toolkit version 4: Software for service-oriented systems. *Journal of Computer Science and Technology*, 21(4):513–520.

[Foster et al., 2005] Foster, I., Czajkowski, K., Ferguson, D., Frey, J., Graham, S., Maguire, T., Snelling, D., and Tuecke, S. (2005). Modeling and managing state in distributed systems: The role of OGSI and WSRF. *Proceedings of the IEEE*, 93(3):604–612.

[Foster et al., 1996] Foster, I., Geisler, J., Nickless, B., Smith, W., and Tuecke, S. (1996). Software infrastructure for the I-WAY high-performance distributed computing experiment. In *hpdc*, page 562. Published by the IEEE Computer Society.

[Foster and Kesselman, 1999] Foster, I. and Kesselman, C. (1999). The globus toolkit. *The grid: blueprint for a new computing infrastructure*, pages 259–278.

[Foster and Kesselman, 2004] Foster, I. and Kesselman, C. (2004). *The grid: blueprint for a new computing infrastructure*. Morgan Kaufmann.

[Foster et al., 2002] Foster, I., Kesselman, C., Lee, C., Lindell, B., Nahrstedt, K., and Roy, A. (2002). A distributed resource management architecture that supports advance reservations and co-allocation. In *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on*, pages 27–36. IEEE.

[Foster et al., 2001] Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–222.

[Foster et al., 2008] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee.

[Gomes et al., 2007] Gomes, A., Ziviani, A., Lima, L., and Endler, M. (2007). Dichotomy: A resource discovery and scheduling protocol for multihop ad hoc mobile grids. *Proceedings of the 1st Workshop on Context-Awareness and Mobility in Grid Computing (WCAMG), 7th IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*.

[Google, 2011] Google (2011). Android. http://www.android.com.

[Graupner et al., 2003] Graupner, S., Pruyne, J., and Singhal, S. (2003). Making the utility data center a power station for the enterprise grid. *Hewlett Packard Laboratories, Tech. Rep. HPL-2003-53, March*, pages 2003–53.

[Grid Engine, 2011] Grid Engine (2011). Open Grid Scheduler. http://gridscheduler.sourceforge.net/.

[Grumiche et al., 2010] Grumiche, R., Vilain, P., and Dantas, M. (2010). Resource selection based on application features. *Proceedings of the 9th International Information and Telecommunication Technologies Symposium*.

[Hamscher et al., 2000] Hamscher, V., Schwiegelshohn, U., Streit, A., and Yahyapour, R. (2000). Evaluation of job-scheduling strategies for grid computing. *Grid Computing - GRID 2000*, pages 191–202.

[Han and Kamber, 2006] Han, J. and Kamber, M. (2006). *Data mining: concepts and techniques*. Morgan Kaufmann.

[Hey and Trefethen, 2002] Hey, T. and Trefethen, A. (2002). The UK e-science core programme and the grid. *Future Generation Computer Systems*, 18(8):1017–1031.

[Jankowski et al., 2006] Jankowski, G., Kovacs, J., Meyer, N., Januszewski, R., and Mikolajczak, R. (2006). Towards checkpointing grid architecture. *Parallel Processing and Applied Mathematics*, pages 659–666.

[Javadi et al., 2006] Javadi, B., Akbari, M., and Abawajy, J. (2006). A performance model for analysis of heterogeneous multi-cluster systems. *Parallel computing*, 32(11-12):831–851.

[Krauter et al., 2002] Krauter, K., Buyya, R., and Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164.

[Levesque, 1990] Levesque, J. (1990). Forge 90: a parallel programming environment. In *Compcon Spring'92. Thirty-Seventh IEEE Computer Society International Conference, Digest of Papers.*, pages 291–294. IEEE.

[Lewis and Grimshaw, 1996] Lewis, M. and Grimshaw, A. (1996). The core Legion object model. In *Proc. 5th IEEE Symp. on High Performance Distributed Computing*, pages 562–571. IEEE Computer Society Press.

[Liu and Zsu, 2009] Liu, L. and Zsu, M. (2009). *Encyclopedia of database systems*. Springer Publishing Company, Incorporated.

[Min and Maheswaran, 2001] Min, R. and Maheswaran, M. (2001). Scheduling advance reservations with priorities in grid computing systems. *Proceedings of Parallel and Distributed Computing and Systems (PDCS'01)*, pages 172–176.

[Mu'alem and Feitelson, 2001] Mu'alem, A. and Feitelson, D. (2001). Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. *Parallel and Distributed Systems, IEEE Transactions on*, 12(6):529–543.

[MySQL, 2011] MySQL (2011). MySQL Manual. http://ftp.ntu.edu.tw/ftp/pub/MySQL/doc/mysql-cluster-excerpt/5.1/en/mysql-cluster-multi-computer.html?ff=nopfpls.

[Németh and Sunderam, 2005] Németh, Z. and Sunderam, V. (2005). A formal framework for defining grid systems. In *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*. IEEE.

[Novotny et al., 2004] Novotny, J., Russell, M., and Wehrens, O. (2004). Gridsphere: a portal framework for building collaborations. *Concurrency and Computation: Practice & Experience*, 16(5):503–513.

[OGE, 2011] OGE (2011). Oracle Grid Engine. http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

[Ohishi, 2006] Ohishi, M. (2006). International virtual observatory alliance. *Proceedings of the International Astronomical Union*, 2(14):528–529.

[Ourgrid, 2011] Ourgrid (2011). OurGrid. http://www.ourgrid.org/.

[Parashar and Pierson, 2007] Parashar, M. and Pierson, J. (2007). When the Grid becomes pervasive: A vision on Pervasive Grids. In *Proceedings of the 8th Hellenic European Research on Computer Mathematics & its Applications Conference*. Citeseer.

[Park et al., 2003] Park, S., Ko, Y., and Kim, J. (2003). Disconnected operation service in mobile grid computing. *Service-Oriented Computing-ICSOC 2003*, pages 499–513.

[PBS, 2011] PBS (2011). PBS Works - Enabling On-Demand Computing. http://www.pbsgridworks.com.

[Pinto et al., 2008] Pinto, L., Mendonça, R., and Dantas, M. (2008). Impact of interconnection networks and application granularity to compound cluster environments. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 468–473. IEEE.

[Platform Computing, 2011] Platform Computing (2011). Platform LSF - The HPC Workload Management Standard. http://www.platform.com/workload-management/high-performance-computing.

[Pourreza et al., 2006] Pourreza, H., Eskicioglu, M., and Graham, P. (2006). Performance assessment of four cluster interconnects on identical hardware: hints for cluster builders. *International Journal of High Performance Computing and Networking*, 4(5):270–285.

[Press et al., 1996] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1996). *Numerical recipes in Fortran 90: the art of parallel scientific computing*.

[PVM Projects, 2011] PVM Projects (2011). PVM: Parallel Virtual Machine. http://www.csm.ornl.gov/pvm/.

[Qin and Bauer, 2009] Qin, J. and Bauer, M. (2009). Job co-allocation strategies for multiple high performance computing clusters. *Cluster Computing*, 12(3):323–340.

[Rajachandrasekar et al., 2009] Rajachandrasekar, R., Nagarajan, R., Sridhar, G., and Sumathi, G. (2009). Mobile Device Interface to Access Computational Grid. In *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, pages 363–365. IEEE.

[Rimal et al., 2009] Rimal, B., Choi, E., and Lumb, I. (2009). A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 44–51. IEEE.

[Roblitz and Reinefeld, 2005] Roblitz, T. and Reinefeld, A. (2005). Coreservation with the concept of virtual resources. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 1, pages 398–406. IEEE.

[Rocha et al., 2010] Rocha, C., Viera, M., Capretz, M., Bauer, M., Augustin, I., and Dantas, M. (2010). A user-centric authentication for advanced resource reservation in mobile grid environments. In *Proceedings of the 2010 International Conference on Grid Computing and Applications, GCA'10*.

[Rossetto et al., 2007] Rossetto, A., Borges, V., Silva, A., and Dantas, M. (2007). SuMMIT-A framework for coordinating applications execution in mobile grid environments. *Proceeding GRID '07 Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pages 129–136.

[Sangket et al., 2010] Sangket, U., Mahasirimongkol, S., Chantratita, W., Tandayya, P., and Aulchenko, Y. (2010). Parallabel: an r library for generalized parallelization of genome-wide association studies. *BMC bioinformatics*, 11(1):217.

[Seymour et al., 2005] Seymour, K., YarKhan, A., Agrawal, S., and Dongarra, J. (2005). Netsolve: Grid enabling scientific computing environments. *Advances in Parallel Computing*, 14:33–51.

[Shaikh et al., 2010] Shaikh, J., Fiedler, M., and Collange, D. (2010). Quality of experience from user and network perspectives. *Annals of Telecommunications*, 65(1):47–57.

[Siddiqui and Fahringer, 2005] Siddiqui, M. and Fahringer, T. (2005). Gridarm: Askalon's grid resource management system. *Advances in Grid Computing-EGC 2005*, pages 122–131.

[Siddiqui et al., 2005] Siddiqui, M., Villazon, A., Prodan, R., and Fahringer, T. (2005). Advanced reservation and co-allocation of grid resources: A step towards an invisible grid. In *9th International Multitopic Conference, IEEE INMIC 2005*, pages 1–6. IEEE.

[Silva and Dantas, 2007] Silva, A. and Dantas, M. (2007). A selector of grid resources based on the semantic integration of multiple ontologies. In *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pages 143 –150.

[Smith et al., 2000] Smith, W., Foster, I., and Taylor, V. (2000). Scheduling with advanced reservations. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 127–132. IEEE.

[Snir, 1998] Snir, M. (1998). *MPI–the Complete Reference: The MPI core*. The MIT Press.

[Sobral and Proença, 2001] Sobral, J. and Proença, A. (2001). A scoopp evaluation on packing parallel objects in run-time. *Vector and Parallel Processing - VECPAR 2000*, pages 114–127.

[Sodan, 2005] Sodan, A. (2005). Loosely coordinated coscheduling in the context of other approaches for dynamic job scheduling: a survey. *Concurrency and Computation: Practice and Experience*, 17(15):1725–1781.

[Song et al., 2000] Song, H., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K., and Chien, A. (2000). The microgrid: a scientific tool for modeling computational grids. *Proceedings of IEEE Supercomputing (SC 2000)*.

[Sulistio, 2008] Sulistio, A. (2008). Advance reservation and revenue-based resource management for grid systems. *Proceedings International Parallel and Distributed Processing Symposium (IPDPS)*.

[Sulistio and Buyya, 2004] Sulistio, A. and Buyya, R. (2004). A grid simulation infrastructure supporting advance reservation. In *16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, pages 1–7. Citeseer.

[Takefusa et al., 2007] Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y., and Sekiguchi, S. (2007). GridARS: an advance reservation-based grid co-allocation framework for distributed computing and network resources. In *Proceedings of the 13th international conference on Job scheduling strategies for parallel processing*, pages 152–168. Springer-Verlag.

[Talia, 2002] Talia, D. (2002). The open grid services architecture: where the grid meets the web. *IEEE Internet Computing*, 6(6):71.

[TeraGrid, 2002] TeraGrid (2002). Grid computing info centre. http://www.teragridforum.org/mediawiki/images/c/cd/Schedwg_-RsrvCoschedReport.pdf.

[Tuecke et al., 2003] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., et al. (2003). Open grid services infrastructure (OGSI). *Global Grid Forum Draft Recommendation*.

[Vahdat-Nejad et al., 2007] Vahdat-Nejad, H., Monsefi, R., and Naghibzadeh, M. (2007). A new fuzzy algorithm for global job scheduling in multiclusters and grids. In *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on*, pages 54–58. IEEE.

[Venugopal et al., 2004] Venugopal, S., Buyya, R., and Winton, L. (2004). A grid service broker for scheduling distributed data-oriented applications on global grids. In *Proceedings of the 2nd workshop on Middleware for grid computing*, pages 75–80. ACM.

[Vidal et al., 2007] Vidal, A., da Silva e Silva, F., Kofuji, S., and Kon, F. (2007). Semantics-based grid resource management. In *Proceedings*

of the 5th international workshop on Middleware for grid computing: held at the ACM/IFIP/USENIX 8th International Middleware Conference, pages 1–6. ACM.

[Viera et al., 2010] Viera, M., Rocha, C., Capretz, M., Bauer, M., and Dantas, M. (2010). Toward advance resource reservation in mobile grid configurations based on user-centric authentication. In *Anais do VIII Workshop em Clouds, Grids e Aplicações, WCGA'2010*. SBC.

[Xiaohui et al., 2006] Xiaohui, W., Zhaohui, D., Shutao, Y., Chang, H., and Huizhen, L. (2006). Csf4: A wsrf compliant meta-scheduler. *World Congress in Computer Science, Computer Engineering, and Applied Computing, GCA*, 6:61–67.

[XtremWeb, 2011] XtremWeb (2011). XtremWeb : the Open Source Platform for Desktop Grids. http://www.xtremweb.net/.

[Yeo et al., 2006a] Yeo, C., Buyya, R., Pourreza, H., Eskicioglu, R., Graham, P., and Sommers, F. (2006a). Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers. *Handbook of Nature-Inspired and Innovative Computing*, pages 521–551.

[Yeo et al., 2006b] Yeo, C., De Assuncao, M., Yu, J., Sulistio, A., Venugopal, S., Placek, M., and Buyya, R. (2006b). Utility Computing and Global Grids. *Arxiv preprint cs/0605056*.

[Zeng et al., 2008] Zeng, W.-Y., Zhao, Y.-L., Zeng, J.-W., and Song, W. (2008). Mobile grid architecture design and application. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1 –4.

Apêndice A

Publicações

Este apêndice visa ilustrar as publicações realizadas ao longo da pesquisa desse trabalho de dissertação.

A.1 Trabalhos Completos Publicados em Anais de Congressos

- **Título:** A User-Centric Authentication for Advanced Resource Reservation in Mobile Grid Environments
- **Evento:** GCA 2010 – The 2010 International Conference on Grid Computing and Applications
- **Local:** Las Vegas, Nevada, EUA
- **Data:** Julho de 2010
- **Autores:** Matheus A. Viera, Cristiano C. Rocha, Miriam Capretz, Michael A. Bauer, Iara Augustin e Mário A. R. Dantas
- **Estrato Qualis/CAPES:** B3

- **Título:** A Context-Aware Authentication Approach Based on Behavioral Definitions
- **Evento:** IKE 2010 – The 2010 International Conference on Information and Knowledge Engineering
- **Local:** Las Vegas, Nevada, EUA
- **Data:** Julho de 2010
- **Autores:** Matheus A. Viera, Cristiano C. Rocha, João Carlos D. Lima, Miriam Capretz, Michael A. Bauer, Iara Augustin e Mário A. R. Dantas

- **Estrato Qualis/CAPES:** B4

- **Título:** Toward Advance Resource Reservation in Mobile Grid Configurations Based on User-Centric Authentication
- **Evento:** WCGA 2010 – 8th Workshop on Clouds, Grids and Applications
- **Local:** Gramado, RS, Brasil
- **Data:** Maio de 2010
- **Autores:** Matheus A. Viera, Cristiano C. Rocha, Miriam Capretz, Michael A. Bauer e Mário A. R. Dantas
- **Estrato Qualis/CAPES:** B5

- **Título:** Uma Arquitetura de Reserva Antecipada de Recursos Centrada no Usuário para Ambientes de Grades Móveis
- **Evento:** I2TS 2009 – 8th International Information and Telecommunication Technologies Symposium
- **Local:** Florianópolis, SC, Brasil
- **Data:** Dezembro de 2009
- **Autores:** Matheus A. Viera, Cristiano C. Rocha, Luis Fernando Friedrich, Iara Augustin e Mário A. R. Dantas

- **Título:** Grid Computing Middleware Survey
- **Evento:** I2TS 2008 – 7th International Information and Telecommunication Technologies Symposium
- **Local:** Foz do Iguaçu, PR, Brasil
- **Data:** Novembro de 2008
- **Autores:** Matheus A. Viera, Cristiano C. Rocha, Denise J. Ferreira, Miguel Cartagena, Rodrigo G. Silva e Mário A. R. Dantas