Carlos Fran Ferreira Dantas

Escalonamento de Tarefas em Grid no Cenário de TV Digital Interativa

Mossoró-RN

2011

Carlos Fran Ferreira Dantas

Escalonamento de Tarefas em Grid no Cenário de TV Digital Interativa

Monografia apresentada à Universidade do Estado do Rio Grande do Norte como um dos pré-requisitos para obtenção do grau de bacharel em Ciência da Computação.

Orientador:

Prof. Sebastião Emídio Alves Filho

Mossoró-RN

2011

Trabalho de conclusão de curso sob o título *"Escalonamento de Tarefas em Grid no Cenário de TV Digital Interativa"*, defendido por Carlos Fran Ferreira Dantas e aprovada 20 de janeiro de 2011, em Mossoró, Rio Grande do Norte, pela banca examinadora constituída por:

Prof. Sebastião Emídio Alves Filho
Orientador

Prof. Dr. Rommel Wladimir de Lima
UERN

Mizael Clistion Souza Elias
UERN

Este trabalho é dedicado à minha mãe
cujo amor e dedicação me inspiram
a melhorar dia após dia.

Agradecimentos

Agradeço a Deus por tudo, por Ele.

Aos meus pais, Francisco Benedito Belo e Maria José Ferreira Dantas, pelo apoio incondicional, pelo amor e orientação.

Ao casal José Ilson e Ineuda Xavier de Oliveira, e familiares, pela imensa contribuição para realização de um sonho.

Aos professores pelas contribuições profissionais e pessoais, em especial aos professores Sebastião Emídio, Marcelino Pereira e a professora Cicília Maia.

Aos meus irmãos, Carliano, Carliesio e Carliolane, pelos incentivos.

Aos meus amigos, Mizael Clistion, Marlos Lima e Natalyany Nunes, pela motivação e auxílio em alguns momentos.

Aos colegas de turma e todos aqueles que contribuíram de alguma forma para a conclusão dessa etapa.

Muito obrigado à todos.

"*Aquilo que não é necessariamente uma escolha,*
não pode ser considerado como mérito
ou como fracasso."
(Milan Kundera)

Resumo

Este trabalho apresenta o algoritmo de escalonamento de tarefas para grid orientado a serviços, com interoperação de dispositivos móveis como provedores de recursos, em redes locais domésticas (HAN – *Home Area Network*) no contexto de TV digital interativa. O algoritmo denominado PUTS (Power-aware User-preference Task-size based Scheduling) realiza balanceamento dos níveis de energia das baterias dos dispositivos móveis que constituem o grid, e permite configurações de preferência do usuário que influênciam nas atribuições das tarefas.

Palavras-chave: Escalonamento de tarefas, Computação em grid, TV Digital

Abstract

This document presents the task scheduling algorithm for service-oriented grid, with interoperation of mobile devices such as resource providers, in home area networks in the context of digital TV. The algorithm, called PUTS (Power-aware User-preference Task-size based Scheduling), performs balancing of the energy levels of the batteries of mobile devices that constitute the grid, and allows the user-preference settings that influence assignment of tasks.

Keywords: Task scheduling, Grid computing, Digital TV

Sumário

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

1 INTRODUÇÃO **14**

1.1 Objetivos 15

1.1.1 Objetivo Geral 15

1.1.2 Objetivos Específicos 15

1.2 Justificativa 15

1.3 Estrutura do Trabalho 16

2 TV DIGITAL INTERATIVA **17**

2.1 Componentes Fundamentais 18

2.2 Arquitetura e Padrões de Sistema de TV Digital 20

2.2.1 Advanced Television System Committee - ATSC 21

2.2.2 Digital Video Broadcasting - DVB 22

2.2.3 Integrated Services Digital Broadcasting - ISDB 23

2.2.4 Sistema Brasileiro de TV Digital - SBTVD 23

2.3 Aplicações em TVDI 25

2.4 Middleware Gringa 26

2.4.1 Arquitetura e Tecnologias do Gringa 27

3 ESCALONAMENTO EM GRIDS COMPUTACIONAIS 29

3.1 Taxonomia dos algoritmos de escalonamento 30

3.1.1 Local vs Global 31

3.1.2 Estático vs Dinâmico 31

3.1.3 Distribuídos vs Não-distribuído 32

3.1.4 Cooperativo vs Não-cooperativo 32

3.1.5 Ótimo vs Sub-ótimo 32

3.1.6 Aproximado vs Heurístico 32

3.2 Aspectos restritos à grids computacionais 33

3.3 Algoritmos de escalonamento 34

3.3.1 *WorkQueue Replication* 35

3.3.2 *XSufferage* 35

3.3.3 *Dynamic Fastest Processor to Lagest Task First* (Dyn-FPLTF) . . . 36

3.3.4 Max-Min e Min-Min 37

3.4 Interoperação entre Dispositivos Móveis e Grids Computacionais 37

3.4.1 Abordagens de Utilização dos Dispositivos Móveis em Grids 38

3.4.2 Aspectos Relevantes para Escalonamento em Grids com Interoperação de Dispositivos Móveis 39

4 ESCALONAMENTO NO MIDDLEWARE GRINGA 42

4.1 Estimando custo de tarefas 42

4.2 O Algoritmo PUTS 43

4.2.1 Consumo de Energia 44

4.2.2 Preferências do Usuário 46

4.2.3 Agrupamento de dispositivos 48

5 ESTUDOS DE CASO 51

5.1 Estudo de Caso 1 – Cenário com poucos recursos 51

5.2 Estudo de Caso 2 – Cenário com heterogeneidade de recursos 53

Considerações Finais **56**

Referências **57**

Lista de Figuras

1 Combinações de formatos de vídeo 19

2 Processo de Transmissão de TV Digital Interativa 19

3 Arquitetura em Camadas de Sistemas de TV Digital 21

4 Padrões de TV Digital Interativa 22

5 Adoção do ISBT-Tb pelos países da América do Sul e Central 24

6 Cenário de Automação Residêncial Utilizando Ginga@Home 26

7 Cenário exemplo de formação do grid em rede local doméstica 27

8 Taxonomia dos Algoritmos de Escalonamento 31

9 Modelo de Escalonamento Hierárquico baseado em proxy 40

10 Estudo de caso 1 - Formação do grid 52

11 Gráfico – Degradação do desempenho - Estudo de caso 1 53

12 Estudo de caso 2 - Formação do grid 54

13 Gráfico – Degradação do desempenho - Estudo de caso 2 55

Lista de Tabelas

1 Meios de Difusão do Sinal Digital de TV 20

2 Balanceamento de carga – Estudo de caso 1 52

3 Balanceamento de carga – Estudo de caso 2 54

Lista de abreviaturas e siglas

CT	Completion Time
CTE	Completion Time Energy
Dyn-FPLTF	Dynamic Fastest Processor to Largest Task First
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
JME	Java Micro Edition
MIDP	Mobile Information Device Profile
PDA	Personal Digital Assistant
PUTS	Power-aware User-preference Task-size based Scheduling
QoS	Quality of Service
RPC	Remote Procedure Call
SMS	Short Message Service
TBA	Time Become Avaliable
TCP	Transmission Control Protocol
UPN	User Preference Node
UPSN	User Preference for Service at Node
UPW	User-Preference Weight
WQR	WorkQueue Replication
XML	eXtensible Markup Language

1 INTRODUÇÃO

Segundo dados da Pesquisa Nacional por Amostra de Domicílios (PNAD, 2009) realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE), a TV está presente em 95,7% dos domicílios brasileiros, sendo o meio de comunicação com maior penetração nas residências no país. Entretanto, pesquisas indicam que a cada ano, várias Tecnologias da Informação e Comunicação (TIC's) como computador de mesa, *notebook*, *netbook* e *smartphones*, estão cada vez mais presentes.

Com a adoção da transmissão do sinal digital de TV pelo Brasil, melhorias na qualidade de imagem e som, a execução de aplicações e a possibilidade de interação do telespectador, tornam-se fatos. No entanto, é de conhecimento espontâneo que as aplicações multimídias requerem bem mais recursos de hardware que as demais.

No ambiente de TV Digital Interativa (TVDI), os *settop boxes* (STB's) responsáveis pela execução das aplicações, possuem restrições de hardware. Desse modo, o middleware Gringa (FILHO, 2010), cujo objetivo é dar suporte ao processamento em forma de grid, utilizando TIC's presentes nos domicílios para acelerar o processamento das aplicações, aumenta o leque de possibilidades para a construção de programas interativos, uma vez que é possível romper as barreiras impostas pelas limitações dos STB's.

Grids computacionais requerem a coordenação de recursos que não estão sujeitos ao controle local e apresentam desafios quanto ao escalonamento. Nesse sentido, análises e comparativos tem sido realizados com o objetivo de avaliar e caracterizar os algoritmos de escalonamento nos cenários de aplicação do paradigma *Grid Computing*.

A interoperação entre grids e dispositivos móveis como provedores de recursos, apresentam mais desafios ao escalonamento. Estes desafios estão associados ao consumo de energia (baterias dos dispositivos) e as características da comunicação sem fio.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo desse trabalho é a aplicação de um algoritmo de escalonamento que considere a heterogeneidade dos dispositivos, a comunicação e o consumo de energia dos mesmos, no ambiente de aplicação do *middleware* Gringa.

1.1.2 Objetivos Específicos

- Descrever o funcionamento do Gringa – *middleware* para execução de aplicações em forma de grid em cenários de TVDI nos domicílios;
- Implementação de uma heurística para o escalonamento de tarefas que considere o consumo de energia e a transferência de dados dos dispositivos.
- Desenvolver um mecanismo que permita ao usuário influênciar indiretamente no escalonamente de acordo com sua preferência.

1.2 Justificativa

A complexidade do problema geral de escalonamento é conhecidamente NP-Completo. Em grids computacionais, o problema de escalonamento apresenta alguns desafios a mais, como a heterogeneidade e o compartilhamento de recursos, o dinamismo do poder computacional oferecido, a transferência de dados e a integração de resultados (SILVA, 2009). Tais desafios remetem a necessidade de políticas de escalonamento adequadas a cada ambiente no qual o paradigma de computação em grid é aplicado.

Em grids que envolvem dispositivos móveis, é relevante considerar no escalonamento, aspectos da comunicação com os nós e o fato desses dispositivos utilizarem baterias.

Frequentemente, os algoritmos de escalonamento em grids não consideram características do ambiente, como por exemplo a heterogeneidade e a comunicação. Como relatam (TSENG; CHIN; WANG, 2009), o tempo de comunicação é frequentemente ignorado no escalonamento em grids fixos (ditos convencionais). Em redes sem fio (*Wireless Networks*), a transferência de dados tem impacto significativo no desempenho das aplicações e no consumo de energia. Desse modo, trabalhos relacionados à grids com

interoperação de dispositivos móveis (atuando como provedores de recursos) consideram os dois aspectos (comunicação e consumo de energia).

O consenso entre pesquisadores afirma que para um escalonador ser capaz de tomar decisões apropriadas, deve-se utilizar um modelo de escalonamento que retrate de forma mais precisa possível as características do ambiente sobre o qual a aplicação será executada. Portanto, este trabalho apresenta a aplicação de um modelo de escalonamento que considera os diferentes aspectos do ambiente de aplicação do middleware Gringa.

1.3 Estrutura do Trabalho

As seções seguintes tratam da fundamentação necessária para a compreensão e a realização do trabalho.

O capítulo 2, TV Digital Interativa, apresenta o sistema de TV digital interativa e descreve o middleware Gringa.

O capítulo 3, Escalonamento de Tarefas em Grids Computacionais, consiste dos conceitos fundamentais relacionados ao escalonamento em grid. Relata os principais algoritmos e os aspectos relacionados a interoperação entre dispositivos móveis e grids.

O capítulo 4, apresenta o algoritmo PUTS (Power-aware User-preference Task-size based Scheduling) proposto neste documento.

No capítulo 5, Estudos de caso, trata dos cenários e resultados dos experimentos realizados.

Por fim, as considerações finais são descritas.

2 TV DIGITAL INTERATIVA

As primeiras pesquisas em TV digital foram realizadas na decáda de 1980 e consolidaram-se na decáda de 1990 com o lançamento comercial dos primeiros padrões, respectivamente, ATSC (*Advanced Television Systems Committee*) nos EUA e DVB (*Digital Video Broadcasting*) na Europa (SOMBRA, 2009; MARQUES, 2008).

Em 2003, o Japão lançou comercialmente o padrão ISDB (*Integrated Service Digital Broadcasting*) e o Brasil, após cerca de 9 anos de pesquisas, instituiu o Sistema Brasileiro de TV Digital (SBTVD) com o Decreto 4.901 que propos o desenvolvimento de um sistema que culminasse em um modelo com todos os serviços que a nova tecnologia permite (TONIETO, 2006).

TV Digital é um sistema de televisão com transmissão, recepção e processamento digitais (PAES; ANTONIAZZI, 2005). Para (TONIETO, 2006), nada mais é, que a transmissão de sinais de televisão na forma digital.

Como relatam (SOMBRA, 2009; PAES; ANTONIAZZI, 2005), a tecnologia digital proporciona uma série de melhorias e novas possibilidades. Vejamos:

- Melhor qualidade de imagem com maior imunidade à ruídos e distorções;
- Melhor qualidade de som, com o tratamento do áudio, permite reforçar graves e agudos;
- Melhor aproveitamento do espectro radioelétrico, proporcionando maior robustez frente a interferências e viabilizando recepção com menores valores de intensidade de campo, maior quantidade de informação transmitida na mesma faixa de frequência;
- Flexibilidade na manipulação e enriquecimento do tratamento e edição de sinais;
- Capacidade de transporte conjunto de múltiplos programas com vídeo, diversos áudios, textos e dados;
- Oferecimento de serviços interativos

Tais itens representam para os usuários/telespectadores maior qualidade de som e imagem, interatividade, recepção do sinal em aparelhos móveis e a possibilidade de optar entre programas transmitidos por uma emissora, em um único canal.

Quanto a qualidade da definição da imagem, (MENDES, 2007) ressalta que está relacionada ao formato de vídeo utilizado na TV digital. Atualmente, os seguintes formatos são adotados:

1. LDTV (*Low Definition Television*) – é um formato de vídeo com baixa definição, utilizado para transmissão de sinais para dispositivos móveis, como celulares e PDA's. É formado por 240 linhas com 320 pixels cada, e formato de tela 4:3.

2. SDTV (*Standard Definition Television*) – formato de vídeo de resolução padrão, equivalente ao sinal de TV analógica. Possui 480 linhas com 640 pixels cada. O SDTV pode ter formato de tela 4:3 (formato convencional) ou 16:9 (*Widescreen*).

3. EDTV (*Enhanced Definition Television*) – formato intermediário entre SDTV e o HDTV. O EDTV possui o formato de tela 16:9 e 480 linhas com 720 pixels cada.

4. HDTV (*High Definition Television*) – atualmente, o melhor formato de vídeo em um sistema de TV digital terrestre. Pode conter 720 linhas com 1280 pixels cada, ou 1080 linhas com 1920 pixels cada. Ambas as configurações em formato de tela 16:9.

Na figura 1, (MENDES, 2007) apresenta algumas combinações desses formatos utilizadas na faixa de 6 MHz (19.3 Mbps) de transmissão de TV digital terrestre (radiodifusão). Nas combinações apresentadas, existe uma reserva da largura de banda para a transferência de informações relativas à interatividade.

2.1 Componentes Fundamentais

O processo de transmissão em TV digital interativa é frequentemente decomposto em 3 partes básicas, como ilustrado na figura 2.

A primeira parte, Difusor, é a responsável por fornecer o conteúdo a ser transmitido e pelo suporte as interações dos telespectadores representados no bloco "Receptor".

No Receptor, o aparelho receptor recebe o conteúdo e oferece a possibilidade do telespetador reagir ou interagir com o difusor. A recepção pode ser feita por dispositivos

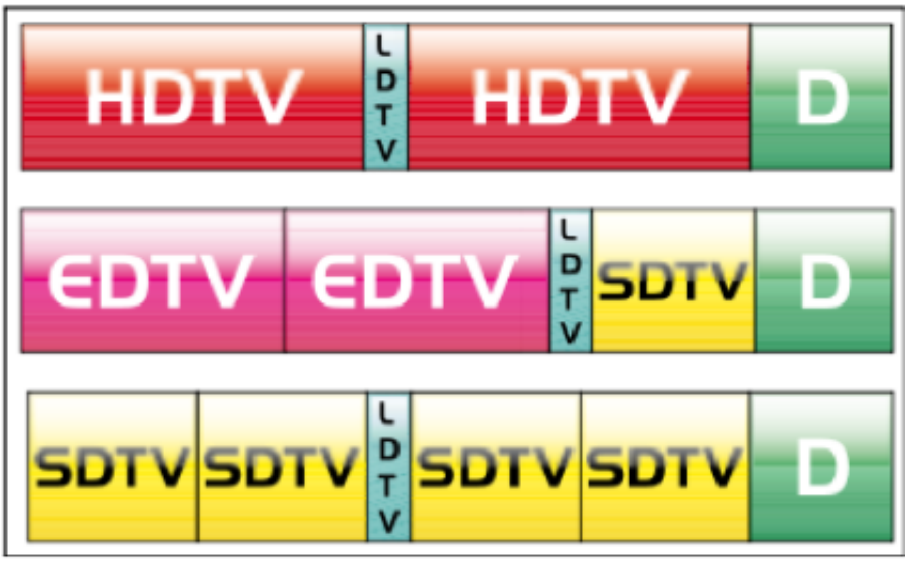

Figura 1: Combinações de formatos de vídeo (MONTEZ; BECKER, 2005)

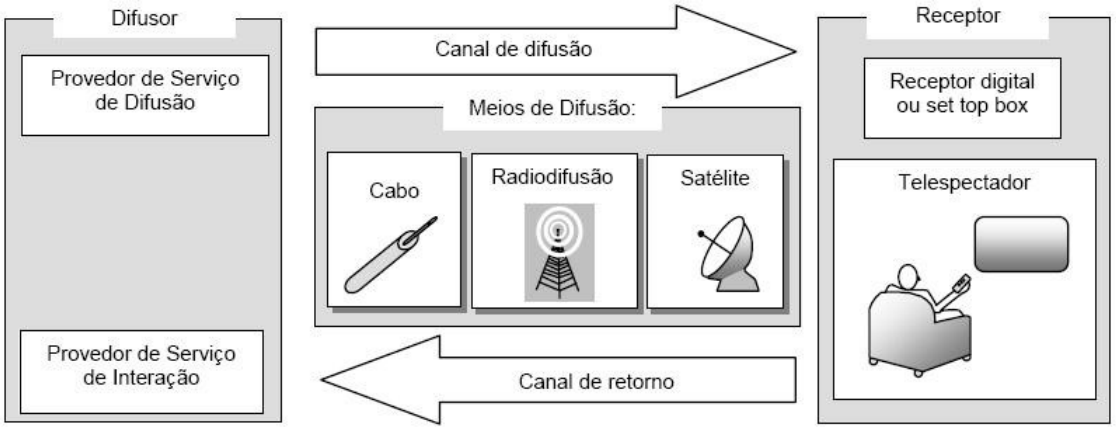

Figura 2: Processo de Transmissão de TV Digital Interativa (MONTEZ; BECKER, 2005).

totalmente digitais ou através de aparelhos analógicos acoplados a unidade conversoras (URD - Unidade Receptora Decodificada, sendo também conhecida pelos termos IRD - *Integrated Receiver Decoder* e STB - *Settop box*) (MONTEZ; BECKER, 2005; PAES; ANTONIAZZI, 2005).

A comunicação entre difusor e receptor é realizado através do Canal de difusão (sentido difusor-receptor) e Canal de retorno (sentido receptor-difusor), respectivamente.

O Canal de difusão é o meio utilizado pelo difusor para a transmissão do conteúdo aos telespectadores, sendo os mais comuns via satélite, cabo e radiodifusão. A tabela 1 (MONTEZ; BECKER, 2005) apresenta uma síntese das vantagens e desvantagens de cada meio.

Meio de Difusão	Vantagens	Desvantagens
Via cabo	Boa largura de banda para o seu canal de difusão e para o canal de retorno.	Refere-se ao alcance de transmissão restrita às residências interligadas fisicamente.
Satélite	Alcance de abrangência de seu sinal.	Dificuldade de oferecer o canal de retorno indispensável para serviços interativos.
Radiodifusão / difusão terrestre	Utilizada pelas TVs abertas nas transmissões convencionais sendo possível a migração lenta dos telespectadores da TV analógica para TV digital e interativa.	Largura de banda disponível, geralmente pouca devido às restrições na frequência espectral.

Tabela 1: Meios de Difusão do Sinal Digital de TV (MONTEZ; BECKER, 2005)

Quanto ao canal de interação (ou retorno), nos sistemas de TV digital interativa que utilizam a comunicação via satélite ou cabo, usam do mesmo meio para a transferência de dados do receptor para o difusor (provedor do serviço de interação).

2.2 Arquitetura e Padrões de Sistema de TV Digital

O sistema de TV digital possui uma arquitetura baseada em camadas, sendo que cada camada oferece serviços a camada superior e utiliza serviços oferecidos pela camada inferior. A figura 3 ilustra as camadas do sistema de TV digital interativa.

A camada de Transmissão (ou Modulação) é responsável por três serviços: a) o serviço de transmissão e recepção, cuja função é amplificar o sinal no difusor e sintonizar o sinal no receptor; b) o serviço de modulação e demodulação do fluxo de transporte codificado; c) o serviço de codificação e decodificação do fluxo de transporte.

Na camada de Transporte, acontece a multiplexação dos vários programas (fluxos de vídeo, áudio e dados) transmitidos pelo difusor em uma único fluxo de transporte. No receptor, a camada de Transporte realiza a demultiplexação do fluxo.

A camada de Compressão é responsável pela compressão e descompressão dos sinais de áudio e vídeo, no lado do difusor e no lado do receptor, respectivamente.

A camada de Middleware é a camada de abstração ou padronização dos serviços ofe-

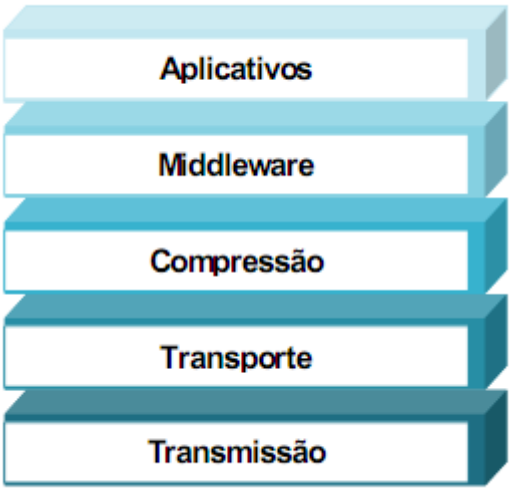

Figura 3: Arquitetura em Camadas de Sistemas de TV Digital (TONIETO, 2006)

recidos à camada de Aplicativos. Esconde peculiaridades das camadas de compressão, transporte e transmissão. É importante ressaltar que essa camada possibilita a portabilidade das aplicações transmitidas para qualquer tipo de receptor (set-top box) que suporte o middleware adotado.

A camada de Aplicativos é responsável pela execução dos aplicativos e representa a camada visível ao usuário.

Segundo (SEDREZ, 2008), cada camada pode ser construída de várias maneiras, por componentes diferentes. Estes componentes são escolhidos de acordo com o sistema de TV digital adotado. A figura 4 apresenta uma visão geral dos componentes adotados nos principais padrões de sistemas de TV digital.

2.2.1 Advanced Television System Committee - ATSC

Criado nos EUA, o padrão ATSC-T está em funcionamento desde novembro de 1998. É o padrão de TV digital adotado pelo Canadá, Coréia do Sul, Taiwan e México, entre outros países (MARQUES, 2008).

O ATSC-T não permite aplicações em dispositivos portáteis, devido a modulação (8VSB - *8-level Vestigial SideBand*) realizada por amplitude, e a inflexibilidade na configuração dos parâmetros de transmissão, que causam uma baixa imunidade a multipercurso afetando a recepção em campo (outdoor) e interiores (indoor) (PAES; ANTONIAZZI, 2005).

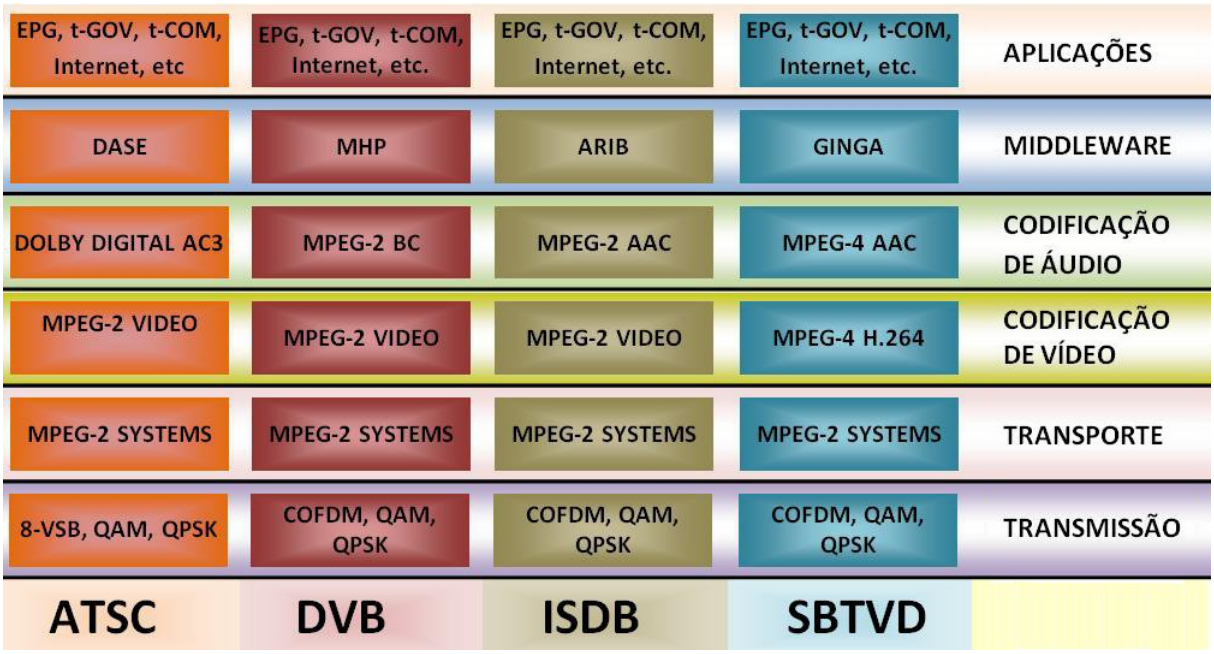

Figura 4: Padrões de TV Digital Interativa. Adaptação, Fonte: Grupo Dev-DTV .

O middleware utilizado é o DASE (*DTV Application Software Enviroment*) que possui suporte as linguagens Java, HTML e Javascript. Apesar do suporte à Java, não há compatibilidade de aplicações com o middleware MHP (MARQUES, 2008).

2.2.2 Digital Video Broadcasting - DVB

Conhecido como o padrão de TV digital europeu, o DVB é o padrão de TV digital adotado pela Austrália, Malásia, Hong-Kong, Índia, África do Sul, Inglaterra, entre outros países.

O DVB utiliza modulação COFDM (*Coded Orthogonal Frequency Division Multiplex*), codificação de áudio e vídeo com o padrão MPEG-2 e permite a transmissão para dispositivos móveis, porém, críticos relatam que não funciona satisfatoriamente (TONIETO, 2006).

O middleware do padrão é o MHP (*Multimedia Home Plataform*) que possui suporte à linguagem Java e a linguagem semelhante ao HTML, conhecida como DVB-HTML (MARQUES, 2008).

2.2.3 Integrated Services Digital Broadcasting - ISDB

Criado em 1999, o ISDB-T é o padrão de transmissão terrestre japonês. Utiliza modulação COFDM e suas maiores vantagens são a grande flexibilidade de operação e potencial de transmissão para dispositivos móveis e portáteis (SOMBRA, 2009).

O middleware do ISBD é o ARIB (*Association of Radio Industries and Businesses*) que permite a execução de aplicações desenvolvidas na linguagem BML (*Broadcast Markup Language*), uma linguagem declarativa baseada em XML (*Extensible Markup Language*) (TONIETO, 2006).

Um diferencial do padrão ISDB é a segmentação de canais, onde um canal digital é subdividido em vários subcanais que permitem a transmissão paralela de vários serviços (TONIETO, 2006).

2.2.4 Sistema Brasileiro de TV Digital - SBTVD

Atualmente, considerado o sistema de TV digital mais moderno do mundo, o Sistema Brasileiro de TV Digital Terrestre (SBTVD-T) tem como base o padrão japonês (ISDB - Integrated Services Digital Broadcasting) e foco na inclusão digital – baixo custo e aberto as possibilidades da interatividade.

O SBTVD, também conhecido como padrão nipo-brasileiro de TV digital, permite transmissão digital em alta definição e em definição padrão (SDTV), transmissão simultânea para o recepção fixa, móvel e portátil, e possui suporte a interatividade.

Segundo dados apresentados por (MARTINS, 2005), os requisitos básicos para atender as necessidades específicas da sociedade brasileira quanto a tv digital são:

- Baixo custo e robustez na recepção (classes C, D e E)
- Flexibilidade e capacidade de evolução (classes A e B)
- Interatividade e novos serviços (Inclusão Digital)

A implatação do SBTV, também conhecido por ISBT-Tb, já ocorreu na maioria das capitais e em diversas cidades do região Sul e Sudeste do país. A previsão é que em 2016, todas as transmissões de TV sejam em sinal digital.

O ISBT-Tb foi adotado em diversos países da América do Sul e Central, como ilustra a figura 5.

Figura 5: Adoção do ISBT-Tb pelos países da América do Sul e Central. Fonte: Site DTV

O padrão nipo-brasileiro de TV digital apresenta a adoção da compressão H.264 e o seu middleware Ginga, como os principais diferenciais perante o padrão ISDB.

O middleware Ginga é resultado de anos pesquisas lideradas pela PUC-RIO (Pontifícia Universidade Católica do Rio de Janeiro) e da UFPB (Universidade Federal da Paraíba).

O Ginga pode ser dividido em três subsistemas principais:

1. Ginga-CC (Common-Core) – oferece o suporte básico para os ambiente declarativo (Ginga-NCL) e o procedural (Ginga-J).

2. Ginga-NCL (Nested Context Language) – provê um ambiente de apresentação para aplicações declarativas desenvolvidas em NCL e Lua, linguagens estas, desenvolvidas no Laboratório de Telemídia da PUC-RIO, assim como, o próprio Ginga-NCL.

3. Ginga-J é mantido pelo Laboratório de Aplicações de Vídeo Digital (LAViD) da Universidade Federal da Paraíba e provê uma infra-estrutura de execução de aplicações Java e extensões especificamente voltadas ao ambiente de TV.

2.3 Aplicações em TVDI

Segundo (MARQUES, 2008), a qual baseia-se esta seção, as aplicações para TV digital interativa são as mais vastas possíveis, variando o grau de interatividade, e pode-se classificá-las em algumas categorias, como as propostas por (MACLIN, 2001):

- **TV avançada** – A TV avançada suporta a transmissão de vários elementos, tais como textos, gráficos e vídeos. O exemplo de aplicação mais abrangente desse categória é o EPG (Eletronic Program Guide), onde a grade da programação de televisão é apresentada como um conjunto dos elementos citados acima.

- **Internet na TV** – Possibilita aos telespectadores o acesso à Internet, o que viabiliza aplicações de mensagens eletrônicas (E-mail) e mensagens instantâneas (instant messaging), dentre outras aplicações de comunicação e acesso à conteúdos disponivéis na Web.

- **TV individualizada** – Aplicações desta categória permitem que o telespectador modifique a forma de apresentação do programa, como por exemplo, escolhendo o ângulo de visão da cena.

- **Vídeo sob demanda** – Conhecidas como VoD (Video-on-Demand), estas aplicações permitem ao telespectador assistirem ao conteúdo disponibilizado pela emissora, no momento em que desejarem.

- ***Personal Video Recorder* (PVR)** – Estas aplicações viabilizam a gravação dos programas de TV selecionados pelo telespectador em função do título, dos atores, do assunto ou de qualquer outro item relacionado à produção.

- ***Walled Garden*** – Visto como portal de aplicações interativas, que usualmente disponibiliza um conjunto de diversas aplicações - jogos, comércio eletrônico, governo televisivo (T-Governament), home-banking (t-banking), etc;

- **Console de jogos** – Uma das possibilidade mais exploradas atualmente, é o desenvolvimento de jogos, sejam para o entretenimento ou com objetivos educacionais.

Como exemplos das possibilidades oferecidades para o desenvolvimento de aplicações, pode-se citar os projetos inovadoras Ginga@Home e o projeto Gringa.

O projeto Ginga@Home (OLIVEIRA, 2010) tem objetivo de centralizar na TV, o controle de certos dispositivos de uma residência. A figura 6 ilustra o cenário de automação residêncial utilizando o Ginga@Home.

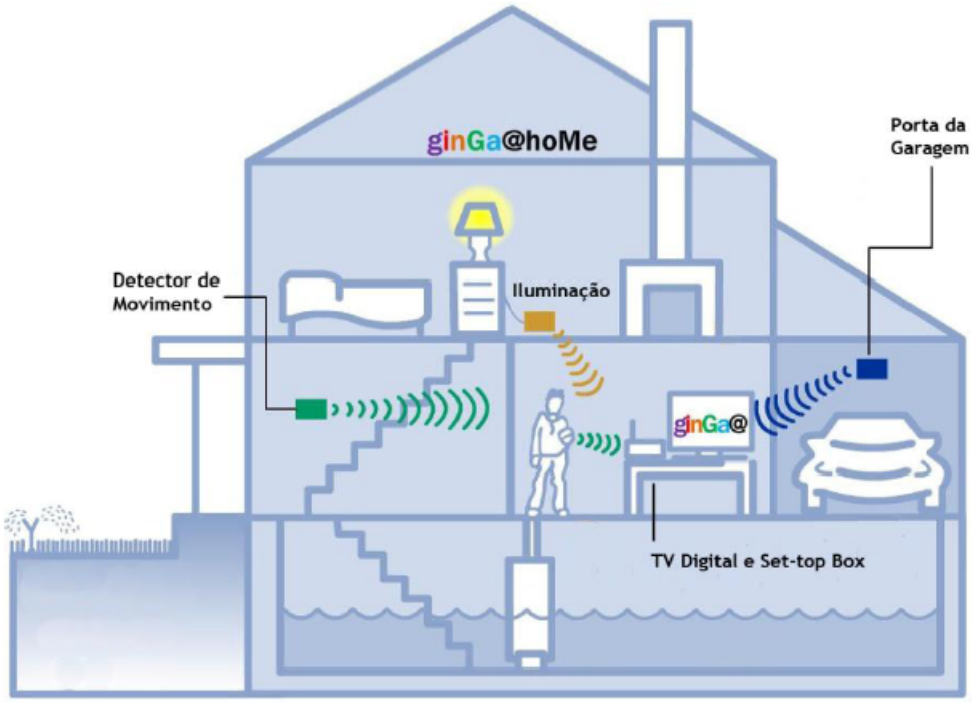

Figura 6: Cenário de Automação Residêncial Utilizando Ginga@Home (OLIVEIRA, 2010).

O Gringa (FILHO, 2010) é o middleware em desensolvimento na UERN (Universidade do Estado do Rio Grande do Norte) a partir do Programa Laboratórios de Experimentação e Pesquisa em Tecnologias Audiovisuais (XPTA.LAB), coordenado pela Sociedade Amigos da Cinemateca e pelas Secretaria do Audiovisual e Secretaria de Políticas Culturais do Ministério da Cultura. A seção seguinte é dedicada ao Gringa, alvo da proposta de escalonamento deste trabalho.

2.4 Middleware Gringa

O Middleware Gringa tem por objetivo facilitar o desenvolvimento e a execução de programas em um grid computacional no contexto de ambientes centrado em TV Digital Interativa.

O grid é formado por equipamentos computacionais presentes em redes locais domésticas, tais como desktops, laptops e smartphones, além dos equipamentos da TVDI, de maneira

que aplicações Ginga desenvolvidas em NCL, Lua e Java, possam interagir. A figura 7 ilustra um exemplo de formação do grid.

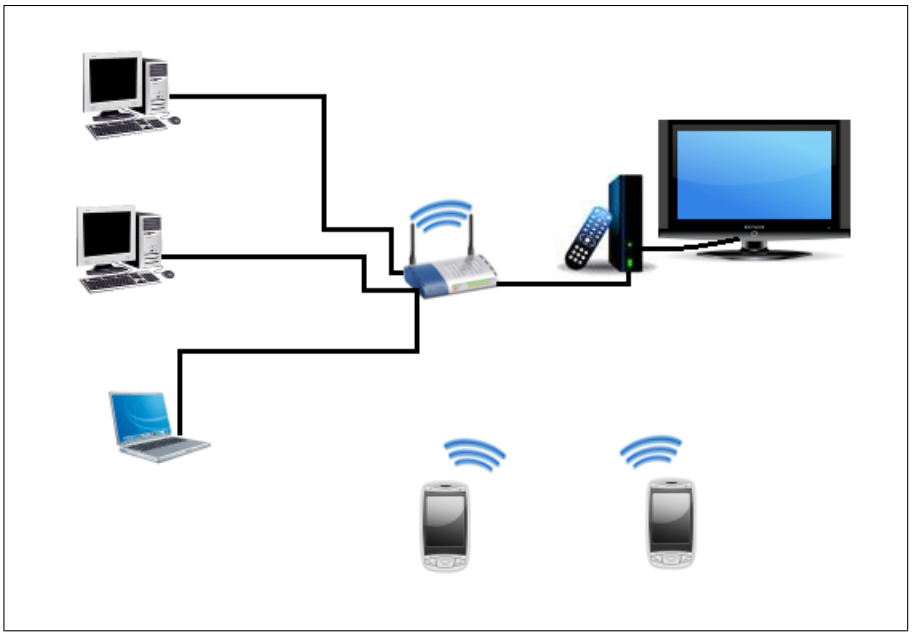

Figura 7: Cenário exemplo de formação do grid em rede local doméstica

2.4.1 Arquitetura e Tecnologias do Gringa

O Gringa é desenvolvido em Java e possui arquitetura orientada à serviços, onde os recursos oferecidos pelos nós do grid são disponibilizados na forma de serviços que podem ser requisitados através de uma interface comum.

Toda a comunicação é realizada atráves do protocolo XML-RPC[1] que utiliza HTTP para o transporte de mensagens codificadas em XML. Os nós Gringa são constituídos de duas camadas de software:

1. **Camada de Comunicação**: responsável por abstrair as chamadas entre os nós do grid. Esta camada é composta por:
   - Adaptador de rede: responsável por recebimento e envio das mensagens através dos protocolos de meio físico e enlaces configurados, tais como Bluetooth, rede cabeada ou sem fio;
   - Codificador/Decodificador HTTP: utilizado para criar/interpretar requisições HTTP para o transporte de mensagens XML-RPC contendo solicitações de

[1] http://www.xmlrpc.com/

serviços.

- Codificador/Decodificador XML-RPC: utilizado para criar/interpretar as mensagens com chamadas à serviços.

2. **Camada de Serviços**: acionada quando uma solicitação é realizada para o provimento de um serviço disponível. Os principais serviços no Gringa são:

   (a) Serviço de diretórios: usados para armazenar informações sobre os nós do grid e sobre os serviços disponibilizados por eles.

   (b) Serviço de escalonamento: que gerencia as requisições (tarefas) e realiza a seleção do melhor nó disponível para execução da tarefa.

   (c) Serviço de configurações: utilizado para gerenciar as informações de nós e serviços no grid, como por exemplo, a inserção, remoção ou atualização de dados sobre as características de um nó ou serviço.

   (d) Serviços de usuário: uma classificação geral das aplicações que podem ser disponibilizadas por todos os nós do grid.

O nó central (coordenador do grid) oferece os serviços descritos nos itens (a), (b) e (c).

O Capítulo seguinte aborda aspectos relacionado ao escalonamento em grids computacionais e o capítulo posterior, trata do escalonamento no Gringa.

3 ESCALONAMENTO EM GRIDS COMPUTACIONAIS

Segundo (STOCKINGER, 2007), existem diferentes ideias do que realmente é um grid computacional. No entanto, a ideia concebida por (FOSTER; KESSELMAN, 1997) é amplamente aceita, e a mesma faz uma metáfora com a rede elétrica, onde a eletricidade é fornecida sob demanda escondendo dos usuários detalhes como a origem da energia e a complexidade da malha de transmissão e distribuição. Em grids computacionais, recursos como processamento, memória e armazenamento em disco, são disponibilizados de forma transparente para oferecer capacidade computacional ao usuário (CIRNE, 2002).

Grid computacional (FOSTER; KESSELMAN, 1997) denota uma infraestrutura distribuída de hardware e software que provê acesso seguro, consistente, pervasivo e de baixo custo a um alto poder computacional. Foster (FOSTER, 2002) ressalta 3 pontos essenciais para a compreensão da definição proposta: (1) a coordenação de recursos que não estão sob o controle local, (2) a utilização de padrões, abertos e interfaces e protocolos de propósito geral, para (3) oferecer QoS não-trivial.

Para (CIRNE, 2002), grid computacional é um plataforma de execução de aplicações paralelas que apresenta características como:

- Heterogeneidade – Componentes de um grid apresentam características diferentes.
- Alta dispersão geográfica – Grids computacionais podem ter escala mundial.
- Compartilhamento – Os recursos que compõem o grid não estão dedicados a plataforma, mas oferecem a possibilidade de utilização temporária.
- Múltiplos domínios administrativos – Consequentemente, diferentes politicas de segurança e compartilhamento entre os domínios.
- Controle distribuído – Em geral, não há uma única entidade que detenha total controle sobre todo o grid.

Cabe destacar, que uma plataforma para execução de aplicações paralelas que não apresenta alguma das características acima, não deve ser desqualificada automaticamente como grid computacional (CIRNE, 2002).

Computação em grid é uma área relativamente nova, que tem despertado o interesse da comunidade científica e das empresas. Um dos tópicos que recebe bastante atenção é o escalonamento (*scheduling*). Grids computacionais apresentam alguns desafios a mais ao problema de escalonamento. Tais desafios são inerentes as características de grids, como a heterogeneidade, escalabilidade e o compartilhamento de recursos (DONG; AKL, 2006; SILVA, 2009).

O escalonamento consiste de um conjunto de regras que definem quando e como serão obtidas as informações dos recursos, como será a distribuição das tarefas e quais recursos serão utilizados para execução. As políticas de escalonamento em grids são constantemente chamadas de escalonadores de aplicação, pois não detêm os recursos utilizados na execução das tarefas, mas é o responsável pela coordenação de recursos que não estão sob controle local da aplicação, e pela distribuição das tarefas entre os dispositivos que compõe o grid.

Alguns algoritmos de escalonamento tradicionais são utilizados em grids, como por exemplo o *Workqueue* e *Round Robin*. No entanto, devido a dinamicidade do ambiente característico de grids computacionais, tais algoritmos não apresentam um desempenho satisfatório. Mesmo assim, são constantemente utilizados como base para o desenvolvimento de algoritmos mais robustos e adaptados às características do ambiente e das aplicações.

3.1 Taxonomia dos algoritmos de escalonamento

A taxonomia hierárquica proposta por (CASAVANT; JON; KUHL, 1988) para os algoritmos de escalonamento aplicados a sistemas distribuídos e paralelos classifica os algoritmos segundo uma visão do sistema. Para (DONG; AKL, 2006), apenas parte dos subconjuntos presentes na taxonomia aplicam-se especificamente a grids. Na figura 8, os subconjuntos aplicados a grids estão destacados em itálico. Logo abaixo, as subseções exploram a taxonomia ressaltando aspectos relacionados a aplicação dos algoritmos em grids.

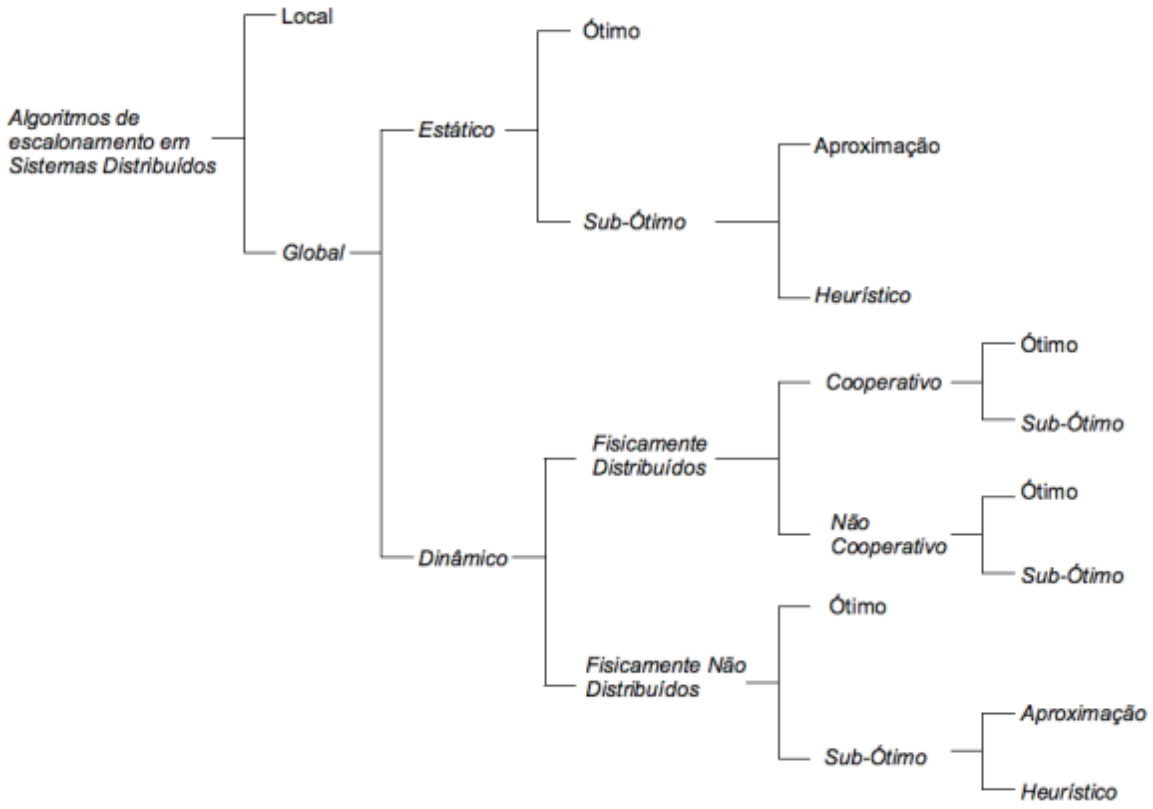

Figura 8: Taxonomia dos Algoritmos de Escalonamentos em Sistemas Distribuídos (ALIAGA, 2009)

3.1.1 Local vs Global

O primeiro nível da hierarquia difere o escalonamento realizado em sistemas como um único processador e o escalonamento realizado em sistemas com mais de um processador. O escalonamento local refere-se aos intervalos de tempo atribuídos para execução em um único processador, enquanto o global, refere-se a qual processador será utilizado para execução. Claramente, os algoritmos globais são os utilizados em grids computacionais, nos quais, o processo de escalonamento a coordena recursos distribuídos que estão disponíveis mas que não estão sob o controle local.

3.1.2 Estático vs Dinâmico

Uma politica de escalonamento é dita estática, quando todas as decisões do escalonamento são fixadas a priori, antes da execução da aplicação. Em geral, são utilizadas quando é possível estimar o custo das tarefas e se tem bastante informações sobre os re-

cursos disponíveis. No escalonamento dinâmico, as decisões são tomadas no momento em que antecede a execução das tarefas da aplicação. Por serem mais adaptativas, reagem melhor a dinamicidade de um grid computacional (JUNIOR et al., 2007).

3.1.3 Distribuídos vs Não-distribuído

No cenário dos algoritmos dinâmicos para escalonamento global, é possível compartilhar a responsabilidade das decisões entre diversos escalonadores distribuídos fisicamente. Essa estratégia é adotada pelo fato de que manter apenas um único escalonador possar torná-lo o gargalo do sistema. Um aspecto positivo para os escalonadores globais não-distribuídos (ou centralizados) é que são mais simples de implementar.

3.1.4 Cooperativo vs Não-cooperativo

Um questão relativa aos algoritmos distribuídos, é se os escalonadores envolvidos trabalharão de forma independente (não-cooperativos), sem a preocupação direta com objetivos gerais do sistema, ou se os mesmos serão cooperativos e terão uma visão do objetivo geral.

3.1.5 Ótimo vs Sub-ótimo

Quando todas as informações sobre as tarefas e estado dos recursos são conhecidos e precisos, e computacionalmente, é possível aplicar o critério de seleção, consegue-se obter menores tempo de execução (*makespan*) e maximizar a utilização dos recursos (*throughput*), são soluções ditas ótimas. Porém, a complexidade dos algoritmos de escalonamento e as dificuldades dos cenários de grids tem focados as pesquisas recentes na busca por soluções sub-ótimas (DONG; AKL, 2006).

3.1.6 Aproximado vs Heurístico

Os algoritmos de aproximação usam modelos formais para encontrar uma solução dita "boa" suficiente, de acordo com as métricas estabelecidas (um conceito do que seria o "ótimo"). As heurísticas consideram parâmetros que afetam indiretamente o sistema. Os algoritmos do tipo heurístico, são normalmente avaliados em cenários reais ou simulações, e mais adaptativos para os cenários de grids, onde os recursos e aplicações são altamente diversificados e dinâmicos (DONG; AKL, 2006).

Como relatado no estudo (DANTAS; FILHO, 2010), as heurísticas dinâmicas são os algoritmos mais utilizados no escalonamento em grids computacionais devido a boa adaptação a dinamicidade e heterogeneidade do ambiente. Normalmente, os algoritmos dinâmicos aplicados em grids, são avaliados em cenários reais ou simulações, afim de caracterizar o quão adaptáveis eles são aos aspectos presentes nos cenários.

3.2 Aspectos restritos à grids computacionais

Como apresentado por (DONG; AKL, 2006) , alguns aspectos relativos à grids computacionais que não são abordados pela taxonomia hierárquica proposta por (CASAVANT; JON; KUHL, 1988). São estes:

- **Funções Objetivo (*Objective Functions*)**

  Em grids, temos dois grupos de usuários: os provedores de recursos, que compartilham os recursos, e os consumidores de recursos, que submetem as aplicações. Os algoritmos podem ser classificados em dois grupos segundo o objetivo de cada grupo de usuários:

  - **Algoritmos centrados na aplicação**

    Visam a eficácia das aplicações dos consumidores de recursos. Uma métrica utilizada é o *makespan*, tempo gasto entre o início da primeira tarefa e o fim da última tarefa da aplicação. Outra métrica, é o custo econômico, que se baseia nos modelos econômicos (RODAMILANS, 2009).

  - **Algoritmo centrados no recurso**

    Relacionam-se com a utilização e o desempenho dos recursos. O *throughput*, ou seja, a capacidade de processar um número de aplicações em um determinado período de tempo e a taxa de utilização do recurso, que é a porcentagem de tempo que os recursos estão ocupados, são usados como métricas desta classe de escalonamento (RODAMILANS, 2009).

- **Adaptativos e Probabilísticos**

  Segundo (SILVA, 2009) um algoritmo adaptativo é aquele no qual o grau de ponderação dos parâmetros utilizados em sua implementação, mudam em tempo de execução, de acordo com o comportamento do sistema em resposta à política de escalonamento adotada. Portanto, um algoritmo que considera sempre a mesma ponderação entre os parâmetros é dito não-adaptativo.

Nos algoritmos probabilísticos, a ideia é gerar, aleatoriamente, a partir do dado inicial, um conjunto de soluções de um problema e dentre estas soluções, selecionar a melhor. (SILVA, 2009) diz que o número de soluções deve ser bastante para possibilitar que o conjunto apresente alguma solução que aproxime da solução ótima.

- **Dependência ou não dependência entre tarefas da aplicação**

  A independência entre as tarefas, ou seja, quando as tarefas não necessitam se comunicarem entre si e podem ser executadas em qualquer ordem, facilita o processo de escalonamento, pois tarefas dependentes aumentam a complexidade do problema impondo restrições às decisões do escalonamento. Este trabalho aborda somente aplicações de tarefas independentes, também chamadas de *Bag of Tasks*.

- **Aplicações que trabalham com muitos dados**

  Algumas aplicações envolvem a geração de um grande volume de dados, ou necessitam como entrada. Em muitos casos, estes dados requerem armazenamento e sistemas de gerenciamento especializados. Em grids, a localização onde uma aplicação será executada e os dados de entrada podem estar em diferentes locais, de forma que há um custo muito alto na transferência destes dados.

- **Modelos não-tradicionais**

  Baseiam-se na premissa que, assim como na natureza e na sociedade humana, os sistemas possuem características similares. Modelos não-tradicionais de escalonamento empregam regras fundamentadas no comportamento em sociedade (como métodos econômicos) e leis/fenômenos da natureza.

- **Limitações de QoS**

  Em grids, ambientes distribuídos heterogêneos não dedicados, a dinamicidade do ambiente aumenta a preocupação das aplicações com relação a QoS (*Quality of Service*). Em geral, os requisitos de QoS impostos pelas aplicações, variam de acordo com a preocupação dos diferentes usuários (tempo de processamento, largura de banda, tamanho de memória, prazo final), mas caracterizam-se por um conjunto de condições minímas necessárias para a execução bem sucedida da aplicação.

3.3 Algoritmos de escalonamento

O ambiente dinâmico e heterogêneo característico de grids, remete aos algoritmos de escalonamento dinâmicos, também chamados de *bin-packing*, pela mudança de valores,

em tempo de execução, dos parâmetros relevantes ao escalonamento, como por exemplo, a latência da comunicação, carga do host e tamanho da tarefa.

Dentro dos principais algoritmos dinâmicos, podemos destacar duas estratégias para realizar o escalonamento das tarefas: 1) realiza o escalonamento sem informações dos recursos e utiliza-se da replicação das tarefas para obter melhor tempo de resposta; 2) realiza o escalonamento com base em informações dos recursos e das tarefas.

A seguir são apresentados alguns dos principais algoritmos de escalonamento utilizados em grids.

3.3.1 *WorkQueue Replication*

Desenvolvido por (SILVA; CIRNE; BRASILEIRO, 2003), o WorkQueue Replication mantém uma fila de tarefas e cada tarefa é alocada a um processador disponível. Apenas uma tarefa por vez, é executa em cada processador. Quando um processador torna-se livre, recebe uma nova tarefa. O *WorkQueue Replication* difere do algoritmo tradicional (sem replicação) a partir do momento em que não há tarefas na fila e pelo menos um dos processadores está livre. Nesse momento, uma das tarefas em execução passa a executar, também no processador livre. Quando a tarefa ou uma de suas replicas conclui a execução, as demais são interrompidas.

O problema desta política é a grande possibilidade que existe de uma tarefa ser alocada para um dos processadores mais lento ou sobrecarregados do grupo. Outro ponto a considerar, é o número de replicas que uma tarefa pode ter. Em geral, limita-se o número de réplicas para tentar minimizar a sobrecarga causada pela replicação. Geralmente, o número máximo de réplicas de uma tarefa é indicado no nome do algoritmo (*WorkQueue Replication* 2x, ou simplesmente, WQR2x).

3.3.2 *XSufferage*

O *XSufferage* desenvolvido por (CASANOVA et al., 2000) é uma extensão do algoritmo Sufferage (IBARRA; KIM, 1977). Estes algoritmos consideram informações das tarefas e recursos para definir em qual dos hosts a tarefa será executada. Considerando apenas as máquinas disponíveis, o algoritmo calcula a perda que cada tarefa teria se não fosse atribuída ao host que oferecer melhor tempo de execução. O valor da perda, chamado de “sufferage”, é a diferença do melhor e do segundo melhor tempo de execução para a tarefa. Após o calculo do “sufferage” de todas as tarefas na fila, a tarefa com maior valor

é alocada ao host que oferece melhor tempo de execução.

A principal diferença entre *Sufferage* e *Xsufferage* é que o segundo considera também informações relativas a transferência de dados, como a disponibilidade atual do link de rede e o tempo para a transferência de dados da tarefa, enquanto o primeiro considera apenas o informações da CPU e tamanho da tarefa (tempo estimado para execução) (JUNIOR et al., 2007).

3.3.3 *Dynamic Fastest Processor to Lagest Task First* (Dyn-FPLTF)

O algoritmo Dyn-FPLTF (SILVA; CIRNE; BRASILEIRO, 2003) foi desenvolvido com base no estático *Fastest Processor to Largest Task First* (FPLTF) (MENASCÉ et al., 1995) e considera três informações para o escalonamento das tarefas:

- Velocidade do host (*Host Speed*) – um valor relativo que representa a velocidade do host dentre o conjunto de máquinas do grid.
- Carga do host (*Host Load*) – é a fração de CPU que não está disponível para execução da aplicação.
- Tamanho da tarefa (*Task Size*) – O tempo necessário para que uma máquina com HostSpeed igual a 1, e HostLoad igual a 0, execute a tarefa.

Para cada máquina o algoritmo reserva uma variável chamada TBA (*Time to Become Avaliable* - tempo para se tornar disponível) que é iniciada com valor zero. As tarefas são organizadas em ordem decrescente de tamanho e alocadas ao host que oferecer o melhor tempo de execução (CT – *Completion Time*) para ela.

$$CT = TBA + TaskCost \tag{3.1}$$

Onde TaskCost é:

$$TaskCost = \frac{\frac{TaskSize}{HostSpeed}}{(1 - HostLoad)} \tag{3.2}$$

A cada tarefa alocada para um host, a variável TBA é incrementada com o custo da tarefa alocada. A execução começa quando todas as máquinas no grid estão em uso. Ao finalizar uma tarefa, todas as tarefas que não estão executando são escalonadas até que todas as máquinas estejam sendo usadas (JUNIOR et al., 2007). Segundo (SILVA; CIRNE;

BRASILEIRO, 2003), o algoritmo apresenta uma excelente performance, no entanto, é complicado colocar em prática porque necessita de muitas informações sobre o ambiente e essas informações muitas vezes são difíceis de se obter e estão frequentemente indisponíveis devido as restrições administrativas.

Algoritmos como o Dyn-FPLTF, que requerem informações tanto dos recursos como das tarefas, são os que melhor se adaptam a dinamicidade do ambiente, porém, são complexos de implementar e em alguns casos inviáveis.

3.3.4 Max-Min e Min-Min

As heurísticas Max-Min e Min-Min utilizam as mesmas informações que o Dyn-FPLTF para realizar o escalonamento.

Na Max-Min, as tarefas são ordenadas de forma decrescente segundo o tamanho e cada uma delas é atribuída ao *host* que oferece o maior MCT (*Minimum Completion Time*) (RODAMILANS, 2009; CASANOVA et al., 2000).

Segundo (REIS, 2005), a Max-Min consegue baixos tempos de resposta quanto maior for o número de pequenas tarefas em relação às grandes. A ideia é a execução de tarefas pequenas nos *hosts* mais rápidos enquanto as máquinas lentas executam as tarefas grandes que receberam e que desbalanceam o sistema.

Na heurística Min-Min, as tarefas são ordenadas de forma crescente segundo o tamanho, e cada uma delas é atribuída ao *host* que oferece o menor MCT (*Minimum Completion Time*) de forma que os recursos são liberados mais rápidos para a execução de outras tarefas (REIS, 2005; CASANOVA et al., 2000).

3.4 Interoperação entre Dispositivos Móveis e Grids Computacionais

Devido ao rápido desenvolvimento nas tecnologias *Wireless* e *Grid Computing*, a integração de dispositivos *Wireless* em grid tem mostrado um enorme potencial para o compartilhamento dos recursos destes dispositivos, cujo uso atual tem sido predominantemente pessoal (BASTOS et al., 2008).

Em geral, os dispositivos *Wireless*, como celulares e PDAs, são caracterizados pelo baixo poder de processamento, pequena capacidade de armazenamento secundário, memória

reduzida, utilização de bateria e comunicação instável e de baixa largura de banda. Porém, a popularidade desses dispositivos, o baixo custo, tamanho reduzido e a mobilidade proporcionada, além de outros recursos que dispõem (como por exemplo, leitores de código e receptores GPS), associados à expansão dos pontos de acesso à redes sem fio, os tornam cada vez mais capazes de participar de redes de grid (MANVI; BIRJE, 2010).

A integração de dispositivos móveis em grid é extremamente problemática, devido as muitas deficiências que a comunicação sem fio e estes dispositivos ainda apresentam. Redes sem fio (*Wireless Networks*) são de baixa largura de banda e instáveis – as propriedades de QoS podem variar em um curto espaço de tempo e existe um consideravél consumo de energia para transmissão de dados no meio utilizado – o ar. Já os dispositivos móveis possuem bem menos recursos disponivéis que os computadores de mesa (*Desktop*).

3.4.1 Abordagens de Utilização dos Dispositivos Móveis em Grids

Como relata (BASTOS et al., 2008), são duas as abordagens de utilização dos dispositivos móveis em grids:

1. Dispositivos móveis podem participar como clientes e/ou provedores no contexto de um grid fixo pré-existente.

   Exemplos dessa abordagem são os trabalhos realizados por:

   - (GRABOWSKI; LEWANDOWSKI; RUSSELL, 2004) - utiliza uma aplicação Web projetada especificamente para atender as requisições dos dispositivos móveis para o GridLab[1]. Permite que dispositivos que utilizam JME no perfil MIDP 1.0 (*Mobile Information Device Profile*), dispositivos com limitações mais restritas de recursos, tenham acesso aos serviços do grid, como a submissão e gerência de tarefas.
   - (GOMES; SILVA; ENDLER, 2007) - Dispositivos móveis interagem como clientes do Integrade[2]. A comunicação é relizadas através de proxy que mapeia as conexões socket TCP (dos dispositivos móveis) para requisições à interface de submissão de aplicações do Integrade, utilizando a tecnologia de objetos distribuídos CORBA (*Common Object Request Broker Architeture*).

2. Dispositivos móveis formam um grid puramente sem fio – denominados *Mobiles Ad hoc Grids*.

[1] www.gridlab.com
[2] www.integrade.org.br

- Middlware MasGrid (*Mobile Ad hoc Service Grid*) (IHSAN; QADIR; IFTIKHAR, 2005) desenvolvido com base em protocolos de conectividade e roteamento existente em redes ad hoc, para fornecer serviços de descoberta e acesso à recursos compartilhados dispositivos mobile.
- Middleware MoGrid utiliza o protocolo denominado DICHOTOMY (GOMES et al., 2007) para formação de ad hoc grid (multi hop.

Em ambas abordagens, proxies são utilizados para a integração dos dispositivos móveis em grid. Tais proxies são implementados na forma de protocolos desenvolvidos especificamente para o propósito, ou com a utilização de aplicações Web.

Na maioria dos trabalhos citados no início desta seção, os dispositivos móveis atuam como clientes de um grid, exceto nos casos em que o mesmo faz parte de um grid puramente *Wireless* (*Ad hoc grids*). Em (BASTOS et al., 2008) propõe-se a utilização de proxies (baseado no protocolo - DICHOTOMY) para a interoperação de wireless grids à grids fixos (ditos convencionais) de forma a expandir os recursos do grid móvel, ou seja, todo o grid móvel passa a utilizar os recursos do grid fixo através do proxy.

3.4.2 Aspectos Relevantes para Escalonamento em Grids com Interoperação de Dispositivos Móveis

Uma grande preocupação em Wireless grids é o consumo de bateria dos dispositivos. Assim, é relevante para o escalonamento considerar aspectos relacionados a esse consumo, de modo que, faz-se necessário um eficiênte balanceamento que permita otimizar o consumo de bateria dos dispositivos móveis e prolongar a permanência dos mesmos no grid.

(AZEVEDO et al., 2009) relata o uso frequente da replicação de tarefas como tentativa de otimizar alguma função objetivo relacionada às limitações dos dispositivos móveis. Porém, nota-se que a replicação provoca uma sobrecarga nos dispositivos e consquentimento o desperdício de recursos.

(TSENG; CHIN; WANG, 2009) apresenta o algoritmo ATCS-MCT (*Apparent Tardiness Cost Setups – Minimum Completion Time*) que considera o tempo de comunicação e atribui multas, caso o nó perca o *deadline* de uma tarefa. O trabalho possui foco na redução do custo financeiro associado a utilização de recursos de outra organização. O trabalho não relata a presença de dispositivos móveis, mas comenta o fato do tempo de comunicação ser ignorado frequentimente na avaliação dos algoritmos.

A proposta de (HUANG et al., 2006) é uma heurística que otimiza o consumo de energia dos dispositivos móveis que interoperam com um grid fixo. O escalonamento é realizado em dois níveis. No primeiro nível (*Overall Scheduler*), tem-se o escalonamento do grid convencional considerando cada agrupamento de dispositivos móveis, coordenados por um proxy (*Wireless Proxy Scheduler*), como um único recurso do grid. Veja a figura 9.

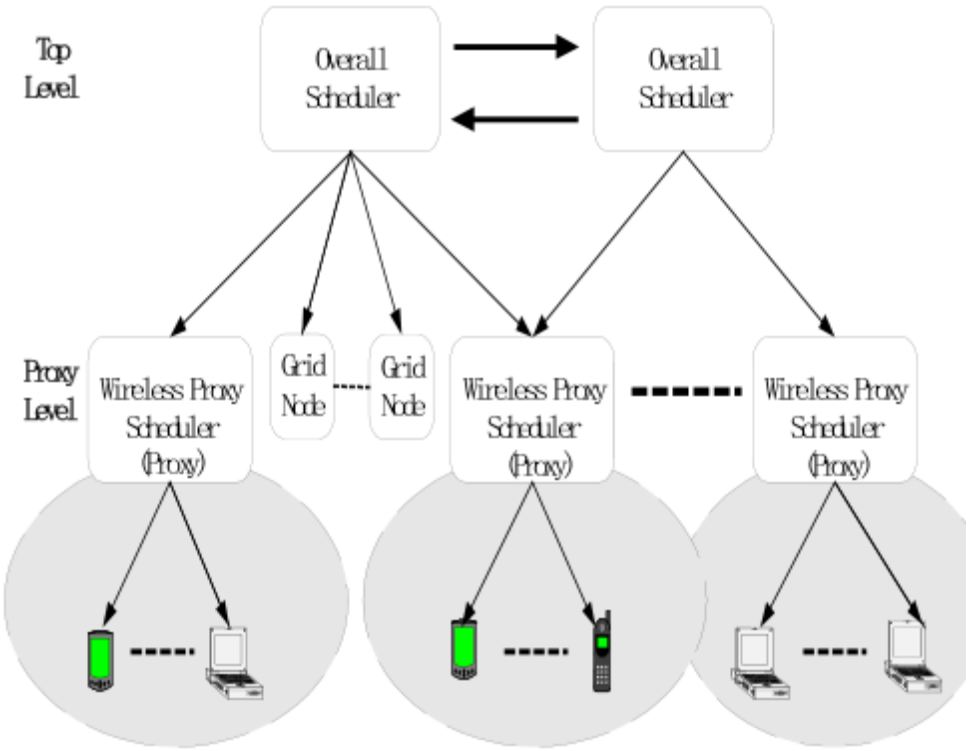

Figura 9: Modelo de Escalonamento Hierárquico baseado em proxy (HUANG et al., 2006)

No *Wireless Proxy Scheduler*, as solicitações são atendidas segundo o algoritmo FIFS (*First Input First Service*) e o escalonamento das tarefas nos nós sem fio é realizado por uma versão modificada do algoritmo Min-Min, que passa a considerar o consumo de energia (bateria do dispositivo).

Os parâmetros do algoritmos são:

1. Máximo de energia da bateria: B(j)
2. Taxa de consumo de energia para a execução de uma tarefa: E(j)
3. Taxa de consumo de energia da comunicação de uma tarefa por unidade de tempo: C(j)
4. Largura de banda da comunicação do dispositivo: BW(j)

Os parâmetros 2 e 3 são simplificados para um modelo real de consumo, não detalhado pelos autores.

A energia consumida na execução de uma simples tarefa i em nó *Wireless* j é determinada por: ETC(i,j) * E(j)

Já a energia consumida para transferência de um dado de tamanho g de um nó j para o nó k é definida por: CMT(j, k) = 1/min(BW(j), BW(k)) * C(j) * | g | .

O algoritmo Min-Min apresentado por (HUANG et al., 2006), encontra para cada tarefa, o nó que oferece o menor consumo de energia (transmissão e execução da tarefa) descritos como:

$$En = ETC(i,j) * E(j) + CMT(j,k) * C(j) * \mid g \mid \tag{3.3}$$

No segundo momento, escolhe-se a tarefa com o *deadline* mais próximo e então, a submete. Após isso, os valores do tempo e energia disponível do nó são atualizados. O conjunto de tarefas do escalonamento é atualizado incluindo novas tarefas e o processo reinicia.

Como resultado, (HUANG et al., 2006) apresenta uma redução no consumo de energia quando 25% ou mais dos recursos nos nós móveis são reservados, ou seja, os dispositivos tornam-se indisponíveis para o escalonamento quando atigem cerca de 75% da sua capacidade. A reserva tem por objetivo, antender as solicitações recebidas e o tratamento de falhas, evitando sobrecarga nos dispositivos.

4 ESCALONAMENTO NO MIDDLEWARE GRINGA

O Middleware Gringa aplica-se nos mais diversos ambientes – desde residências com poucos dispositivos conectados à ele, até ambientes públicos, onde o número de dispositivos pode ser relativamente grande. Assim, em certos momentos ou cenários, o grid será puramente sem fio, e em outros será misto ou puramente fixo – dispositivos conectado por rede cabeada. Desse modo, é extremamente relevante considerar no escalonamento as características de processamento dos dispositivos e a comunicação utilizada, assim como, o consumo de bateria nos dispositivos móveis envolvidos.

Os cenários de aplicação do middleware Gringa podem ser caracterizados como a interoperação de dispositivos móveis com o grid fixo em rede doméstica, formado pelo set-top box e outros dispositivos interligados a ele. O set-top box centraliza todos os serviços relacionados a administração do grid e atua como proxy na interoperação dos dispositivos móveis.

4.1 Estimando custo de tarefas

O custo computacional de um algoritmo é o número de operações com vírgula flutuante requeridas para a sua execução. No entanto, o número exata de operações efetuadas não é essencial, bastando apenas determinar a ordem de grandeza do algoritmo em função de um parâmetro que está associado à dimensão do problema (QUARTERONI, 2006). Por exemplo, para determinar o custo de produto de uma matriz quadrada de ordem n por um vetor v, são necessários n produtos e *n-1* adições ($n(2n-1)$).

A velocidade de um computador é frequentemente medida pelo número máximo de operações com vírgula que pode executar num segundo (flops). Porém, o desempenho de um algoritmo é dependente de outros aspectos, como o tempo de acesso a memória, e não somente da velocidade do processamento e do número estimado de operações do

algoritmo.

No algoritmo de escalonamento PUTS (descrito na seção seguinte), o custo das tarefas são estimados com base no método do algoritmo Dyn-FPLTF e definidos como *taskTime*, incluindo o tempo de transferência dos dados pela rede.

O valor de taskTime é o tempo estimado para transferir (TimeTransf) e executar a tarefa (TaskCost), os quais são defenidos da seguinte maneira:

$$TaskCost = \frac{\frac{TaskSizeInMflops}{HostSpeed}}{(1 - HostLoad)} \tag{4.1}$$

E TimeTransf:

$$TimeTransf = \frac{TaskSizeInBytes}{TxWidth} \tag{4.2}$$

Os valores de *HostSpeed* e *HostLoad* são mensurados com a execução de benchmarks, que no Gringa é utilizado o algoritmo denominado Linpack (IDA, 2005).

TxWidth é a largura de banda da comunicação estimada pelo benchmark Ping-Pong (LUSZCZEK et al., 2005).

Desta forma, o custo de uma tarefa é estimado como o tempo total necessário para a transferência da tarefa para o nó e o tempo de execução estimado com base no estado atual do dispositivo (velocidade, carga e largura de banda).

4.2 O Algoritmo PUTS

A proposta do algoritmo PUTS (Power-aware User-preference Task-size based Scheduling) é considerar aspectos relacionados ao tamanho da tarefa e as preferências dos usuários quanto a disponibilidade dos nós e dos serviços nos nós. O algoritmo atribui a tarefa ao nó que oferece o menor valor para:

$$CTE * UPW \tag{4.3}$$

Onde UPW é um peso associado às preferências do usuário sobre a utilização dos serviços disponibilizados no nó (detalhes da fórmula são mostrados posteriormente).

O Completion Time Energy (CTE) engloba paramêtros relativos ao processamento, comunicação e consumo de energia nos dispositivos. O valor do CTE de cada dispositivo

para um tarefa é definido por:

$$CTE = TBA + TE \tag{4.4}$$

Para TE (*Time Energy*) igual a:

$$TE = taskTime * BTR \tag{4.5}$$

O BTR (*Battery Rate*) é um peso relativo ao consumo de energia no dispositivo, calculado segundo a metodologia descrita na seção seguinte. O objetivo do fator BTR é aumentar o custo das tarefas (*taskTime*) nos nós, à medida que o nível de energia nas baterias dos mesmos ficam mais próximos de zero.

A soma dos tempos estimados para transferir e executar a tarefa é o valor utilizado para atualizar a variável TBA de cada nó, no momento de atribuição e da conclusão de uma tarefa por ele.

4.2.1 Consumo de Energia

Ao entrar no grid, o coordenador registra o nível de energia e o instante de entrada do nó. A cada atualização das informações de estado (carga, nível de energia, memória livre, outros) de um nó, o coordenador do grid verifica se houve alterações no nível de energia do dispositivo.

No momento em que é verificada uma alteração (algoritmo 1), calcula-se então a taxa de consumo da bateria no dispositivo, representado pela variável *txPerda* definida em percentuais por segundo. Nesse instante, os valores são atualizados para detectar novas mudanças no nível de energia.

Algoritmo 1 calcTxPerda – Cálculo de txPerda

```
  {instante de entrada do nó no grid}
levelbattery ← newLevelbattery; {registra nível de bateria}
timePreviousLB ← currentTime; {registra o instante do nível de bateria registrado}

 {Durante as atualizações das informações sobre os recursos do nó}
if newLevelbattery ≠ levelbattery then
  txPerda ⇐ (levelbattery − newLevelbattery)/(currentTime − timePreviousLB);
  levelbattery ← newLevelbattery;
  timePreviousLB ← currentTime;
end if
```

Enquanto não há alterações no nível de energia de um dispositivo, a taxa de consumo (txPerda) é constante em um valor extremamente pequeno, não influênciando no cálculo de BTR.

O BTR (Battery Rate) é calculado considerando o tempo da tarefa, o nível de energia e a taxa de consumo no dispositivo.

O tempo da tarefa e a taxa de consumo são utilizados para estimar o valor do nível de energia após a conclusão da tarefa (equação 4.6).

$$estimatedLB = levelbattery - (txPerda * taskTime); \tag{4.6}$$

Quando o nível estimado no término da tarefa é maior que o atual, significa que o dispositivo está carregado e sua taxa de perda *txPerda* é negativa.

BTR então é definido por:

$$BTR = \frac{100\%}{estimatedLB} \tag{4.7}$$

O valor de BTR aumenta gradativamente a medida que o nível de bateria do dispositivo diminui. Considere, por exemplo, um dispositivo cujo nível atual da bateria é 100% e o valor do nível estimado é igual a 50%, o valor de BTR multiplicará o custo da tarefa em 2 (100%/50%) e quando o nível estimado for 10%, o custo da tarefa (*taskTime*) será multiplicado por 10 (100%/10%).

Quando o valor de BTR é menor ou igual a zero, devido ao valor estimado ser negativo, a tarefa não é atribuída ao nó em questão.

Dispositivos que consomem bateria terão um peso maior (BTR), determinado pela taxa de consumo e nível de energia atual.

Os dispositivos conectados à rede elétrica que não utilizam bateria, mantém estáticos os valores de *levelbattery* e *txPerda*, em 100 e $2 * 10^{-1074}$, respectivamente.

O valor estimado de bateria para o instante após a conclusão de uma tarefa, pode ser utilizado para estabelecer uma reserva da bateria, determinando até quando o dispositivo pode receber solicitações. Se no instante da atribuição de uma tarefa, o valor de estimatedLB for menor que o percentual da reserva desejada, a tarefa não deve ser atribuída ao nó (Algoritmo 2 realiza o cálculo de BTR para uma tupla <*nó*, *tarefa*>).

Em suma, o objetivo do BTR é balancear o consumo de energia das baterias e manter

Algoritmo 2 calcBTR – Cálculo de BTR

Require: $taskTime$ {custo da tarefa no nó}
{cálculo do nível de bateria após o término da tarefa}
$estimatedLB \leftarrow (levelbattery - txPerda * taskTime)$;
{verifica o limite mínimo do nível de bateria}
if $estimatedLB \leq 0$ **or** $estimatedLB \leq reservationBattery$) **then**
$estimatedLB \Leftarrow -1$;
end if
{verifica o limite máximo}
if $estimatedLB > 100$ **then**
$estimatedLB \Leftarrow 100$;
end if
$BTR \leftarrow 1/(estimatedLB/100)$;
return BTR;

os nós móveis por mais tempo no grid.

Os dispositivos mais rápido (processamento, rede) se sobressaem, caso tenham uma taxa de consumo de energia competitiva.

4.2.2 Preferências do Usuário

Devido à característica dos cenários, onde os dispositivos não estão exclusivamente para operação no grid, e pela existência de um custo associado ao serviço que não é considerado pelo algoritmo de escalonamento, como por exemplo, o custo financeiro para envio de um SMS ou de utilização da conexão com a Internet (que pode variar entre dispositivos), adotaram-se parâmetros configuráveis a critério do usuário, que influenciam na disponibilidade do dispositivo e dos serviços no oferecidos no mesmo. Os parâmetros são:

UPN – User Preference Node (Nó de preferência do usuário)

UPSN – User Preference for Service at Node (Preferência do usuário pelo serviço no nó)

Estes parâmetros podem assumir os valores no intervalo de 0 à 1, de acordo com a preferência do usuário pela disponibilidade deste nó ou serviço oferecido no nó em questão.

No escalonamento, os valores de UPN e UPSN representam um peso no *Completion Time Energy* (CTE) e podem ser definidos de modo a priorizar a configuração de

disponibilidade do nó (equação 4.8), ou do serviço no nó (equação 4.9). Vejamos:

$$UPW = \frac{2}{UPN + UPN * UPSN} \tag{4.8}$$

$$UPW = \frac{2}{UPSN + UPSN * UPN} \tag{4.9}$$

A solicitação é atribuída ao dispositivo que oferece menor valor para $CTE * UPW$, segundo o procedimento do algoritmo 3.

Algoritmo 3 puts – seleciona o melhor nó dentre o grupo para executar a tarefa

Require: *task* {Tarefa a ser submetida} *listNodes* {lista de nós com o serviço}
  $puts \leftarrow MAX\_VALUE$;
  $incrTba \leftarrow 0$; {valor utilizado para atualizar o TBA do nó selecionado}
  $btr \leftarrow 0$;
  $upw \leftarrow 0$;
  $te \leftarrow 0$; custo com fator BTR
  $timeTransf \leftarrow 0$;
  $taskCost \leftarrow 0$;
  $node$
  $bestNode$

  while $listNodes.hasNextNode()$ **do**
    $node \leftarrow listNodes.nextNode()$;
    $timeTransf \leftarrow (task.sizeInBytes/node.txWidth)$;
    $taskCost \leftarrow (task.sizeInMflops/node.hostSpeed)/(node.hostLoad/node.hostSpeed)$;
    $btr \leftarrow calcBTR(timeTransf + taskCost)$;
    if $btr > 0$ **then**
      $te \leftarrow (timeTransf + taskCost) * btr$;
      $upw \leftarrow 2/(node.upn + node.upn * node.upsnService(task.serviceName))$;
      $cteNode \leftarrow node.tba + te$;
      $putsNode \leftarrow cteNode * upw$;
      if $cteNode \leq puts$ **then**
        $puts \leftarrow putsNode$;
        $bestNode \leftarrow node$;
        $incrTba \leftarrow (timeTransf + taskCost)$;
      end if
    end if
  end while
  {atribuição da tarefa}
  $bestNode.taskCount \leftarrow bestNode.taskCount + 1$;
  $bestNode.tba \leftarrow bestNode.tba + incrTba$;
  $task.deadline \leftarrow incrTba$;
  {Envia a tarefa para o nó selecionado}
  $sendTask(bestNode, task)$;

Quando o valor de UPN ou UPSN, é igual a zero, significa que o dispositivo ou serviço, está indisponível no momento.

4.2.3 Agrupamento de dispositivos

Além de considerar a transferência de dados, consumo de energia nos dispositivos e as preferências do usuário, o algoritmo organiza os dispositivos em dois grupos de acordo com o valor do HostSpeed (estimado em Mflops com a execução do benchmark Linpack). Assim, há o grupo denominado "grupo 0", cujos dispositivos possuem HostSpeed menor que o limiar, e o grupo denominado "grupo 1" constituído dos dispositivos com HostSpeed igual ou maior que o limiar dos grupos.

Cada dispositivo do "grupo 0" recebe apenas uma tarefa por vez. Isto evita que em um cenário onde o grid é formado apenas por nós de baixo poder de processamento, o algoritmo sobrecarregue algum destes nós com várias tarefas de uma única vez, pelo fato de o dispositivo apresentar melhores condições se comparado aos outros dispositivos.

Para melhor compreender o caso citado acima, considere um cenário com 3 nós (smartphones), onde 2 deles estão com baixos valores para o nível de bateria e preferências do usuário (UPN e UPSN), e o outro dispositivo está com valores máximos. Nesta situação, o algoritmo sem a divisão dos dispositivos em grupos e sem a restrição imposta ao grupo com HostSpeed inferior ao limiar, atribuiria tarefas até que todos os nós estejam ocupados, permitindo que o nó com melhores valores dos paramêtros considerados, receba mais de um tarefa, sobrecarregando-o.

O algoritmo 4 impõe a restrição, limitando o número de tarefas nos nós, evitando a sobrecarga.

As requisições são primeiramente atendidas pelos dispositivos com capacidade superior ao limiar dos grupos. Somente quando não há nós deste grupo disponível, é que o algoritmo tenta escalonar a tarefa para um dos nós livres do grupo com HostSpeed abaixo do limiar (veja algoritmo 5).

As tarefas são mantidas no grid até que algum nó possa executá-la ou a aplicação cliente é encerrada (remove todas as suas tarefas no grid).

Algoritmo 4 getListNodes – Obtém lista de nós com o serviço

Require: *serviceName* {nome do serviço solicitado}
Require: *group* {identificação do grupo desejado}

```
listNodesInGrid {lista de todos os nós com o serviço}
listNodes
 {percorre a lista de nós}
while listNodesInGrid.hasNode() do
  node = listNodesInGrid.nextNode();
  if group = 1 then
    if node.group = 1 and node.upn > 0
    and node.upsnService(serviceName) > 0 then
      listNodes.add(node);
    end if
  else if group = 0 then
    if node.group = 0 and node.taskCount = 0
    and node.upn > 0 and node.upsnService(serviceName) > 0 then
      listNodes.add(node);
    end if
  end if
end while
return listNodes;
```

Algoritmo 5 selectGroup – selectiona o grupo de nós para o qual a tarefa será escalonada

Require: bagOfTasks {lista de tarefas ordenadas decrescentemente pelo custo de execução}
nodes {lista vazia para armazenar referências aos nós}
task {variável para referenciar uma tarefa}
{percorre a lista de tarefas}
while $bagOfTasks > 0$ **do**
  $task = bagOfTasks.nextTask()$
  {pegar a lista de nós do grupo 1 com o serviço}
  $nodes \leftarrow getListNodes(1, serviceName)$
  {verifica se existe algum nó livre}
  $hasFreeNode = falso$;
  while $nodes > 0$ **and not** $hasFreeNode$ **do**
    {Obtém o próximo nó da lista}
    $node \leftarrow nodes.nextNode()$;
    if $node.tba < 4$ **then**
      $hasFreeNode = verdadeiro$
    end if
  end while
  if $hasFreeNode$ **then**
    {função que escolhe o nó mais adequado}
    $scheduler(task, nodes)$;
  else
    {pegar a lista de nós livre do grupo 0 com o serviço}
    $nodes \leftarrow getListNodes(0, serviceName)$
    if $nodes > 0$ **then**
      $scheduler(task, nodes)$;
    end if
  end if
end while

5 ESTUDOS DE CASO

Com o objetivo de verificar a eficácia do algoritmo e a degradação do fator UPW (relativo à preferência de usuário) no desempenho, foram realizados 2 estudos de caso, utilizando uma aplicação de multiplicação de matrizes de números reais. Para cada cenário, foram realizadas 30 execuções da aplicação, com matrizes (A e B) de ordem 400x260 e 260x400, respectivamente, geradas aleatóriamente.

A aplicação divide o número de linhas da matriz A em 50 partes não uniformes, e submete uma cópia da matriz B junto à uma das partes da matriz A, como uma tarefa ao Gringa.

Os estudos de caso são apresentados a seguir.

5.1 Estudo de Caso 1 – Cenário com poucos recursos

Para o estudo de caso 1, o grid é formado por 3 nós, como ilustra a figura 10.

Os valores do *HostSpeed* dos dispositivos são 180 Mflops para o **Netbook** e 5,5 Mflops para os *smartphones*.

O limiar de grupos foi definido em 100 Mflops, e portanto, o *grupo 1* é constituído do **Netbook** e o *grupo 0* dos *smartphones*.

No primeiro cenário, a configuracão de preferência de usuário foi mantida de forma a não influênciar no escalonamento (valores de UPN e UPSN iguais a 1). Já no segundo cenário, houve alterações na configuração de preferência de usuário no **Netbook**, de modo que, o valor do CTE obtido do nó para as tarefas tivesse um acréscimo de 17,19 vezes o valor.

O gráfico (figura 11) apresenta as médias das 30 execuções em cada cenário.

A diferença existente entre o desempenho do *netbook* em relação aos *smartphones*, não permitiu que a configuração de preferências de usuário influênciasse na decisão, visto que

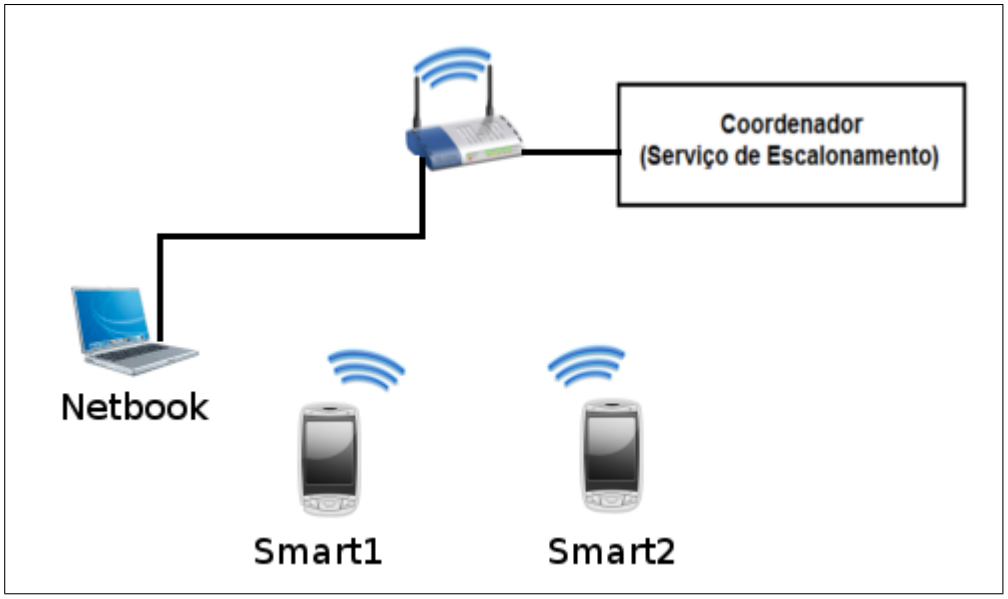

Figura 10: Estudo de caso 1 - Formação do grid

	N^o de tarefas		**Tempo utilizado (s)**		**Tam. das tarefas (Mflops)**	
	Cenários					
Dispositivos	**1**	**2**	**1**	**2**	**1**	**2**
Smart1	0,867	0,967	16,396	18,650	4,144	4,073
Smart2	1,000	0,933	16,438	20,43	3,064	7,829
Netbook	48,133	48,100	10,214	11,772	2,022	1,930

Tabela 2: Balanceamento de carga – Estudo de caso 1

as tarefas são escalonadas primeiramente para os dispositivos do grupo 1. Assim, tarefas foram enviadas para o **Netbook** até que o mesmo torna-se ocupado (TBA ultrapassa o valor da frequência de atualização das informações – que nos experimentos foi determinada em 4 segundos). Nesse instante, o algoritmo passa à atribuir tarefas para os dispositivos do grupo 0, que por serem homogenêos, apresentam desempenhos diferentes em virtude de suas respectivas cargas.

Como apresenta a tabela 2, a carga dos dispositivos é mantida em praticamente o mesmo número de tarefas e tempo de utilização do recurso pelo grid.

O **Smart1**, tem um pequeno aumento no número de tarefas e consequentemente, é utilizado pelo grid por um intervado de tempo maior. Já o **Smart2**, recebe um pouco menos tarefas, porém maiores, o que eleva o tempo de processamento.

Como o **Netbook** recebe um número razoavelmente grande das tarefas do conjunto,

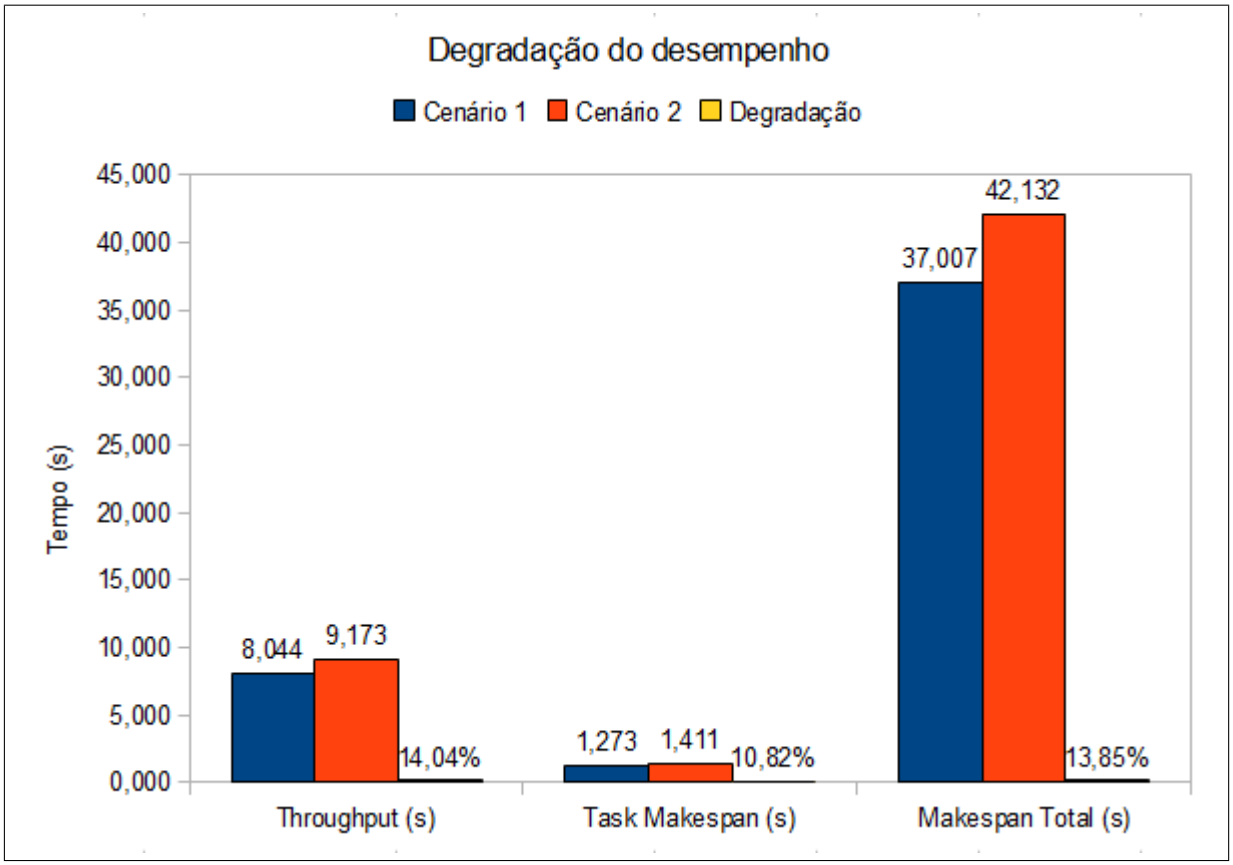

Figura 11: Gráfico – Degradação do desempenho - Estudo de caso 1

tende a aproximar o tamanho médio das tarefas recebidas com o tamanho médio das tarefas da aplicação (2 Mflops).

5.2 Estudo de Caso 2 – Cenário com heterogeneidade de recursos

Neste estudo de caso, o grid é mais heterogêneo, formado por um número maior de dispositivos, nos dois agrupamentos. A figura 12 ilustra a formação do grid.

Os valores do *HostSpeed* dos dispositivos **Desktop1** e **Desktop2** são iguais, média de 1370 Mflops.

Seguindo a mesma metodologia do primeiro estudo, no cenário 1, as configurações de preferência de usuário são mantidas em valores que não influênciam no escalonamento. No cenário 2, os valores de UPN e UPSN são configurados no **Desktop2** de modo a incrementar 7,33 vezes o valor do CTE (*Completion Time Energy*) obtido do nó para as tarefas. Tal configuração impõe que uma tarefa só será atribuída ao nó quando 8,33 vezes

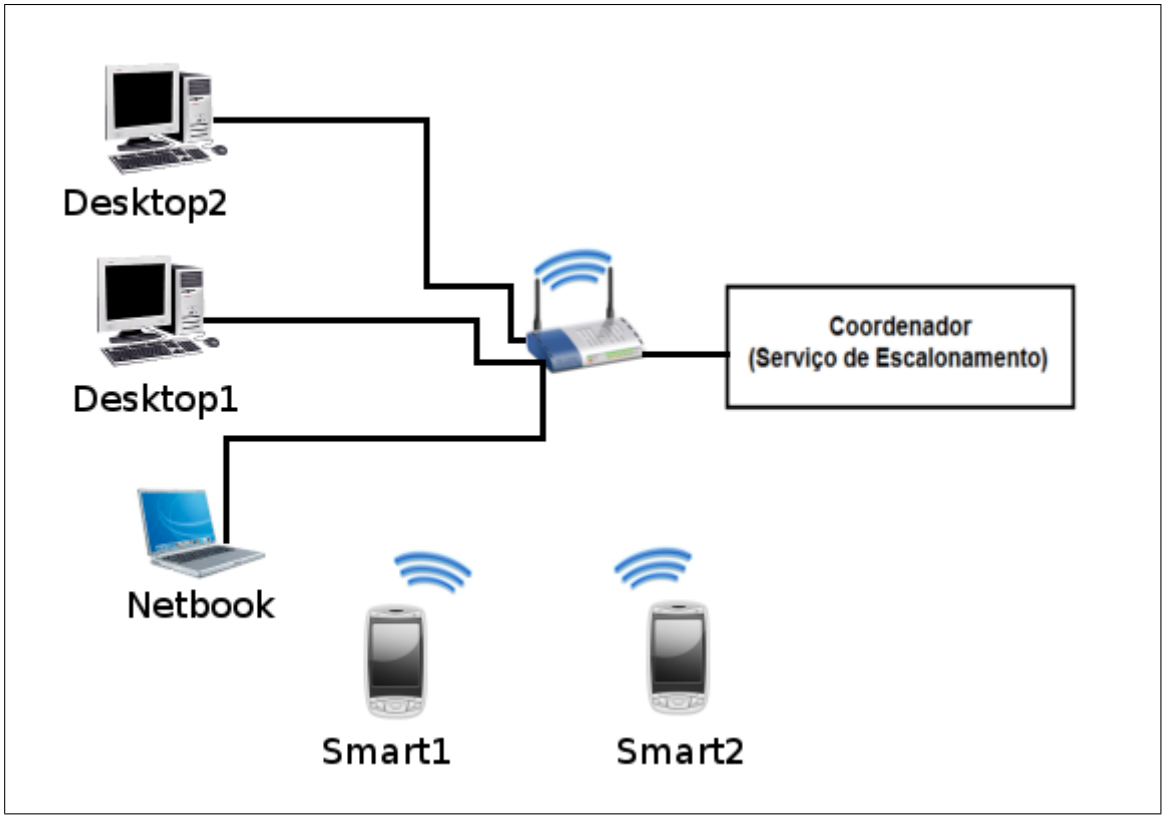

Figura 12: Estudo de caso 2 - Formação do grid

o valor do CTE representar o menor valor dentre todos os nós do grupo.

O gráfico (figura 13) apresenta os resultados.

A degradação de 147,54% no makespan da tarefa é explicada pela atribuíção das tarefas para nós que estão com mais carga, se comparados ao experimento anterior. Na tabela 3 os valores da distribuíção das tarefas são apresentados.

Devido a configuração de preferência de usuário, o número de tarefas atribuídas para

	N^o de tarefas		**Tempo de utilização (s)**	
	Cenários			
Dispositivos	**1**	**2**	**1**	**2**
Smart1	0,00	0,00	0,000	0,000
Smart2	0,00	0,00	0,000	0,000
Netbook	14,10	18,03	2,634	5,147
Desktop1	17,63	31,27	1,395	2,438
Desktop2	18,43	0,70	1,200	0,083

Tabela 3: Balanceamento de carga – Estudo de caso 2

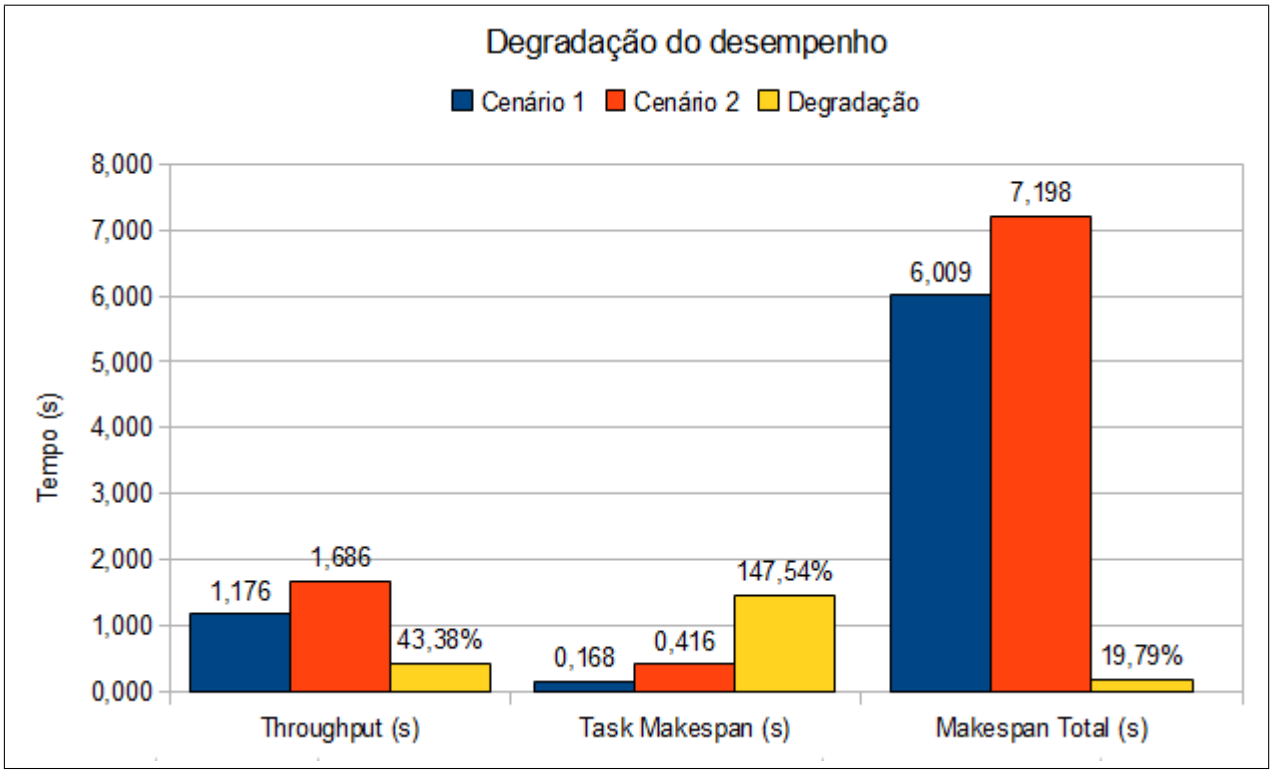

Figura 13: Gráfico – Degradação do desempenho - Estudo de caso 2

o *Desktop2* reduziu em 96,2%, o que comprova a eficácia do mecanismo.

A diferença entre os tempos de execução nos estudos de caso estão claramente associados a capacidade dos dispositivos que formam o grid.

Considerações Finais

O algoritmo PUTS utiliza as informações do nível de energia e a configuração de preferências de usuário para reduzir o tamanho das tarefas atribuídas aos nós, o que dependendo da carga do grid, reduz o número de tarefas e o tempo que o dispositivo é utilizado pelo grid.

Há dificuldades na obtenção de informações sobre o tamanho das tarefas e estado dos recursos. Tais dificuldades estão associados à complexidade de estimar o tempo de execução das tarefas e ao custo para execução dos algoritmos de benchmarks. No entanto, o custo associado à execução de benchmarks são compensados pela adaptação que o algoritmo PUTS realiza. O mesmo, reage satisfatoriamente à dinamicidade do ambiente e as mudanças de cenários.

De modo geral, a configuração de preferências de usuário, só tem impacto maior quando dois nós possuem o serviço disponível e estão em condições semelhantes de execução da tarefa, ou seja, o valor do CTE (*Completion Time Energy*) dos nós para uma determinada tarefa são semelhantes. Nestas situações o valor de UPW (*User-Preference Weight*) tem impacto decisivo para a atribuição.

Dentre os trabalhos futuros, espera-se elaborar uma escala de valores para UPW que venha facilitar a configuração pelo usuário. Além disto, realizar uma adaptação do algoritmo para escalonamento de tarefas dependentes, a inclusão da técnica de replicação de tarefas e o suporte a orquestração de serviços.

Referências

ADOcÃO do ISBT-Tb pelos países da América do Sul e Central. Disponível em: <http://www.dtv.org.br/index.php/onde-ja-tem-tv-digital/veja-aqui-os-paises-da-america-do-sul-que-ja-adotaram-o-padrao-isdb-tb/>.

ALIAGA, A. H. M. *Análise de desempenho de algoritmos de a escalonamento em simuladores de grades*. dez. 2009.

AZEVEDO, A. T. et al. Performance evaluation of a discovery and scheduling protocol for multihop ad hoc mobile grids. *J ournal of the Brazilian Computer Society*, scielo, v. 15, p. 15–29, 12 2009.

BASTOS, B. F. et al. *Interoperação de Grades Móveis Ad hoc com Grades Fixas*. 2008.

CASANOVA, H. et al. Heuristics for scheduling parameter sweep applications in grid environments. In: PUBLISHED BY THE IEEE COMPUTER SOCIETY. *Heterogeneous Computing Systems Workshop*. [S.l.], 2000. p. 349.

CASAVANT, T. L.; JON; KUHL, G. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, p. 141–154, 1988.

CIRNE, W. Computational grids: Architectures, tecnologies and applications. In: *Terceiro Workshop em Sistemas Computacionais de Alto Desempenho*. [S.l.: s.n.], 2002.

DANTAS, C.; FILHO, S. A. Um estudo sobre algoritmos de escalonamento para grids computacionais. In: *EPOCA2010*. [S.l.: s.n.], 2010.

DONG, F.; AKL, S. *Scheduling algorithms for grid computing: State of the art and open problems*. Kingston, Ontario, jan. 2006.

FILHO, S. E. A. *(Proposta) Gringa - Um middleware para computação distribuída em um ambiente de TV digital interativa*. Dissertação (Mestrado) — UERN/UFERSA - Mestrado em Ciência da Computação, fev. 2010.

FILHO, S. E. A.; ANTONIO, M.; DANTAS, C. F. F. *Gringa - Manual do Usuário*. [S.l.], out. 2010. Disponível em: <http://di.uern.br/xpta/gringa-manual.pdf>.

FOSTER, I. What is the grid? a three point checklist. *GRID today*, v. 1, n. 6, p. 32–36, 2002.

FOSTER, I.; KESSELMAN, C. The Grid: Blueprint for a New Computing Infrastructure. 1998. *Morgan-Kaufmann: San Francisco. Gailly, J.-l. and M. Adler, zlib-A Massively Spiffy Yet Delicately Unobtrusive Compression Library. Geist, AG, JA Kohl, and PM Papadopoulos, CU tolerance, Visualization, and Steering of Parallel Applications. International Journal of Supercomputer Application*, v. 11, n. 3, p. 224–35, 1997.

GOMES, A. T. A. et al. Dichotomy: A resource discovery and scheduling protocol for multihop ad hoc mobile grids. *Cluster Computing and the Grid, IEEE International Symposium on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 719–724, 2007.

GOMES, D. S.; SILVA, F. J. da Silva e; ENDLER, M. *Integrando Dispositivos Móveis ao Middleware Integrade*. 2007.

GOOGLE Groups: Desenvolvimento para TV Digital. Disponível em: <http://groups.google.com/group/devdtv?hl=pt-BR>.

GRABOWSKI, P.; LEWANDOWSKI, B.; RUSSELL, M. Access from j2me-enabled mobile devices to grid services. In: *Proceedings of Mobility Conference 2oo4, Singapore*. [S.l.: s.n.], 2004.

HUANG, C. et al. Power-aware hierarchical scheduling with respect to resource intermittence in wireless grids. In: IEEE. *Machine Learning and Cybernetics, 2006 International Conference on*. [S.l.], 2006. p. 693–698.

IBARRA, O.; KIM, C. Heuristic algorithms for scheduling independent tasks on nonidentical processors. {*J*ournal of the ACM (JACM), ACM, v. 24, n. 2, p. 280–289, 1977. ISSN 0004-5411.

IDA, C. O. *Linpack*. maio 2005. Disponível em: <http://www.inf.ufrgs.br/gppd/disc/cmp134/trabs/T1/001/benchmarks/Benchmarks-linpack.htm>.

IHSAN, I.; QADIR, M. A.; IFTIKHAR, N. Mobile ad-hoc service grid - MASGRID. In: ARDIL, C. (Ed.). *WEC (5)*. [S.l.]: Enformatika, Çanakkale, Turkey, 2005. p. 124–127.

JUNIOR, J. N. F. et al. Avaliação de algoritmos de escalonamento em grids para diferentes. configurações de ambiente. In: *WPerformance - V Workshop em Desempenho de Sistemas Computacionais e de Comunicação*. [S.l.: s.n.], 2007.

LUSZCZEK, P. et al. Introduction to the HPC challenge benchmark suite. *SC2005 (submitted), Seattle, WA*, Citeseer, 2005.

MACLIN, B. What every marketer needs to know about itv. 2001. Disponível em: <http://portal.acm.org/citation.cfm?id=1109194>.

MANVI, S.; BIRJE, M. A Review on Wireless Grid Computing. *International Journal of Computer and Electrical Engineering*, v. 2, n. 3, jun. 2010.

MARQUES, V. C. *Desenvolvimento de uma aplicação para TV Digital utilizando o Middleware nacional Ginga-NCL voltado ao aprendizado de crianças*. 2008.

MARTINS, R. B. *Sistema Brasileiro de TV Digital Metodologia para Escolha do Modelo de Referência*. 2005. Disponível em: <http://www.telebrasil.org.br/arquivos/tvdigital.pdf>.

MENASCÉ, D. A. et al. *Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures*. 1995.

MENDES, L. L. *SBTVD: Uma visão sobre a TV digital no Brasil*. V, n. 12, out. 2007.

MONTEZ, C.; BECKER, V. *TV digital interativa: conceitos, desafios e perspectivas para o Brasil*. 2005.

OLIVEIRA, B. J. D. de. *Um estudo de caso entre Ginga-J e Ginga-NCL no âmbito de aplicações interativas residentes*. jan. 2010.

PAES, A.; ANTONIAZZI, R. Padrões de Middleware para TV Digital. *CEP*, v. 24210, p. 240, 2005.

PESQUISA nacional por amostra de domicílios 2009. 2009.

PORTAL do Software Público Brasileiro - Middleware Ginga. jan. 2011. Http://www.softwarepublico.gov.br/. Disponível em: <http://www.softwarepublico.gov.br/dotlrn/clubs/ginga/>.

QUARTERONI, A. *Cálculo científico com Matlab e Octave*. [S.l.]: Springer, 2006.

REIS, V. Q. dos. *Escalonamento em grids computacionais: estudo de caso*. Dissertação (Mestrado) — Instituto de Ciências e Matemáticas e de Computação - ICMC-USP, 2005.

RODAMILANS, C. *Análise de desempenho de algoritmos de escalonamento de tarefas em grids computacionais usando simuladores*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2009.

SEDREZ, F. de M. *Desenvolvimento de um aplicativo para TV Digital Interativa utilizando a tecnologia Java TV*. 2008.

SILVA, B. F. da. *TMRorR : um novo algoritmo de escalonamento para o OurGrid que combina o uso de informação e replicação*. Dissertação (Mestrado) — Faculdade de Informática - Programa de Pós-graduação em Ciência da Computação, 2009.

SILVA, D. P. D.; CIRNE, W.; BRASILEIRO, F. V. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: *Applications on Computational Grids, in Proc of Euro-Par 2003*. [S.l.: s.n.], 2003. p. 169–180.

SOMBRA, E. L. *Utilização do Middleware Ginga NCL no desenvolvimento de Aplicações para TV Digital: Um estudo de caso sobre um comercial de Supermercado*. dez. 2009.

STOCKINGER, H. Defining the grid: a snapshot on the current view. *The {Journal of* Supercomputing, Springer, v. 42, n. 1, p. 3–17, 2007. ISSN 0920-8542.

TONIETO, M. T. *Sistema Brasileiro de TV Digital - SBTVD - Uma análise Política e Tecnológica na Inclusão Social*. Dissertação (Mestrado) — Diretoria de Pesquisa e Pós-Graduação (DIPPG). Centro Federal de Educação Tecnológica do Ceará (CEFET-CE), dez. 2006.

TSENG, L.; CHIN, Y.; WANG, S. The anatomy study of high performance task scheduling algorithm for Grid computing system. *Computer Standards & Interfaces*, Elsevier, v. 31, n. 4, p. 713–722, 2009.