UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Eduardo Camilo Inacio

UM MÉTODO DE SELEÇÃO DE RECURSOS EM AMBIENTES DISTRIBUÍDOS BASEADO EM ONTOLOGIA E ORIENTAÇÃO AO CONTEXTO

Florianópolis – SC
2013

Eduardo Camilo Inacio

UM MÉTODO DE SELEÇÃO DE RECURSOS EM AMBIENTES DISTRIBUÍDOS BASEADO EM ONTOLOGIA E ORIENTAÇÃO AO CONTEXTO

Monografia submetida à Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Ph. D. Mario Antonio Ribeiro Dantas

Florianópolis – SC

2013

Eduardo Camilo Inacio

UM MÉTODO DE SELEÇÃO DE RECURSOS EM AMBIENTES DISTRIBUÍDOS BASEADO EM ONTOLOGIA E ORIENTAÇÃO AO CONTEXTO

Esta monografia foi julgada adequada para a obtenção do Grau de Bacharel em Sistemas de Informação, e aprovado em sua forma final pela Universidade Federal de Santa Catarina.

Florianópolis, 27 de Maio de 2013.

_____

Prof. Dr. Leandro José Komosinski
Coordenador do Curso de Sistemas de Informação

Banca Examinadora:

_____

Prof. Ph. D. Mario Antonio Ribeiro Dantas
Orientador

_____

Prof. Dr. José Leomar Todesco
Universidade Federal de Santa Catarina

_____

Prof.ª. Dr ª. Lúcia Helena Martins Pacheco
Universidade Federal de Santa Catarina

Dedico este trabalho
a minha esposa, Alessandra,
pelo apoio e compreensão,
e aos meus pais, José e Liege,
pelo incentivo aos estudos.

AGRADECIMENTOS

Este trabalho foi realizado no Laboratório de Pesquisas em Sistemas Distribuídos (LaPeSD) da Universidade Federal de Santa Catarina (UFSC) sob orientação do professor Mario Antonio Ribeiro Dantas, a quem devo meus mais profundos agradecimentos pelo auxílio e motivação dispensados.

Ao todo grupo docente do curso de Sistemas de Informação, pelos conhecimentos transmitidos, que muito colaboraram para minha formação.

Ao pesquisador e amigo, Matheus Anversa Viera, pelas opiniões e sugestões em diversos momentos de dificuldade durante a realização deste trabalho.

Aos demais colegas do LaPeSD que estiveram envolvidos de alguma forma neste trabalho. Apesar de não tê-los citados a todos aqui, saibam da minha sincera gratidão.

“O cientista não é o homem que fornece as verdadeiras respostas;
é quem faz as verdadeiras perguntas”.
Claude Lévi-Strauss

RESUMO

O processo de seleção de recursos é um dos mecanismos mais importantes de um ambiente distribuído, pois influencia diretamente na otimização do seu uso. Em sistemas gerenciadores de recursos e *middlewares* de computação distribuída, é comum se observar a comparação exata de valores, baseada em palavras-chaves, como método de seleção. Este tipo de método restringe a capacidade de otimização de recursos apenas para usuários com alto conhecimento do ambiente distribuído.

Por meio do uso de uma ontologia de domínio, é possível tornar o conhecimento do ambiente distribuído acessível ao processo de seleção. Com esse conhecimento, o processo de seleção pode descobrir alternativas para o atendimento de requisições, aumentando a capacidade do sistema. Tornando o processo de seleção sensível ao contexto, pode se otimizar a utilização dos recursos pela consideração de informações não especificadas pelo usuário.

Neste trabalho é proposto um método de seleção de recursos em ambientes distribuídos que faz uso de ontologia e de sensibilidade ao contexto. Pela utilização deste método, é possível melhorar o processo de seleção através da redução do tempo de espera pelo atendimento de requisições e do tempo total, percebido pelo usuário, para a execução de um conjunto de requisições.

A validação do método se deu por meio da realização de um conjunto de experimentos. Esses experimentos foram divididos em três cenários e realizados em um ambiente distribuído controlado. O ambiente foi criado no Laboratório de Pesquisas em Sistemas Distribuídos (LaPeSD) da Universidade Federal de Santa Catarina (UFSC). Nos testes, o método proposto foi comparado ao método baseado em palavras-chave, implementado pelo sistema gerenciador de recursos HTCondor.

Para todos os cenários de teste aplicados, os resultados obtidos com o método proposto neste trabalho foram melhores que os observados com o método baseado em palavras-chave. Estes resultados foram atribuídos a ampliação da capacidade de seleção, proporcionada pelo uso da ontologia, e a otimização na utilização dos recursos, consequente da orientação do processo ao contexto do ambiente e da requisição do usuário.

Palavras-chave: sistemas distribuídos, ontologia, sensibilidade ao contexto, seleção de recursos.

ABSTRACT

The resource matching process is one of the most important mechanism of a distributed environment, because it directly influences the optimization of its use. In resource management systems and distributed computing middlewares, it is commonly observed the keyword-based resource selection as selection method. This kind of method restricts the capacity of resource optimization only to users with high knowledge of the distributed environment.

Through the use of a domain ontology, it is possible to make knowledge of the distributed environment accessible to the selection process. With this knowledge, the matching process can uncover alternatives for answering requests, increasing system capacity. Making the matching process context-aware, can optimize the use of resources by considering information not specified by the user.

This work proposes a method of resource matching in distributed environments that make use of ontology and context awareness. By using this method, it is possible to improve the matching process by reducing the waiting time for servicing of requests and the total time perceived by the user to perform a set of requests.

The method validation was made by performing a set of experiments. These experiments were divided into three sets and performed in a controlled distributed environment. The environment has been created at Research Laboratory of Distributed Systems (LaPeSD) at Federal University of Santa Catarina (UFSC). On the tests, the proposed method was compared to the keyword-based method, implemented by the resource management system HTCondor.

For all test scenarios applied, the results obtained with the method proposed in this work were better than those observed with the keyword-based method. These results were attributed to expansion of matching capacity, provided by the use of ontology, and the optimization of resources use, resulting from the process orientation to the context of environment and user request.

Keywords: Distributed systems, ontology, context awareness, resource matching.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API - Application Programming Interface

CAP - Context-Aware Packet

CAPEUS - Context-Aware Packets Enabling Ubiquituous Services

CIM - Common Information Model

CML - Context Modelling Language

DMTF - Distributed Management Task Force

LaPeSD - Laboratório de Pesquisas em Sistemas Distribuídos

MIMD - Multiple Instruction Multiple Data

MISD - Multiple Instruction Single Data

MPI - Message Passing Interface

MPP - Massively Parallel Processors

NUMA - Non-Uniform Memory Access

OMG - Object Management Group

OWL - Web Ontology Language

PVM - Parallel Virtual Machine

RDF - Resource Description Framework

RDFS - Resource Description Framework Schema

RMS - Resource Management System

RTT - Round-Trip Time

SIMD - Single Instruction Multiple Data

SISD - Single Instruction Single Data

SMP - Symmetric Multiprocessors

SOAP - Simple Object Access Protocol

SSI - Single System Image

UFSC - Universidade Federal de Santa Catarina

UML - Unified Modelling Language

URI - Uniform Resource Identifier

VO - Virtual Organization

W3C - World Wide Web Consortium

WWW - World Wide Web

XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A computação distribuída e paralela é vista como uma alternativa interessante para a execução de sistemas com demanda de alto desempenho. A possibilidade de compartilhamento de recursos através da agregação de computadores interconectados por uma rede é uma das principais motivações para a criação e utilização de sistemas distribuídos.

Contudo, a agregação de recursos computacionais e a coordenação das suas ações implica em superar alguns desafios. A capacidade de tratar a heterogeneidade do ambiente, que pode se apresentar em termos de equipamentos, sistemas operacionais, linguagens de programação, entre outros, é um deles. Outro exemplo está relacionado a capacidade de administrar os recursos disponíveis, incluindo a seleção adequada dos recursos necessários para cada requisição enviada ao ambiente.

O processo de seleção de recursos é de especial interesse porque influencia diretamente na otimização da utilização do ambiente. O método utilizado pelo processo de seleção determina como as requisições serão distribuídas no ambiente, afetando diretamente o tempo para o atendimento e para a execução destas. Este processo se torna ainda mais complexo a medida que aumentam o número de requisições. Nestas situações, é comum observar requisições aguardando em filas de espera pela liberação de recursos.

Um dos métodos mais utilizados em processos de seleção de recursos em ambientes distribuídos é a comparação exata de valores baseada em palavras-chave. Este método é encontrado em grande parte dos sistemas gerenciadores de recursos e *middlewares* de computação distribuída e consiste da verificação de equivalência entre os valores estipulados para atributos pré-definidos (palavras-chave) da requisição e dos recursos do ambiente computacional distribuído.

Este mecanismo de busca não é naturalmente capaz de identificar o significado das informações manipuladas no processo. Isso implica que nestes processos de seleção, a capacidade de atendimento de requisições é prejudicada, por exemplo, pela impossibilidade de identificação de recursos com atributos e valores diferentes do especificado na requisição, mas ainda assim, equivalentes semanticamente.

Observando a importância do processo de seleção de recursos em ambientes distribuídos e as oportunidades de melhoria provenientes da aplicação de ontologia e de orientação ao contexto neste processo, decidiu-se pela realização deste trabalho.

1.1 Objetivo Geral

Propor um método para otimizar a seleção de recursos em ambientes distribuídos quanto ao tempo de espera pelo atendimento de requisições e ao tempo total percebido pelos usuários para a execução de um conjunto de requisições, também conhecido como *wall-clock time*, por meio da aplicação de uma ontologia de domínio e de orientação ao contexto.

1.2 Objetivos Específicos

O atendimento do objetivo geral deste trabalho engloba o atendimento dos seguintes objetivos específicos:

a. Realizar um estudo de ontologias e informações de contexto relacionados a ambientes distribuídos;
b. Construir ou adequar uma ontologia para a representação do conhecimento do domínio de ambientes distribuídos suficiente para a seleção de recursos segundo a proposta deste trabalho;
c. Propor um conjunto de informações de contexto adequados ao processo de seleção de recursos em ambientes distribuídos;
d. Propor um método de seleção de recursos em ambientes distribuídos baseado em ontologia e orientado ao contexto;
e. Desenvolver um protótipo para seleção de recursos em ambientes distribuídos utilizando o método proposto;
f. Avaliar o desempenho do método de seleção de recursos proposto, considerando o tempo de espera pelo atendimento de requisições e o tempo total percebido pelos usuários para a execução de um conjunto de requisições, comparado com um processo de seleção que utilize o método de comparação exata de valores, baseada em palavras-chave.

1.3 Escopo

Este trabalho apresentará um método e os resultados da aplicação deste método na seleção de recursos em ambientes distribuídos utilizando ontologia e orientação ao contexto. O protótipo que será desenvolvido para demonstrar a aplicação do método conterá apenas a implementação necessária para o processo de seleção. Isto implica que a administração do ambiente, incluindo desde a coleta das informações sobre os recursos até a alocação de um recurso para a execução de uma tarefa, não fará parte da implementação do protótipo. Estas funções serão realizadas por um *middleware* de computação distribuída e um sistema gerenciador de recursos.

Aspectos mais especificamente relacionados ao escalonamento das requisições, como comportamentos de filas, consideração de prioridades, migração de processos, entre outros, não foram consideradas como parte do método de seleção proposto. Considerou-se aqui que estas funções estão implementadas em outras camadas do sistemas distribuído, como no *middleware* ou no sistema gerenciador de recursos.

As requisições que serão enviadas para o ambiente, com o objetivo de validar a proposta deste trabalho, serão geradas automaticamente por uma aplicação a ser implementada. O protótipo não contará, portanto, com uma interface gráfica de usuário para a submissão ou monitoramento das tarefas. As métricas para avaliação do processo de seleção, serão externalizadas pela aplicação geradora de requisições e pelo próprio protótipo do processo de seleção.

Por fim, não estão sendo considerados neste trabalho aspectos de autenticação e autorização de acesso aos recursos, assim como questões relacionadas a segurança das comunicações.

1.4 Organização do Trabalho

Este trabalho foi assim organizado. Os capítulos 2 e 3 apresentam uma revisão da literatura sobre, respectivamente, computação distribuída e ontologia e orientação ao contexto. O detalhamento da proposta deste trabalho é realizado no capítulo 4. No capítulo 5 é apresentado o ambiente e os resultados dos experimentos de validação da proposta. Por fim, no capítulo 6 é apresentada a conclusão e sugestões de trabalhos futuros.

2 COMPUTAÇÃO DISTRIBUÍDA

Este capítulo abordará a computação distribuída como uma alternativa para incrementar o desempenho de aplicações. Serão apresentadas as características e os principais componentes presentes nestes sistemas. Os sistemas distribuídos serão analisados sob o aspecto dos elementos de hardware, através de um estudo das arquiteturas computacionais e das configurações de sistemas distribuídos, e sob o aspecto dos elementos de software, através da análise dos pacotes e sistemas disponíveis para a construção e utilização dos ambientes.

2.1 Visão Geral

A cada dia, novas aplicações e serviços digitais são disponibilizados. Para a sociedade moderna, apenas o correto funcionamento destes, muitas vezes, não é mais suficiente. Os sistemas computacionais precisam executar com desempenho cada vez mais elevado. Para aumentar o desempenho de um sistema pode-se atuar em nível de hardware, utilizando processadores ou dispositivos de armazenamento, temporários ou permanentes, mais rápidos; em nível de software, utilizando melhores algoritmos; ou empregando o paradigma de computação distribuída ou paralela (DANTAS, 2005).

As duas primeiras alternativas mencionadas têm esbarrado em limitações físicas e de custo. A tecnologia de fabricação de processadores, por exemplo, atingiu limites físicos de velocidade e miniaturização de componentes eletrônicos. Com relação a utilização de algoritmos, muitas vezes o custo para a atualização de um sistema computacional torna sua realização inviável financeiramente. Eis então que a computação distribuída surge como uma alternativa mais razoável para melhorar o desempenho de alguns sistemas computacionais.

Os sistemas distribuídos, são definidos por Tanenbaum e Steen (2006) como “uma coleção de computadores independentes que se apresenta aos seus usuários como um sistema único e coerente”. Essa definição aborda o aspecto da agregação de recursos computacionais para a execução de sistemas.

Para Coulouris *et al.* (2012), “um sistema distribuído é aquele no qual os componentes de hardware e software localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens”. A definição de Coulouris *et al.* enfatiza a importância da comunicação entre computadores e processos em um sistema

distribuído. Ainda segundo Coulouris *et al.* (2012), o compartilhamento de recursos é o principal motivo para a construção e utilização de sistemas distribuídos. Como recursos, podem ser compreendidos tanto dispositivos físicos, como discos e impressoras, quanto entidades de software, como arquivos e bancos de dados.

A partir das definições apresentadas aqui compreende-se que um sistema distribuído é composto não apenas por componentes de hardware, mas também por componentes de software. As características destes dois tipos de componentes, no que diz respeito à sistemas distribuídos, serão apresentadas nas seções a seguir.

2.2 Arquiteturas Computacionais

Um dos desafios inerentes a construção de sistemas distribuídos é a heterogeneidade. É uma necessidade comum a grande parte dos sistemas distribuídos executar sobre um ambiente de hardware com componentes de configurações diferenciadas. Cabe então ao desenvolvedor do sistema conhecer as configurações disponíveis de forma a tirar o melhor proveito das capacidades do ambiente.

Em virtude da diversidade de arquiteturas de computadores existentes, diversas propostas de taxonomias foram realizadas ao longo dos anos com o objetivo facilitar seu estudo e projeto. A classificação mais aceita atualmente é conhecida como taxonomia de Flynn (FLYNN, 1972). Essa taxonomia considera dois parâmetros para determinar a classificação de uma arquitetura computacional: o número de instruções executadas em paralelo e o conjunto de dados para os quais as instruções são submetidas (DANTAS, 2005). Como resultado, tem-se quatro classes de arquiteturas: *Single Instruction Single Data* (SISD), *Single Instruction Multiple Data* (SIMD), *Multiple Instruction Single Data* (MISD) e *Multiple Instruction Multple Data* (MIMD).

A classe SISD corresponde aos computadores pessoais, com processador único. Como processador único, entenda-se aqui chips com um único núcleo (*single core*). Estes computadores são capazes de executar apenas uma instrução de um programa a cada ciclo do processador. Os computadores com arquiteturas do tipo SIMD também são capazes de executar apenas uma instrução por ciclo de processador. Contudo, possuem facilidades de hardware que permitem a execução da mesma instrução sobre diferentes conjuntos de dados. A arquitetura MISD corresponde a computadores com capacidade de executar múltiplas

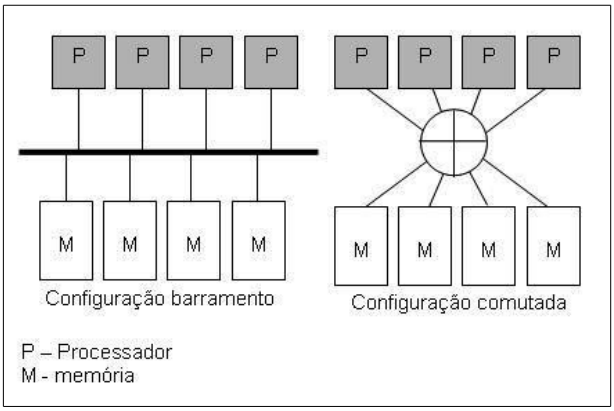
instruções por ciclo de processador sobre um único conjunto de dados. Esta configuração não é normalmente utilizada na construção de computadores, por não oferecer um grande ganho em termos de otimização de recursos. Por fim, a classe MIMD representa os computadores capazes de executar múltiplas instruções por ciclo de processador sobre múltiplos conjuntos de dados. Este tipo de arquitetura compreende um grande número de sistemas computacionais paralelos, incluindo desde os computadores pessoais com processadores de múltiplos núcleos (*multi-core*), até mesmo configurações de *clusters* e *grids* computacionais.

As seções a seguir apresentarão uma subdivisão das arquiteturas MIMD, segundo a interconexão entre processadores e memórias: multiprocessadores e multicomputadores.

2.2.1 Multiprocessadores

Multiprocessadores são arquiteturas caracterizadas pelo compartilhamento de uma única memória entre vários processadores (DANTAS, 2005). Esta memória, fisicamente, pode ser uma placa ou uma composição de várias placas, interconectadas de forma a dar a percepção de unicidade perante o sistema.

Figura 1 - Exemplos de configurações genéricas de multiprocessadores



Fonte: (DANTAS, 2005)

A interconexão entre os processadores e a memória através do sistema local faz com que esta arquitetura seja conhecida como fortemente acoplada. Esta interconexão pode ser na forma de um barramento, onde, geralmente, os processadores estão em uma mesma placa mãe, ou realizada através de um elemento comutador, onde utiliza-se de um equipamento de interconexão especial para interligar computadores individuais de forma a

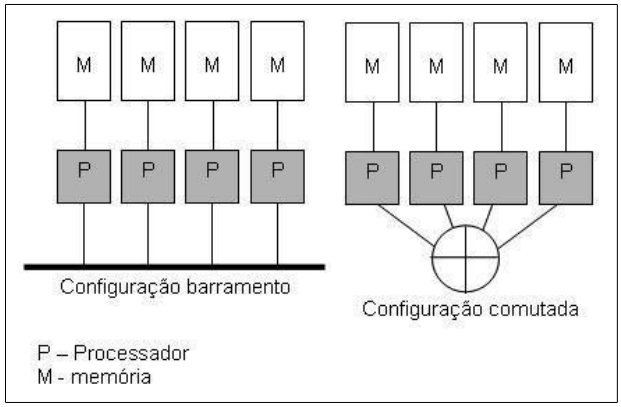
compartilhar a memória, conforme mostra a Figura 1. Em ambos os casos, observa-se um compartilhamento global da memória.

O compartilhamento global da memória entre os processadores traz consigo a possibilidade de utilização de instruções comuns de acesso a memória. Por outro lado, esta mesma condição reflete numa restrição da escalabilidade do sistema, onde existe um número determinado de processadores que podem ser interconectados. Exemplos de configurações segundo este tipo de arquitetura são ambientes de multiprocessadores simétricos (*Symmetric Multiprocessors – SMP*) e ambientes de acesso não-uniforme à memória (*Non-Uniform Memory Access - NUMA*).

2.2.2 Multicomputadores

Arquiteturas de multicomputadores se diferenciam das arquiteturas de multiprocessadores principalmente pelo fraco acoplamento entre os processadores e a memória. Nesta arquitetura, não existe o compartilhamento global da memória, como nos ambientes multiprocessados. A comunicação entre os processos é realizada através da troca de mensagens.

Figura 2 - Exemplos de configurações genéricas de multicomputadores.



Fonte: (DANTAS, 2005)

A interconexão entre os processadores nas arquiteturas de multicomputadores também pode ser realizada através de barramentos ou elementos comutadores. A Figura 2 apresenta estes dois tipos de configuração.

Este tipo de arquitetura é mais simples de ser implementado, visto que pode ser

realizado através da simples interconexão de computadores pessoais através de uma rede de computadores doméstica, por exemplo. Contudo, a decisão pelo meio de interconexão entre os processadores tem peso maior sobre o desempenho do sistema, pois é através deste meio que serão trafegadas todas as mensagens entre os processos executando nos diversos processadores. Ambientes de processadores massivamente paralelos (*Massively Parallel Processors – MPP*), *clusters* e *grids* são exemplos de configurações de multicomputadores. As próximas seções trataram com mais detalhes estas duas últimas configurações.

2.2.2.1 Clusters

Segundo Dantas (2005), as configurações de *clusters* são agrupamentos de inúmeros computadores, de forma dedicada ou não, para a execução de aplicações específicas de uma organização. Seu principal objetivo é disponibilizar os recursos não utilizados nestes computadores para melhorar o desempenho das aplicações.

Entre os aspectos que caracterizam os *clusters* computacionais, talvez o mais evidente seja o limite geográfico. Este tipo de configuração é geralmente realizado no domínio de um laboratório, um departamento ou uma organização.

Outro aspecto importante é quanto a disponibilidade dos recursos. Recursos não-dedicados, ou seja, que executam tanto tarefas submetidas à configuração quanto tarefas locais, mono processadas, apresentam uma boa relação custo-benefício, por otimizar a utilização destes. Por sua vez, os recursos dedicados conferem mais disponibilidade e são preferenciais quando da necessidade de execução de aplicações vitais de uma organização.

A escalabilidade é um ponto forte das configurações de *clusters* computacionais. Novos recursos podem ser interligados ao ambiente à medida que aumenta a demanda por desempenho ou disponibilidade computacional. Contudo, o meio de interconexão pode se tornar um limitador na configuração. Portanto, este deve ser selecionado de acordo com as necessidades das aplicações que serão executadas no ambiente.

2.2.2.2 Grids

Uma *grid* é “um ambiente computacional distribuído paralelo que permite o compartilhamento, a seleção, a agregação de recursos autônomos e geograficamente

distribuídos" (DANTAS, 2005). Trata-se de uma configuração de alto desempenho, onde recursos são disponibilizados em larga escala, extrapolando os limites de uma organização.

As configurações de *grids* computacionais evoluíram e ganharam notoriedade sob a influência da Internet, a rede mundial de computadores. Originalmente utilizadas apenas para aplicações científicas, atualmente estes ambientes têm sido explorados também comercialmente. A capilaridade da Internet permite que qualquer computador pessoal do mundo, interconectado a rede, compartilhe e utilize recursos e serviços.

Uma característica marcante deste tipo de configuração é a heterogeneidade, encontrada em diversos níveis, como hardware, sistema operacional, linguagens de programação e aplicações (COULOURIS et al., 2012). Outro aspecto importante está relacionado a segurança e ao controle dos recursos disponibilizados no ambiente. Estas e outras questões são tratadas através do que se chama de *middleware*, que é uma camada de software localizada entre as aplicações de usuário e o sistema operacional. Mais sobre *middleware* e outros aspectos dos ambientes de software que suportam estas configurações serão tratados nas seções a seguir.

2.3 Ambientes de Software

A utilização adequada de um ambiente computacional distribuído por parte de uma aplicação requer o suporte de ambientes de software especializados. Entre estes encontram-se bibliotecas para utilização na programação das aplicações, protocolos de comunicação padronizados e sistemas mais complexos, como os sistemas gerenciadores de recursos e ambientes de *middleware*.

As bibliotecas para programação paralela e distribuída são pacotes de software, escritos em diversas linguagens de programação, cuja função é auxiliar no desenvolvimento de aplicações paralelas e distribuídas. Normalmente, sua utilização envolve a inclusão dos pacotes na aplicação, possibilitando ao desenvolvedor utilizar funções e procedimentos implementados nestes. Desta forma, o desenvolvedor não precisa criar em todo o projeto os mecanismos necessários para executar sua aplicação de forma paralela e distribuída.

Existem várias bibliotecas para programação de aplicações paralelas e distribuídas. Alguns exemplos são o PVM (PVM, 2013) e MPI (MPIFORUM, 2013). O *Parallel Virtual Machine* (PVM) é um pacote de software cujo propósito é permitir que uma

coleção de computadores interligados por uma rede seja utilizada como um grande e único computador paralelo. O *Message Passing Interface* (MPI) é uma especificação de uma interface para passagem de mensagens, cujo objetivo é estabelecer um padrão para a troca de mensagens nos sistemas de execução paralela.

O uso de bibliotecas auxilia o desenvolvedor de aplicações paralelas e distribuídas, contudo, muitas vezes o deixa restrito a comunicação entre aplicações desenvolvidas com uma mesma linguagem de programação. Visando a possibilidade de realizar comunicação entre aplicações independente de linguagem ou paradigma de programação, protocolos de comunicação foram propostos. Entre eles, talvez, o mais popular na Internet atualmente seja o SOAP (W3C, 2007) na implementação de *Web Services*.

Os *Web Services* são sistemas que permitem que serviços sejam oferecidos na *Web* e consumidos por outras aplicações independente de diferenças de configurações de hardware, de sistemas operacionais ou de linguagens de programação, através da troca de mensagens XML (*eXtensible Markup Language*) (BRAY et al., 2008) segundo o protocolo SOAP (*Simple Object Access Protocol*) (W3C, 2007).

A construção de ambientes distribuídos, como *clusters* e *grids*, geralmente faz uso de ambientes de software mais complexos, como os sistemas gerenciadores de recursos e os ambientes de *middleware*. As próximas seções abordarão estes ambientes com mais detalhes.

2.3.1 Sistemas Gerenciadores de Recursos

As principais motivações para o uso de sistemas gerenciadores de recursos (*Resource Management Systems – RMS*) são a necessidade de um escalonamento global em um ambiente multicomputado, como em uma configuração de *grid* ou *cluster*, e o provimento de transparência e facilidades na utilização dos ambientes (DANTAS, 2005).

Nestes pacotes de software é comum encontrar recursos como balanceamento de carga, execução paralela, migração de processos, suporte para sistemas operacionais e arquiteturas de computadores heterogêneas, facilidades de submissão e controle de tarefas, administração de políticas de uso, entre outros.

É possível encontrar pacotes RMS tanto comerciais quanto gratuitos. Entre estes últimos, alguns disponibilizam inclusive seu código-fonte. Alguns exemplos de RMS comerciais são Oracle Grid Engine (ORACLE, 2013a), IBM Platform LSF (IBM, 2013) e

PBS Professional (PBSWORKS, 2013). Como exemplos de RMS gratuitos temos o TORQUE (ADAPTIVECOMPUTING, 2013) e o HTCondor (UW-MADISON, 2013). A seção a seguir trará mais detalhes sobre o HTCondor.

2.3.1.1 HTCondor

O HTCondor é um sistema de gerenciamento de recursos especializado para aplicações de computação intensiva (UW-MADISON, 2013). Através dele, é possível construir ambientes de *cluster* e de *grid,* utilizando computadores com sistemas operacionais e arquiteturas heterogêneas, dedicados ou não, para a execução de trabalhos (*jobs*) de forma paralela e distribuída.

Entre as facilidades providas pela ferramenta estão um mecanismo de enfileiramento de *jobs*, políticas de escalonamento, esquemas de priorização de *jobs*, monitoramento e gestão de recursos.

A arquitetura básica dos ambientes criados utilizando o HTCondor consiste de um nó fazendo o papel de gestor central, recebendo, escalonando e retornando os resultados dos *jobs* submetidos; e um conjunto de nós fazendo os papéis de submissores, executores ou ambos, compondo o que é denominado de *pool* de recursos.

As trocas de informações entre os nós acontecem através do uso de um mecanismo chamado *ClassAd*. Este mecanismo consiste de uma espécie de anúncio, onde diversas informações são passadas para descrever recursos disponíveis e necessários.

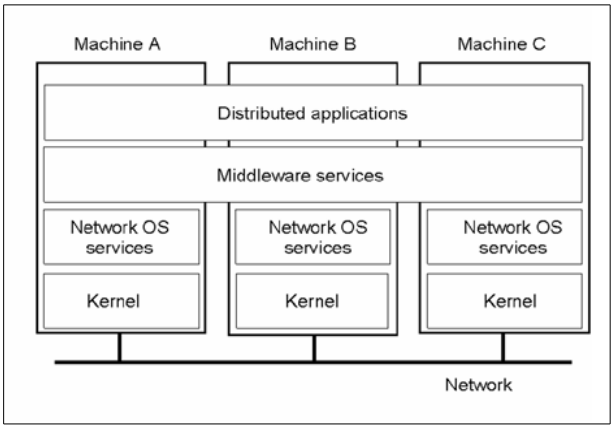
Na versão 7.8, o HTCondor ainda não oferece uma interface gráfica amigável para a interação com a ferramenta. Seu principal meio de utilização é através de um terminal de linha de comando, encontrado nos principais sistemas operacionais. Um subconjunto das funcionalidades está disponível também através de *Web Services*, que podem ser ativados na instalação.

2.3.2 Ambientes de Middleware

O *middleware* pode ser entendido como uma camada de software, localizada entre as aplicações do usuário e o sistema operacional, cuja função, segundo Dantas (2005), é prover para os usuários dos *clusters* e *grids* computacionais transparência segura de acesso,

facilidades de submissão das aplicações e gerenciamento de recursos e serviços. Outra função importante dos ambientes de *middleware*, conforme Coulouris *et al.* (2012) é mascarar a heterogeneidade das tecnologias de rede, elementos de hardware, sistemas operacionais e linguagens de programação. A Figura 3 apresenta a localização do *middleware* em uma configuração distribuída.

Figura 3 - Localização do middleware em uma configuração distribuída.



Fonte: (KOOLKAMPUS, 2013)

Um tipo de *middleware* bastante utilizado, principalmente em configurações de *clusters*, é o Sistema de Imagem Única (*Single System Image – SSI*). Este tipo de sistema se caracteriza pela oferta de uma visão unificada dos recursos e serviços disponibilizados no ambiente distribuído. Alguns exemplos são o OSCAR (OPENCLUSTERGROUP, 2013) e o Sprite (BERKELEY, 2013).

Em configurações de *grid*, os ambientes de *middleware* mais utilizados são o Globus (GLOBUS, 2013) e o Legion (VIRGINIA, 2013). O primeiro, se caracteriza pela quantidade de recursos e serviços disponibilizados para auxiliar na construção e utilização dos ambientes. Enquanto o segundo se diferencia dos demais por oferecer um ambiente de software orientado a objeto, com a disponibilização de uma máquina virtual representando a configuração do *grid* para a execução das aplicações do usuário.

2.4 Considerações

Neste capítulo, a computação distribuída for apresentada como uma alternativa para a execução de aplicações com necessidades de alto desempenho e alta disponibilidade.

Conforme os conceitos apresentados, estes ambientes são caracterizados por computadores interconectados, trocando mensagens, orientados a execução de uma tarefa. Foi observado também que estes ambientes são compostos por elementos de hardware e de software

Sobre os elementos de hardware, foram apresentadas as arquiteturas computacionais, segundo a taxonomia de Flynn (FLYNN, 1972): SISD, SIMD, MISD e MIMD. Em função da possibilidade de prover configurações de alto desempenho e alta disponibilidade, a arquitetura MIMD foi mais detalhada em termos de multiprocessadores e multicomputadores. O primeiro foi apresentado como um conjunto de processadores compartilhando um memória, enquanto que no segundo caso, cada processador possui sua própria memória.

As configurações de *clusters* e *grids*, formas de implementação de arquiteturas de multicomputadores, foram apresentadas em seguida. Os *clusters* foram apresentados como um agregado de recursos para a execução de aplicações de uma organização. Por sua vez, os *grids* foram vistos como a agregação e compartilhamento de recursos e serviços geograficamente distribuídos.

Quanto aos elementos de software que compõe um sistema distribuído, foram vistos que estes podem se apresentar como bibliotecas para auxílio na programação de aplicações, protocolos padronizados ou ambientes mais complexos, como sistemas gerenciadores de recursos e ambientes de middleware.

3 ONTOLOGIA E SENSIBILIDADE AO CONTEXTO

Este capítulo inicia com a apresentação de ontologia, como forma de representação de conhecimento. Serão observadas definições e os principais componentes e linguagens de representação. Em seguida, se apresenta a computação sensível ao contexto, referindo-se aos sistemas que utilizam de informações de contexto para oferecer funcionalidades aos usuários. Os conceitos envolvidos neste assunto, assim como algumas classificações serão apresentadas.

3.1 Ontologia

A ontologia é uma área de estudo da filosofia, onde o foco é a natureza e a estrutura da "realidade" (GUARINO; OBERLE; STAAB, 2009). Na ciência da computação, a ontologia é utilizada fundamentalmente para a representação de conhecimento. Esta compreensão pode ser obtida a partir das definições de alguns autores.

Uma das definições mais aceitas foi proposta em 1993, por Gruber: "uma ontologia é uma especificação explícita de uma conceitualização" (GRUBER, 1993). Uma conceitualização, segundo Genesereth e Nilsson (1987), corresponde aos "objetos, conceitos e outras entidades, que assume-se existir em alguma área de interesse, e os relacionamentos entre eles". Observa-se aqui um aspecto importante da ontologia aplicada na ciência da computação: o foco na representação do conhecimento de um domínio específico e não de todo o universo possível.

Outra definição, criada por Borst, traz a ontologia como uma "especificação formal de uma conceitualização compartilhada" (BORST, 1997). Esta definição afirma o caráter formal da ontologia no âmbito computacional e a consequente capacidade de compartilhamento.

Na prática, uma ontologia é um artefato de engenharia, composto por um vocabulário específico, utilizado para descrever um certo domínio, mais um conjunto de suposições explícitas relacionadas ao significado pretendido para as palavras do vocabulário (GUARINO, 1995). É uma estrutura formal, criada com o intuito de representar conhecimento e permitir a comunicação deste entre pessoas ou sistemas computacionais. Contudo, é importante observar que, para haver a interoperabilidade entre as partes, é

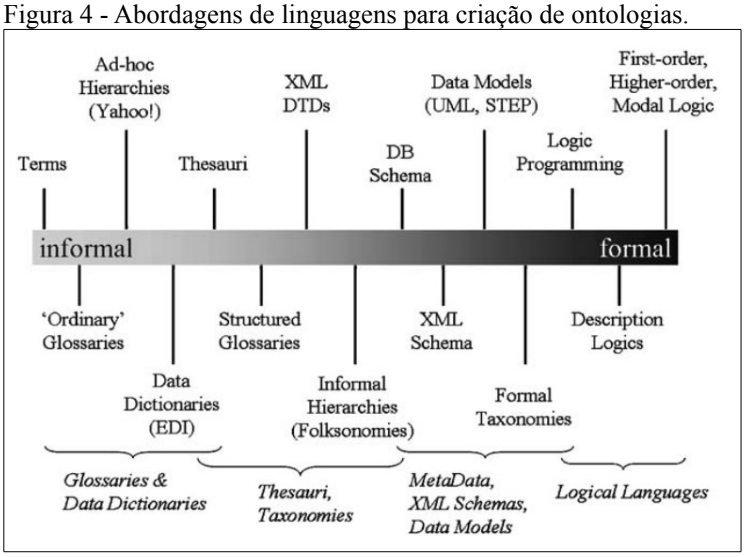
necessário que estas estejam de acordo com os termos e axiomas representados pela ontologia.

A área de pesquisa em aplicação de ontologias nos sistemas computacionais tem sido impulsionada pelo que é chamado de Web Semântica. Proposta pelo criador da Web, Tim Berners-Lee (BERNERS-LEE, 2000), a Web Semântica seria uma Web processável não apenas por pessoas, como é na sua maior parte, até o momento, mas também por agentes computacionais. Uma das premissas para a realização desta e de outras aplicações de ontologia em sistemas computacionais é o formato de representação.

3.1.1 Linguagens

Ontologias são representadas com o uso de linguagens criadas especificamente para este fim. Existe uma grande variedade de linguagens para a criação de ontologias (EUZENAT; SCHVAIKO, 2007). O principal aspecto que influencia na sua escolha é sua capacidade de expressão e de formalização. A Figura 4 apresenta diferentes abordagens para linguagens relacionando-as com sua característica formal.

Figura 4 - Abordagens de linguagens para criação de ontologias.



Fonte: (GUARINO; OBERLE; STAAB, 2009)

A decisão por abordagens mais formais, segundo Guarino *et al.* (2009), envolve um compromisso entre expressividade e eficiência. Linguagens que seguem abordagens de lógicas de primeira ordem, de alta ordem e modal são muito expressivas. Porém, o processo

de raciocínio (*reasoning*) sobre ontologias criadas com estes tipos de linguagens, quando possível, pode ser intratável. Como alternativa, existem linguagens que seguem abordagens que são subconjuntos menos rigorosos de lógicas de primeira ordem, como lógicas de descrição e lógicas de programação. Estas linguagens costumam apresentar raciocinadores semânticos (*reasoners*) decidíveis e mais eficientes. Por esta razão, costumam ser preferidas para a construção de ontologias.

3.1.1.1 Componentes

Apesar de apresentarem particularidades na forma de representação de ontologias, em grande parte das linguagens é possível observar componentes comuns. Entre os principais componentes encontrados podemos citar as classes, atributos, relações e indivíduos.

Classes, ou conceitos, são os principais componentes de uma ontologia (EUZENAT; SCHVAIKO, 2007). Os blocos fundamentais das principais linguagens existentes. Podem ser utilizados tanto para classificar indivíduos ou objetos como também outras classes do domínio. São os elementos que permitem representar os principais termos do vocabulário.

Atributos, ou propriedades, correspondem aos elementos que definem características, aspectos ou partes de um indivíduo ou objeto. Seus valores podem ser tanto dados simples como nomes, números, etc., quanto dados complexos, como outros indivíduos, por exemplo.

Relações são elementos que descrevem como os objetos de uma ontologia se relacionam. São os principais responsáveis pela semântica de uma ontologia. Os tipos mais comuns de relacionamentos entre objetos são o *IS-A* (é um) e *IS-PART-OF* (é parte de). O primeiro define uma hierarquia, na qual uma classe é um subtipo da classe superior. Enquanto que o segundo tipo, define uma relação de composição, na qual um objeto é formado através da agregação de outros.

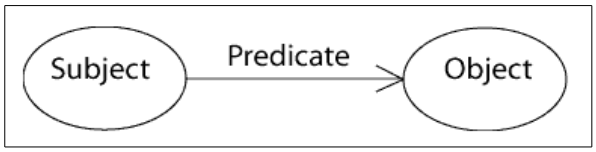
Indivíduos, ou objetos, são elementos que representam indivíduos particulares ou concretos de um domínio. São instâncias das classes definidas na ontologia. Para alguns pesquisadores, a presença de indivíduos nas ontologias não é obrigatória. Outros consideram que a existência de indivíduos transforma a ontologia em uma base de conhecimento.

3.1.1.2 RDF (Resource Description Framework)

O *Resource Description Framework* (RDF) é uma linguagem para a representação de informação sobre recursos na *World Wide Web* (WWW) (MANOLA; MILLER, 2004). Sua estrutura a torna especialmente adequada para a representação de dados e afirmações simples.

A estrutura básica do RDF é uma tripla, composta por um sujeito, um predicado e um objeto. O sujeito se refere ao recurso sendo descrito. O predicado corresponde a propriedade alvo da declaração. O objeto denota o valor da propriedade sendo declarada. Uma tripla RDF, ou um conjunto delas, pode ser representado através de um grafo direcionado, onde os nós de origem correspondem aos sujeitos, os nós de destino correspondem aos objetos e os arcos correspondem aos predicados. A Figura 5 apresenta um grafo direcionado de uma declaração RDF.

Figura 5 - Representação de um tripla RDF.



Fonte: (KLYNE; CARROLL, 2004)

Cada elemento da tripla RDF é considerado um recurso, e estes são identificados utilizando *Uniform Resource Identifiers* (URIs). O uso do URI implica que cada recurso seja identificado unicamente dentro da representação, sem ambiguidades. Apesar de usualmente o URI apresentar valores que sugerem uma localização na Web, esta não é obrigatória.

A representação gráfica é mais adequada para um pequeno conjunto de declarações e para ser analisado por pessoas. Para uso computacional, a escrita normativa do RDF utiliza uma sintaxe estendida da *eXtensible Markup Language* (XML) (BRAY et al., 2008), chamada RDF/XML.

O RDF oferece os mecanismos básicos para a construção de vocabulários de um domínio, através das declarações de recursos, propriedades e valores. Contudo, algumas aplicações possuem necessidades específicas em termos de declarações e restrições, não ofertadas pelo RDF básico. Um extensão do RDF, chamada *RDF Schema* (RDFS) (BRICKLEY; GUHA, 2004), vem para suprir parte dessa necessidade.

O RDFS é uma linguagem para descrição de vocabulários. Basicamente, o RDFS

é ele próprio um vocabulário, cujos termos conferem a semântica de determinar, por exemplo, uma hierarquia de conceitos dentro de um domínio. O RDFS não contém termos do vocabulário de um domínio específico. Ele simplesmente provê facilidades para a construção de outros vocabulários.

A simplicidade das declarações do RDF faz dele um mecanismo bastante adequado e prático para a representação de dados. Contudo, mesmo utilizando-se das extensões oferecidas pelo RDFS, existem algumas capacidades ausentes. Não é possível, por exemplo, descrever restrições de cardinalidade, especificar a transitividade de uma propriedade, denotar a equivalência entre classes, entre outras capacidades. Para cobrir estas lacunas, outras linguagens foram propostas.

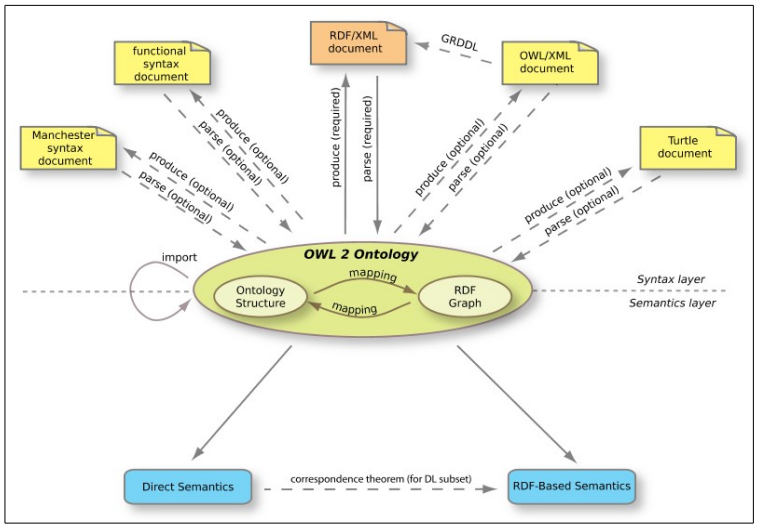
3.1.1.3 OWL (Web Ontology Language)

A *Web Ontology Language* (OWL) (W3C OWL WORKING GROUP, 2009) é uma linguagem para descrição de ontologias desenvolvida pelo *World Wide Web Consortium* (W3C) (W3C, 2013) com o objetivo de facilitar a representação e o compartilhamento de informações pela Web e tornar estas informações mais acessíveis para sistemas (HITZLER et al., 2009). Seu desenvolvimento faz parte do esforço de criação da Web Semântica.

Esta linguagem teve influências de outras linguagens propostas anteriormente, como a XML, RDF e DAML+OIL. Diversos conceitos, recursos e sintaxes destas linguagens antecessoras foram mantidos na OWL por uma questão de compatibilidade, visto que estas já estavam sendo utilizadas para construção de ontologias compartilhadas na Web (HORROCKS; PATEL-SCHNEIDER; VAN HARMELEN, 2003).

A Figura 6 apresenta a estrutura da linguagem OWL na sua segunda versão. Na figura observa-se a ontologia, representada pela elipse central; na parte superior encontram-se as sintaxes disponíveis na especificação da OWL2 para a expressão de ontologias; e na parte inferior, as especificações semânticas, que agregam significado às ontologias (W3C OWL WORKING GROUP, 2009).

Figura 6 - Estrutura da OWL2.



Fonte: (W3C OWL WORKING GROUP, 2009)

Além de oferecer diferentes sintaxes e especificações semânticas para a construção de ontologias, a OWL2 permite ainda a utilização de sub-linguagens ou subconjuntos sintáticos da especificação, chamados *profiles* (W3C OWL WORKING GROUP, 2009). O objetivo deste recurso é permitir o aproveitamento da expressividade e das restrições de cada *profile* nos diversos cenários de uso de ontologias.

3.2 Computação Sensível ao Contexto

A computação sensível ao contexto refere-se aos sistemas que fazem uso de informações de contexto para oferecer informações ou serviços relevantes ao usuário (DEY; ABOWD, 1999). Este capítulo inicia definindo contexto, no âmbito computacional. São apresentadas as principais propostas de categorias e abordagens de modelagem de informações de contexto. Em seguida são apresentadas as principais definições para sistemas sensíveis ao contexto, finalizando com as propostas de classificação para estes sistemas.

3.2.1 Contexto

As primeiras definições de contexto consistiam basicamente de enumerações de exemplos e de sinônimos para o termo (DEY; ABOWD, 1999). Um exemplo é o trabalho de Schilit e Theimer (SCHILIT; THEIMER, 1994), considerado o primeiro a utilizar o termo

"computação sensível ao contexto" (*context-aware computing*), onde contexto é definido como localização, identidades de pessoas e objetos próximos e as alterações nestes objetos. A determinação de que um tipo de informação é ou não uma informação de contexto, a partir destas definições, baseadas em exemplos e sinônimos, pode ser difícil em muitos casos. Além disso, enumerar todos os possíveis tipos de informações de contexto seria inviável, na prática.

Uma definição com escopo mais genérico seria uma solução para este problema. Uma das mais aceitas, é a de Dey e Abowd (1999) em que estes afirmam que contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade, sendo a entidade uma pessoa, lugar ou objeto, considerada relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação.

Sendo assim, pode-se observar que o contexto é um conjunto de informações obtidas do ambiente em que a aplicação está inserida e não explicitamente fornecidas pelo usuário. E que o propósito da coleta desta informações é tornar a interação do usuário com o sistema mais eficiente. Como meio de melhor estudar e utilizar essas informações, algumas propostas de classificação de informações de contexto foram feitas. As principais serão apresentadas nas seções a seguir.

3.2.1.1 Categorias de Informações de Contexto

As informações de contexto, segundo Schilit *et al.* (1994), podem ser dividas em contexto computacional, contexto de usuário e contexto físico. Uma característica desta classificação é que ela se baseia nas entidades referidas pelas informações e não exatamente no tipo de informação de contexto. Esta classificação foi estendida no trabalho de Chen e Kotz (2000), com a inclusão do contexto temporal.

Uma outra proposta de classificação, que considera o tipo da informação como critério para determinar a categoria a qual esta pertence, foi feita por Dey *et al.* (2001). Segundo os autores, as informações de contexto podem ser classificadas em quatro categorias essenciais: identidade, localização, estado (ou atividade) e tempo. A categoria de identidade refere-se a informações de contexto que permitem identificar unicamente uma entidade. A categoria de localização corresponde a informações de posicionamento, elevação, orientação e relações espaciais entre entidades. A categoria de estado condiz com as informações de contexto que identificam características intrínsecas de uma entidade. E a categoria tempo

aplica-se as informações de contexto que auxiliam na caracterização temporal de uma situação.

A classificação das informações de contexto permite ao desenvolvedor identificar que tipo de informação é mais relevante para o sistema sendo desenvolvido. A partir desta identificação, o desenvolvimento do sistema pode ser otimizado destacando determinado aspecto. Outro fator que influencia o desenvolvimento de sistemas sensíveis ao contexto é a forma como as informações são modeladas. As seções a seguir apresentam algumas abordagens.

3.2.1.2 Modelagem de Informações de Contexto

A complexidade inerente de sistemas sensíveis ao contexto levou ao estudo de técnicas de modelagem e raciocínio sobre informações de contexto. Com estes estudos foram identificados diversos requisitos para a modelagem das informações de contexto. Segundo Bettini *et al.* (2010), os principais requisitos são heterogeneidade e mobilidade, relacionamentos e dependências, histórico, imperfeição, raciocínio, usabilidade de formalismos de modelagem e provisionamento eficiente do contexto.

O requisito de heterogeneidade está relacionado a capacidade de suportar diferentes fontes de informações de contexto. A mobilidade refere-se a capacidade de adaptação à ambientes dinâmicos, comuns em aplicações sensíveis ao contexto executadas em dispositivos móveis. A capacidade de capturar as relações existentes entre informações de contexto corresponde aos requisitos de relacionamento e dependência. A dependência pode ser considerada, de fato, como um tipo de relacionamento. O requisito de histórico representa a necessidade de acesso a registros temporais das informações de contexto. O tratamento das diferenças de qualidade e exatidão das informações de contexto, relacionadas a natureza dinâmica e heterogênea das mesmas, é definido pelo requisito de imperfeição. O raciocínio é o requisito relacionado a capacidade de verificação de consistência de um modelo contextual e de raciocínio, ou derivação de novos fatos, sobre o contexto. O requisito de usabilidade de formalismos de modelagem corresponde a facilidade de construção e manipulação de modelos contextuais tanto por pessoas quanto por sistemas. Por fim, o requisito de provisionamento eficiente do contexto refere-se a eficiência de acesso as informações de contexto.

As principais técnicas de modelagem de informações de contexto, conforme

descrito por Bettini *et al.* (2010), não atendem, individualmente, a todos os requisitos mencionados. Cada técnica contempla um subconjunto destes requisitos, o que a torna mais ou menos eficiente para determinados tipos de sistemas. Nas seções a seguir, serão apresentadas algumas características destas técnicas.

3.2.1.3 Modelagem Baseados em Fatos

As abordagens de modelagem baseada em fatos surgiram a partir de tentativas de criação de modelos formais que suportassem processamento de consultas e raciocínio sobre informações de contexto. Esta abordagem oferece um mapeamento facilitado entre as informações de contexto e as construções da modelagem. Sua aplicação busca satisfazer, particularmente, os requisitos de heterogeneidade, histórico, raciocínio e usabilidade (BETTINI et al., 2010).

Um exemplo deste tipo de abordagem é a *Context Modelling Language* (CML), que oferece uma notação gráfica para a descrição dos tipos de informação, suas classificações, metadados referentes a qualidade das informações e as dependências entre os diferentes tipos de informação (HENRICKSEN; INDULSKA, 2004).

As principais vantagens da CML são seu suporte para vários estágios do processo de engenharia de software, facilitado pela sua notação gráfica, e sua capacidade superior, em comparação com outras abordagens, de tratamento de imperfeições e histórico em informações de contexto. A obrigação da representação das informações de contexto como fatos atômicos, impossibilitando a declaração de estruturas hierárquicas, é uma das principais desvantagens da CML.

3.2.1.3.1 Modelagem Espacial

As informações espaciais têm importância especial em grande parte das aplicações sensíveis ao contexto. Uma prova disto é a presença deste tipo de informação nas principais definições e classificações de contexto vistas nas seções anteriores. A abordagem de modelagem espacial surge com o intuito de oferecer um mecanismo mais adequado que os propostos anteriormente para manipulação deste tipo de informação, visando principalmente os requisitos de mobilidade, histórico e provisionamento eficiente (BETTINI et al., 2010).

Muitos modelos espaciais são, na sua essência, modelos baseados em fatos, diferenciados pela organização das informações de contexto em termos de localização. Esta pode ser a localização física das entidades, dos sensores que capturam as informações, entre outras. A informação de localização pode ser pré-definida, quando as entidades são estáticas, ou obtidas a partir de sistemas de posicionamento, quando as entidades são móveis.

O modelo espacial tem sua principal aplicação em sistemas móveis. Estes sistemas apresentam um grande potencial de uso de informações de contexto para oferecer uma melhor interação com o usuário, em função do dinamismo ambiental inerente a sua execução. A principal desvantagem dos modelos espaciais está no esforço necessário para capturar e manter atualizados os dados de localização.

3.2.1.3.2 Modelagem Baseada em Ontologia

Considerando que informações de contexto podem ser caracterizadas como um tipo de conhecimento, o uso de ontologias para a representação deste conhecimento torna-se uma alternativa interessante. Os modelos baseados em ontologia exploram o poder de representação e de raciocínio automatizado, suportado atualmente por uma variedade de ferramentas. Isso permite expressar dados complexos de contexto, não representáveis com a mesma facilidade através de outras abordagens; compartilhar e integrar informações de contexto entre diferentes sistemas, através da formalização da semântica do contexto; verificar a consistência dos relacionamentos que descrevem o contexto; e inferir novas informações de contexto a partir dos dados e relacionamentos mais básicos capturados. Esta abordagem visa atender, principalmente, aos requisitos de heterogeneidade, relacionamento e raciocínio (BETTINI et al., 2010).

A OWL, em suas diversas variações, é a linguagem tipicamente utilizada para a construção de modelos baseados em ontologia (BETTINI et al., 2010). Isto se deve a popularização no uso da linguagem para representação de conhecimento em aplicações de diversos domínios e ao suporte ferramental disponível para criação, verificação e raciocínio sobre ontologias criadas com a linguagem. Várias ontologias utilizando linguagem OWL foram criadas com a proposta de representar informações de contexto. Destacam-se, entre estas, a *SOUPA* (CHEN et al., 2004), voltada para a modelagem de contextos em ambientes pervasivos, e a *CONON* (ZHANG; GU; WANG, 2005), cujo foco são ambientes de casas

inteligentes (*smart homes*).

Uma das principais vantagens no uso de ontologias para modelagem de informações de contexto está na expressividade formal, que por sua vez, favorece a inferência de novas informações. O suporte ferramental também pode ser considerado como um fator importante para a decisão pela utilização deste tipo de modelagem. Contudo, esta abordagem tem como principais desvantagens o baixo suporte para modelagem de aspectos temporais; questões de desempenho no processo de raciocínio, que dependem da expressividade da linguagem utilizada; e questões de escalabilidade, encontradas quando a ontologia é populada com um grande número de indivíduos.

3.2.1.3.3 Comparação entre as Abordagens

Nenhuma das abordagens de modelagem de informações de contexto apresentadas nas seções anteriores cobre todos os requisitos descritos na introdução desta seção (Seção 3.2.1.2). Cada abordagem contempla um subconjunto dos requisitos, tornando-a mais adequada para a modelagem de determinados sistemas.

Modelos baseados em fatos se destacam pela capacidade de lidar com a heterogeneidade e o histórico das informações de contexto, pelo compromisso entre o poder de expressividade e a eficiência de raciocínio e pelo suporte ao processo de engenharia de software. Têm como principal desvantagem a ausência de suporte para construções mais complexas, como hierarquia de informações de contexto.

A abordagem de modelagem espacial é principalmente útil para sistemas móveis, onde informações de localização podem ser utilizadas para oferecer uma melhor interação entre os usuários e o sistema. Esta abordagem apresenta como principal dificuldade a captura e atualização dos dados de localização.

Por fim, modelos baseados em ontologias oferecem vantagens em termos de representatividade de dados de contexto complexos, além de suporte para raciocínio automatizado sobre o contexto. Aspectos temporais, de desempenho e de escalabilidade, no entanto, são desvantagens deste tipo de abordagem.

A Tabela 1 apresenta uma comparação entre as abordagens de modelagem de informações de contexto no que tange o atendimento dos requisitos básicos apresentados. Com o objetivo de obter uma abordagem capaz de contemplar um conjunto maior de

requisitos, permitindo uma maior flexibilidade, alguns trabalhos, como (AGOSTINI; BETTINI; RIBONI, 2009) e (HENRICKSEN, 2004), propõe a combinação de duas ou mais abordagens, gerando assim uma modelagem híbrida.

Tabela 1 - Comparação das abordagens de modelagem contextual

Requisitos	**Abordagens de Modelagem**		
	Baseada em Fatos	**Espacial**	**Baseada em Ontologia**
Heterogeneidade	Sim	Parcial	Sim
Mobilidade	Parcial	Sim	Não
Relacionamento	Parcial	Parcial	Sim
Histórico	Sim	Sim	Não
Imperfeição	Parcial	Parcial	Não
Raciocínio	Parcial	Não	Sim
Usabilidade	Sim	Não	Parcial
Eficiência	Parcial	Sim	Não

Fonte: Adaptado de (BETTINI et al., 2010)

3.2.2 Computação Sensível ao Contexto

A primeira definição de "computação sensível ao contexto" (*context-aware computing*) foi feita em 1994 por Schilit e Theimer (1994), onde estes a descrevem como sendo o software que se adapta de acordo com seu local de uso, com a coleção de pessoas e objetos próximos, assim como às mudanças nestes objetos no tempo. Essa definição desconsidera as aplicações que apenas exibem informações de contexto para o usuário, restringindo àquelas que se alteram em função do contexto.

Uma outra definição, mais abrangente que a anterior, foi proposta em (DEY; ABOWD, 1999) e descreve um sistema como sensível ao contexto se este usa o contexto para prover informação e/ou serviços relevantes para o usuário, onde a relevância depende da tarefa do usuário. Esta definição compreende um número maior de sistemas sensíveis ao contexto visto que não requer que a aplicação se altere.

Sistemas sensíveis ao contexto são sistemas computacionais que fazem uso de informações de contexto para facilitar a interação com o usuário. Estas aplicações oferecem informações e funcionalidades, ou recursos, diferenciados de acordo com o contexto em que o usuário se encontra. As principais propostas para classificação de sistemas sensíveis ao contexto, de acordo com os recursos fornecidos, serão apresentadas nas seções a seguir.

3.2.2.1 Classificação de Sistemas Sensíveis ao Contexto

Com o objetivo de auxiliar na definição de tipos para aplicações sensíveis ao contexto, algumas propostas de categorização foram realizadas. As principais propostas definem categorias baseadas nos recursos ou no comportamento das aplicações.

A primeira tentativa de definir uma taxonomia para sistemas sensíveis ao contexto foi feita em (SCHILIT; ADAMS; WANT, 1994). Este trabalho define duas dimensões ortogonais através das quais um aplicação pode ser classificada. Uma dimensão diferencia as aplicações entre as que apresentam informações e as que executam comandos. A outra dimensão diferencia as aplicações entre as que as tarefas são executadas manualmente ou automaticamente. Aplicações que destacam informações referentes a entidades próximas para seleção manual do usuário são classificadas como aplicações de *seleção baseada em proximidade*. Aplicações que acrescentam, removem ou alteram as conexões entre componentes de acordo com o contexto do usuário, de forma automática, são classificadas como aplicações de *reconfiguração contextual automática*. Aplicações que apenas apresentam comandos para o usuário, conforme o contexto, são classificadas como aplicações de *comando contextual*. Por fim, aplicações que executam comandos automaticamente segundo o contexto do usuário são classificadas como aplicações de *ações disparadas segundo o contexto*. A Tabela 2 apresenta as classificações segundo as dimensões propostas.

Tabela 2 - Dimensões de classificação de sistemas sensíveis ao contexto

	Acionamento Manual	**Acionamento Automático**
Apresentam Informações	Seleção Baseada em Proximidade	Reconfiguração Contextual Automática
Executam Comandos	Comandos Contextuais	Ações Disparadas Segundo o Contexto

Fonte: Adaptado de (SCHILIT; ADAMS; WANT, 1994)

Uma outra taxonomia para características de sistemas sensíveis ao contexto foi proposta por (PASCOE, 1998). Esta proposta se diferencia da anterior, principalmente, pelo foco nas características centrais e não nas classes de aplicações. Segundo (PASCOE, 1998), a característica mais básica de sistemas sensíveis ao contexto é a *detecção contextual*. Esta característica representa a habilidade de detecção e apresentação das informações de contexto para o usuário. A *adaptação contextual* é a característica que corresponde a habilidade de

executar ou modificar um serviço automaticamente conforme o contexto do usuário. A habilidade de localizar e explorar recursos e serviços relevantes ao contexto do usuário está relacionada a característica chamada de *descoberta de recursos contextuais*. A última característica, *expansão contextual*, refere-se a habilidade de associar dados digitais ao contexto do usuário.

Tabela 3 - Equivalências entre taxonomias de sistemas sensíveis ao contexto

Schilit, Adams e Want (1994)	**Pascoe (1998)**	**Dey e Abowd (1999)**
Seleção Baseada em Proximidade	Detecção Contextual	Apresentação de Informações e Serviços para o usuário
Comando Contextual	*(sem equivalente)*	
Ações Disparadas Segundo o Contexto	Adaptação Contextual	Execução Automática de Serviços
Reconfiguração Contextual Automática	Descoberta de Recursos Contextuais	*(sem equivalente)*
(sem equivalente)	Expansão Contextual	Etiquetagem de Contexto para Visualização Posterior

Fonte: Baseado no trabalho de (DEY; ABOWD, 1999)

Combinando os trabalhos de Schilit *et al.* (1994) e Pascoe (1998), uma terceira proposta de taxonomia foi feita por Dey e Abowd (1999). Esta taxonomia foi pensada levando-se em consideração as principais diferenças entre as taxonomias anteriores e focando nas características que deveriam ser suportadas pelos sistemas sensíveis ao contexto. Segundo Dey e Abowd (1999), existem três características principais: *apresentação* de informações e serviços para o usuário, *execução automática* de serviços e *etiquetagem* de contexto para visualização posterior. A característica de *apresentação* é uma combinação da *seleção baseada em proximidade* e de *comando contextual* de Schilit *et al.*, com a noção de apresentação de contexto da característica *detecção contextual* proposta por Pascoe. *Execução automática* é uma combinação de *ações disparadas segundo o contexto*, de Schilit *et al.*, com a *adaptação contextual* de Pascoe. Por fim, a *etiquetagem* de contexto corresponde a *expansão contextual* de Pascoe. Na Tabela 3 é possível observar a equivalência entre as taxonomias.

Observa-se nos três trabalhos citados nesta seção uma preocupação com a apresentação e execução de serviços para o usuário de acordo com o contexto em que este se encontra. Esta é, de fato, a característica mais encontrada em sistemas sensíveis ao contexto.

3.3 Considerações

Este capítulo procurou apresentar os principais conceitos e características referentes a ontologia e computação sensível ao contexto.

A ontologia foi vista como um estudo da "realidade", vinculado a filosofia, com a sua aplicação na ciência da computação como mecanismo para representação formal e compartilhamento de conhecimento entre pessoas e máquinas.

Observou-se que os avanços nas pesquisas em aplicação de ontologia nos sistemas computacionais tem sido motivados pela proposta da Web Semântica, uma Web cujas informações poderiam ser processadas não apenas por pessoas, mas também por agentes computacionais.

Este capítulo explicita também que a construção de ontologias requer o uso de linguagens e que a formalidade desta afeta a expressividade e a eficiência de uma ontologia. Estas linguagens, apesar de suas diferenças, apresentam um conjunto de componentes comuns, compreendidos como necessários para as representações mais básicas de conhecimento: as classes, atributos, relações e instâncias.

Como exemplo de linguagens, foram apresentadas o *Resource Description Framework* (RDF) e a *Web Ontology Language* (OWL). O RDF é uma linguagem baseada na construção de triplas, particularmente adequada para a representação de dados e afirmações simples. A OWL é uma linguagem baseada no RDF, cujo propósito é a representação de ontologias. Ela oferece construções para definição de hierarquia de conceitos, restrições de cardinalidade, entre outras.

Em seguida, foi abordada a computação sensível ao contexto, referindo-se aos sistemas que utilizam informações de contexto para oferecer funcionalidades aos usuários.

Mostrou-se que as pesquisas iniciais definiam contexto através de enumerações e que este tipo de definição dificultava a identificação das informações como sendo ou não de contexto. Então, uma definição mais ampla foi apresentada, que relaciona o contexto com as informações das entidades relevantes para a interação do usuário com o sistema. Foram apresentadas também propostas de categorização de informações de contexto, que visam permitir uma melhor utilização destas no desenvolvimento de sistemas.

Observou-se que existe um conjunto de requisitos básicos encontrados em sistemas sensíveis ao contexto. Para o atendimento destes requisitos, diversas abordagens de

modelagem de informações de contexto foram propostas. Cada abordagem, no entanto, como foi apresentado, não é capaz de, individualmente, atender a todos os requisitos.

Abordagens de modelagem baseada em fatos se destacam pela capacidade de lidar com heterogeneidade e histórico, porém não permitem a representação de dados complexos. Modelagens espaciais têm grande aplicabilidade em sistemas móveis ou baseados em localização. No entanto, agregam ao sistema a complexidade de captura e atualização dos dados de localização. Por fim, as modelagens baseadas em ontologias exploram a expressividade e suporte ao raciocínio de linguagens como a OWL, porém apresentam questões de desempenho, escalabilidade e falta de suporte para histórico.

As principais definições de computação sensível ao contexto foram apresentadas. Foram apresentadas também as principais propostas de classificação de sistemas sensíveis ao contexto. As três propostas apresentadas se sobrepõe em diversos aspectos, diferenciando-se basicamente pelo foco nas classes ou nas características das aplicações e pelas considerações de algumas características.

4 PROPOSTA

A proposta deste trabalho, conforme observado anteriormente, é melhorar o processo de seleção de recursos em ambientes distribuídos, considerando a diminuição do tempo de espera das requisições por recursos e do tempo total, percebido pelos usuários, para a execução de um conjunto de requisições, ou *wall-clock time*, através da aplicação de ontologia e de orientação ao contexto.

Este capítulo é dedicado ao detalhamento desta proposta. Na seção a seguir são apresentados alguns trabalhos relacionados com a presente proposta. Depois, é apresentado um cenário atual de seleção de recursos em ambientes distribuídos e o cenário segundo a proposta deste trabalho.

4.1 Trabalhos Relacionados

Muitos trabalhos foram desenvolvidos na área de seleção de recursos em ambientes distribuídos. Nesta seção, serão apresentados alguns destes trabalhos.

O HTCondor (UW-MADISON, 2013) é um sistema gerenciador de recursos desenvolvido pela Universidade de Wisconsin-Madison, especializado para aplicações de computação intensiva. Este sistema implementa um mecanismo de seleção de recursos baseado em comparação exata de valores, utilizando palavras-chave. Cada elemento, ou nó, do ambiente publica periodicamente, no nó central do HTCondor, seus recursos através de um formato chamado ClassAd. O ClassAd é basicamente um documento composto por palavras-chave, representando o nome dos recursos, e seus valores. Para requisitar a execução de uma aplicação em um ambiente gerenciado pelo HTCondor, o usuário envia também um ClassAd para o nó central, especificando os recursos necessários. O processo de seleção do HTCondor, então, faz uma comparação entre o ClassAd da requisição e os ClassAds dos recursos, para identificar um nó compatível para o atendimento da requisição. As requisições que não encontram recursos compatíveis disponíveis, são colocadas em uma fila, que é reprocessada periodicamente.

No trabalho de Silva e Dantas (2007), uma abordagem de seleção de recursos em ambientes de *grid* computacional é proposta. Esse trabalho propõe também a integração de ontologias, construídas pelas *Virtual Organizations* (VO). A proposta consiste da construção

de uma ontologia de referência para o ambiente, que é utilizada para a integração das diversas ontologias. Essa integração é feita com intervenção de um especialista da própria VO. A seleção dos recursos no ambiente é realizada através de uma pesquisa, definida também por uma ontologia proposta no referido trabalho, em que os termos da ontologia integrada são utilizados. Para a obtenção de melhores resultados no processo de seleção, o elemento da arquitetura proposta, chamado *Matchmaker*, responsável pela comparação da requisição com os recursos disponíveis, utiliza um mecanismo de expansão de consultas. Esse mecanismo consiste da adição de novos critérios na consulta fornecida, a partir do conhecimento representado na ontologia integrada, visando aumentar os resultados da busca.

Uma outra abordagem de descobrimento semântico de recursos em *grids* computacionais é proposta por Somasundaram *et al.* em (2006), e aprimorada por Amarnath *et al.* em (2009). Esta abordagem propõe uma arquitetura onde existe uma camada responsável pela interpretação semântica (*knowledge layer*) dos recursos e das requisições. A camada de conhecimento proposta nestes trabalhos faz uso de uma ontologia, que descreve o ambiente distribuído, para construir uma base de conhecimento através das informações dos recursos disponibilizadas pelas camadas inferiores da arquitetura (*middleware layers*). Além disso, esta camada é responsável também pela descoberta de recursos compatíveis com a requisição do usuário. Os resultados apresentados nestes trabalhos demonstram uma quantidade de acertos muito superior comparado a processos de seleção baseados em comparações exatas com palavras-chave.

A arquitetura de seleção de serviços orientada ao contexto CAPEUS (*Context-Aware Packets Enabling Ubiquitous Services*) é proposta por Samulowitz *et al.* em (2002). Esse trabalho estabelece uma abordagem baseada em um documento chamado *Context-Aware Packet* (CAP) que descreve tanto os serviços quanto as requisições de um ambiente computacional ubíquo. O mecanismo de seleção de serviços proposto neste trabalho considera o contexto do usuário no momento da requisição. Sendo assim, utilizado o exemplo apresentado no trabalho, se o usuário requisita um serviço de impressão ao ambiente, entre as avaliações realizadas pelo mecanismo de seleção, estará a proximidade física do serviço em relação ao usuário, ou seja, será dada prioridade ao serviço de impressão mais próximo do usuário.

O presente trabalho considera a aplicação de ontologia e orientação ao contexto na seleção de recursos em ambientes distribuídos. Entende-se que essa proposta pode trazer

melhores resultados que o processo de seleção do HTCondor (UW-MADISON, 2013) por agregar conhecimento do domínio ao processo de seleção. Mecanismos de seleção, baseados em comparações exatas com palavras-chave, são naturalmente incapazes de compreender os relacionamentos semânticos entre os atributos e valores manipulados.

O processo de seleção aqui proposto compartilha com os trabalhos de Silva e Dantas (2007), Somasundaram *et al.* (2006) e Amarnath *et al.* (2009) o método de utilização de ontologia para representação do conhecimento do ambiente distribuído. Os diferenciais da presente proposta com relação a estas é a utilização explícita de informações do contexto, do ambiente e da requisição, no processo de seleção e uma menor interferência dos usuários no processo.

Por fim, o presente trabalho se diferencia do trabalho de Samulowitz *et al.* (2002), principalmente, pelo domínio explorado e por utilizar ontologia ao invés de uma comparação baseada em palavras-chave para fazer a seleção dos recursos. A Tabela 4 apresenta alguns parâmetros de comparação entre os trabalho analisados.

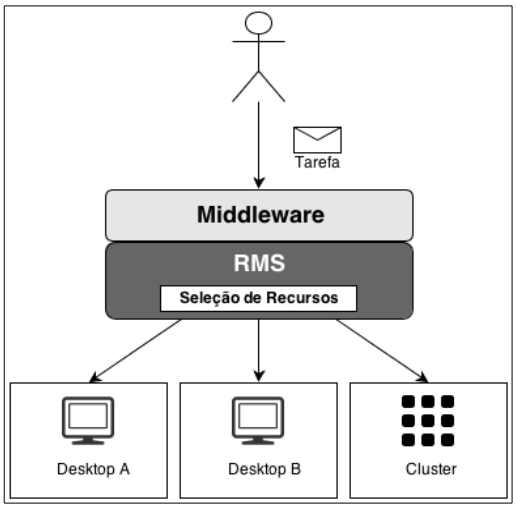
Tabela 4 - Comparação entre os trabalhos relacionados.

Parâmetros	**UW-Madison (2013)**	**Silva e Dantas (2007)**	**Somasundaram *et al.* (2006) e Amarnath *et al.* (2009)**	**Samulowitz, Michahelles e Linnhoff-Popien (2002)**
Objetivo	Seleção de recursos em ambientes distribuídos	Seleção de recursos em ambientes distribuídos	Seleção de recursos em ambientes distribuídos	Seleção de serviços em ambientes ubíquos
Método de seleção	Comparação exata de palavras-chave	Baseado em ontologia, com expansão de consultas	Baseado em ontologia, com inferência de conhecimento	Comparação exata de palavras-chave
Interferência do usuário	Envio da requisição	Envio da requisição e construção da ontologia	Envio da requisição e decisão final sobre recurso que será utilizado, entre os compatíveis	Envio da requisição
Linguagem de representação da ontologia	-	OWL	OWL	-
Reasoner	-	Jena Generic Rule Reasoner (JGRR)	Algernon	-
Orientação ao contexto	Não	Não	Não	Sim

4.2 Cenário Atual

A maneira mais comum de interação com um ambiente distribuído é através de sistemas gerenciadores de recursos (*Resource Management System – RMS*) ou *middlewares* de computação distribuída. Os RMSs normalmente oferecem funcionalidades mais voltadas para a coordenação dos recursos enquanto os *middlewares*, que pode inclusive fazer uso de um ou mais RMSs na sua execução, tendem a oferecer mais facilidades em termos de utilização dos ambientes, como interface gráfica de usuário, capacidade de reserva de recursos, controles de acesso, entre outras. A Figura 7 apresenta uma possível configuração de ambiente distribuído com um *middleware* atuando sobre um RMS.

Figura 7 - Configuração de ambiente distribuído com middleware e RMS



Em ambos os casos, no entanto, o usuário que deseja ter sua tarefa executada precisa informar no momento da requisição um conjunto de parâmetros que definem a demanda de recursos da sua tarefa. Estes parâmetros variam de acordo com as características do ambiente e da tarefa em questão. Exemplos comuns de parâmetros são a quantidade de memória livre, espaço em disco rígido, número de processadores, entre outros.

O processo de seleção implementado em grande parte dos RMSs e *middlewares* de computação distribuída consiste de um comparação exata de valores, baseada em palavras-chave, semelhante ao que ocorre nos bancos de dados relacionais. Por meio desta busca são localizados nós do ambiente que possuem recursos de acordo com a especificação do usuário. Em alguns casos é possível até mesmo utilizar critérios de comparação mais flexíveis, como

seleção de nós com oferta de recursos maior ou igual ao especificado na requisição, por exemplo.

Este tipo de abordagem delega muito da responsabilidade sobre a seleção de recursos para o usuário. A comparação exata baseada nas palavras-chaves definidas nos critérios de execução da tarefa impõe uma restrição muito forte sobre os recursos elegíveis. Desta forma, caso o usuário não tenha um elevado conhecimento do ambiente em que deseja executar sua tarefa, que o torne capaz de definir critérios suficientes para selecionar várias alternativas entre os recursos disponíveis, é possível que este não faça o melhor uso possível do ambiente e, até mesmo, não tenha sua requisição atendida imediatamente.

Situações deste tipo, implicam em um maior tempo de espera pelo atendimento das requisições e, por consequência, um maior tempo total de execução de um conjunto de requisições percebido pelo usuário. É objetivando reduzir o impacto dessas duas possíveis consequências do método de seleção de recursos, que o cenário, detalhado na seção à seguir, é proposto.

4.3 Cenário Proposto

Este trabalho propõe a utilização de ontologia e de orientação ao contexto no processo de seleção de recursos para diminuir o tempo de espera pelo atendimento de requisições e tempo total, percebido pelos usuários, para a execução de um conjunto de requisições, ou *wall-clock time*. Em termos práticos, a ideia consiste de um módulo de seleção de recursos que possa ser integrado a um *middleware* de computação distribuída ou a um RMS. A Figura 8 apresenta uma configuração em que a seleção de recursos baseada em ontologia e orientada ao contexto é integrada em um *middleware* de computação distribuída executando sobre um RMS.

Um módulo de seleção de recursos segundo esta proposta possui, ao menos, três atividades: representar o conhecimento do ambiente distribuído utilizando uma ontologia de domínio, localizar os recursos compatíveis com a requisição do usuário a partir do conhecimento representado na ontologia e identificar, entre os recursos compatíveis, qual o recurso mais adequado, visando um menor tempo de atendimento da requisição e um menor *wall-clock time*, utilizando informações do contexto em que a requisição está sendo realizada. Na Figura 9, o módulo de seleção proposto é detalhado com relação a estas três atividades.

Figura 8 - Processo de seleção de recursos proposto

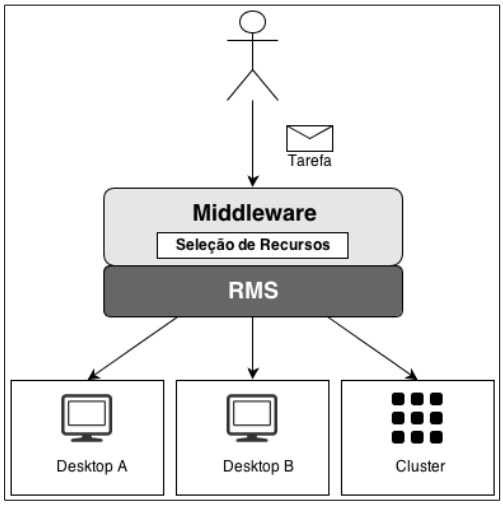
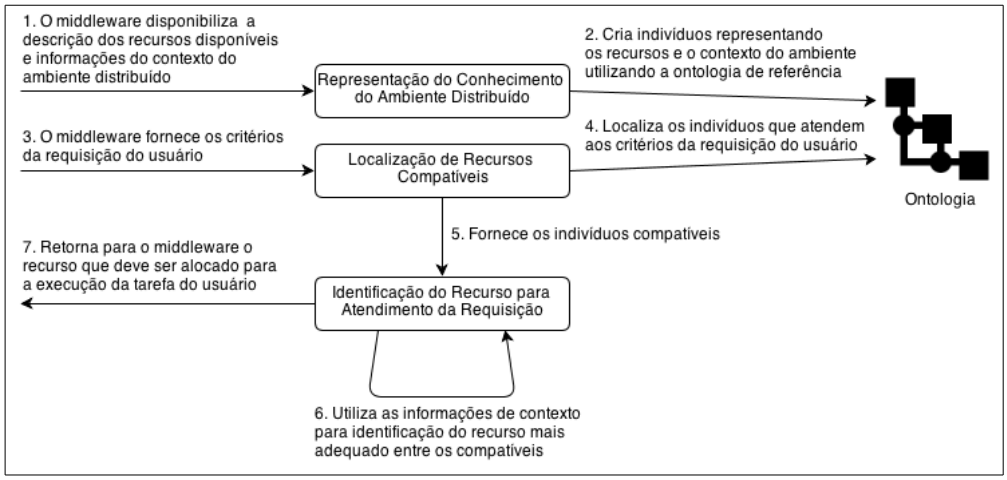


Figura 9 - Atividades realizadas pelo módulo de seleção de recursos proposto



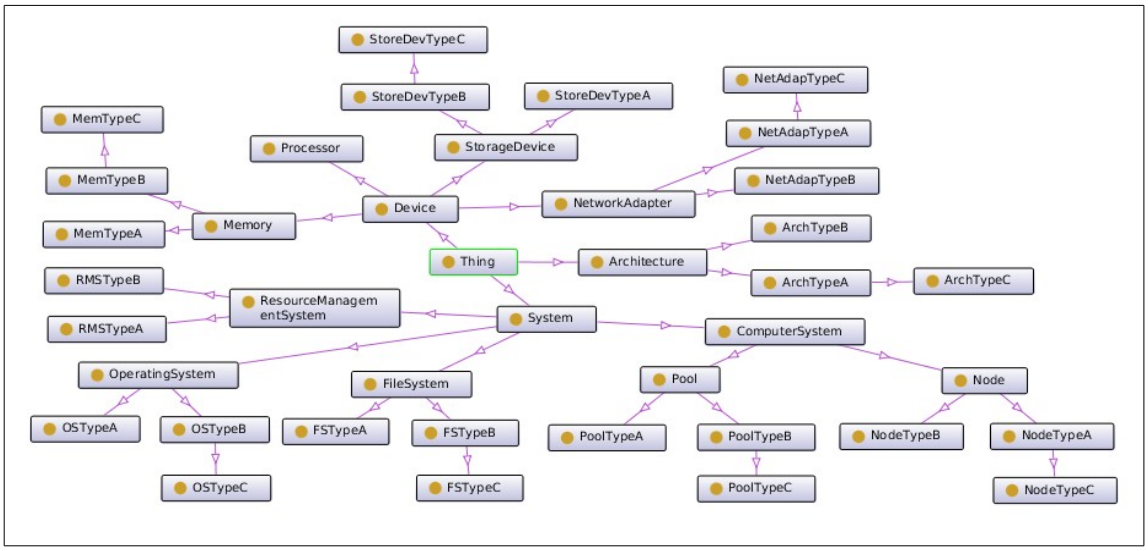
O papel da ontologia na seleção de recursos proposta está na representação do conhecimento sobre o ambiente distribuído. Num primeiro momento utiliza-se uma ontologia de referência para criar indivíduos representando os recursos disponíveis e as informações de contexto. Considera-se aqui que estas informações são fornecidas para o processo de seleção pelo *middleware* ou pelo RMS que administra o ambiente, fugindo do escopo deste trabalho o detalhamento de como estas são coletadas.

Uma vez criados estes indivíduos e os relacionamentos entre eles, segundo a ontologia de referência, o processo de seleção passa a contar com uma base de conhecimento sobre o ambiente. Desta forma, esta base de conhecimento é utilizada para descobrir recursos

compatíveis com os critérios da requisição do usuário, utilizando um raciocinador semântico (*reasoner*), que considera não apenas a comparação exata dos valores dos critérios com os valores dos recursos disponíveis, mas também o significado destes valores. A descoberta de recursos equivalentes, possibilitada pelo uso da ontologia, é um dos principais motivos pelo qual se espera que esta proposta reduza o tempo de espera pelo atendimento de requisições.

Observa-se então que a construção da ontologia de referência influencia diretamente no processo de seleção proposto. A Figura 10 apresenta um exemplo de uma ontologia descrevendo um ambiente distribuído. As setas interligando os termos na ontologia apresentada na figura representam relações de hierarquia do tipo IS-A (é um), no qual lê-se que *OSTypeA* é um *OperatingSystem*, por exemplo. No capítulo 5 é apresentada a ontologia de referência que foi construída e utilizada no protótipo de validação da proposta deste trabalho.

Figura 10 - Exemplo de ontologia representando um ambiente distribuído



Em um ambiente distribuído é comum a existência de vários elementos, ou nós, compartilhando recursos. Sendo assim, é possível que a seleção do recurso para a execução da tarefa solicitada pelo usuário encontre mais de um nó compatível no ambiente. Este trabalho propõe que para identificar o nó que deve ser utilizado para a execução da tarefa entre os possíveis, previamente selecionados, sejam consideradas as informações de contexto do ambiente.

Diversas informações de contexto poderiam ser utilizadas nesta etapa do processo. Exemplos são a carga e o número de núcleos do processador, a latência e o porcentual de

perda de pacotes da rede, a memória livre, o espaço disponível em disco, entre outras. Cada variável de contexto deve ser avaliada de acordo com sua característica e contribuição para o processo de seleção. Vamos considerar o número de núcleos do processador como um exemplo. O processo deve priorizar elementos com o menor número de núcleos, entendendo que todos os nós previamente selecionados têm ao menos a quantidade solicitada pelo usuário. Desta forma, evita-se o desperdício de recursos e possibilita-se que uma tarefa posterior, que demande mais núcleos de processador, possa ser atendida.

Este trabalho propõe ainda a atribuição de pesos para cada variável de contexto utilizada no processo. Assim, é possível diferenciar a relevância de cada variável no processo de decisão. A atribuição dos pesos poderia ser realizada de uma forma fixa, ou seja, definida pelo desenvolvedor do módulo de seleção de recursos ou através de alguma variável de configuração da aplicação, ou de uma forma dinâmica, podendo utilizar mecanismos mais complexos, como técnicas de inteligência artificial, por exemplo. Através deste mecanismo, seria possível definir, por exemplo, que o número de núcleos do processador é um fator mais relevante que o espaço disponível em disco, na decisão final.

Com a aplicação da orientação ao contexto nesta decisão final sobre o recurso que será alocado para a execução da tarefa do usuário, espera-se contribuir tanto para a redução do tempo de espera de atendimento de requisições como para a redução do *wall-clock time*. A alocação dos recursos, considerando critérios contidos ou não na requisição do usuário, permite que a utilização do ambiente seja otimizada e, por consequência, que mais requisições possam ser atendidas em um menor intervalo de tempo.

O capítulo a seguir descreve o ambiente e os experimentos realizados para a comprovação da proposta descrita nesta seção.

5 AMBIENTE E RESULTADOS EXPERIMENTAIS

Para validar a proposta deste trabalho, apresentada no capítulo anterior, foi desenvolvido um protótipo funcional que implementa um processo de seleção de recursos em ambientes distribuídos baseado em ontologia e orientado ao contexto. Este protótipo foi submetido a um conjunto de testes, caracterizados em três cenários de interesse para a avaliação da proposta.

Este capítulo inicia com uma apresentação do ambiente de testes, seguido pela descrição do protótipo. Por fim, são detalhados os três cenários de teste considerados, com a análise dos seus respectivos resultados.

5.1 Ambiente

Os experimentos de validação da proposta deste trabalho foram realizados em um ambiente distribuído controlado construído no Laboratório de Pesquisas em Sistemas Distribuídos (LaPeSD) da Universidade Federal de Santa Catarina (UFSC). O LaPeSD atua em projetos de pesquisa nas áreas de bancos de dados distribuídos, computação de alto desempenho, *middleware* de computação distribuída, sistemas multimídia distribuídos, entre outros.

Tabela 5 - Configuração das máquinas utilizadas no experimento

Nome	Processador	Nº de Núcleos	Memória RAM	Disco Rígido	Sistema Operacional	Física/ Virtual
lapesd010	Intel Core i5-2400 3,10GHz	4	4 GB	500 GB	Ubuntu 12.04 Desktop 64-bit	Física
lapesd-5	Intel Core i5-2400 3,10GHz	4	3 GB	500 GB	Windows XP Professional SP3 32-bit	Física
vnode01	Intel Core i5-2400 3,10GHz	1	512 MB	8 GB	Ubuntu 12.04 Server 64-bit	Virtual
vnode02	Intel Core i5-2400 3,10GHz	1	192 MB	8 GB	Windows XP Professional SP1 64-bit	Virtual
vnode03	Intel Core i5-2400 3,10GHz	1	512 MB	8 GB	CentOS 6.3 32-bit	Virtual
vnode04	Intel Core i5-2400 3,10GHz	1	128 MB	20 GB	FreeBSD 8.3 64-bit	Virtual

O ambiente distribuído utilizado foi composto de duas máquinas físicas e quatro máquinas virtuais, totalizando seis elementos, ou nós. A Tabela 5 apresenta as configurações

destes nós. As máquinas virtuais vnode01, vnode02, vnode03 e vnode04 foram instanciadas na máquina física lapesd010.

A opção pelo uso de máquinas virtuais se deu por vários motivos. O primeiro deles foi a facilidade de criação de máquinas com características heterogêneas para a composição do ambiente. A virtualização oferece uma alternativa conveniente para configuração de máquinas com diferentes sistemas operacionais, quantidade de processadores, tamanhos de memória volátil e permanente, entre outras.

A relação custo-benefício também foi considerada na escolha. A possibilidade de obter virtualmente cinco nós de execução (quatro virtuais e um físico) no ambiente distribuído utilizando apenas uma máquina física permitiu otimizar o uso dos equipamentos disponíveis. Deve ser observado aqui que esta decisão foi vantajosa especificamente neste experimento porque não há preocupação com o desempenho das aplicações que serão executadas. Em ambientes de execução de aplicações reais, uma análise mais detalhada deve ser realizada.

Por fim, o isolamento provido pela virtualização de alguns elementos do ambiente foi considerado suficiente para a realização dos experimentos e a validação da proposta. Apesar de compartilharem os recursos físicos da máquina hospedeira, cada máquina virtual interage de forma independente com o ambiente, através da troca de mensagens via suas interfaces de rede virtuais. Novamente, como a proposta deste trabalho atua sobre a seleção de recursos e não sobre o desempenho da execução das aplicações, o uso destes recursos foi considerado como suficiente.

A administração deste ambiente distribuído foi realizada utilizando o *middleware* de computação distribuída em desenvolvimento no LaPeSD, executando sobre o sistema gerenciador de recursos HTCondor (UW-MADISON, 2013), versão 7.8. Em função da etapa em que se encontra o desenvolvimento do *middleware* do LaPeSD, no momento da realização deste trabalho, ainda não há referências para sua implementação. Diversas foram as razões para a adoção destas duas ferramentas.

Como este experimento pretende validar um processo de seleção de recursos, é de particular interesse contar com um ambiente computacional heterogêneo. Esta característica permite avaliar o processo de seleção com um conjunto mais variado de indivíduos. Portanto, o suporte à ambientes computacionais distribuídos heterogêneos, como o utilizado neste experimento, é fundamental. E este requisito é atendido pelo HTCondor e pelo *middleware* do LaPeSD.

A arquitetura de implementação do *middleware* também contribuiu para sua escolha. O *middleware* de computação distribuída do LaPeSD é uma aplicação Web, orientada a componentes, seguindo uma abordagem de linha de produtos de software. Essa abordagem facilita o desenvolvimento de um conjunto de sistemas de um mesmo domínio, com variações em suas funcionalidades, atribuídas ao conjunto de componentes utilizados na sua construção. Tendo em vista que neste experimento pretende-se construir um protótipo funcional que implemente o processo de seleção de recursos proposto, a facilidade de integração do protótipo através deste tipo de abordagem foi vista como favorável.

Além do aspecto da arquitetura, a existência de componentes já desenvolvidos para a comunicação entre o *middleware* e o HTCondor, também influenciou na decisão. Quanto ao uso do HTCondor, mais especificamente, a existência de documentação suficiente, a característica de ser um sistema de código-fonte aberto e a experiência de utilização do mesmo por parte da equipe do LaPeSD, foram outros fatores considerados.

A instalação dos softwares no ambiente experimental se deu da seguinte forma. A máquina lapesd010 foi escolhida como o elemento principal do ambiente. Essa escolha foi motivada pela quantidade de recursos disponíveis ser vista como suficiente para suportar a execução dos experimentos. Nesta máquina foram instalados o servidor central do HTCondor, o servidor web Tomcat (APACHE, 2013), para a execução do *middleware* de computação distribuída, e o VirtualBox (ORACLE, 2013b), para a virtualização das máquinas vnode01, vnode02, vnode03 e vnode04.

Nas demais máquinas do ambiente (lapesd-5, vnode01, vnode02, vnode03 e vnode04), foram instalados o HTCondor, configurado como nó de execução de tarefas, e o Tomcat, para a execução do componente de comunicação do *middleware* com o HTCondor. Esta comunicação é realizada através de Web Services e permite a execução de operações como submissão e monitoramento de tarefas e monitoramento do ambiente.

A instalação do HTCondor na máquina com sistema operacional FreeBSD (vnode04) gerou alguns problemas. Apesar de a documentação do HTCondor afirmar ter suporte para o FreeBSD e este possuir a versão 7.8 na sua coleção de pacotes instaláveis (*ports*), após a compilação do código-fonte, a aplicação não inicializava. A solução foi substituir o pacote *patch,* instalado por padrão no sistema operacional, pelo *gpatch*, e só então fazer a compilação e instalação do HTCondor no FreeBSD.

Para a realização dos experimentos, duas versões do *middleware* foram

instanciadas. Uma delas contendo o protótipo que implementa a seleção de recursos utilizando ontologia e orientação ao contexto e outra versão que delega a seleção para o sistema gerenciador de recursos, que neste caso foi o HTCondor. O objetivo é utilizar uma mesma infraestrutura de software para a comparação das duas abordagens de seleção de recursos, minimizando assim possíveis ruídos nas medições provocados por diferenças na comunicação com o ambiente distribuído.

A geração e o envio das requisições para o ambiente, na realização dos experimentos, foi realizada por uma aplicação desenvolvida exclusivamente para este fim. Esta aplicação se comunica com o *middleware* através de Web Services e seu propósito é basicamente enviar sequencialmente um conjunto de requisições com critérios previamente definidos. Outra função desta aplicação é monitorar a conclusão das tarefas no ambiente, computando o tempo total utilizado para a conclusão do conjunto de tarefas enviados. Mais detalhes sobre os tipos de requisições, a quantidade de iterações e o sequenciamento das invocações serão dados em seções seguintes, quando serão descritos os cenários de teste.

Antes de apresentar os cenários utilizados para a validação desta proposta, é detalhado na próxima seção o protótipo construído, que implementa um processo de seleção de recursos utilizando ontologia e orientação ao contexto.

5.2 Protótipo

O protótipo do módulo de seleção de recursos foi desenvolvido em linguagem de programação Java, versão 1.7. Esse requisito foi uma consequência da decisão pelo uso do *middleware* de computação distribuída do LaPeSD. Para aproveitar os serviços já disponibilizados pelo *middleware* na realização dos experimentos, foi necessário implementar o protótipo como um componente da arquitetura do *middleware*, respeitando uma interface de componente de seleção de recursos previamente definida.

A arquitetura do *middleware* impõe a implementação de uma interface com dois métodos para componentes de seleção de recursos. A definição da interface é apresentada no Quadro 1. O método *feed* é invocado pelo *middleware* para alimentar o componente de seleção com as informações sobre os nós. Ele é invocado periodicamente, em intervalos de dois segundos, objetivando manter o mecanismo de seleção com uma imagem mais atual possível do estado do ambiente. Por sua vez, o método *perform* é utilizado para fazer a

seleção de um nó, representado pela classe *ExecutingUnit*, para a execução de uma tarefa, representada pelo parâmetro *job*.

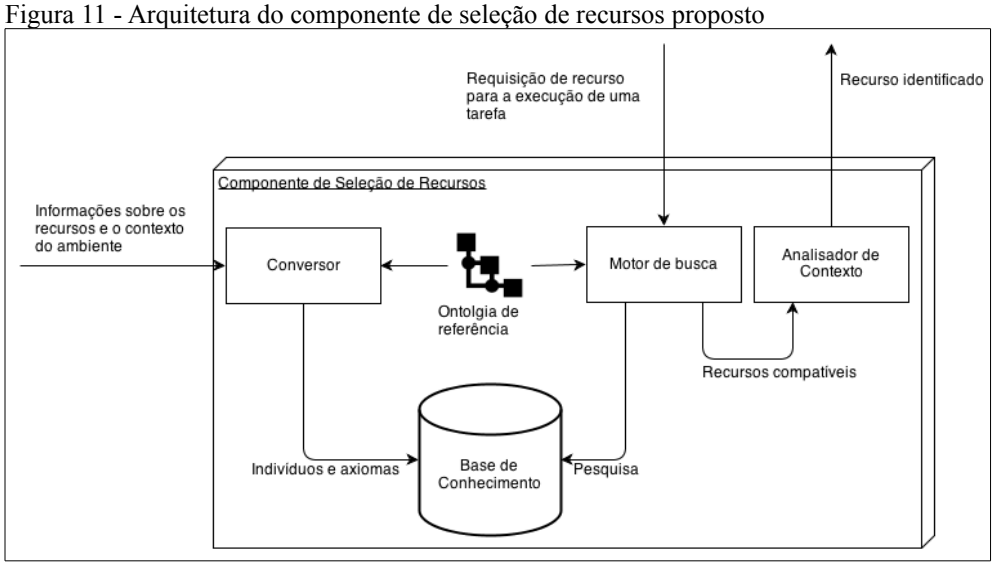
Quadro 1 - Definição da interface de componente de seleção de recursos do middleware

```
public interface ResourceSelector {

        ExecutingUnit perform(Job job) throws ResourceSelectorException;

        void feed(Collection<Node> nodes) throws ResourceSelectorException;

}
```

O desenvolvimento do protótipo levou em consideração as três atividades apresentadas na proposta deste trabalho: representação do conhecimento do ambiente distribuído, localização de recursos compatíveis e identificação do recurso para atendimento da requisição. A arquitetura do componente é apresentada na Figura 11.

O conversor é o elemento da arquitetura do componente responsável pela tradução das informações dos recursos e do contexto do ambiente para alimentar a base de conhecimento, utilizando a linguagem OWL. Neste protótipo, utilizou-se a API (*Application Programming Interface*) OWLAPI (OWLAPI, 2013) para manipular os elementos da OWL. A escolha se deu por esta ser uma API de código aberto, por oferecer suporte à quase totalidade das construções da linguagem OWL2 e por suportar diferentes *reasoners*.

Figura 11 - Arquitetura do componente de seleção de recursos proposto



No conversor, todas as informações recebidas do ambiente distribuído são

convertidas para indivíduos e axiomas da linguagem OWL e armazenados na base de conhecimento. Em virtude do tamanho do ambiente experimental e pela dinamicidade das informações, optou-se neste trabalho pelo armazenamento da base de conhecimento na memória principal. Ou seja, não foi utilizado nos experimentos nenhum sistema de gerenciamento de banco de dados baseado em persistência permanente.

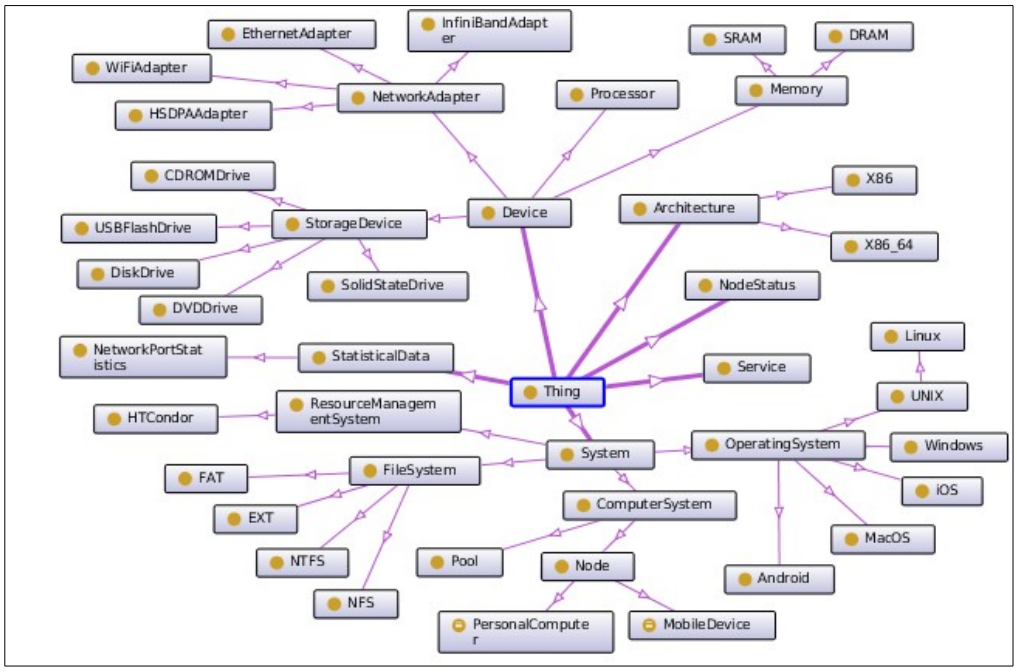
A responsabilidade de buscar na base de conhecimento os indivíduos que atendem aos critérios passados na requisição do usuário é atribuída neste componente ao elemento chamado motor de busca. Inicialmente, este elemento prepara uma expressão OWL que define as restrições definidas na requisição do usuário. Esta expressão é então passada para um *reasoner* que, utilizando inferência, localiza os indivíduos compatíveis na base de conhecimento. Utilizou-se neste protótipo o *reasoner* Hermit (OXFORD, 2013). Os motivos foram o suporte nativo para a OWLAPI e a capacidade de processamento de ontologias escritas em linguagem OWL.

Tanto o conversor quanto o motor de busca do componente fazem uso de uma ontologia de referência para executar suas funções. Esta ontologia de referência, cuja hierarquia é apresentada na Figura 12, representa os conceitos e os seus relacionamentos pertinentes a um ambiente distribuído. A ontologia construída neste protótipo foi escrita em linguagem OWL, utilizando o editor Protégé (STANFORD, 2013). O Protégé é um software de código aberto que oferece uma interface gráfica de usuário amigável para a construção de ontologias.

Os conceitos e relacionamentos representados nesta ontologia foram baseados nas características do domínio e em outros trabalhos publicados. A principal fonte externa foi o CIM (*Common Information Model*) (DMTF, 2013). O CIM é uma proposta de definições comuns para o gerenciamento de sistemas de informação, redes, aplicações e serviços. Seu foco principal é padronizar o vocabulário utilizado no domínio de sistemas computacionais. Este modelo segue um paradigma de orientação a objetos e é representado em UML (*Unified Modelling Language*) (OMG, 2013). Outros trabalhos também considerados na construção desta ontologia foram (SILVA; DANTAS, 2007) e (AMARNATH et al., 2009).

É importante enfatizar que este trabalho não tem a pretensão de definir uma ontologia completa, com todos os conceitos e relacionamentos de um ambiente de computação distribuída. A ontologia criada se limitou a definir um conjunto de conceitos relevantes ao processo de seleção de recursos e ao ambiente utilizado neste trabalho.

Figura 12 - Ontologia de referência



Uma vez identificados pelo motor de busca na base de conhecimento, os indivíduos compatíveis com a requisição do usuário são encaminhados para o analisador de contexto. O analisador de contexto é o elemento que identifica entre os indivíduos pré-selecionados, qual é mais adequado para a execução da tarefa. Esta decisão é realizada com base em informações do contexto disponibilizadas periodicamente pelo *middleware* em conjunto com as informações dos recursos do ambiente.

No analisador de contexto deste protótipo, foram consideradas as seguintes variáveis de contexto computacional: espaço disponível no sistema de arquivos, memória virtual livre, porcentual de carga do processador, número de núcleos disponíveis no processador, porcentual de perda de pacotes na rede, atraso de rede, considerando como métrica o valor do *Round-Trip Time* (RTT), e o sistema operacional. Esta lista não representa a totalidade de variáveis de contexto possíveis em um ambiente distribuído. Foram escolhidas estas pela disponibilidade nativa das informações através do *middleware* utilizado e por terem sido consideradas suficientes para a validação da proposta deste trabalho.

O processo de identificação implementado no analisador de contexto deste protótipo consiste da ordenação dos indivíduos com base em um indicador. Este indicador representa uma comparação das variáveis de contexto entre dois indivíduos. A comparação é

feita aos pares em função da utilização da interface *Comparator* da API Java para a ordenação dos indivíduos na implementação deste protótipo. São priorizados os indivíduos com menor indicador a cada comparação.

Este indicador é calculado para cada indivíduo do par sendo comparado pelo somatório da multiplicação, da normalização pelo máximo, dos valores de cada variável de contexto, por um peso. O Quadro 2 ilustra um exemplo do cálculo do indicador para a variável de contexto número de núcleos disponíveis no processador entre as máquinas lapesd010 e vnode01.

Quadro 2 - Exemplo de cálculo do indicador utilizado pelo analisador de contexto

```
a = Número de núcleos disponíveis no processador da máquina lapesd010 = 4
b = Número de núcleos disponíveis no processador da máquina vnode01 = 1
maximo = max(a, b) = max(4, 1) = 4
a_normalizado = a / maximo = 4 / 4 = 1,00
b_normalizado = b / maximo = 1 / 4 = 0,25
indicador_a = indicador_a + (a_normalizado x peso)
indicador_b = indicador_b + (b_normalizado x peso)
```

A normalização dos valores tem por finalidade equalizar a influência de cada variável, independente da magnitude dos seus valores absolutos, no indicador final. Os pesos, por outro lado, permitem controlar explicitamente esta influência. Através deste mecanismo é possível, por exemplo, atribuir mais importância a uma determinada variável de contexto no cálculo do indicador, provocando alterações na forma como a seleção do indivíduo é realizada.

Neste protótipo, os pesos foram atribuídos de forma fixa, na própria implementação. Os valores dos pesos de cada variável de contexto podem ser observados na Tabela 6. Apesar de todas as variáveis mencionadas terem participado dos experimentos, apenas duas, número de núcleos disponíveis do processador e sistema operacional, tiveram seus pesos particularmente calibrados. Isso ocorreu em função dos cenários de teste utilizados para a validação da proposta, descritos na próxima seção. O sistema operacional recebeu valor de peso igual a 2,0 para favorecer a decisão do usuário sobre o sistema operacional desejado. O número de núcleos disponíveis do processador, por sua vez, recebeu valor de peso igual a 1,5 considerando que este é um recurso caro ao sistema, comparado aos demais, que deve ser otimizado, evitando desperdícios.

Tabela 6 - Pesos das variáveis de contexto no cálculo do indicador

Variável de contexto	Peso da variável no cálculo do indicador
Espaço disponível no sistema de arquivos	1,0
Memória virtual livre	1,0
Porcentual de carga do processador	1,0
Número de núcleos disponíveis do processador	1,5
Porcentual de perda de pacotes na rede	1,0
Atraso de rede (RTT)	1,0
Sistema operacional	2,0

Como observado anteriormente, os indivíduos com menor indicador são priorizados. Essa abordagem leva em consideração que o motor de busca seleciona indivíduos que possuam recursos equivalentes ou superiores ao requisitado pelo usuário. A razão para isso é procurar atender um número maior de requisições. Desta forma, quanto menor o valor do indicador, significa que o recurso está mais próximo do requisitado pelo usuário. Por consequência, menos recursos são desperdiçados, podendo ser utilizados para as próximas requisições.

O analisador de contexto finaliza a operação do componente de seleção de recursos, entregando ao *middleware* a identificação do nó que possui os recursos suficientes para a execução da tarefa. A alocação do recurso e o gerenciamento da execução da tarefa é responsabilidade do *middleware* e não faz parte do escopo deste trabalho.

O detalhamento dos cenários e das métricas consideradas nos experimentos realizados para a validação deste trabalho é realizado na próxima seção.

5.3 Cenários de Teste

Os experimentos realizados neste trabalho se concentraram em alguns cenários específicos visando demonstrar que, através da aplicação de ontologia e de orientação ao contexto, o processo de seleção de recursos em um ambiente distribuído pode ser melhorado com relação a redução do tempo de espera pelo atendimento das requisições e a redução do tempo total percebido pelo usuário, ou *wall-clock time*, para a execução de um conjunto de tarefas.

O tempo de espera pelo atendimento das requisições consiste do intervalo entre o momento do envio da requisição para o ambiente distribuído e o momento da alocação de um

recurso para o atendimento da requisição. O cálculo desta métrica levou em consideração as diferenças entre as duas instâncias do *middleware* utilizadas no experimento.

A instância do *middleware* que delega a seleção para o sistema gerenciador de recursos, no caso o HTCondor, faz o cálculo do tempo de espera pelo intervalo entre o momento do envio da requisição do *middleware* para o HTCondor e o momento da alocação do recurso para a execução da tarefa. Como nesta instância do *middleware* não há nenhum processamento de seleção sobre a requisição, esta é simplesmente encaminhada para o HTCondor. O tempo de permanência da requisição no *middleware* é irrelevante e foi desconsiderado. O valor obtido com o cálculo nesta instância representa, então, na prática, o tempo de espera das requisições na fila do HTCondor. Toda a requisição que não encontra recursos disponíveis para a execução no HTCondor, são encaminhadas para esta fila. As requisições desta fila são analisadas periodicamente, em intervalos de vinte segundos.

Na instância do *middleware* que utiliza o protótipo desenvolvido neste trabalho como componente de seleção de recursos, o cálculo do tempo de espera é realizado pelo intervalo entre a chegada da requisição no *middleware* e a identificação do recurso para a sua execução. Não foi considerado aqui o momento da alocação do recurso para o cálculo do tempo de espera para reduzir a interferência do HTCondor no cálculo. O HTCondor não disponibiliza uma forma de desativar ou ignorar seu processo de seleção de recursos. Desta forma, mesmo quando o *middleware* informa especificamente qual máquina deve ser alocada para a execução da tarefa, o HTCondor, ainda assim, executa seu processo de seleção. Inclusive, mantendo sua periodicidade mínima de vinte segundos entre cada processamento da sua fila. Se o momento da alocação do recurso fosse considerado aqui, o tempo de espera observado com o uso da proposta deste trabalho poderia ter um acréscimo de até, aproximadamente, vinte segundos por tarefa, provocado por esta interferência do HTCondor. Sendo assim, o valor observado no tempo de espera da instância do *middleware* que utiliza a seleção com ontologia e orientação ao contexto corresponde ao tempo em que a requisição ficou retida na fila de requisições do *middleware*.

A fila de requisições do *middleware* do LaPeSD opera sob o princípio *first in, first out*, onde a primeira requisição a entrar na fila é a primeira requisição a ser atendida. O *middleware* processa todas as requisições da fila, em intervalos de três segundos, sequencialmente. Apesar de bastante simplificado, o mecanismo implementado no *middleware* poderia ser comparado a um mecanismo de escalonamento de curto prazo *(short-*

term scheduling), onde as tarefas estão prontas para serem executadas, aguardando apenas a disponibilidade de recursos.

O *wall-clock time*, neste trabalho, é a métrica que representa o tempo decorrido entre o envio da primeira requisição de um conjunto para o ambiente e a conclusão da execução da última tarefa deste mesmo conjunto. É o tempo percebido pelo usuário do ambiente, que envia um conjunto de requisições e aguarda a sua finalização.

O cálculo do *wall-clock time* não diferenciou as instâncias do *middleware* utilizadas no experimento. Isso porque, para este cálculo, é necessário considerar o momento da finalização da última tarefa, de um conjunto, enviada para o ambiente. Desta forma, não foi possível aqui isolar a instância do *middleware* que contém o processo de seleção proposto neste trabalho, das interferências da fila e do processo de seleção do HTCondor. Esta limitação provocou um ruído nos resultados obtidos nesta instância do *middleware*.

A medida adotada para minimizar o impacto deste ruído, que como observado anteriormente, pode ser de até, aproximadamente, vinte segundos por requisição, foi utilizar um tempo de execução de tarefa maior. Foram enviados, então, para o ambiente, em todos os cenários, tarefas que mantém um nó ocupado por oitocentos segundos, o que faz com que um possível ruído corresponda a 2,5% do *wall-clock time* obtido. Contudo, nos experimentos, foram observados outros ruídos provocados pelo HTCondor, que prejudicaram as medições da instância do *middleware* com seleção baseada em ontologia e orientada ao contexto. Detalhes sobre estes ruídos serão apresentados nas próximas seções, junto com as avaliações dos resultados.

As medidas do tempo de espera pelo atendimento da requisição e do *wall-clock time* foram realizadas para todos os cenários considerados neste experimento. Em cada cenário, conjuntos pré-definidos de requisições, com critérios específicos, foram submetidos para cada instância do *middleware*. O sequenciamento do envio das requisições foi mantido igual para as duas instâncias, de forma que a comparação das métricas pudesse ser feita considerando a mesma condição de carga do ambiente. Este procedimento foi repetido por dez vezes, ou seja, cada instância do *middleware* recebeu dez conjuntos de requisições. A consideração de que é necessário ter um volume maior de amostras para a avaliação dos resultados, foi a razão para esta decisão.

Este experimento se dividiu em três cenários. O primeiro cenário explora especificamente uma situação em que o uso da ontologia permite atender uma quantidade

maior de requisições, através da descoberta de recursos equivalentes. No segundo cenário, é observado a otimização no uso dos recursos, proporcionado pela orientação ao contexto do processo de seleção proposto, auxiliando também no aumento da capacidade de atendimento de requisições. Por fim, o último cenário não foca em uma situação específica, mas sim no comportamento geral do ambiente diante de vários conjuntos de requisições. Os detalhes de cada cenário, assim como os resultados obtidos, são apresentados nas próximas seções.

5.3.1 Cenário de Teste 1: Validação do Uso da Ontologia

Este cenário de teste foi construído especificamente para validar o uso da ontologia segundo o processo de seleção de recursos proposto neste trabalho. Espera-se neste cenário que, através do uso do conhecimento do ambiente distribuído, representado na ontologia, o processo de seleção proposto seja capaz de atender uma quantidade maior de requisições e, desta forma, reduzir o tempo de espera e o *wall-clock time*.

Na ontologia criada neste protótipo, baseando-se nos trabalhos de Amarnath *et al.* (2009) e Silva e Dantas (2007), definiu-se que um sistema operacional Linux é uma subclasse do sistema operacional UNIX, conforme pode ser observado na Figura 12 (seção 5.2). Isso sugere que, o processo de seleção de recursos proposto, na presença de requisições que tenham como critério o sistema operacional UNIX, poderia identificar nós disponíveis no ambiente distribuído com sistema operacional Linux como equivalentes e elegíveis para o atendimento das requisições. Processos de seleção que fazem comparações exatas, baseadas em palavras-chave, não teriam naturalmente condições de lidar com este tipo de condição.

Quadro 3 - Critérios da requisição utilizada no cenário de teste 1.

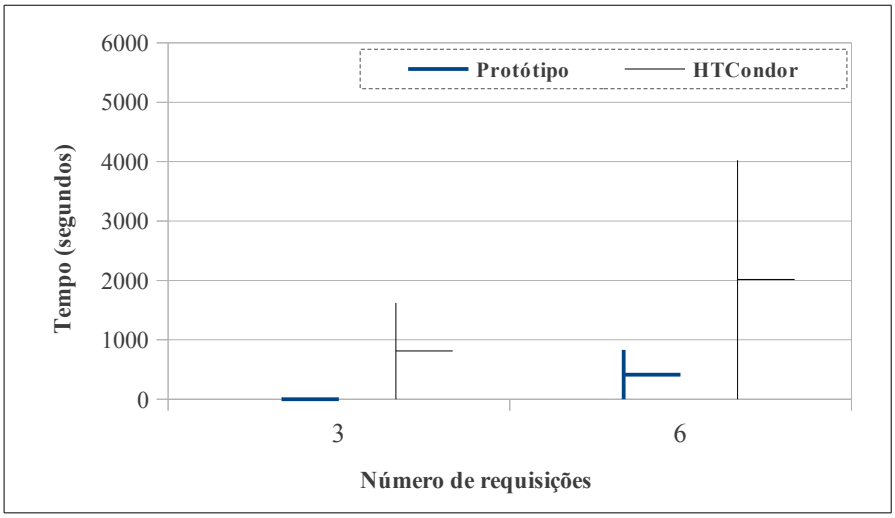
```
Sistema Operacional = UNIX
Arquitetura = X86_64
Largura do Barramento de Dados = 64 bits
Número de Núcleos Disponíveis = 1
```

A situação descrita é explorada neste cenário de teste. São enviados para o ambiente dois conjuntos de requisições iguais, cujo critério de principal interesse é o sistema operacional UNIX. O primeiro conjunto contém três requisições e o segundo seis. Essas quantidades foram definidas pensando na capacidade máxima esperada da ocupação do ambiente, diante deste tipo de requisição, segundo o processo de seleção de recursos proposto.

O Quadro 3 apresenta os critérios da requisição.

A Figura 13 apresenta um gráfico sumarizando os resultados do tempo de espera pelo atendimento de requisições segundo este cenário de teste. A linha contínua do gráfico representa o tempo de espera observado nos conjuntos de requisições enviados para a instância do *middleware* que possui o protótipo com a implementação do processo de seleção de recursos baseado em ontologia e orientado ao contexto. Enquanto que a linha pontilhada mostra os resultados da mesma métrica obtida na instância do *middleware* que delega a seleção dos recursos para o HTCondor.

Figura 13 - Resultados do tempo de espera no cenário de teste 1.

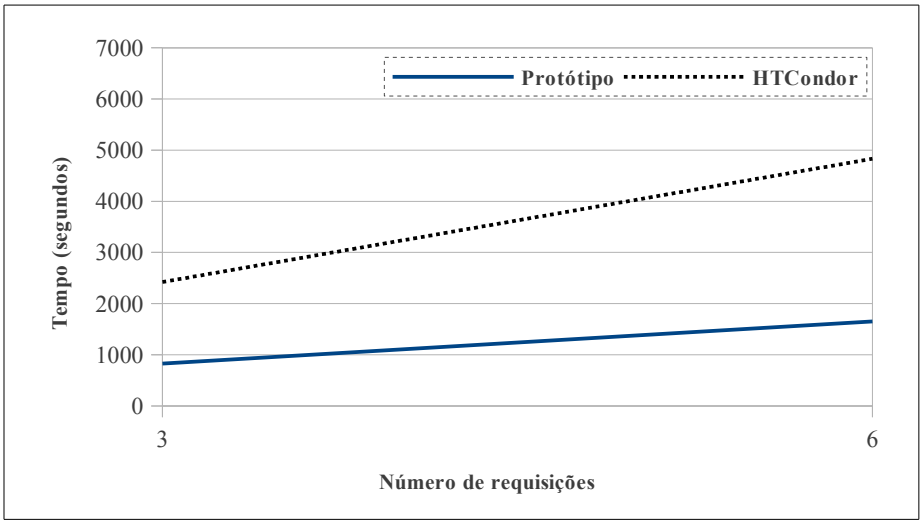


Os resultados deixam evidente que o processo de seleção de recursos proposto teve um desempenho superior no tempo de espera pelo atendimento de requisições em comparação com o processo que faz comparações exatas, baseadas em palavras-chave, representado nestes experimentos pelo processo de seleção do HTCondor. A explicação para este resultado é simples.

Observando a Tabela 5 (seção 5.1), o ambiente distribuído utilizado nos experimentos contém apenas um nó com o sistema operacional UNIX (vnode04). No entanto, existem três nós com sistema operacional Linux (lapesd010, vnode01 e vnode03) que poderiam ser utilizados, segundo a definição da ontologia. Como existem restrições adicionais nas requisições, envolvendo arquitetura e largura do barramento de dados do processador de 64 bits, o nó vnode03 deve ser desconsiderado pelo processo de seleção. Sendo assim, era esperado pelo processo de seleção que utiliza ontologia reconhecer três nós para o

atendimento das requisições enviadas neste cenário de teste, um com sistema operacional UNIX e dois com sistema operacional Linux.

Figura 14 - Resultados do wall-clock time no cenário de teste 1.



Pela mesma razão, como pode ser visto na Figura 14, houve uma redução no *wall-clock time* do cenário, ou seja, o tempo percebido por um usuário para finalização da execução dos conjuntos de requisições enviados para a instância do *middleware* com a seleção de recursos baseada em ontologia e orientada ao contexto foi menor. Sob outro ponto de vista, pode se entender, a partir dos resultados obtidos, que, neste cenário de teste, o processo de seleção proposto fez um melhor aproveitamento dos recursos disponíveis, atendendo a um número maior de requisições em um mesmo intervalo de tempo.

5.3.2 Cenário de Teste 2: Validação da Orientação ao Contexto

O objetivo deste cenário de teste é validar a orientação ao contexto aplicada ao processo de seleção de recursos proposto neste trabalho. O principal papel da orientação ao contexto nesta proposta é identificar o recurso adequado ao atendimento de uma requisição a partir de um conjunto de recursos, previamente selecionados, elegíveis para tal função. O resultado esperado na execução dos experimentos neste cenário de teste é um aumento na capacidade de atendimento de requisições, provocado agora pela redução no desperdício de recursos. Assim como no cenário anterior, serão utilizados como métrica para esta avaliação o tempo de espera pelo atendimento de requisições e o *wall-clock time*.

Entre as sete variáveis de contexto tratadas no protótipo de seleção de recursos implementado neste trabalho, optou-se por explorar, neste cenário de teste, o número de núcleos disponíveis no processador. Esta decisão foi motivada pelo entendimento de que a economia deste recurso é particularmente especial em ambientes distribuídos. A execução paralela de sistemas, apesar de não fazer parte especificamente do escopo deste trabalho, é uma das principais formas de utilização de ambientes distribuídos quando se deseja obter aumento de desempenho na execução de sistemas. Sendo assim, alocar recursos com uma quantidade de núcleos de processador maior do que a necessária para a execução de uma tarefa pode fazer com que requisições posteriores tenham que aguardar pela liberação de recursos ocupados, porém, na prática, subutilizados.

Em processos de seleção de recursos baseados em comparação exata, representado nos experimentos deste trabalho pelo processo implementado no HTCondor, uma informação como o número de núcleos disponíveis de um processador é percebida como um simples valor numérico. Este tipo de mecanismo é naturalmente incapaz de considerar o significado ou o contexto vinculado a esta informação. Desta forma, não há expectativas de que a seleção seguindo este método seja capaz de otimizar a utilização dos recursos.

Quadro 4 - Critérios dos dois tipos de requisições do cenário de teste 2.

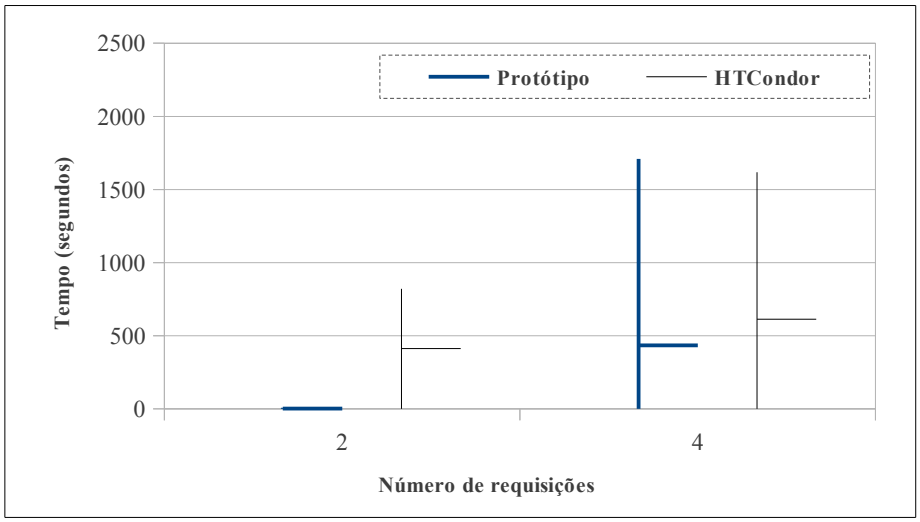
Requisição A:	Sistema Operacional = Linux
	Arquitetura = X86_64
	Largura do Barramento de Dados = 64 bits
	Número de Núcleos Disponíveis = 1
Requisição B:	Sistema Operacional = Linux
	Arquitetura = X86_64
	Largura do Barramento de Dados = 64 bits
	Número de Núcleos Disponíveis = 4

Para validar o processo de seleção proposto neste trabalho contra a situação descrita acima, foi criado este cenário de teste. Num primeiro momento, foi criado um conjunto com duas requisições, enviadas uma após a outra, sendo que a primeira (Requisição A) tem como principal critério uma quantidade de núcleos de processador inferior a segunda (Requisição B). Os demais critérios, conforme pode ser observado no Quadro 4, restringem os nós elegíveis, do ambiente distribuído utilizado nos experimentos, a apenas dois: lapesd010 e vnode01. Um segundo teste considerou um conjunto com quatro requisições, semelhante ao primeiro conjunto, sendo que a primeira e a terceira requisição solicitam menos núcleos de

processador (Requisição A) que a segunda e a quarta requisição (Requisição B). A razão para este segundo conjunto de teste é analisar o comportamento do processo de seleção tratando requisições na fila de espera.

Os resultados do tempo de espera pelo atendimento das requisições dos dois conjuntos de teste executados neste cenário são apresentados no gráfico da Figura 15. Pode-se verificar que em termos médios (linhas horizontais), o protótipo que utiliza ontologia e orientação ao contexto para fazer a seleção de recursos, representado pela linha contínua, teve melhores resultados que a seleção por comparação exata, utilizada pelo HTCondor, representado pela linha pontilhada.

Figura 15 - Resultados do tempo de espera no cenário de teste 2.



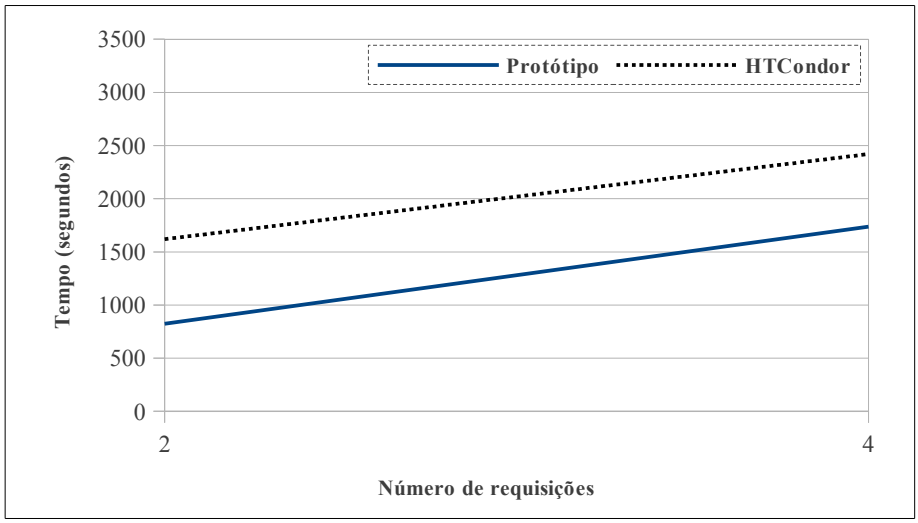
A razão para este melhor resultado no processo de seleção proposto neste trabalho é atribuído a orientação ao contexto. Como esperado, o analisador de contexto do protótipo, ao receber a primeira requisição, que estabelece como critério um núcleo de processador, define que, entre os nós lapesd010 e vnode01, o nó vnode01 tem recursos suficientes para a execução. Desta forma, na chegada da segunda requisição, que estabelece como critério quatro núcleos de processador, o nó lapesd010 ainda está disponível para o atendimento desta requisição. A seleção realizada pelo HTCondor, por outro lado, alocou para a primeira requisição o nó com quatro núcleos (lapesd010). Sendo assim, a segunda requisição teve que aguardar na fila pela liberação do nó lapesd010, provocando um tempo de espera maior, conforme observado nos resultados.

Houve, no entanto, nos resultados, um evento inesperado. Nos testes com

conjuntos de quatro requisições, a seleção proposta apresentou um valor máximo de tempo de espera superior ao valor máximo observado no HTCondor. Esta ocorrência foi provocada por um atraso na alocação dos recursos por parte do HTCondor. Como mencionado anteriormente, o HTCondor pode impor um tempo de espera de até vinte segundos em cada requisição, mesmo quando esta passa pelo processo de seleção proposto.

No incidente observado no teste, a primeira requisição teve o recurso alocado pelo HTCondor com atraso, fazendo com que sua finalização ocorresse após a finalização da segunda requisição, que não teve o atraso na alocação. Como resultado, quando a terceira requisição foi retirada da fila e avaliada pelo processo de seleção, apenas o recurso liberado pela segunda requisição, o lapesd010, com quatro núcleos de processador, estava disponível. Essa seleção fez com que a quarta requisição precisasse aguardar pela finalização da terceira, que estava utilizando o único recurso do ambiente que atende aos seus critérios. Esta sequência de eventos justifica o valor máximo superior observado no gráfico.

Figura 16 - Resultados do wall-clock time no cenário de teste 2.



Os benefícios do uso da orientação ao contexto no processo de seleção neste cenário também foi percebido no *wall-clock time* dos testes. A Figura 16 apresenta em forma de gráfico os resultados do *wall-clock time* para os dois conjuntos de teste. O tempo de execução dos conjuntos de requisições foi superior no HTCondor, conforme esperado, porque as decisões tomadas pelo processo de seleção fizeram com que recursos fossem subutilizados enquanto requisições com demandas maiores não puderam ser atendidas.

5.3.3 Cenário de Teste 3: Comportamento Geral

No terceiro e último cenário de teste deste trabalho, o objetivo é avaliar o comportamento do processo de seleção proposto comparado a um processo de seleção baseado em comparação exata, representado pelo HTCondor, diante de um conjunto diferenciado de requisições. Diferente dos cenários de teste anteriores, não é forçado aqui uma situação específica, determinada pelo tipo e pela ordenação das requisições. Espera-se observar neste cenário mais diverso a mesma melhora, ainda que em menores proporções, no tempo de espera pelo atendimento de requisições e no *wall-clock time*.

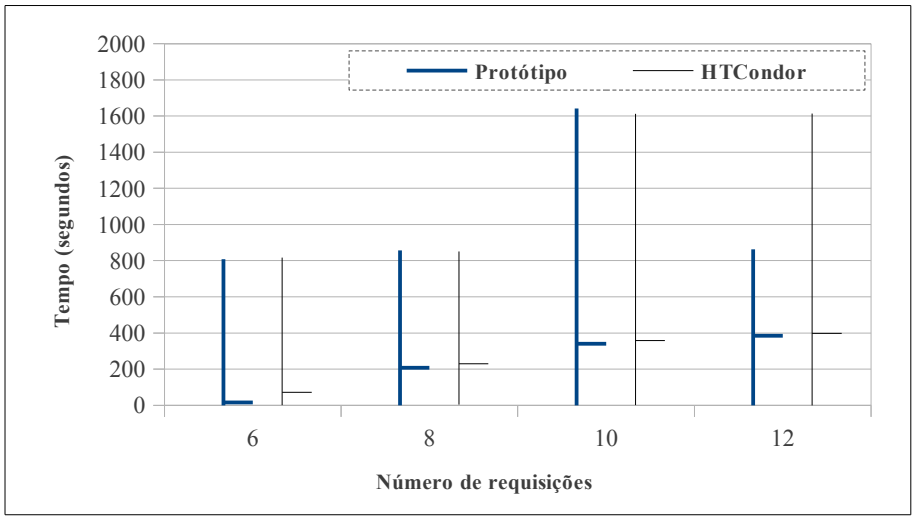
Quadro 5 - Critérios dos tipos de requisições considerados no cenário de testes 3.

Requisição A:	Sistema Operacional = UNIX
	Arquitetura = X86_64
	Largura do Barramento de Dados = 64 bits
	Número de Núcleos Disponíveis = 1
Requisição B:	Sistema Operacional = Linux
	Arquitetura = X86
	Largura do Barramento de Dados = 32 bits
	Número de Núcleos Disponíveis = 1
Requisição C:	Sistema Operacional = Linux
	Arquitetura = X86_64
	Largura do Barramento de Dados = 64 bits
	Número de Núcleos Disponíveis = 1
Requisição D:	Sistema Operacional = Linux
	Arquitetura = X86_64
	Largura do Barramento de Dados = 64 bits
	Número de Núcleos Disponíveis = 4
Requisição E:	Sistema Operacional = Windows
	Arquitetura = X86
	Largura do Barramento de Dados = 32 bits
	Número de Núcleos Disponíveis = 4
Requisição F:	Sistema Operacional = Windows
	Arquitetura = X86_64
	Largura do Barramento de Dados = 64 bits
	Número de Núcleos Disponíveis = 1

Foram utilizados neste cenário quatro conjuntos de teste: com seis, oito, dez e doze requisições. Cada conjunto de teste foi enviado dez vezes para cada instância do *middleware*. Os tipos e a ordenação das requisições presentes em cada conjunto de teste e em cada iteração foram aleatórios. As únicas restrições impostas no cenário foram que os tipos de requisições fossem selecionados a partir de um conjunto pré-definido e que o mesmo conjunto

de requisições fosse enviado para as duas instâncias do *middleware*. A primeira restrição visa evitar requisições que não tenham recurso compatível no ambiente e a segunda, garantir que os resultados obtidos em cada instância sejam comparáveis, por se basearem em uma mesma situação de carga do ambiente distribuído. A relação com as requisições permitidas é apresentada no Quadro 5.

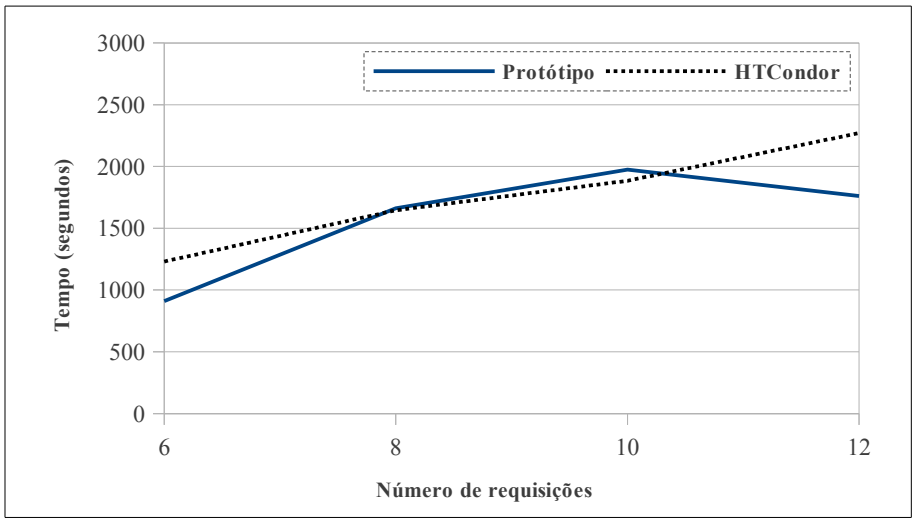
Figura 17 - Resultados do tempo de espera no cenário de teste 3.



A Figura 17 apresenta os resultados do tempo de espera pelo atendimento de requisições neste cenário de teste. O resultados do *wall-clock time* do cenário são apresentados na Figura 18.

Observa-se nos resultados do tempo de espera pelo atendimento de requisições que, na média (linha horizontal), o processo de seleção proposto teve tempos menores que o processo de seleção do HTCondor. Apesar da pequena diferença entre os tempos médios na maior parte dos conjuntos de teste do cenário, o resultado é visto como muito positivo, porque, o fato de considerar o contexto no processo de seleção, por si só, já é observado como uma melhora no processo. Considerando que, assim como aconteceu nos outros cenários de teste, as medidas obtidas com o protótipo tiveram alguma influência, proporcionada por algum atraso na alocação e liberação de recursos pelo HTCondor, estima-se que em condições ideais, os resultados do processo de seleção proposto possam ser ainda melhores.

Figura 18 - Resultados do wall-clock time no cenário de teste 3.



Nos resultados do *wall-clock time* deste cenário de teste, é possível observar mais claramente a melhor distribuição dos recursos por parte do protótipo implementado neste trabalho. Para os conjuntos de teste com seis e doze requisições, o tempo total do teste no protótipo teve uma diferença significativa, para menos, comparado ao tempo total com a seleção do HTCondor. Nestes dois conjuntos de teste, mais especificamente, estima-se que uma distribuição ideal manteria todos os recursos ocupados durante todo o período de teste, considerando que entre as seis e doze requisições houvessem uma mesma quantidade de cada tipo de requisições. Ou seja, considerando que houvesse uma requisição com critérios adequados para cada nó do ambiente distribuído, é esperado que um processo de seleção tenha capacidade de atender todas as requisições. Na média, este objetivo foi atingido pelo processo de seleção baseado em ontologia e orientado ao contexto.

5.4 Considerações

Este capítulo detalhou o ambiente distribuído utilizado, a implementação do processo de seleção proposto e os cenários de teste e resultados obtidos na realização deste trabalho.

O ambiente distribuído utilizado neste trabalho, criado no Laboratório de Pesquisa em Sistemas Distribuídos (LaPeSD) da Universidade Federal de Santa Catarina (UFSC), consistiu de seis elementos, ou nós, com características heterogêneas em termos de sistema operacional, processador, etc. A utilização de máquinas virtuais no ambiente contribuiu para a

execução dos experimentos pelas facilidades de configuração, isolamento e otimização do uso de equipamentos inerentes ao processo de virtualização.

A administração do ambiente foi realizada pelo *middleware* do LaPeSD, atuando sobre o sistema gerenciador de recursos HTCondor. A escolha desta configuração favoreceu o desenvolvimento do protótipo com a implementação da proposta, que resultou em um componente que pôde ser facilmente integrado a composição da instância do *middleware* utilizada nos experimentos.

A composição do protótipo de seleção de recursos desenvolvido neste trabalho pode ser resumida em três elementos principais: conversor, motor de busca e analisador de contexto. O conversor é o elemento responsável pela tradução das informações e do contexto do ambiente distribuído para indivíduos e axiomas OWL para armazenamento na base de conhecimento. A identificação dos recursos compatível com a requisição do usuário na base de conhecimento é responsabilidade do motor de busca. Foi utilizado o *reasoner* HermiT como parte do motor de busca construído no protótipo. Por fim, os indivíduos localizados pelo motor de busca são submetidos ao analisador de contexto, que baseado em um conjunto de variáveis de contexto, identifica o recurso para o atendimento da requisição.

Três cenários de teste foram construídos para a validação da proposta deste trabalho. Para todos os cenários, as medidas do tempo de espera pelo atendimento de requisições e o *wall-clock time*, que é o tempo percebido pelo usuário para a execução de um conjunto de requisições no ambiente distribuído, foram considerados como métricas para comparação entre as abordagens de seleção de recursos.

O primeiro cenário explorou a representação de equivalência entre sistemas operacionais para expandir as possibilidades de alocação de recursos para requisições. Este tipo de avaliação não é naturalmente possível através de processos de seleção baseados em comparação exata de valores a partir de palavras-chave, como no HTCondor.

Os benefícios da utilização da orientação ao contexto na seleção de recursos foi observado na segundo cenário de teste. A avaliação foi feita considerando a otimização do uso de núcleos de processadores. Os experimentos mostraram que o processo de seleção do HTCondor, por considerar a informação como um simples valor numérico, foi incapaz de compreender o contexto por trás da utilização do recurso e, por consequência, de otimizar sua alocação.

No último cenário de teste, avaliou-se o processo de seleção diante de uma carga

variada de requisições, sem forçar nenhuma situação específica. Observou-se neste cenário, em termos médios, melhores resultados no processo de seleção proposto em comparação com o processo de seleção do HTCondor.

Concluindo, os resultados obtidos foram considerados suficientes para a validação do método proposto. A realização dos experimentos em um ambiente controlado e o estabelecimento de cenários de teste focados em aspectos específicos da abordagem proposta favoreceram a avaliação. Por fim, a realização de testes em um cenário mais geral serviu para fortalecer a proposta.

6 CONCLUSÕES E TRABALHOS FUTUROS

A realização deste trabalho foi motivada pelo desafio de melhorar o processo de seleção de recursos em ambientes distribuídos, considerando a redução do tempo de espera pelo atendimento de requisições e do tempo total percebido pelo usuário para a execução de um conjunto de requisições, ou *wall-clock time*. Objetivando resolver esse problema, se propôs um método de seleção de recursos baseado em ontologia e orientado ao contexto.

Através do desenvolvimento de um protótipo funcional, implementando o método de seleção proposto; da construção de uma ontologia de domínio, contendo um conjunto de conceitos e relacionamentos identificados como mais relevantes para a seleção de recursos em ambientes distribuídos; e da utilização de algumas informações de contexto disponíveis no ambiente, foram realizados vários experimentos, divididos em três cenários de teste, para validação do método proposto neste trabalho.

Os experimentos destes três cenários de teste foram realizados em um ambiente controlado, criado no Laboratório de Pesquisas em Sistemas Distribuídos (LaPeSD) da Universidade Federal de Santa Catarina (UFSC). Foram utilizadas duas instâncias do *middleware* de computação distribuída do LaPeSD na realização dos experimentos. Uma instância integrada com o protótipo de seleção de recursos desenvolvido neste trabalho e outra, delegando a seleção de recursos para o sistema gerenciador de recursos HTCondor. Vale ressaltar que a configuração do ambiente distribuído utilizado neste experimento ocasionou alguns problemas, principalmente com relação a instalação do HTCondor.

Em todos os cenários de teste avaliados, o método de seleção proposto obteve, na média, melhores resultados que o método de seleção baseado em comparação exata de valores, através de palavras-chave, representado, nos experimentos, pelo processo de seleção implementando no sistema gerenciador de recursos HTCondor. A esses resultados se atribui o fato do aumento da capacidade de atendimento, através da identificação de recursos equivalentes proporcionada pela ontologia criada, e pela otimização no uso dos recursos do ambiente, consequente da orientação do processo ao contexto do ambiente e da requisição do usuário.

Durante a realização dos experimentos, se percebeu uma grande influência nos resultados do método de seleção proposto neste trabalho, provocado por atrasos na seleção, alocação e liberação de recursos do HTCondor, que chegavam, em algumas situações, a até

vinte segundos por requisição enviada para o ambiente. Mesmo definindo no *middleware* o recurso para a execução da tarefa, não foi possível desabilitar ou ignorar o processamento realizado no HTCondor. A medida adotada foi utilizar um tempo de execução maior nas tarefas, oitocentos segundos, de forma a minimizar o impacto destes atrasos nos resultados dos testes.

A partir dos resultados obtidos, concluiu-se que o método proposto é viável e oferece uma melhora em termos de redução do tempo de espera pelo atendimento de requisições e de redução do *wall-clock time*. Esta melhora foi mais evidente nos dois primeiros cenários de teste, onde situações específicas foram exploradas. Contudo, mesmo em proporções menores, foram observados melhores resultados também no terceiro cenário de teste, o qual contou com uma carga de requisições mais variada no ambiente.

Considerando os atrasos mencionados anteriormente, provocados pelo HTCondor, acredita-se que em condições ideais, os resultados obtidos com o método proposto podem ser ainda melhores. É importante considerar também que, além de ter sua requisição atendida mais rapidamente, o usuário tem sua interação com o ambiente facilitada, através do método de seleção proposto, visto que parte do conhecimento necessário para fazer uso dos recursos foi colocado no processo de seleção.

Um direcionamento para futuras pesquisas envolvendo a evolução do método proposto neste trabalho poderia ser no sentido da ampliação da ontologia, através da consideração de novos conceitos e relacionamentos, e das informações de contexto consideradas no processo. Neste trabalho, utilizou-se um pequeno conjunto de variáveis de contexto computacional, escolhido em função da disponibilidade no ambiente. Entende-se que outros tipos de informações de contexto, como variáveis temporais, refletindo o histórico de seleções realizadas, por exemplo, poderiam resultar em melhorias.

Adicionalmente, a utilização de técnicas de inteligência artificial para a definição dos pesos utilizados na avaliação das informações de contexto seria uma alternativa interessante em comparação com a definição estática realizada no protótipo funcional construído para os experimentos deste trabalho.

Por fim, a comparação do método proposto com outros métodos de seleção de recursos ou considerando outros tipos de métricas, traria novas contribuições para essa área de pesquisa.

REFERÊNCIAS

ADAPTIVECOMPUTING. **TORQUE Resource Manager**. Disponível em: <http://www.adaptivecomputing.com/products/open-source/torque/>. Acesso em: 6 fev. 2013.

AGOSTINI, A.; BETTINI, C.; RIBONI, D. Hybrid Reasoning in the CARE Middleware for Context-Awareness. **International Journal of Web Engineering and Technology**, v. 5, n. 1, p. 3-23, 2009.

AMARNATH, B. R. et al. Ontology-based Grid resource management. **Journal Software—Practice & Experience**, p. 1419-1438, 2009.

APACHE. **Apache Tomcat**. Disponível em: <http://tomcat.apache.org/>. Acesso em: 9 maio. 2013.

BERKELEY, U. OF. **The Sprite Operating System**. Disponível em: <http://www.eecs.berkeley.edu/Research/Projects/CS/sprite/sprite.html>. Acesso em: 7 fev. 2013.

BERNERS-LEE, T. **Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web**. 1. ed. [S.l.] HarperBusiness, 2000. p. 256

BETTINI, C. et al. A survey of context modelling and reasoning techniques. **Pervasive and Mobile Computing**, v. 6, n. 2, p. 161-180, abr. 2010.

BORST, N. W. **Construction of engineering ontologies**. [S.l.] Institute of Telematica and Information Technology, 1997.

BRAY, T. et al. **Extensible Markup Language (XML) 1.0 (Fifth Edition)**. Disponível em: <http://www.w3.org/TR/2008/REC-xml-20081126/>. Acesso em: 17 nov. 2012.

BRICKLEY, D.; GUHA, R. V. **RDF Vocabulary Description Language 1.0 : RDF Schema**. Disponível em: <http://www.w3.org/TR/rdf-schema>.

CHEN, G.; KOTZ, D. **A survey of context-aware mobile computing research**. [S.l: s.n.]. Disponível em: <http://www.cs.dartmouth.edu/~dfk/papers/chen-survey-tr.pdf>. Acesso em: 8 out. 2012.

CHEN, H. et al. SOUPA: standard ontology for ubiquitous and pervasive applications. **The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.**, p. 258-267, 2004.

COULOURIS, G. et al. **Distributed Systes: Concepts and Design**. Fifth Edit ed. Boston: Addison-Wesley, 2012. p. 1067

DANTAS, M. **Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais**. Rio de Janeiro: Axcel Books do Brasil, 2005. p. 278

DEY, A.; ABOWD, G.; SALBER, D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. **Human-Computer Interaction**, v. 16, n. 2, p. 97-166, 1 dez. 2001.

DEY, A. K.; ABOWD, G. D. Towards a better understanding of context and context-awareness. **HUC '99 Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing**, p. 304-307, 1999.

DMTF, D. M. T. F.-. **Common Information Model**. Disponível em: <http://dmtf.org/standards/cim>. Acesso em: 10 maio. 2013.

EUZENAT, J.; SCHVAIKO, P. **Ontology Matching**. [S.l.] Springer-Verlag Berlin Heidelberg, 2007.

FLYNN, M. J. Some Computer Organizations and Their Effectiveness. **IEEE Transactions on Computers**, v. C-21, n. 9, p. 948-960, set. 1972.

GENESERETH, M.; NILSSON, N. **Logical foundations of artificial intelligence**. [S.l: s.n.].

GLOBUS. **Globus**. Disponível em: <http://www.globus.org/>. Acesso em: 7 fev. 2013.

GRUBER, T. A translation approach to portable ontology specifications. **Knowledge acquisition**, v. 5, n. April, p. 199-220, 1993.

GUARINO, N. Formal ontology, conceptual analysis and knowledge representation. **International journal of human computer studies**, 1995.

GUARINO, N.; OBERLE, D.; STAAB, S. What is an Ontology. **Handbook on Ontologies**, p. 1-17, 2009.

HENRICKSEN, K. Towards a hybrid approach to context modelling, reasoning and interoperation. **Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning And Management, in conjunction with UbiComp. 2004**, 2004.

HENRICKSEN, K.; INDULSKA, J. A software engineering framework for context-aware pervasive computing. **Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the**, p. 77-86, 2004.

HITZLER, P. et al. **OWL 2 Web Ontology Language Primer**. Disponível em: <http://www.w3.org/TR/owl2-primer/>. Acesso em: 11 fev. 2012.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; VAN HARMELEN, F. From SHIQ and RDF to OWL: the making of a Web Ontology Language. **Web Semantics: Science, Services and Agents on the World Wide Web**, v. 1, n. 1, p. 7-26, dez. 2003.

IBM. **IBM Platform LSF**.

KLYNE, G.; CARROLL, J. J. **Resource Description Framework (RDF): Concepts and Abstract Syntax**. Disponível em: <http://www.w3.org/TR/rdf-concepts>. Acesso em: 11 fev. 2012.

KOOLKAMPUS. **Software Concepts: Middleware**. Disponível em: <http://koolkampus.com/engineering-notes-1/information-technology/software-concepts-middleware/>. Acesso em: 7 fev. 2013.

MANOLA, F.; MILLER, E. **RDF Primer**. Disponível em: <http://www.w3.org/TR/rdf-primer/>. Acesso em: 11 fev. 2012.

MPIFORUM. **Message Passing Interface Forum**. Disponível em: <http://www.mpi-forum.org/>. Acesso em: 3 fev. 2013.

OMG, O. M. G.-. **UML**. Disponível em: <http://www.uml.org/>. Acesso em: 10 maio. 2013.

OPENCLUSTERGROUP. **OSCAR**. Disponível em: <http://svn.oscar.openclustergroup.org/trac/oscar>. Acesso em: 7 fev. 2013.

ORACLE. **Oracle Grid Engine**. Disponível em:

<http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html>. Acesso em: 6 fev. 2013a.

ORACLE. **Oracle VM Virtualbox**. Disponível em: <https://www.virtualbox.org/>. Acesso em: 9 maio. 2013b.

OWLAPI. **OWL API**. Disponível em: <http://owlapi.sourceforge.net/>. Acesso em: 10 maio. 2013.

OXFORD, U. OF. **HermiT Reasoner**. Disponível em: <http://www.hermit-reasoner.com/>. Acesso em: 10 maio. 2013.

PASCOE, J. Adding Generic Contextual Capabilities to Wearable Computers. **Wearable Computers, 1998. Digest of Papers. Second International Symposium on**, p. 92-99, 1998.

PBSWORKS. **PBS Professional**. Disponível em: <http://www.pbsworks.com/Product.aspx?id=1>. Acesso em: 6 fev. 2013.

PVM. **PVM: Parallel Virtual Machine**. Disponível em: <http://www.csm.ornl.gov/pvm/>. Acesso em: 3 fev. 2013.

SAMULOWITZ, M.; MICHAHELLES, F.; LINNHOFF-POPIEN, C. Capeus: An architecture for context-aware selection and execution of services. **New Developments in Distributed Applications and Interoperable Systems**, v. 70, p. 1-14, 2002.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-aware computing applications. **Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on**, p. 85-90, 1994.

SCHILIT, B. N.; THEIMER, M. M. Disseminating Active Map Information to Mobile Hosts. **Network, IEEE**, v. 8, n. 5, p. 22-32, 1994.

SILVA, A. P. .; DANTAS, M. A. R. A Selector of Grid Resources based on the Semantic Integration of Multiple Ontologies. **19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07)**, p. 143-150, out. 2007.

SOMASUNDARAM, T. S. et al. Semantic-based Grid Resource Discovery and its Integration with the Grid Service Broker. **2006 International Conference on Advanced Computing and Communications**, p. 84-89, 2006.

STANFORD. **The Protégé Ontology Editor**. Disponível em: <http://protege.stanford.edu/>. Acesso em: 10 maio. 2013.

TANENBAUM, A. S.; STEEN, M. V. **Distributed Systems: Principles and Paradigms**. Second Edi ed. Amsterdam: Pearson Prentice Hall, 2006. p. 705

UW-MADISON. **HTCondor - High Throughput Computing**. Disponível em: <http://research.cs.wisc.edu/htcondor/index.html>. Acesso em: 6 fev. 2013.

VIRGINIA, U. OF. **The Legion Project**. Disponível em: <http://legion.virginia.edu/>. Acesso em: 7 fev. 2013.

W3C. **SOAP Version 1.2**. Disponível em: <http://www.w3.org/TR/soap/>. Acesso em: 3 fev. 2013.

W3C. **World Wide Web Consortium**. Disponível em: <http://www.w3c.org>. Acesso em: 19 maio. 2013.

W3C OWL WORKING GROUP. **OWL 2 Web Ontology Language Document Overview**.

Disponível em: <http://www.w3.org/TR/owl2-overview/>.

ZHANG, D.; GU, T.; WANG, X. Enabling context-aware smart home with semantic web technologies. **International Journal of Human-friendly Welfare Robotic Systems**, v. 6, n. 1, p. 12-20, 2005.

ANEXO A – ONTOLOGIA OWL

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY distributed-system "http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#" >
]>
<rdf:RDF xmlns="http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#"
     xml:base="http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:distributed-system="http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl">
        <rdfs:comment xml:lang="en">Ontology representing distributed environments.</rdfs:comment>
    </owl:Ontology>
    <!--
    ////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasArchitecture -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasArchitecture">
        <rdfs:comment xml:lang="en">Relation between processor and its architecture.</rdfs:comment>
        <rdfs:range rdf:resource="&distributed-system;Architecture"/>
        <rdfs:domain rdf:resource="&distributed-system;Processor"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasFileSystem -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasFileSystem">
        <rdfs:comment xml:lang="en">Relation between a node and its file system.</rdfs:comment>
        <rdfs:range rdf:resource="&distributed-system;FileSystem"/>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasMemory -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasMemory">
        <rdfs:comment xml:lang="en">Relation between a node and its memory device.</rdfs:comment>
        <rdfs:range rdf:resource="&distributed-system;Memory"/>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasNetworkAdapter -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasNetworkAdapter">
        <rdfs:comment xml:lang="en">Relation between a node and its network adapter.</rdfs:comment>
        <rdfs:range rdf:resource="&distributed-system;NetworkAdapter"/>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasNetworkPortStatistics -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasNetworkPortStatistics">
                <rdfs:comment  xml:lang="en">Relation  between  a  network  adapter  and  its  statistical
```

```
data.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;NetworkAdapter"/>
        <rdfs:range rdf:resource="&distributed-system;NetworkPortStatistics"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasNode -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasNode">
        <rdfs:comment xml:lang="en">Relation between a pool and its node.</rdfs:comment>
        <rdfs:range rdf:resource="&distributed-system;Node"/>
        <rdfs:domain rdf:resource="&distributed-system;Pool"/>
        <owl:inverseOf rdf:resource="&distributed-system;isNodeOf"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasNodeStatus -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasNodeStatus">
        <rdfs:comment xml:lang="en">Relation between a node and its status.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
        <rdfs:range rdf:resource="&distributed-system;NodeStatus"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasOperatingSystem -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasOperatingSystem">
        <rdfs:comment xml:lang="en">Relation between a node and its operating system.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
        <rdfs:range rdf:resource="&distributed-system;OperatingSystem"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasProcessor -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasProcessor">
        <rdfs:comment xml:lang="en">Relation between a node and its processor.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
        <rdfs:range rdf:resource="&distributed-system;Processor"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasServiceOffering -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasServiceOffering">
        <rdfs:comment xml:lang="en">Relation between a system and the service it offers.</rdfs:comment>
        <rdfs:range rdf:resource="&distributed-system;Service"/>
        <rdfs:domain rdf:resource="&distributed-system;System"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasStorageDevice -->
    <owl:ObjectProperty rdf:about="&distributed-system;hasStorageDevice">
        <rdfs:comment xml:lang="en">Relation between node and its storage device.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
        <rdfs:range rdf:resource="&distributed-system;StorageDevice"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#isNodeOf -->
    <owl:ObjectProperty rdf:about="&distributed-system;isNodeOf">
                <rdfs:comment  xml:lang="en">Relation  between  a  node  and  the  pool,  in  which  it  is
included.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Node"/>
        <rdfs:range rdf:resource="&distributed-system;Pool"/>
    </owl:ObjectProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#isResourceManagedBy -->
    <owl:ObjectProperty rdf:about="&distributed-system;isResourceManagedBy">
        <rdfs:comment xml:lang="en">Relation between a pool and the resource management system that manages
its resources.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Pool"/>
        <rdfs:range rdf:resource="&distributed-system;ResourceManagementSystem"/>
    </owl:ObjectProperty>
```

```
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#isServiceOfferredBy -->
    <owl:ObjectProperty rdf:about="&distributed-system;isServiceOfferredBy">
                <rdfs:comment xml:lang="en">Relation between a service and the system that offers
it.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Service"/>
        <rdfs:range rdf:resource="&distributed-system;System"/>
        <owl:inverseOf rdf:resource="&distributed-system;hasServiceOffering"/>
    </owl:ObjectProperty>
    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasAvailableSpace -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasAvailableSpace">
        <rdfs:comment xml:lang="en">Relation between a file system and its available space.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;FileSystem"/>
        <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasAverageRTT -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasAverageRTT">
         <rdfs:comment xml:lang="en">Relation between a network statistical data and its average round-trip
time.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;NetworkPortStatistics"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasDataWidth -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasDataWidth">
        <rdfs:comment xml:lang="en">Relation between a processor and its data width.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Processor"/>
        <rdfs:range>
            <rdfs:Datatype>
                <owl:oneOf>
                    <rdf:Description>
                        <rdf:type rdf:resource="&rdf;List"/>
                        <rdf:first>32</rdf:first>
                        <rdf:rest>
                            <rdf:Description>
                                <rdf:type rdf:resource="&rdf;List"/>
                                <rdf:first>64</rdf:first>
                                <rdf:rest rdf:resource="&rdf;nil"/>
                            </rdf:Description>
                        </rdf:rest>
                    </rdf:Description>
                </owl:oneOf>
            </rdfs:Datatype>
        </rdfs:range>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasFamily -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasFamily">
        <rdfs:comment xml:lang="en">Relation between a processor and its family.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Processor"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
```

```
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasFreePhysicalMemory -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasFreePhysicalMemory">
            <rdfs:comment xml:lang="en">Relation between a operating system and its free physical
memory.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;OperatingSystem"/>
        <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasFreeVirtualMemory -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasFreeVirtualMemory">
            <rdfs:comment xml:lang="en">Relation between a operating system and its free virtual
memory.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;OperatingSystem"/>
        <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasLoadPercentage -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasLoadPercentage">
        <rdfs:comment xml:lang="en">Relation between a processor and its load percentage.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Processor"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasNumberOfEnabledCores -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasNumberOfEnabledCores">
            <rdfs:comment xml:lang="en">Relation between a processor and its number of enabled
cores.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Processor"/>
        <rdfs:range rdf:resource="&xsd;nonNegativeInteger"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasPacketLossPercentage -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasPacketLossPercentage">
            <rdfs:comment xml:lang="en">Relation between a network statistical data and its packet loss
percentage.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;NetworkPortStatistics"/>
        <rdfs:range rdf:resource="&xsd;double"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasPermanentAddress -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasPermanentAddress">
           <rdfs:comment xml:lang="en">Relation between a network adapter and its permanent address (MAC
Address)</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;NetworkAdapter"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </owl:DatatypeProperty>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#hasServiceAccessURI -->
    <owl:DatatypeProperty rdf:about="&distributed-system;hasServiceAccessURI">
        <rdfs:comment xml:lang="en">Relation between a service and its URI.</rdfs:comment>
        <rdfs:domain rdf:resource="&distributed-system;Service"/>
        <rdfs:range rdf:resource="&xsd;anyURI"/>
    </owl:DatatypeProperty>
    <!--
    ///////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////
     -->
```

```
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Android -->
<owl:Class rdf:about="&distributed-system;Android">
    <rdfs:subClassOf rdf:resource="&distributed-system;OperatingSystem"/>
    <rdfs:comment xml:lang="en">Class of Android operating systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Architecture -->
<owl:Class rdf:about="&distributed-system;Architecture">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment xml:lang="en">Class of processor architectures</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#CDROMDrive -->
<owl:Class rdf:about="&distributed-system;CDROMDrive">
    <rdfs:subClassOf rdf:resource="&distributed-system;StorageDevice"/>
    <rdfs:comment xml:lang="en">Class of CD ROM drives.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#ComputerSystem -->
<owl:Class rdf:about="&distributed-system;ComputerSystem">
    <rdfs:subClassOf rdf:resource="&distributed-system;System"/>
    <rdfs:comment xml:lang="en">Class of physical or virtual computers.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#DRAM -->
<owl:Class rdf:about="&distributed-system;DRAM">
    <rdfs:subClassOf rdf:resource="&distributed-system;Memory"/>
    <rdfs:comment xml:lang="en">Class of dynamic random-access memory devices.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#DVDDrive -->
<owl:Class rdf:about="&distributed-system;DVDDrive">
    <rdfs:subClassOf rdf:resource="&distributed-system;StorageDevice"/>
    <rdfs:comment xml:lang="en">Class of DVD drives.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Device -->
<owl:Class rdf:about="&distributed-system;Device">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment xml:lang="en">Class of computer hardware devices</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#DiskDrive -->
<owl:Class rdf:about="&distributed-system;DiskDrive">
    <rdfs:subClassOf rdf:resource="&distributed-system;StorageDevice"/>
    <rdfs:comment xml:lang="en">Class of hard disk drives.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#EXT -->
<owl:Class rdf:about="&distributed-system;EXT">
    <rdfs:subClassOf rdf:resource="&distributed-system;FileSystem"/>
    <rdfs:comment xml:lang="en">Class of EXT file systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#EthernetAdapter -->
<owl:Class rdf:about="&distributed-system;EthernetAdapter">
    <rdfs:subClassOf rdf:resource="&distributed-system;NetworkAdapter"/>
    <rdfs:comment xml:lang="en">Class of Ethernet network adapters.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#FAT -->
<owl:Class rdf:about="&distributed-system;FAT">
    <rdfs:subClassOf rdf:resource="&distributed-system;FileSystem"/>
    <rdfs:comment xml:lang="en">Class of FAT file systems.</rdfs:comment>
</owl:Class>
```

```
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#FileSystem -->
<owl:Class rdf:about="&distributed-system;FileSystem">
    <rdfs:subClassOf rdf:resource="&distributed-system;System"/>
    <rdfs:comment xml:lang="en">Class of file systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#HSDPAAdapter -->
<owl:Class rdf:about="&distributed-system;HSDPAAdapter">
    <rdfs:subClassOf rdf:resource="&distributed-system;NetworkAdapter"/>
    <rdfs:comment xml:lang="en">Class of HSDPA network adapters.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#HTCondor -->
<owl:Class rdf:about="&distributed-system;HTCondor">
    <rdfs:subClassOf rdf:resource="&distributed-system;ResourceManagementSystem"/>
    <rdfs:comment xml:lang="en">Class of HTCondor resource management systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#InfiniBandAdapter -->
<owl:Class rdf:about="&distributed-system;InfiniBandAdapter">
    <rdfs:subClassOf rdf:resource="&distributed-system;NetworkAdapter"/>
    <rdfs:comment xml:lang="en">Class of InfiniBand network adapters.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Linux -->
<owl:Class rdf:about="&distributed-system;Linux">
    <rdfs:subClassOf rdf:resource="&distributed-system;UNIX"/>
    <rdfs:comment xml:lang="en">Class of Linux operating systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#MacOS -->
<owl:Class rdf:about="&distributed-system;MacOS">
    <rdfs:subClassOf rdf:resource="&distributed-system;OperatingSystem"/>
    <rdfs:comment xml:lang="en">Class of MacOS operating systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Memory -->
<owl:Class rdf:about="&distributed-system;Memory">
    <rdfs:subClassOf rdf:resource="&distributed-system;Device"/>
    <rdfs:comment xml:lang="en">Class of computer memory devices</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#MobileDevice -->
<owl:Class rdf:about="&distributed-system;MobileDevice">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="&distributed-system;Node"/>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&distributed-system;hasNetworkAdapter"/>
                    <owl:someValuesFrom>
                        <owl:Class>
                            <owl:unionOf rdf:parseType="Collection">
                                <rdf:Description rdf:about="&distributed-system;HSDPAAdapter"/>
                                <rdf:Description rdf:about="&distributed-system;WiFiAdapter"/>
                            </owl:unionOf>
                        </owl:Class>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&distributed-system;hasOperatingSystem"/>
                    <owl:someValuesFrom>
```

```
                        <owl:Class>
                            <owl:unionOf rdf:parseType="Collection">
                                <rdf:Description rdf:about="&distributed-system;Android"/>
                                <rdf:Description rdf:about="&distributed-system;iOS"/>
                            </owl:unionOf>
                        </owl:Class>
                    </owl:someValuesFrom>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">Class of mobile nodes of execution</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#NFS -->
<owl:Class rdf:about="&distributed-system;NFS">
    <rdfs:subClassOf rdf:resource="&distributed-system;FileSystem"/>
    <rdfs:comment xml:lang="en">Class of NFS file systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#NTFS -->
<owl:Class rdf:about="&distributed-system;NTFS">
    <rdfs:subClassOf rdf:resource="&distributed-system;FileSystem"/>
    <rdfs:comment xml:lang="en">Class of NTFS file systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#NetworkAdapter -->
<owl:Class rdf:about="&distributed-system;NetworkAdapter">
    <rdfs:subClassOf rdf:resource="&distributed-system;Device"/>
    <rdfs:comment xml:lang="en">Class of computer network adapters</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#NetworkPortStatistics -->
<owl:Class rdf:about="&distributed-system;NetworkPortStatistics">
    <rdfs:subClassOf rdf:resource="&distributed-system;StatisticalData"/>
    <rdfs:comment xml:lang="en">Class of statistical data for computer networks.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Node -->
<owl:Class rdf:about="&distributed-system;Node">
    <rdfs:subClassOf rdf:resource="&distributed-system;ComputerSystem"/>
    <rdfs:comment xml:lang="en">Class of executing nodes.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#NodeStatus -->
<owl:Class rdf:about="&distributed-system;NodeStatus">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment xml:lang="en">Class of status of node.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#OperatingSystem -->
<owl:Class rdf:about="&distributed-system;OperatingSystem">
    <rdfs:subClassOf rdf:resource="&distributed-system;System"/>
    <rdfs:comment xml:lang="en">Class of operating systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#PersonalComputer -->
<owl:Class rdf:about="&distributed-system;PersonalComputer">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="&distributed-system;Node"/>
                <owl:Restriction>
```

```
                    <owl:onProperty rdf:resource="&distributed-system;hasNetworkAdapter"/>
                    <owl:someValuesFrom rdf:resource="&distributed-system;NetworkAdapter"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&distributed-system;hasOperatingSystem"/>
                    <owl:someValuesFrom>
                        <owl:Class>
                            <owl:unionOf rdf:parseType="Collection">
                                <rdf:Description rdf:about="&distributed-system;MacOS"/>
                                <rdf:Description rdf:about="&distributed-system;UNIX"/>
                                <rdf:Description rdf:about="&distributed-system;Windows"/>
                            </owl:unionOf>
                        </owl:Class>
                    </owl:someValuesFrom>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:comment xml:lang="en">Class of personal computer nodes of execution.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Pool -->
<owl:Class rdf:about="&distributed-system;Pool">
    <rdfs:subClassOf rdf:resource="&distributed-system;ComputerSystem"/>
    <rdfs:comment xml:lang="en">Class of agglomeration of nodes.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Processor -->
<owl:Class rdf:about="&distributed-system;Processor">
    <rdfs:subClassOf rdf:resource="&distributed-system;Device"/>
    <rdfs:comment xml:lang="en">Class of computer processors.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#ResourceManagementSystem -->
<owl:Class rdf:about="&distributed-system;ResourceManagementSystem">
    <rdfs:subClassOf rdf:resource="&distributed-system;System"/>
    <rdfs:comment xml:lang="en">Class of Resource Management Systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#SRAM -->
<owl:Class rdf:about="&distributed-system;SRAM">
    <rdfs:subClassOf rdf:resource="&distributed-system;Memory"/>
    <rdfs:comment xml:lang="en">Class of static random-access memory devices.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Service -->
<owl:Class rdf:about="&distributed-system;Service">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment xml:lang="en">Class of computer services.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#SolidStateDrive -->
<owl:Class rdf:about="&distributed-system;SolidStateDrive">
    <rdfs:subClassOf rdf:resource="&distributed-system;StorageDevice"/>
    <rdfs:comment xml:lang="en">Class of solid state drives.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#StatisticalData -->
<owl:Class rdf:about="&distributed-system;StatisticalData">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment xml:lang="en">Class of statistical data.</rdfs:comment>
</owl:Class>
```

```
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#StorageDevice -->
<owl:Class rdf:about="&distributed-system;StorageDevice">
    <rdfs:subClassOf rdf:resource="&distributed-system;Device"/>
    <rdfs:comment xml:lang="en">Class of computer storage devices.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#System -->
<owl:Class rdf:about="&distributed-system;System">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:comment xml:lang="en">Class of systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#UNIX -->
<owl:Class rdf:about="&distributed-system;UNIX">
    <rdfs:subClassOf rdf:resource="&distributed-system;OperatingSystem"/>
    <rdfs:comment xml:lang="en">Class of UNIX operating systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#USBFlashDrive -->
<owl:Class rdf:about="&distributed-system;USBFlashDrive">
    <rdfs:subClassOf rdf:resource="&distributed-system;StorageDevice"/>
    <rdfs:comment xml:lang="en">Class of USB flash drives.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#WiFiAdapter -->
<owl:Class rdf:about="&distributed-system;WiFiAdapter">
    <rdfs:subClassOf rdf:resource="&distributed-system;NetworkAdapter"/>
    <rdfs:comment xml:lang="en">Class of WiFi network adapters.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Windows -->
<owl:Class rdf:about="&distributed-system;Windows">
    <rdfs:subClassOf rdf:resource="&distributed-system;OperatingSystem"/>
    <rdfs:comment xml:lang="en">Class of Windows operating systems.</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#X86 -->
<owl:Class rdf:about="&distributed-system;X86">
    <rdfs:subClassOf rdf:resource="&distributed-system;Architecture"/>
    <rdfs:comment xml:lang="en">Class of X86 processor architectures</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#X86_64 -->
<owl:Class rdf:about="&distributed-system;X86_64">
    <rdfs:subClassOf rdf:resource="&distributed-system;Architecture"/>
    <rdfs:comment xml:lang="en">Class of X86 processor architectures</rdfs:comment>
</owl:Class>
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#iOS -->
<owl:Class rdf:about="&distributed-system;iOS">
    <rdfs:subClassOf rdf:resource="&distributed-system;OperatingSystem"/>
    <rdfs:comment xml:lang="en">Class of iOS operating systems.</rdfs:comment>
</owl:Class>
<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////
 -->
<!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#AMD64 -->
<owl:NamedIndividual rdf:about="&distributed-system;AMD64">
    <rdf:type rdf:resource="&distributed-system;X86_64"/>
```

```
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#ARM -->
    <owl:NamedIndividual rdf:about="&distributed-system;ARM">
        <rdf:type rdf:resource="&distributed-system;Architecture"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Allocated -->
    <owl:NamedIndividual rdf:about="&distributed-system;Allocated">
        <rdf:type rdf:resource="&distributed-system;NodeStatus"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Available -->
    <owl:NamedIndividual rdf:about="&distributed-system;Available">
        <rdf:type rdf:resource="&distributed-system;NodeStatus"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#GenericX86 -->
    <owl:NamedIndividual rdf:about="&distributed-system;GenericX86">
        <rdf:type rdf:resource="&distributed-system;X86"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#GenericX86_64 -->
    <owl:NamedIndividual rdf:about="&distributed-system;GenericX86_64">
        <rdf:type rdf:resource="&distributed-system;X86_64"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#IA-32 -->
    <owl:NamedIndividual rdf:about="&distributed-system;IA-32">
        <rdf:type rdf:resource="&distributed-system;X86"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#IA-64 -->
    <owl:NamedIndividual rdf:about="&distributed-system;IA-64">
        <rdf:type rdf:resource="&distributed-system;X86_64"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Intel64 -->
    <owl:NamedIndividual rdf:about="&distributed-system;Intel64">
        <rdf:type rdf:resource="&distributed-system;X86_64"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Matched -->
    <owl:NamedIndividual rdf:about="&distributed-system;Matched">
        <rdf:type rdf:resource="&distributed-system;NodeStatus"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#PowerPC -->
    <owl:NamedIndividual rdf:about="&distributed-system;PowerPC">
        <rdf:type rdf:resource="&distributed-system;Architecture"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#SPARC -->
    <owl:NamedIndividual rdf:about="&distributed-system;SPARC">
        <rdf:type rdf:resource="&distributed-system;Architecture"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Unavailable -->
    <owl:NamedIndividual rdf:about="&distributed-system;Unavailable">
        <rdf:type rdf:resource="&distributed-system;NodeStatus"/>
    </owl:NamedIndividual>
    <!-- http://lapesd.inf.ufsc.br/ontologies/distributed-system.owl#Unknown -->
    <owl:NamedIndividual rdf:about="&distributed-system;Unknown">
        <rdf:type rdf:resource="&distributed-system;Architecture"/>
        <rdf:type rdf:resource="&distributed-system;NodeStatus"/>
    </owl:NamedIndividual>
</rdf:RDF>
```