Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Fabiana Fortes Rodrigues

Política de escalonamento de recursos computacionais em clusters de física de altas energias

Rio de Janeiro
2012

Fabiana Fortes Rodrigues

Política de escalonamento de recursos computacionais em clusters de física de altas energias

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Orientadores: Prof. Dr. Alexandre Sztajnberg

Prof. Dr. Alberto Franco de Sá Santoro

Rio de Janeiro

2012

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/A

R696 Rodrigues, Fabiana Fortes.

Política de escalonamento de recursos computacionais em clusters de física de altas energias/ Fabiana Fortes Rodrigues. - 2012.

96 f. : il.

Orientadores:.Alexandre Sztajnberg, Alberto Franco de Sá Santoro.

Dissertação (Mestrado) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Arquitetura de computador - Teses. 2. Processamento paralelo – Computadores - Teses. I. Sztajnberg, Alexandre. II. Santoro, Alberto Franco de Sá. III. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística. IV. Título.

CDU 004.2

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta dissertação, desde que citada a fonte.

_____ _____

Assinatura Data

Fabiana Fortes Rodrigues

Política de Escalonamento de Recursos Computacionais em Clusters de Física de Altas Energias

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro. Área de concentração: Grades Computacionais

Aprovada em 30 de Outubro de 2012.

Banca examinadora:

Prof. Dr. Alexandre Sztajnberg (Orientador)

Instituto de Matemática e Estatística – UERJ

Prof. Dr. Alberto Franco de Sá Santoro (Orientador)

Instituto de Física – UERJ

Prof. Dr. Carlos Antônio de Moura

Instituto de Matemática e Estatística – UERJ

Prof. Dr. Marcio Portes de Albuquerque

Centro Brasileiro de Pesquisas Físicas - CBPF

Rio de Janeiro

2012

DEDICATÓRIA

Dedicamos este projeto a Deus, minha família, amigos e orientadores pelo apoio, força, incentivo, companheirismo e amizade. Sem eles nada disso seria possível.

AGRADECIMENTOS

A Deus por me amparar nos momentos difíceis, me dar força interior para superar as dificuldades, mostrar os caminho nas horas incertas e me suprir em todas as minhas necessidades.

Ao meu orientador Prof Alexandre Sztajnberg, pelo apoio, incentivo e colaboração excepcionais. Por sua ajuda nos momentos mais críticos, por acreditar no futuro deste projeto e contribuir para o meu crescimento profissional. Sua participação foi fundamental para a realização deste trabalho.

Ao meu orientador Prof. Alberto Santoro por ter me aberto as portas da pós-graduação e pela oportunidade de participar do projeto do CERN, pelo seu exemplo de vida que é uma inspiração para todos que amam a ciência.

Ao Dr. Marcos Dias de Assunção um dos integrantes do projeto GridSim, pela ajuda e pelo o código que foi o ponto de partida para o desenvolvimento do *broker* neste trabalho.

À minha família, a qual amo muito, pelo carinho, paciência e incentivo.

À FAPERJ pela bolsa de mestrado, que contribuiu para a execução deste trabalho.

Aos amigos que fizeram parte desses momentos sempre me ajudando e incentivando.

À equipe da T2_HEPGRID_UERJ pela amizade e apoio neste projeto.

Aos demais professores do IME-UERJ pelo convívio e aprendizado além da força e apoio moral na realização deste projeto.

"Todo o futuro da nossa espécie, todo o governo das sociedades, toda a prosperidade moral e material das nações dependem da ciência, como a vida do homem depende do ar. Ora, a ciência é toda observação, toda exatidão, toda verificação experimental. Perceber os fenômenos, discernir as relações, comparar as analogias e as dessemelhanças, classificar as realidades, e induzir as leis, eis a ciência; eis, portanto, o alvo que a educação deve ter em mira. Espertar na inteligência nascente as faculdades cujo concurso se requer nesses processos de descobrir e assimilar a verdade."

Rui Barbosa

RESUMO

RODRIGUES, Fabiana Fortes. **Política de escalonamento de recursos computacionais em clusters de física de altas energias**. 2012. 111 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2012.

Este trabalho apresenta uma proposta para permitir o uso compartilhado dos recursos computacionais utilizados em um cluster de forma a atender simultaneamente aos quatro experimentos do CERN. A abordagem adotada utiliza o conceito de contratos, onde os requisitos e restrições de cada experimento são descritos em perfis, e uma política de alocação de recursos é definida para manter a utilização dos recursos de forma a atender aos perfis. Propomos um modelo de arquitetura para gerenciar o uso compartilhado de um cluster pelas quatro Organizações Virtuais do LHC. Este modelo de arquitetura é composto de elementos comuns a um cluster típico da *Tier-2*, acrescidos de funcionalidades para controlar a admissão de novos *jobs* de todas as Organizações Virtuais do LHC. Este modelo monitora a utilização de recursos do cluster por cada OV, aloca recursos para cada *job* de acordo com uma política estabelecida para o cluster de forma a procurar respeitar os requisitos de cada uma delas. Definimos um algoritmo para o escalonamento de *jobs*, que utiliza mecanismos de preempção para controlar a alocação de nós do cluster dependendo o uso corrente e acumulado de recursos por cada OV. Este algoritmo é executado em um dos elementos da arquitetura batizado de *broker*, dado que o mesmo intermedeia a chegada de novos *jobs* e a alocação dos nós do cluster, e tem com objetivo manter o controle dos contratos de cada OV. A simulação da arquitetura proposta foi feita no simulador de grades GridSim e os resultados obtidos foram avaliados.

Palavras-chave: Cluster. Organização Virtual. Compartilhamento de recursos.

ABSTRACT

This paper presents a proposal to allow the shared use of computing resources available in a cluster in order to support simultaneously the four experiments at CERN. The approach uses the concept of contracts, where the requirements and restrictions of each experiment are described in profiles, and a resource allocation policy is set to maintain the use of resources to meet those profiles. We propose an architecture model to manage the shared use of a cluster by the four LHC Virtual Organizations. This architectural model is composed of elements common to a typical *Tier-2* cluster plus features to control the admission of new *jobs* from all LHC Virtual Organizations. This model monitores the resource usage for each cluster VO, allocates resources to each *job* according to an established policy for the cluster in order to seek compliance with requirements of each VO. We define an algorithm for scheduling *jobs*, that uses preemption mechanisms for controlling the allocation of cluster nodes depending on the current use of resources and accumulated by each VO. This algorithm runs in one of the elements of architecture named *broker*, since it mediates the influx of new *jobs* and the allocation of the cluster nodes, and is aiming to keep control of contracts for each VO. The simulation of the proposed architecture was done in GridSim simulator grid and the results were evaluated.

Keywords: Cluster. Virtual Organization. Resource sharing

LISTA DE FIGURAS

Figura 1 - Cluster Virtualizado 24
Figura 2 - Espaços de Trabalho Virtuais[1] 26
Figura 3 - Distribuição do espaço em disco de uma T2 [14] 38
Figura 4 - Um cluster participando dos 4 experimentos do LHC 43
Figura 5 - Fluxograma do algoritmo de escalonamento 47
Figura 6 - *Jobs* processados na T2_TW_Taiwan 53
Figura 7 - *Jobs* processados na T2_IN_TIFR 53
Figura 8 - *Jobs* processados na T2_KN_KNU 54
Figura 9 - *Jobs* processados na T2_HU_Budapest 54
Figura 10 - *Jobs* processados na T2_GR_Ioannina 54
Figura 11 - *Jobs* processados na T2_US_Caltech 55
Figura 12 - *Jobs* processados na T2_US_MIT 55
Figura 13 - *Jobs* processados na T2_US_Nebraska 55
Figura 14 - *Jobs* processados na T2_US_Wisconsin 56
Figura 15 - *Jobs* processados na T2_US_UCSD 56
Figura 16 - *Jobs* processados na TW-FTT 56
Figura 17 - *Jobs* processados na GOEGRID 57
Figura 18 - *Jobs* processados na IN2P3-LPSC 57
Figura 19 - *Jobs* processados na IN2P3-LPC 57
Figura 20 - *Jobs* processados na IN2P3-LAPP 58
Figura 21 - *Jobs* processados na IN2P3-CPPM 58
Figura 22 - *Jobs* processados na TOKYO-LCG2 58
Figura 23 - *Jobs* processados na SIGNET 59
Figura 24 - *Jobs* processados na AGLT2 59
Figura 25 - *Jobs* processados na MWT2_UC 59
Figura 26 - *Jobs* processados na WT2 60
Figura 27 - Arquitetura do GridSim [28] 73
Figura 28 - Modelagem proposta 74
Figura 29 - Diagrama das classes *RunBrokerRamp*, *HEPGridPolicy*, *RampWorkload* e *WorkloadRamp* 77
Figura 30 - Rampa de *jobs* submetidos 78
Figura 31 - Diagrama das classes *Broker*, *BrokerNoPolicy* e *BrokerWLCG* 84
Figura 32 - Evolução dos elementos da arquitetura de acordo c/ a política ao longo de 50000 s 86
Figura 33 - Tempo de Espera para Execução – *Site* A Cluster I 90
Figura 34 - Tempo de Espera para Execução – *Site* A Cluster II 90
Figura 35 - Tempo de Espera para Execução – *Site* A Cluster III 91
Figura 36 - Tempo de Espera para Execução – *Site* B Cluster I 91
Figura 37 - Tempo de Espera para Execução – *Site* B Cluster II 92
Figura 38 - Tempo de Espera para Execução – *Site* B Cluster III 93

Figura 39 - Tempo Total no Cluster – *Site* A Cluster I.....96
Figura 40 - Tempo Total no Cluster – *Site* A Cluster II.....97
Figura 41 - Tempo Total no Cluster – *Site* A Cluster III.....97
Figura 42 - Tempo Total no Cluster – *Site* B Cluster I.....98
Figura 43 - Tempo Total no Cluster – *Site* B Cluster II.....98
Figura 44 - Tempo Total no Cluster – *Site* B Cluster III.....99
Figura 45 - *Jobs* Expirados – *Site* A Cluster I.....102
Figura 46 - *Jobs* Expirados – *Site* A Cluster II.....103
Figura 47 - *Jobs* Expirados – *Site* A Cluster III.....103
Figura 48 - *Jobs* Expirados – *Site* B Cluster I.....104
Figura 49 - *Jobs* Expirados – *Site* B Cluster II.....104
Figura 50 - *Jobs* Expirados – *Site* B Cluster III.....105

LISTA DE TABELAS

Tabela 1 - Disponibilidade para os serviços essenciais em uma T2 [15] 34
Tabela 2 - Capacidade computacional do CMS[16] 35
Tabela 3 - Capacidade computacional do ATLAS[16].....36
Tabela 4 - Capacidade computacional do ALICE[16].....36
Tabela 5 - Capacidade computacional do LHCb[16].....36
Tabela 6 - Capacidade de processamento do grupo A em HEPSPEC 50
Tabela 7 - Capacidade de processamento do grupo B em HEPSPEC 50
Tabela 8 - Tempo de processamento de 86% dos *jobs* 61
Tabela 9 - Duração Máxima do *Job* X Número Máximo de *Jobs* (Grupo A).....62
Tabela 10 - Porcentagem do cluster comprometida com cada OV em caso de pico (Grupo A).....62
Tabela 11 - Núcleos comprometidos com cada OV em caso de pico (Grupo A).....63
Tabela 12 - Duração Máxima do *Job* X Número Máximo de *Jobs* (Grupo B).....63
Tabela 13 - Porcentagem do cluster comprometida com cada OV em caso de pico (Grupo B).....63
Tabela 14 - Núcleos comprometidos com cada OV em caso de pico (Grupo B).....64
Tabela 15 - *Sites* com 1 único cluster e que atendem 1% do CMS.....65
Tabela 16 - *Sites* com 1 único cluster e que atendem 4% do CMS.....65
Tabela 17 - *Sites* com 1 único cluster e que atendem a 1% do ATLAS 66
Tabela 18 - *Sites* com 1 único cluster e que atendem a 5% do ATLAS 66
Tabela 19 - *Sites* com 1 único cluster e que atendem a 1% do ALICE 67
Tabela 20 - *Sites* com 1 único cluster e que atendem a 8% do ALICE 67
Tabela 21 - *Sites* com 1 único cluster e que atendem a 3%, 4% e 5% do LHCb 68
Tabela 22 - *Sites* com 1 único cluster e que atendem a 7% do LHCb 68
Tabela 23 - *Sites* com 1 único cluster e que atendem aos 4 experimentos 69
Tabela 24 - Número de núcleos por OV do grupo A 79
Tabela 25 - Número de núcleos por OV do grupo B 79
Tabela 26 - Grupo A - Perfil dos *Sites*.....80
Tabela 27 - Grupo A - Contrato *Site* I.....80
Tabela 28 - Grupo A - Contrato *Site* II.....80
Tabela 29 - Grupo A - Contrato *Site* III.....81
Tabela 30 - Grupo B - Perfil dos *Sites*.....81
Tabela 31 - Grupo B - Contrato *Site* I.....81
Tabela 32 - Grupo B- Contrato *Site* II.....82
Tabela 33 - Grupo B - Contrato *Site* III.....82
Tabela 34 - Tempo de espera para execução.....95
Tabela 35 - Tempo total no cluster.....101
Tabela 36 - *Jobs* com *proxy* expirado 106

LISTA DE ABREVIATURAS E SIGLAS

ALICE	A Large Ion Collider Experiment
AOD	Analysis Object Data
ATLAS	A Toroidal LHC Apparatus
CE	Compute Element
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
EGEE	Enabling Grid for E-sciencE
GridFTP	Grid File Transfer Protocol
GUMS	Grid User Management System
HEP	High-Energy Physics
JDL	*Job* Description File
LHC	Large Handron Collider
LHCb	Large Hadron Collider beauty
MMV	Monitor de Máquina Virtual
MoU	Memorando de Entendimento
MV	Máquina Virtual
NFS	Network File System
OSG	Open Science Grid
OV	Organização Virtual
PhEDEx	Physics Experiment Data Export
POSIX	Portable Operating System Interface,
RFIO	Remote File Input/Output protocol
SAM	*Site* Availability Monitoring
SE	Storage Element
SI2K	SpecInt2000
SPEC	Standard Performance Evaluation Corporation
SL4	Scientific Linux 4
SRM	Storage Resource Manager
T1	*Tier-1*
T2	*Tier-2*
T3	Tier-3
WLCG	Worldwide LHC Computing Grid
TFC	Trivial File Catalog
WM	CMS Workflow Management

SUMÁRIO

	INTRODUÇÃO	15
1	**CONCEITOS BÁSICOS**	20
2	**TRABALHOS RELACIONADOS**	28
3	**REQUISITOS DE UMA T2**	32
3.1	**Requisitos gerais para uma T2 do LHC**	32
3.2	**Requisitos específicos para cada OV**	37
3.2.1	CMS	37
3.2.2	ATLAS	39
3.2.3	ALICE	40
3.2.4	LHCb	40
3.3	**Demanda por armazenamento em disco**	40
4	**ARQUITETURA E POLÍTICA PROPOSTAS**	41
4.1	**Arquitetura proposta**	42
4.2	**Política proposta**	45
4.3	**Contrato e Perfis**	50
5	**SIMULAÇÃO**	70
5.1	**O Simulador GridSim**	70
5.2	**Modelagem da arquitetura no simulador**	74
5.3	**Workload em Rampa**	75
5.4	**Perfil do Site**	79
5.5	**A política**	83
5.6	**Logs**	87
6	**RESULTADOS**	88
6.1	**Tempo de Espera de Execução**	89
6.2	**Tempo Total no Cluster**	96
6.3	***Jobs* expirados**	102
6.4	**Comentários**	107
7	**CONCLUSÃO**	108
	REFERÊNCIAS	112

INTRODUÇÃO

O LHC (*Large Handron Collider*) é o maior acelerador de partículas do mundo. Encontra-se em um túnel de 27Km de circunferência, a 100m de profundidade, na fronteira entre a França e a Suíça [8]. Ele foi construído pelo CERN (*European Organization for Nuclear Research*) para dar suporte aos testes de várias previsões relacionadas à física de altas energias, incluindo a existência do bóson de Higgs, e a grande família de novas partículas previstas pela super-simetria. Dois feixes de partículas subatômicas chamadas *hádrons* viajam em direções opostas dentro do acelerador. Físicos usarão o LHC para recriar as condições logo após o *Big Bang*, colidindo os dois feixes. Grupos de físicos em todo o mundo analisarão, nos próximos anos, as partículas criadas nas colisões usando quatro detectores de partículas. Cada detector está relacionado a um experimento do LHC que são: CMS [9], ATLAS [10], ALICE [11], LHCb [12]. Esses experimentos de certa forma são competidores entre si. A metodologia e os resultados parciais obtidos por um projeto não são compartilhados com os demais.

O LHC, que já está em operação, tem a perspectiva de gerar anualmente 15 Petabytes (15 milhões de Gigabytes) de dados anualmente e milhares de cientistas ao redor do mundo irão acessar e analisar esses dados. Em uma primeira estimativa, a análise desses dados, incluindo comparação com simulações teóricas, requer aproximadamente 100.000 CPUs para todos os experimentos [34].

Para organizar a infraestrutura necessária para os quatro experimentos do LHC foi criada a WLCG (*Worldwide LHC Computing Grid*). A WLCG é uma colaboração de mais de 170 centros computacionais em 36 países cuja missão é prover recursos computacionais para o armazenamento, a distribuição e a análise dos dados gerados pelo LHC. Ela foi construída sobre infraestruturas de grades existentes: primeiramente a OSG (*Open Science Grid*) [33] nos EUA e América do Sul, e em seguida da EGEE (*Enabling Grid for E-sciencE*) na Europa e outros lugares.

A arquitetura da WLCG é subdividida em camadas (*tiers*), sendo cada camada responsável por processar e armazenar dados oriundos das camadas acima, sendo que a primeira camada processa os dados vindos do próprio detector. A arquitetura da WLCG é subdividida nas seguintes camadas:

- *Tier-0*: Os dados gerados pelas colisões nos detectores são filtrados, para que somente os eventos relevantes possam ser estudados. Nesta primeira etapa, os dados brutos emergentes do sistema de aquisição de dados de cada detector são gravados na *Tier-0* localizada no CERN. Os dados oriundos do detector depois de serem filtrados passam por um processo onde os dados brutos são reconstruídos e transformados em dados do tipo RECO (*Reconstructed Data*). A reconstrução dos dados é realizada nesta camada onde uma cópia dos dados reconstruídos é armazenada. A *Tier-0* distribui uma segunda cópia dos dados brutos para os *sites* na *Tier-1* associados com o experimento. Os dados do tipo RECO também são copiados para os *sites Tier-1*;
- *Tier-1*: Tem a responsabilidade de fazer a reconstrução dos dados e de gerenciar o armazenamento permanente de dados (brutos, reconstruídos (RECO), simulados e processados) e fornecer capacidade computacional para reprocessamento e para processos de análise que requerem acesso a uma grande quantidade de dados. Os dados do tipo RECO são processados e transformados em dados to tipo AOD (*Analysis Object Data*). Os dados do tipo AOD são utilizados pelos físicos para análise;
- *Tier-2*: O papel dos *sites* na *Tier-2* é prover capacidade computacional e serviço de armazenamento de dados apropriado para simulações de eventos de Monte Carlo e análise para usuários finais. Uma *Tier-2* obtém os dados conforme o necessário das *Tiers-1* e os dados gerados pelas *Tiers-2* são enviados para as *Tiers-1* para o armazenamento de dados permanente.

Cada *site* na WLCG representa uma instituição que possui um ou mais clusters que pertencem à grade da OSG ou da EGEE, sendo a maioria localizada na *Tier-2*. Cada *site* pode estar associado a grupos de pesquisadores de um ou mais experimentos e para isso deve fornecer a infraestrutura necessária para atender aos mesmos.

Os *sites* que participam de dois ou mais experimentos normalmente optam por ter um cluster separado e dedicado para cada um dos experimentos. Essa opção é feita geralmente porque cada experimento possui requisitos de hardware e software específicos e, adicionalmente, a participação do *site* em um dado experimento está condicionada a aderência a contratos restritivos, tais como

políticas de utilização e segurança, e disponibilidade de recursos, muitas vezes mutuamente exclusivas. Além disso, por questões estratégicas do LHC, como mencionado anteriormente, os experimentos não compartilham informações ou resultados. Assim sendo, o isolamento entre os experimentos é importante.

Ter um cluster separado para cada experimento pode resultar numa subutilização dos recursos de hardware, que precisam ser replicados para cada experimento, e na duplicação de recursos humanos, pois neste caso também existe redundância de atividades de gerência. Além disso, os participantes de um experimento não podem utilizar todos os recursos do *site* em benefício do mesmo, ainda que os clusters de outros experimentos neste *site* estejam ociosos.

O objetivo desse trabalho é apresentar uma proposta para otimizar o uso dos recursos computacionais pertencentes a um *site* típico da *Tier-2* da WLCG, permitindo que atividades de experimentos diferentes sejam executadas compartilhando recursos de um mesmo cluster e ainda assim atendendo-se todas as restrições inerentes ao experimento. A abordagem adotada utiliza o conceito de contratos, onde os requisitos e restrições de cada experimento são descritos em perfis, e uma política de alocação de recursos é definida para manter a utilização dos recursos de forma a atender aos perfis.

Diferentes níveis de qualidade podem ser definidos para oferecer ou retirar recursos dinamicamente de uma dada OV, de forma a se respeitar, por um lado, os requisitos mínimos de cada experimento em períodos de picos e, por outro, permitir que recursos extras sejam alocados a esta mesma atividade em períodos de ociosidade.

A gerência de contratos deve contemplar mecanismos para impor as restrições de segurança e as políticas de uso de cada experimento. Para isso, os recursos físicos do cluster devem ser monitorados bem como as diversas atividades dos experimentos. Desta forma será possível tomar decisões de alocação de recursos de forma consistente e, caso seja necessário utilizar os mecanismos de gerência para inicializar, suspender ou mesmo mover processos entre os recursos dentro do cluster. A gerência de contratos estende naturalmente o papel do *front-end* de um cluster típico do WLCG - que tem o papel de autenticar e gerenciar o escalonamento de *jobs* de um único experimento, monitorando e escalonando *jobs* dos diversos experimentos. De acordo com [29], cada recurso ou *site* deve negociar

com cada OV quais recursos estão disponíveis para serem usados por cada OV através de Acordos de Nível de Serviço.

Propomos um modelo de arquitetura para gerenciar o uso compartilhado de um cluster pelas quatro OVs do LHC. Este modelo de arquitetura é composto de elementos comuns a um cluster típico da *Tier-2*, acrescidos de funcionalidades para controlar a admissão de novos *jobs* de todas as OVs, monitorar a utilização de recursos do cluster por cada OV, alocar recursos para cada *job* de acordo com uma política estabelecida para o cluster de forma a procurar respeitar os requisitos de cada uma delas. Definimos um algoritmo para o escalonamento de *jobs*, que utiliza filas e mecanismos de preempção para controlar a alocação de nós do cluster dependendo o uso corrente e acumulado de recursos por cada OV. Este algoritmo é executado em um dos elementos da arquitetura batizado de *broker*, dado que o mesmo intermedeia a chegada de novos *jobs* e a alocação dos nós do cluster, e tem com objetivo manter o controle dos contratos de cada OV.

Para avaliar a proposta, implementamos os elementos mais importantes da arquitetura, incluindo o *broker* e o algoritmo de escalonamento no simulador de grades computacionais GridSim. O cluster e os contratos foram configurados e parametrizados em três cenários utilizando-se o número de *jobs* submetidos que efetivamente ocorrem nos quatro experimentos. As configurações dos três cenários foram submetidas a cargas em rampa de forma a se testar o pior caso de uso, ou seja, quando os quatro experimentos precisam do máximo de recursos possível ao mesmo tempo. Os resultados das simulações são apresentados e discutidos.

A partir dos resultados das simulações observamos que a política proposta é aplicável para se reduzir o número de *jobs* com *proxy* expirados (ou seja, menos *jobs* são abortados), nos cenários com um cluster dimensionado com 100% e 75% de recursos do total necessário para cada VO. O preço da aplicação da política é o aumento do tempo de permanência no cluster e o tempo de espera para iniciar a execução não excedeu 30% em todos os cenários. Para os cenários com clusters com número reduzido de máquinas, tipicamente 50% do total, a política não se mostrou eficiente. Houve um aumento do número de *jobs* com *proxy* expirados. Estes resultados facilitam a tomada de decisão pelo cluster que atende a 75% de recursos do total, caso haja menos recursos computacionais disponíveis visto que os resultados obtidos neste cenário foram satisfatórios. A melhor opção continua

sendo ter o cluster dimensionado com 100% de recursos do total necessário, pois este cenário obteve os melhores resultados com a política.

Organização do Texto

A dissertação está organizada da seguinte maneira: No Capítulo 1 são descritos conceitos básicos para a realização do projeto e a motivação da proposta. No Capítulo 2 temos os trabalhos relacionados com o tema desta proposta. No Capítulo 3 temos os requisitos gerais e específicos para um *site* fazer parte de uma T2 do LHC. No Capítulo 4 temos a arquitetura e política propostas, contratos e perfis e como foi feito o dimensionamento do cluster. No Capítulo 5 são descritas as características do GridSim, o simulador utilizado neste trabalho, e os detalhes da simulação de como foi feita a simulação. No Capítulo 6 temos os resultados obtidos. No Capítulo 7 temos as conclusões e possíveis trabalhos futuros para essa proposta.

1 CONCEITOS BÁSICOS

Neste capítulo apresentamos os conceitos e terminologias empregadas nas tecnologias de suporte ao WLCG. Estes conceitos e terminologias são importantes para a compreensão da proposta apresentada neste trabalho e são citados nas próximas seções.

O WLCG utiliza grades computacionais, que são sistemas distribuídos com arquiteturas bem definidas [40], com a inclusão de especificidades para atender ao LHC:

- **Grade computacional:** a idéia de uma Grade Computacional é análoga à grade de energia elétrica que provê acesso ubíquo à energia elétrica, independente do tipo de aparelho a ser ligado, sem grande necessidade de adaptação. A grade de energia elétrica disponibiliza energia elétrica sob demanda e esconde do usuário detalhes como a origem da energia e a complexidade da malha de transmissão e distribuição. Ou seja, se temos um equipamento elétrico, simplesmente o conectamos na tomada para que ele receba energia. Um componente chave para isso é a padronização de tomadas, conectores e a forma de entrega de energia. Uma grade computacional, nesta linha, seria uma rede na qual o usuário se conecta para obter recursos computacionais (ciclos, armazenamento, software, periféricos). Uma grade computacional pode ser caracterizada, então, como um conjunto heterogêneo de componentes de computadores, *clusters* de computadores e software interconectados e localizados em diversos *sites*, hospedado por diferentes organizações. A grade precisa de uma infraestrutura que realiza a integração e a coordenação de recursos, que podem ser compartilhados por usuários com perfis diferentes, para realizar processamento arbitrariamente complexo. Para isso, esta infraestrutura precisa dar suporte a requisitos e prover mecanismos para a distribuição e execução paralela e distribuída de *jobs*, segurança, política de uso, controle de acesso, compartilhamento e transferência de dados, e monitoramento de recursos, avaliando a adesão às políticas estabelecidas [35].
- **Cluster:** é o termo usado como referência, neste trabalho, a um conjunto de computadores interconectados por uma rede de alta velocidade, sistema de armazenamento local, sistema de fornecimento de energia, e refrigeração,

dentro de um domínio administrativo. Um cluster geralmente adota um sistema operacional e um conjunto de software e bibliotecas para atender ao seu perfil de usuário. Mesmo havendo heterogeneidade entre os nós de computação que compõem um cluster, a configuração e a administração dos sistemas de software e *middleware* oferecem um ambiente consistente aos seus usuários, facilitando o uso dos recursos oferecidos e a execução paralela de aplicações. Com mais detalhes, em um cluster o sistema operacional é responsável por garantir o funcionamento primordial das seguintes funções [45]:

- Controle de recursos: realizar um controle de entrada e saída de cada nó do cluster, e assim, garantir a consistência dos dados executados.
- Monitoramento: verificar a disponibilidade de cada nó do cluster, assim como a quantidade de carga que cada nó poderá receber.
- Contabilidade: funciona como um medidor de desempenho do cluster seja ela, para calcular custos da operação ou até mesmo medir desempenho.
- Gerenciamento de processos: o sistema operacional armazena em fila de prioridades todos os processos que serão executados, levando em consideração as tarefas executadas por diferentes níveis de usuários.
- Planejamento: organização da melhor maneira pela qual as tarefas serão distribuídas entre os nós do cluster.

Existem vários tipos de cluster, no entanto há alguns que são mais conhecidos, como:

- Cluster de Alto Desempenho: Utilizados para melhorar o desempenho das tarefas computacionais. Fraciona-se o processamento total entre os computadores buscando-se um melhor desempenho.
- Cluster de Alta Disponibilidade: Tem a finalidade de aumentar a disponibilidade dos serviços que estão sendo executados em conjunto. Está relacionado ao conceito de redundância, ou seja, a mesma tarefa sendo executada por outros computadores

- Cluster para Balanceamento de Carga: Utilizado para controlar a distribuição de carga entre os demais computadores, mantendo-as equilibradas. Precisa de um monitoramento constante na sua comunicação para que em caso de falha não haja uma interrupção dos serviços disponibilizados pelo cluster.

- **Organização Virtual:** é um conjunto de indivíduos ou instituições que coordenam o compartilhamento de recursos em uma grade, restringindo ou autorizando acesso direto a computadores, software, dados, e outros recursos. O acesso aos recursos computacionais é controlado pela entidade que provê o recurso para uma determinada Organização Virtual. A permissão de acesso aos recursos é definida para cada usuário através de grupos que determinam quais recursos de uma Organização Virtual ele tem acesso.

- **SPEC:** é uma organização sem fins lucrativos cujos membros incluem fornecedores de hardware, companhias de software, universidades, centros de pesquisa cujo objetivo é manter um conjunto de benchmarks padrão para sistemas computacionais [39]. Um benchmark é um programa que executa um conjunto de operações e retorna algum tipo de resultado, descrevendo o comportamento do computador testado. Os benchmarks medem a velocidade (a rapidez com que as operações completaram), o *throughput* (quantas operações por unidade de tempo foram completadas).

- **HEPSPEC:** um benchmark criado especificamente para os *sites* da WLCG. Ele foi desenvolvido após se perceber que o benchmark SpecInt2000 (SI2K) não estava mais atendendo as aplicações voltadas para física de altas energias [26]. Este benchmark é utilizado para medir a capacidade dos recursos computacionais dos *sites* da WLCG. Neste trabalho, esse benchmark é utilizado para o dimensionamento dos clusters em cada um dos cenários apresentados. O benchmark utilizado pela maioria dos processadores é o SI2K. O fator de conversão simplificado entre o HEPSPEC e o SI2K tem uma margem de erro de 5% e 1 HEPSPEC é equivalente a 250 SI2K.

Além das tecnologias tradicionalmente utilizadas nas Grades Computacionais, entre eles sistemas de *middleware* para submissão de *jobs*, existem projetos no WLCG experimentando o uso de virtualização para tornar mais

flexível o uso de clusters físicos (Capítulo 2), na mesma linha de computação em nuvem. Assim, apresentamos os conceitos principais:

- **Virtualização:** é uma técnica que permite que a visão da aplicação dos recursos computacionais seja separada da infraestrutura subjacente. A virtualização provê um mecanismo pelo qual cada usuário pode ter seu próprio ambiente computacional. No contexto do LHC, cada OV pode disponibilizar uma imagem com seu próprio ambiente de execução customizado e tais ambientes serão implantados e ativados nas máquinas físicas quando agendados, assim permitindo cada OV agendar *jobs* em seu próprio cluster privado.
- **Monitor de Máquina Virtual (MMV):** MMV ou Hypervisor é a camada do software de virtualização que permite que a plataforma hospedeira execute múltiplas máquinas virtuais onde cada máquina virtual pode rodar sua própria cópia de um sistema operacional diferente, ou múltiplas versões do mesmo sistema operacional. A política de alocação de recursos pode ser implementada na arquitetura proposta para fazer a alocação das máquinas virtuais de cada experimento. Em [4] as máquinas virtuais são alocadas de acordo com a OV do *job* que irá ser executado. Para isto, eles utilizaram o MMV para inicializar instâncias das máquinas virtuais. Cada instância de máquina virtual pode ser considerada um processo Linux que pode ser inicializado ou cancelado. Assim, uma máquina virtual inicializada com um MMV é abstraída no sistema hospedeiro como um processo. A política proposta pode ser utilizada para decidir quais instâncias de máquinas virtuais serão inicializadas. Assim, uma biblioteca de imagens de sistemas operacional, cada uma configurada especificamente para determinada aplicação, pode ser disponibilizada e, de acordo com a necessidade ser ativada dinamicamente. Desta forma, não é mais necessário dedicar uma máquina física para uma única instalação de sistema. O algoritmo proposto pode ser utilizado para inicializar ou cancelar máquinas virtuais de acordo com os limites de utilização estabelecidos para cada OV.

A integração da arquitetura de grades computacionais, com as tecnologias de virtualização empregadas em computação em nuvem, aliada à organização imposta e as políticas de uso adotadas pela WLCG pode permitir um uso mais flexível de recursos computacionais. A Figura 1 mostra como os conceitos de virtualização podem ser aplicados para o caso de clusters físicos participantes de

experimentos de HEP e ajuda a identificar outros elementos típicos de um cluster do WLCG.

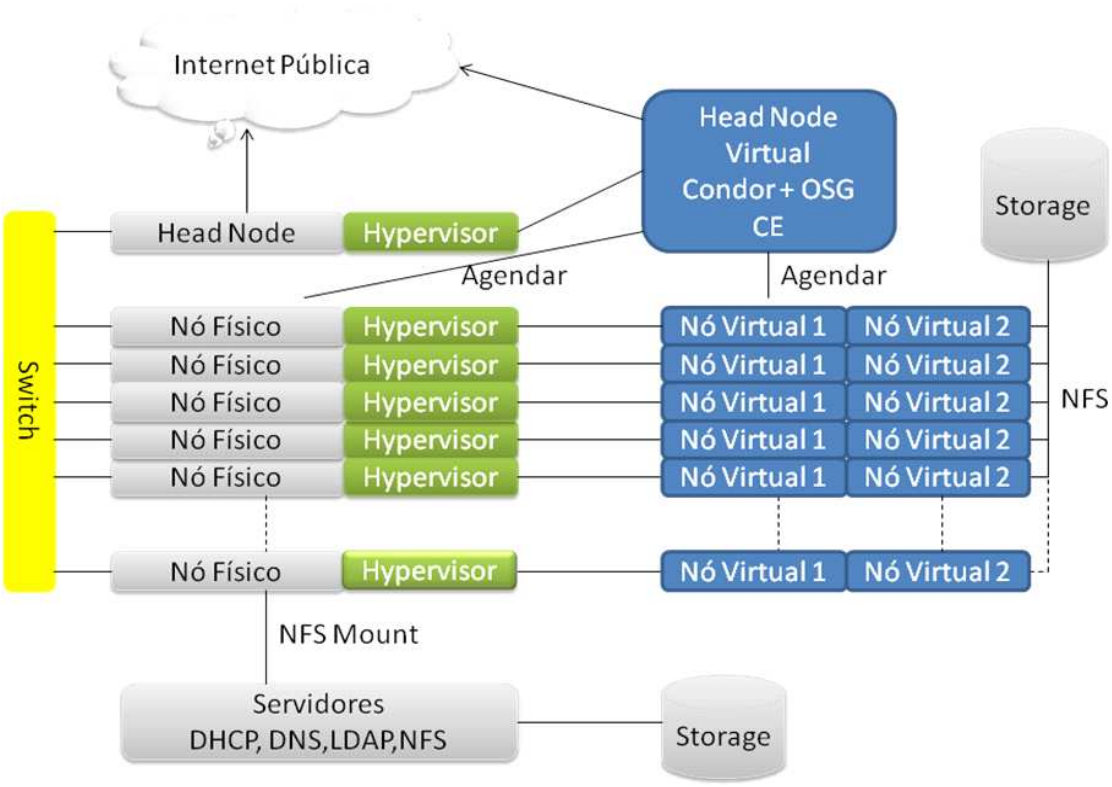

Figura 1 - Cluster Virtualizado

A abordagem é implantar uma MMV (ou *Hypervisor*) em cada nó físico do cluster e permitir que uma ou mais máquinas virtuais sejam ativadas para executar efetivamente as atividades e *jobs*. Por exemplo, é possível carregar imagens pré-configuradas com o sistema Linux e as bibliotecas e softwares específicos para cada OV.

Duas vantagens adicionais das MMVs são a sua capacidade de gerenciamento das MVs e o isolamento imposto entre as MVs hospedadas. É possível ativar, parar, inativar e migrar MVs entre hospedeiros dinamicamente dentro de um *cluster*. Na proposta apresentada em [4] as máquinas virtuais são alocadas de acordo com a OV do *job* que irá ser executado. A característica de isolamento permite, dentro dos limites de cada implementação, impor o isolamento entre as MVs, fazendo com que uma instância não tenha ciência que outras instâncias coexistem no mesmo recurso físico e não permitem o acesso direto a espaços comuns de memória ou a sistemas de armazenamento entre as mesmas.

Em nossa proposta estas características são usadas em conjunto com uma política para gerenciar o uso de recursos do cluster de forma compartilhada entre os experimentos dos quatro experimentos do LHC. Como será discutido adiante a política avaliada requer que *jobs* de uma determinada OV sejam retirados de execução temporariamente até que seja possível realocá-los novamente. Na arquitetura proposta a unidade de alocação sé uma imagem do sistema contendo um ou mais *Jobs* de uma OV.

Nesta linha, é útil a definição de um Espaço de Trabalho Virtual, que inclui a definição de um ambiente de execução em termos dos requisitos de hardware, configuração de software e propriedades de isolamento. O objetivo de definir um espaço de trabalho é capturar os requisitos para um ambiente de execução necessário aos usuários e, então, utilizar ferramentas automatizadas para achar, configurar, e prover um ambiente que melhor atende esses requisitos. Espaços de Trabalho Virtuais, assim, refinam a camada do ambiente de execução na arquitetura de um cluster: ao invés de mapear *jobs* diretamente para recursos de hardware, os *jobs* são mapeadas para espaços de trabalho pré-configurados que podem então ser mapeadas para recursos da grade.

Aprofundando um pouco mais a abordagem do uso de MVs em ambientes de grades, destacamos o exemplo da OSG, organizada em torno de Organizações Virtuais que permitem que cientistas colaborem usando diversos recursos computacionais distribuídos geograficamente. Devido à natureza diversa das OV, assim como os desafios envolvidos em prover o ambiente computacional adequado para cada OV, MV podem ser usadas como um mecanismo para construir clusters com ambientes de software homogêneos. Por exemplo, cada OV pode prover suas próprias imagens de MVs, e é possível prover a cada OV um subconjunto de máquinas físicas no cluster dedicado às MVs que executam exclusivamente seus *jobs*. Na Figura 2 temos um cluster que é subdividido em dois Espaços de Trabalho Virtuais. Cada espaço corresponde ao ambiente computacional de uma OV.

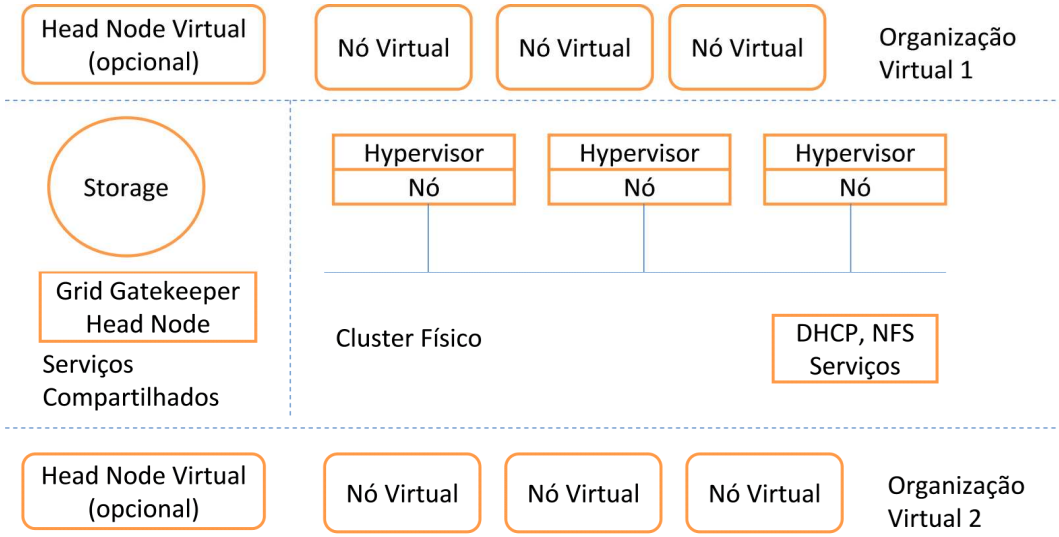

Figura 2 - Espaços de Trabalho Virtuais[1]

Tipicamente um cluster associado ao LHC que atende a uma OV é composto por diversos nós, onde os *jobs* são processados, e servidores. Os principais servidores em um cluster são os seguintes:

- CE (*Compute Element*), servidor responsável por fazer a interface entre o cluster e a grade. Os *jobs* da grade chegam ao cluster através do CE. Em uma grade computacional um cluster é representado pelo seu CE.
- SE (*Storage Element*), servidor que provê acesso uniforme aos recursos de armazenamento de dados local. Ele controla servidores de discos ou *arrays* de discos.
- GridFTP (*Grid File Transfer Protocol*), servidor responsável pela transferência de arquivos entre *sites* ou clusters diferentes. Um *job* de grade pode solicitar acesso a um arquivo remoto.
- GUMS (*Grid User Management System*), servidor que faz o mapeamento entre as credenciais de usuário da grade e as contas de usuários locais dentro do cluster.
- Fronted, servidor responsável por coletar informações, faz a negociação entre os recursos e as requisições dos usuários e submete os *jobs* para serem executados nos nós.

Durante a operação um cluster que atende a uma OV o CE recebe o pedido do usuário da grade solicitando a execução dos *jobs*. As permissões do certificado

de grade do usuário são verificadas pela GUMS e se ele estiver autorizado, ele é mapeado para uma conta local Linux que é comum a todos os nós, e então ganha acesso aos recursos do cluster local. O *job* entra na fila do escalonador onde aguarda ser redirecionado para ser executado em um nó do cluster. A partir dos requisitos pré-estabelecidos pelo usuário, como por exemplo, quantidade de memória ou versão do sistema operacional, o *job* é encaminhado para o melhor nó disponível. Dependendo das características do *job* ela poderá acessar arquivos locais ou remotos, precisando acessar o SE ou utilizar a GridFTP para transferência remota de arquivos. Em ambos os casos o certificado do usuário deverá ter autorização para realizar essas operações.

Em um cluster que atende a uma OV usando MV ocorre basicamente o mesmo que em um cluster comum. A diferença é que o ambiente de controle e execução de *jobs* e os próprios *jobs* são executado sobre máquinas virtuais e não diretamente sobre o hardware.

Como discutido no capítulo anterior, um cluster que atende a apenas uma OV pode representar desperdício de recursos computacionais. Supondo que um usuário deseje utilizar um recurso pertencente a sua OV, mas ele não está disponível. Ele poderia utilizar um recurso cuja localização geográfica seja mais próxima, mas os recursos disponíveis mais próximos pertencem a outra OV. Mesmo que este recurso esteja localizado dentro do mesmo prédio e esteja disponível, o usuário não poderá acessá-lo porque não teria permissão de acesso a esses recursos e mesmo que ele tenha autorização de acesso, provavelmente não terá o ambiente operacional customizado para atender sua demanda, isto é, dependendo do tipo da requisição ele não terá disponível o conjunto de softwares para rodar sua aplicação e poderá não ter permissão para fazer transferência de arquivos entre *sites*.

2 TRABALHOS RELACIONADOS

A utilização e o gerenciamento de recursos computacionais em grade ou cluster são abordados em diversos trabalhos, porém não encontramos um trabalho específico que aborda o assunto apresentado neste trabalho. No entanto, trabalhos que focam em pontos específicos relacionados à presente pesquisa contribuíram e inspiraram as soluções adotadas nesta dissertação.

O GRUBER [20] é um *broker* que utiliza SLAs (*Service Level Agreements*) para decidir para qual *site* o *job* será enviado para execução e o horário da execução. Cada *job* possui atributos, como por exemplo, OV, grupo dentro da OV, tempo de processamento requerido e espaço em disco requerido. Nos *sites* são definidas as SLAs que especificam o número de CPUs e a quantidade de espaço em disco que cada *site* torna disponível para as diversas OVs. As OVs definem SLAs especificando o conjunto de contribuições de todos os *sites* para esta OV que é mantida disponível para diferentes grupos dentro desta OV. No GRUBER são implementados algoritmos que detectam recursos disponíveis na grade. São detectados os *sites* disponíveis para uma dada OV. Para cada *site* disponível, é verificado se a OV ultrapassou a utilização média dentro daquele *site* conforme o acordado na SLA. Verifica-se então naquele que primeiro atender esta condição, se os picos de utilização são menores que a carga total e se este pico está de acordo com a SLA do *site*. Esta abordagem, no entanto, não considera as políticas locais de um cluster que pode ser compartilhado entre múltiplas OVs. A proposta de nosso trabalho permite que cada cluster tenha sua própria política que estipula a porcentagem dos recursos utilizados por cada OV.

O trabalho [21] aborda uma política hierárquica de alocação de *jobs* que é implementada dentro do próprio *site*. A política permite que a capacidade computacional do *site* seja dividida entre várias OVs. Dentro de cada OV os recursos podem ser subdivididos entre diversos grupos de uma mesma OV. Cada um tem uma porcentagem fixa comprometida dentro do recurso sendo que existe um algoritmo que calcula dinamicamente a prioridade de um *job* de acordo com o fluxo de submissão dos usuários. Esta variação fica próxima ao limite fixo pré-estabelecido. Esta política é eficiente nos casos onde se exige um controle mais rígido da porcentagem de recursos alocados para cada OV. A proposta de nosso

trabalho visa manter uma porcentagem fixa de recursos consumidos por cada OV nos casos em que há uma demanda por execução de *jobs* de diversas OV diferentes maior que a disponibilidade recursos. Caso o *site* esteja ocioso, uma OV poderá utilizar recursos que estavam comprometidos para outra OV. Caso a outra OV solicite seus recursos que estão sendo usados por outra, é disparado um mecanismo para fazer a preempção dos *jobs*.

O trabalho [22] menciona dentre as razões pelas quais os nós de trabalho dentro de um *site* devem ser virtualizados. Ele aborda um cluster que pode ser utilizado por diferentes comunidades. Ele menciona 3 cenários onde o primeiro se trata de um grupo de 3 clusters independentes, o segundo é um cluster compartilhado estaticamente particionado e o terceiro se trata de um cluster particionado dinamicamente, onde máquinas virtuais são alocadas e os usuários tem disponíveis diversos ambientes de execução para seus *jobs* .

Em [4] e [5], por exemplo, é apresentada a arquitetura de cluster onde o ambiente de execução dos *jobs* é carregado dinamicamente no nó, através da criação de MV nos nós de trabalho. As MV são alocadas de acordo com a OV à qual o *job* pertence. As MV são criadas a partir de um processo que monitora a fila do escalonador de *jobs*. Em ambos os trabalhos, é explicado como isso é feito para uma OV. Este trabalho se propõe a utilizar uma política de alocação de recursos onde os *jobs* são alocados para o recurso de acordo com a OV à qual ele pertence. Não abordaremos neste trabalho a aplicação desta política em ambientes virtualizados, mas trataremos esta questão em trabalhos futuros.

O trabalho [29] apresenta uma política de uso para o compartilhamento de recursos computacionais entre diversas OVs. Ele apresenta 3 cenários possíveis de compartilhamento de recursos. Ele aborda como essas políticas afetam o agendamento de recursos para a OV e para o recurso, neste caso o próprio *site*. No primeiro cenário, é estabelecido um limite superior rígido dos recursos disponíveis para cada OV. Uma nova submissão é aceita somente se este limite não foi ultrapassado. No segundo cenário o limite superior também é estabelecido, porém ele é aplicado apenas em caso de disputa por recursos. No terceiro cenário são estabelecidos dois limites de utilização de recursos. O primeiro é que eles chamam de limite de época e o segundo de limite de explosão. Um *job* é admitido somente se a média de utilização para esta OV é menor que o limite de época ou se existem

recursos disponíveis e a média de utilização de recursos para esta OV é menor que o limite de explosão para esta OV.

O algoritmo proposto no presente trabalho (Capítulo 4) foi inspirado em [29]. Ele consiste na utilização do segundo cenário acrescido de um limite de tempo de processamento total para cada OV. Cada OV tem uma cota diária de tempo de ocupação de CPU. Além disso, cada OV tem um número máximo de CPUs que podem ser alocadas simultaneamente para seu experimento com o objetivo de impedir que o cluster seja todo ocupado por uma única OV em detrimento das outras. Caso o cluster não esteja completamente ocupado e não haja novas requisições, este limite pode ser ultrapassado.

O projeto *CernVM Virtual Software Appliance* desenvolve e mantém uma única aplicação de software virtual como uma plataforma comum para os quatro experimentos do LHC [2]. A *CernVM* pode ser usada como interface de usuário para testar, desenvolver código e submeter *jobs*, e como ambiente de hospedagem de *jobs* equivalente aos tradicionais nós de trabalho em um sistema batch. Na *CernVM* cada experimento possui um repositório com todo o software necessário para execução dos *jobs*. Este software é transferido quando solicitado e é armazenado em um servidor local durante o primeiro acesso. A abordagem da *CernVM* é interessante para uma futura aplicação da política proposta no presente trabalho em um cluster real.

O uso de virtualização vem sendo estudado em HEP (*High Energy Physics*), embora não empregado em larga escala, principalmente porque seu uso ainda não foi completamente homologado pelo WLCG. O trabalho [6] apresenta os resultados de um benchmarking do software do ATLAS em uma série de cenários de virtualização para determinar todas as vantagens possíveis dessa abordagem.

O *Nimbus* é um projeto que fornece um toolkit de computação em nuvem voltado para fins científicos. No trabalho [7] é mostrado como recursos de computação em nuvem podem ser usados no framework do *AliEn Grid*, desenvolvido pelo ALICE, para realização de simulação, reconstrução e análise de dados dos físicos. Foi implantado o software virtual desenvolvido pelo projeto *CernVM* nos recursos que usam o *Nimbus* para permitir a implantação de máquinas virtuais nos recursos remotos.

O estudo dos trabalhos relacionados acima foi usado como inspiração em vários pontos da proposta da arquitetura e na formulação da política de alocação de recursos apresentada no Capitulo 4.

3 REQUISITOS DE UMA T2

No caso do LHC, além do compartilhamento de recursos entre atividades de diferentes experimentos, ou seja, diferentes OVs, uma questão importante a ser abordada é a qualidade de serviço que deve ser fornecida às OVs que usarão o cluster. Um *site* que deseja pertencer a uma OV relacionada a um dos experimentos do LHC deve atender aos requisitos mínimos de hardware, software e rede impostos por esta OV. O cluster que deseja pertencer as quatro OVs do LHC deverá, em princípio, atender aos requisitos mínimos de todas elas.

Para gerenciar os requisitos e as políticas de acesso dos quatro experimentos, o uso de contratos torna-se interessante. O uso de contratos garante, por exemplo, que os *jobs* de uma única OV não ocupem todos os nós do cluster, pois se surgir uma demanda de outra OV repentinamente ela poderia não ser atendida. Supondo que tivéssemos um cluster com 400 nós físicos e que atendesse aos quatro experimentos. Se em um dado momento tivesse no cluster uma grande quantidade de *jobs* pertencentes a uma única OV, esses *jobs* poderiam ocupar o cluster inteiro. Se chegarem *jobs* das outras três OV eles devem ter a garantia de que serão executados, pelo menos em um volume mínimo.

Assim sendo, é necessário entender quais são os requisitos do LHC e os requisitos e demandas de cada um dos quatro experimentos. Estes requisitos devem ser padronizados de alguma forma para serem comparados e para serem usados como parâmetros das políticas, gerenciando os contratos.

3.1 Requisitos gerais para uma T2 do LHC

A adesão de uma instituição ou *site* à grade computacional do LHC é regulamentada através de um Memorando de Entendimento (MoU) [15], que estabelece os objetivos, direitos e deveres dos membros da colaboração. Este memorando define os níveis de recursos computacionais que cada membro irá oferecer aos experimentos do LHC, e como cada instituição deve respeitar os critérios mínimos de níveis de serviço especificados para *Tier-1* ou *Tier-2*. Através dele, as necessidades computacionais são previstas, bem como devem ser ampliadas com o tempo. Cada instituição deve comprometer determinada

capacidade de processamento, rede e armazenamento de dados para uso colaborativo com o LHC.

De acordo com o parágrafo 4.4 do MoU “se, por qualquer motivo, os níveis dos recursos computacionais comprometidos por uma instituição para um dado experimento do LHC não estiverem sendo plenamente utilizados [....] ela é encorajada a disponibilizar uma parte ou totalidade dos níveis dos recursos computacionais em questão a um ou mais outros experimentos do LHC que ela suporta”. Assim, a presente proposta adere aos requisitos estabelecidos pelo CERN.

Os recursos computacionais efetivamente utilizados por uma instituição são monitorados pelos administradores da colaboração WLCG e a cada ano, os recursos comprometidos são avaliados de forma a se verificar se são suficientes para atender aos pedidos submetidos por um ou mais experimentos. Caso não sejam, novas contribuições serão procuradas.

Cada *site Tier-2* deve ser capaz de contribuir, pelo menos, com uma pequena porcentagem dos recursos necessários dos experimentos do LHC à que elas servem. De acordo com [16] a quantidade de recursos necessária para uma dada OV pode variar anualmente de acordo com uma avaliação realizada periodicamente.

Os serviços a serem prestados por cada *site* são: fornecimento de armazenamento em disco gerenciado, proporcionando armazenamento permanente e/ou temporário de dados para arquivos e bases de dados; fornecimento de acesso aos dados armazenados por outros centros da WLCG; operação de um centro de análise para o usuário final; prestação de outros serviços, como por exemplo, simulação de acordo com os requisitos acordados com os experimentos; largura de banda e serviços para a troca de dados com os centros *Tier-1*. A Tabela 1 mostra os requisitos de disponibilidade para uma T2 do LHC e seus parâmetros definem os níveis mínimos de serviço.

Tabela 1 - Disponibilidade para os serviços essenciais em uma T2 [15]

Serviço	**Prazo máximo para responder a problemas operacionais**		**Disponibilidade média anual**
	Horário nobre	Outros períodos	
Ambiente de suporte à execução de análises de usuário	2 horas	72 horas	95%
Outros serviços	12 horas	72 horas	95%

Baseado em [29] podemos dizer que o relacionamento entre um *site* e as respectivas OVs que ele atende são governados por SLAs. No caso do LHC podemos dizer que esta SLA corresponde à quantidade de recursos computacionais acordada entre o *site* e os respectivos experimentos. Além disso, cada *site* deve manter seus recursos computacionais disponíveis. A disponibilidade dos serviços de cada *site* é calculada através de um algoritmo que coleta os resultados de *jobs* de testes implantados dentro de cada *site* como parte da infraestrutura do WLCG.

Cada experimento tem uma meta anual de consumo de CPU e de espaço em disco que deve ser satisfeita pelo conjunto de *sites* Tiers1 e Tiers2 ao redor do mundo. Cada *site* deve atender a uma porcentagem mínima de cada experimento que ele atende. Na tabela abaixo temos a meta anual de consumo total calculada para cada OV. Todos os *sites* em conjunto devem ser capazes de atender as metas anuais de cada experimento. Por exemplo, a soma da capacidade de processamento de todas as T2 do CMS no ano de 2012 deve ser de 315.000 HEPSPEC. A capacidade de armazenamento de dados para todas as T2 do CMS no ano de 2012 deve ser de 26 PB. Nas Tabelas 2, 3, 4 e 5 temos a capacidade computacional total exigida para cada experimento em termos de CPU e espaço em disco. Essa capacidade foi estabelecida para os anos de 2010 a 2013. Os parâmetros representados nas tabelas correspondem a:

- CERN CPU (kHS06) – capacidade de processamento esperada para a camada T0;
- CERN disk (PB) – capacidade de armazenamento em disco esperada para a camada T0;
- CERN tape (PB) – capacidade de armazenamento em fita esperada para a camada T0;
- T1 CPU (kHS06) – capacidade de processamento esperada para a camada T1;
- T1 disk (PB) – capacidade de armazenamento em disco esperada para a camada T1;
- T1 tape (PB) – capacidade de armazenamento em fita esperada para a camada T1;
- T2 CPU (kHS06) – capacidade de processamento esperada para a camada T2;
- T2 disk (PB) – capacidade de armazenamento em disco esperada para a camada T2;

Tabela 2 - Capacidade computacional do CMS[16]

CMS	2010	2011	2012	2013
CERN CPU (kHS06)	96,6	106,1	120	120
CERN disk (PB)	4,1	4,5	5,5	4,5
CERN tape (PB)	14,6	21,6	23	23
T1 CPU (kHS06)	100,5	150,7	145	145
T1 disk (PB)	13,4	19,5	22	27
T1 tape (PB)	23,3	52,4	51	59
T2 CPU (kHS06)	195	319,5	315	315
T2 disk (PB)	9,2	19,9	26	26

Tabela 3 - Capacidade computacional do ATLAS[16]

ATLAS	2010	2011	2012	2013
CERN CPU (kHS06)	67	75	73	73
CERN disk (PB)	3,9	7	9	10
CERN tape (PB)	8,9	12,2	18	18
T1 CPU (kHS06)	192	226	259	273
T1 disk (PB)	21,9	24,8	27	30
T1 tape (PB)	14,2	30,1	36	40
T2 CPU (kHS06)	240	278	266	289
T2 disk (PB)	20,9	37,6	47	53

Tabela 4 - Capacidade computacional do ALICE[16]

ALICE	2010	2011	2012	2013
CERN CPU (kHS06)	46,8	62	116	116
CERN disk (PB)	5,5	6,1	14,3	14,1
CERN tape (PB)	6,3	6,8	20	25
T1 CPU (kHS06)	57,6	117	160	157
T1 disk (PB)	10,8	7,9	10,8	8,7
T1 tape (PB)	16,3	13	21	28
T2 CPU (kHS06)	89,6	121	145	142
T2 disk (PB)	12,6	6,6	8,3	8,3

Tabela 5 - Capacidade computacional do LHCb[16]

LHCb	2010	2011	2012	2013
CERN CPU (kHS06)	23	21	34	33
CERN disk (PB)	1,29	1,5	3,5	4
CERN tape (PB)	1,8	2,5	6	8
T1 CPU (kHS06)	44	65	113	110
T1 disk (PB)	3,29	3,5	9,5	11,1
T1 tape (PB)	2,4	3,47	6	8
T2 CPU (kHS06)	38	36	43	43
T2 disk (PB)	0,02	0,02	0,02	0,02

3.2 Requisitos específicos para cada OV

Os quatro experimentos possuem características semelhantes no que diz respeito à obtenção dos dados e sua distribuição entre os experimentos. Segundo [17], o primeiro processamento dos eventos obtidos pelos detectores ocorre nas instalações da *Tier-0*. Os dados brutos são arquivados no CERN e copiados, juntamente com os primeiros dados processados, para as instalações das *Tiers-1* ao redor do mundo. Estas instalações arquivam os dados brutos, provêem a capacidade de reprocessamento e a análise programada dos dados processados por grupos de física. Os *datasets* produzidos pelos grupos de física são copiados para as instalações *Tier-2* para análises futuras. As *Tiers-2* provêm capacidade de simulação para os experimentos com os dados simulados alojados nas *Tiers-1*.

Na submissão de *jobs* nos experimentos [17] o usuário deve prover uma JDL (*job-description-file*) onde todas as informações sobre o *job* estão especificadas. Isto inclui o arquivo executável a ser executado e seus parâmetros, os arquivos a serem organizados no nó de execução, os requisitos e as preferências para o nó de execução (e.g. memória disponível, tempo máximo de CPU concedido para um *job* e etc.) e os arquivos para serem na saída.

3.2.1 CMS

O CMS *Workflow Management* (WM) gerencia o processamento de dados em larga escala permitindo ao usuário final fazer análises dos dados. Ele suporta reconstrução de dados, atividades de calibração, produção de Monte Carlo, produção AOD e análises. No CMS os *jobs* são direcionados para os *sites* de acordo com a localização dos dados que serão alocados.

A colaboração do CMS utiliza duas ferramentas para que a transferência e o acesso aos dados sejam feitos de forma eficiente, o PhEDEx e o FroNTier. O PhEDEx provê a localização de dados e o sistema de transferência de arquivos para o CMS. O PhEDEx gerencia e otimiza transferência de dados conectando os *sites* através do SRM e usando o serviço FTS para agendá-las. Um conjunto de agentes independentes cuida da transferência de dados, verificação da consistência dos dados, remoção de dados e monitoramento da transferência.

No CMS cada *Tier-2* está associada a três ou mais grupos de análises físicas

ou de detectores. O número de grupos de física associados a uma *Tier-2* vai depender da capacidade computacional disponível (processamento, armazenamento e rede). Um número restrito de pessoas em cada um desses grupos é responsável por decidir quais *datasets* podem ser armazenados no espaço associado ao grupo. O PhEDEx mantém controle da propriedade dos dados nesta área tornando mais fácil o uso correto dos dados no *site*. O espaço em disco local é dedicado à comunidade dos físicos locais e cada *Tier-2* tem suas próprias regras. Cada *site* possui seu próprio *Data Manager* que é responsável por analisar as transferências de dados ou solicitações de exclusão e, segundo os compromissos do *site* e a situação do armazenamento local, aprová-las ou negá-las. Por exemplo, o administrador do *site* pode solicitar a exclusão de um arquivo que não está sendo usado pelos usuários locais desde que exista uma cópia deste arquivo em outro *site*. O administrador deve garantir também que a distribuição do espaço em disco, de acordo com a Figura 3, seja respeitada. No CMS cada usuário está associado a uma *Tier-2* baseado em sua localização e proximidade geográfica.

A título de ilustração, a distribuição de espaço em disco em uma *Tier-2* com 200 TB é definida de acordo com a Figura 1.

Figura 3 - Distribuição do espaço em disco de uma T2 [14]

Uma *Tier-2* do CMS deve ter configurado em seu cluster os sistemas de *middleware* da OSG, da EGEE ou ARC. Os *jobs* executados pela OV CMS nos *sites* precisam também do framework de software de simulação, reconstrução e análise instalados no cluster. A instalação desses pacotes é centralizada e feita através de

jobs específicos da grade. Os *sites* devem prover mecanismos para garantir a disponibilidade de software para todos os nós de trabalho no cluster.

Dado que uma instalação única de *job* por versão rodará em apenas um nó, o *site* deve certificar-se de que a instalação foi propagada para o resto do cluster. A configuração usual define uma área de instalação de software que é compartilhada entre todos os nós dentro da *Tier-2* (por exemplo, via NFS). Enquanto os dados são transferidos entre *sites* usando SRM, outros protocolos são utilizados pelo *job* para acessar dados locais como, por exemplo, POSIX, RFIO e dCache. Isto é configurado em cada *site* através do TFC (*Trivial File Catalog*) e um arquivo XML mapeando os nomes lógicos dos arquivos para os nomes físicos dos arquivos. O CMSSW, software do CMS utilizado para execução dos *jobs*, olha no TFC para saber onde estão os dados locais e como obtê-los. O monitoramento do *site* é feito através da análise de *jobs* de testes chamados de *JobRobot* que são submetidos ao cluster para a avaliação de desempenho do *site* e do SAM (*Site Availability Monitoring*) que realiza diversos testes na infra-estrutura do *site* e classifica os diversos *sites* de acordo com sua disponibilidade. O CMS *dashboard* provê ferramentas para checar o status dos *jobs* de análise e de produção e o nível de atividades em qualquer *site* do CMS. O PhEDEx provê mecanismos para monitorar a taxa, o volume e a qualidade das transferências, os erros detectados e a razão destes erros, latência e etc.

3.2.2 ATLAS

O detector ATLAS produz em operação uma quantidade de dados equivalente a mais de 1 petabyte por segundo. Esses dados são filtrados de forma que apenas os eventos interessantes são armazenados para futuras análises. A grade é utilizada para a reconstrução destes eventos. Segundo [17] a simulação de eventos, reconstrução e análise, é feita pelo framework Athena. No ATLAS, a maior parte dos *jobs* são de produção e de análise (análise de usuários e de grupos de física) e eles são processados usando-se duas ferramentas desenvolvidas para o experimento: PanDA para produção distribuída e análise distribuída e Ganga para análise de usuário distribuída.

No ATLAS a transferência e a localização de arquivos são feitas através do sistema de gerenciamento de dados distribuídos chamado de DQ2 (DonGuijote 2) [17]. O DQ2 consiste em três componentes principais: um catálogo de réplicas

uniforme, um serviço de transferência de arquivos confiável utilizando banco de dados MySQL para gerenciar transferências usando gridftp e SRM e uma ferramenta de clientes em Python para procura e replicação. Os SEs dos *sites* do ATLAS apresentam a interface SRM com um sistema de armazenamento como o dCache por exemplo. O conjunto de softwares é instalado no *site* em uma área compartilhada através de *jobs* de grid.

3.2.3 ALICE

O ALICE possui um conjunto de softwares para a simulação (AliRoot), análise (ROOT), autenticação na grade (AliEn).

Para um *site* fazer parte da colaboração do ALICE é necessário em termos hardware ter no mínimo processadores Pentium III 2GHz , 1024 MB de RAM, kernel do linux de no mínimo 2.4 e qualquer versão do linux em arquitetura i386, ia64 ou Opteron.

Durante a pesquisa deste trabalho, não foram encontrados outros dados relevantes relacionados à OV ALICE.

3.2.4 LHCb

Durante a pesquisa não foram encontrados dados específicos para a OV LHCb. Observamos que os todos os *sites* que atendem ao LHCb atendem simultaneamente outras OVs do LHC e utilizam a infraestrutura dessas OVs.

3.3 Demanda por armazenamento em disco

Na grade da WLCG os *jobs* dos experimentos podem ser classificadas em *jobs* de análise ou de produção. Os *jobs* de análise precisam ter acesso aos dados espalhados pela grade e os de produção, precisam publicar os resultados obtidos na grade. A maioria dos *sites* que pertencem a uma OV disponibiliza espaço em disco para o armazenamento de dados de sua OV. No caso de múltiplas OV em um único *site* seria interessante ter um mecanismo de armazenamento de dados que seja comum a todos os experimentos. Assim, no caso do LHC, o armazenamento poderia ser compartilhado em quatro discos virtuais e cada disco teria seu acesso protegido de usuários pertencentes a outras OV. É de extrema importância que os dados não sejam compartilhados entre OV. Neste trabalho não abordamos este aspecto, mas é importante considerá-lo em trabalhos futuros.

4 ARQUITETURA E POLÍTICA PROPOSTAS

Os desafios para obter o compartilhamento de clusters entre as quatro OVs respeitando-se as limitações e requisitos de cada uma delas podem ser resumidos nos seguintes pontos:

- Deve ser observada alguma ociosidade de hardware nos clusters pretendentes a “hospedar” atividades dos quatro experimentos, enquanto eles executam atividades de apenas uma ou duas OVs, por exemplo.

- Os recursos de hardware serão compartilhados, mas o isolamento deve ser garantido, como se os *jobs* de cada experimento estivessem efetivamente executando em clusters fisicamente separados. Dados em memória e em disco não devem ser compartilhados.

- Requisitos mínimos de disponibilidade e quantidade de recursos devem ser garantidos para os *jobs* de cada OV. Por outro lado, sempre que houver disponibilidade de recursos, o uso dos mesmos por uma dada OV deve ser facilitado.

Neste capítulo a solução proposta é apresentada. Esta solução inclui uma arquitetura, baseada no uso de máquinas virtuais, que torna possível a administração das quatro OVs em um mesmo cluster, bem como a alocação, o controle e a gerência de *jobs* das quatro OVs no cluster de máquinas físicas. Os mecanismos de gerência de máquinas virtuais, tais como a paralisação e a migração de MVs, são utilizado para permitir liberação de recursos, sem prejuízo dos *jobs* executados por uma OV quando a mesma estiver usando recursos excedentes do cluster físico.

Além da arquitetura, é proposta uma política para a alocação de *jobs* aos recursos do cluster que leva em consideração os contratos de cada OV e procura conciliar os perfis de cada uma delas enquanto tenta garantir que os requisitos mínimos sejam respeitados.

Para descrever e dimensionar os perfis de cada OV e parametrizar o algoritmo da política de alocação de recursos um estudo de vários *sites* dos quatro experimentos e registros da utilização dos mesmos foi também realizada.

4.1 Arquitetura proposta

A Figura 4 apresenta a arquitetura proposta, onde um cluster virtualizado pode executar *jobs* dos quatro experimentos do LHC. Temos os nós do cluster, onde as MVs serão inicializadas e os *jobs* serão executados. A arquitetura é composta dos seguintes elementos:

- Repositório local, com as imagens das máquinas virtuais de cada experimento - A partir deste repositório, o MMV inicializa a imagem correspondente à OV do *job* escalado para executar e ocupar recursos no nó designado;
- Computing Element (CE), o nó responsável por receber as requisições dos experimentos, autenticar o usuário de cada experimento, monitorar suas credenciais e interagir com o módulo de gerência de contratos;
- Contratos, registro onde são estabelecidos os limites de uso do cluster segundo o perfil de cada OV, usados como parâmetros da política de alocação de recursos;
- Gerente de Contratos, elemento que administra os limites estabelecidos pelo contrato de acordo com a demanda de execução de novos *jobs*. Neste módulo este incluído o *broker*, que recebe informações do sistema de monitoramento e aciona o algoritmo que implementa a política de alocação de recurso do cluster. Ele interage com o sistema de provisionamento de máquinas virtuais e os sistemas de *middleware* da instalação para efetivamente alocar e iniciar a execução dos *jobs* selecionados.
- Provisionamento de máquinas virtuais, módulo responsável por gerenciar as máquinas virtuais. Neste módulo podem ser usadas MMVs de prateleira, ou sistemas configurados especificamente para o LHC, como o CernVM, discutido no Capítulo 2, que inclui mecanismos para a atualização da imagem do sistema de cada OV com os sistemas de software e bibliotecas necessárias para a execução de *jobs*.

Na arquitetura proposta, quando um novo *job* de grade chega ao *site*, ele é inicialmente encaminhado para o Computing Element (CE), onde é feita a autenticação do usuário, suas credenciais são avaliadas e os temporizadores relacionados a esta credencial são ativados. Em seguida o *job* é encaminhado ao módulo de Gerenciador de Contratos.

O Gerenciador de Contratos controla a submissão dos *jobs* de uma OV para um recurso baseado no estado atual de utilização do cluster e de acordo com os contratos, conforme citado anteriormente. A gerência de contratos foi desenvolvida com o objetivo de minimizar latências e preservar todos os requisitos de cada contrato.

Destacamos ainda que como cada *site* pode ajustar os contratos de acordo com características específicas, indicando como os requisitos mínimos são atendidos (por exemplo, reservando recursos para cada experimento), e como o serviço de gerência atua em situações de ociosidade, por um lado, e em situações de aumento de demanda que ocorrem dinamicamente (desalocando recursos-extra de alocados a um experimento e "devolvendo" o mesmo para o experimento cuja demanda aumentou).

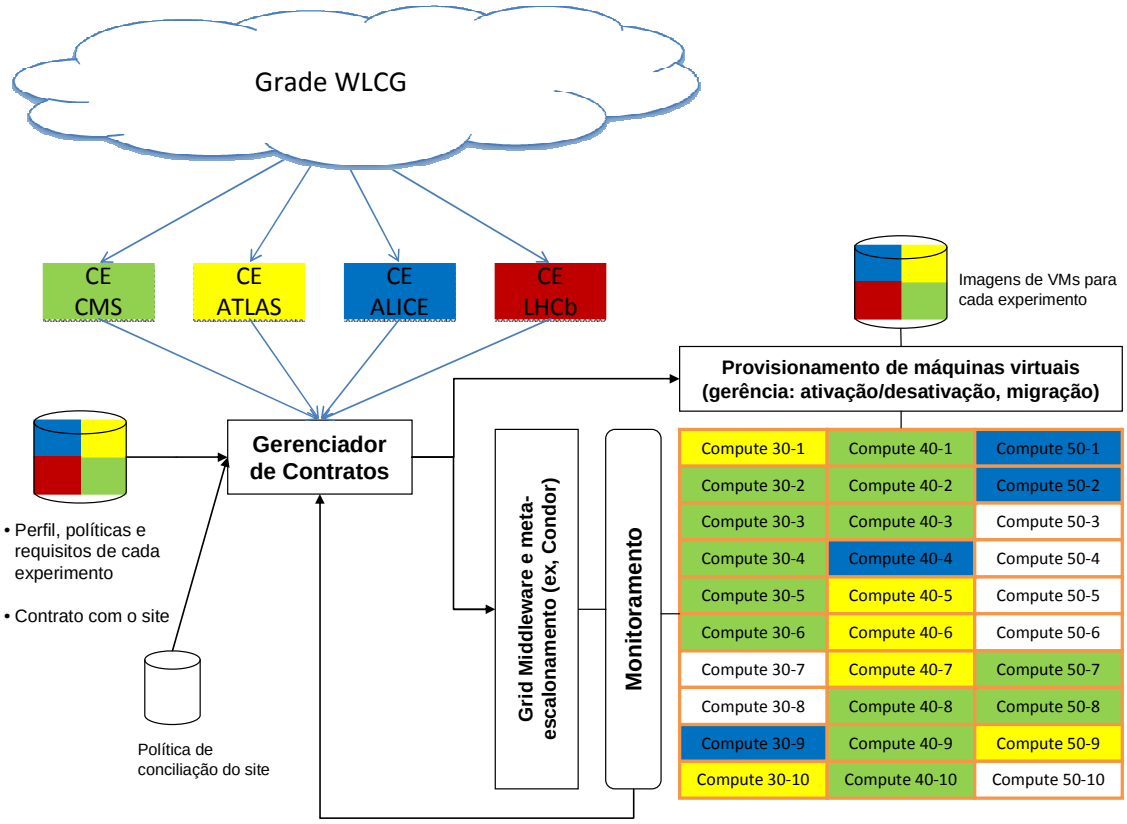

Figura 4 - Um cluster participando dos 4 experimentos do LHC

Dentro do Gerenciador de Contratos, um *broker* local, que não deve ser confundido com o *broker* disponível no *Tier-1*, é o responsável por receber *jobs* de usuários já credenciados pelo respectivo CE e alocar recursos.

Um *broker* típico da *Tier-1* é responsável por selecionar o *site* da *Tier-2* mais adequado para uma nova carga de *jobs* submetida, atuando em um nível acima de um *broker* local. Este *broker* seleciona o melhor *site*, ou recursos de um *site*, para execução de um *job* baseado em parâmetros informados pelo usuário no momento da submissão, como por exemplo, o sistema operacional, localização de arquivos e a disponibilidade do *site*, que é avaliada a cada hora através de *jobs* de testes que verificam o desempenho de cada serviço dentro do *site*.

O *broker* proposto para a arquitetura atua como um escalonador-despachante localizado dentro do *site* e atua sobre os *jobs* já admitidos e credenciados no cluster. Assim que um *job* chega, ele é encaminhado para uma fila de entrada e imediatamente submetido à política implementada no *broker*. A política avalia vários quesitos, de acordo com a OV ao qual o *job* pertence, e decide se vai ser possível ou não encaminhar este *job* para ser executado nos nós virtuais do cluster.

A política implementada pelo algoritmo de alocação de recursos verifica os limites estabelecidos pelos contratos dos diversos experimentos para decidir se o *job* será alocado a uma máquina virtual. Caso a decisão seja positiva, o mecanismo de provisionamento de máquinas virtuais verifica no repositório local qual imagem de máquina virtual será carregada. A MV é então inicializada em algum nó disponível no cluster e os sistemas de *middleware* são acionados para enviar e inicializar o *job* no sistema recém-carregado.

Caso não existam recursos livres para a criação de novas máquinas virtuais, o algoritmo de alocação de recursos verifica qual *job* poderá ser suspenso para ceder recursos para a execução do *job* em questão, caso este seja de uma OV com maior prioridade no contexto atual. A prioridade de cada OV é avaliada de acordo com os perfis estabelecidos nos contratos e o estado atual de utilização de cada OV obtida com a ajuda o sistema de monitoramento. Se o *job* em questão não é associado a uma OV com prioridade no momento, este é movido para uma fila de espera onde aguarda a próxima oportunidade para ser processado. Detalhes da política serão discutidos na próxima seção.

4.2 Política proposta

A política de alocação de recursos proposta permite que várias OVs compartilhem um mesmo cluster físico. Em um dado instante cada OV pode estar associada a um subconjunto de máquinas físicas deste cluster, que pode variar de acordo com a carga de *jobs* submetida, com a disponibilidade de recursos do cluster, com o histórico e uso atual que outras OVs estão fazendo do cluster físico e ao contrato estabelecido e o perfil de cada uma das quatro OVs. O *broker* local aloca *jobs* para um determinado nó de acordo com os parâmetros de utilização de recursos estabelecidos para cada organização virtual dentro do *site*.

Um contrato, assim como os perfis de cada OV, pode ser descrito com base nas informações de restrição e SLAs de cada experimento. As informações destas descrições são usadas como parâmetros do algoritmo que implementa a política de alocação de recursos. No contexto de nossa proposta não desenvolvemos uma meta-descrição para os contratos nem desenvolvemos rotinas para mapear meta-descrições diretamente em parâmetros usados pelo algoritmo, pois não era parte do escopo. Exemplos de soluções para gerenciamento de contratos podem ser avaliados em [41]

Os contratos para a arquitetura proposta foram elaborados baseados em parâmetros como o número máximo de CPUs ocupadas por cada experimento ($C_{\max_{ov}}$), em porcentagem, e o tempo de processamento total sem que ocorra preempção de *jobs* ($T_{\max_{ov}}$), de cada experimento.

Cada OV tem um tempo de processamento máximo $T_{\max_{ov}}$. Quando o valor do $T_{\max_{ov}}$ é ultrapassado, a política de alocação de processos executada no *broker* adota medidas de preempção e realocação de processos de acordo com as prioridades de cada OV.

Para cada um dos experimentos foi atribuído o tempo de 1 hora. Cada *job* tem um tempo de processamento $t_{p_{ov}}$. O tempo total de processamento, $T_{total_{ov}}$, corresponde ao somatório dos tempos de processamento de todos os *jobs* para cada OV. Onde *n* corresponde ao número de *jobs*

$$T_{total_{ov}} = \sum_{j=1}^{n} t_{p_{ov}} \qquad (1)$$

Em um cluster com todas as CPUs ocupadas e com demanda para novas execuções, se existir um *job* em execução de uma OV que ultrapassou seu $T_{\max_{ov}}$, ou seja, $T_{\max_{ov}} \leq T_{total_{ov}}$, um *job* desta OV pode ser suspenso para que o *job* de outra OV que não tenha ultrapassado o limite seja executado.

Cada OV possui um número máximo de CPUs, $C_{\max_{ov}}$, que podem ser ocupadas. Caso o cluster esteja cheio, o número de CPUs ocupadas $C_{busy_{ov}}$ para uma determinada OV deve ser menor ou igual ao número máximo permitido $C_{busy_{ov}} \leq C_{\max_{ov}}$ para esta OV. Caso o cluster esteja vazio, o número máximo de CPUs ocupadas por uma determinada OV pode ser ultrapassado.

Cada *job* deve ser executado dentro de um prazo limite que corresponde à duração do *proxy* do usuário da grade. De acordo com [32] o *proxy* geralmente tem a duração de 12 horas. Um *proxy* é um certificado temporário que permite a autenticação de um usuário para a utilização dos serviços de uma grade.

A política acionada dentro do *broker* deve ser imposta de alguma forma. É possível prever filas e módulos genéricos com filtros que seriam utilizados de forma específica em cada *site*. Para isso seria necessário prover mecanismos de meta-programação onde as políticas e o uso das filas seriam descritos ou extraídos a partir da descrição dos contratos e, através de mecanismos de integração, como programação generativa [36] ou AOP [42], um algoritmo seria sintetizado. Porém, tal estudo também está fora do escopo da presente proposta, embora seja um aspecto importante para que a mesma possa ser considerada para uso efetivo.

Na arquitetura proposta, a política de alocação de recursos é implementada através de um algoritmo específico que utiliza os parâmetros discutidos anteriormente. Na Figura 5 temos o fluxograma do algoritmo de escalonamento descrito e discutido a seguir.

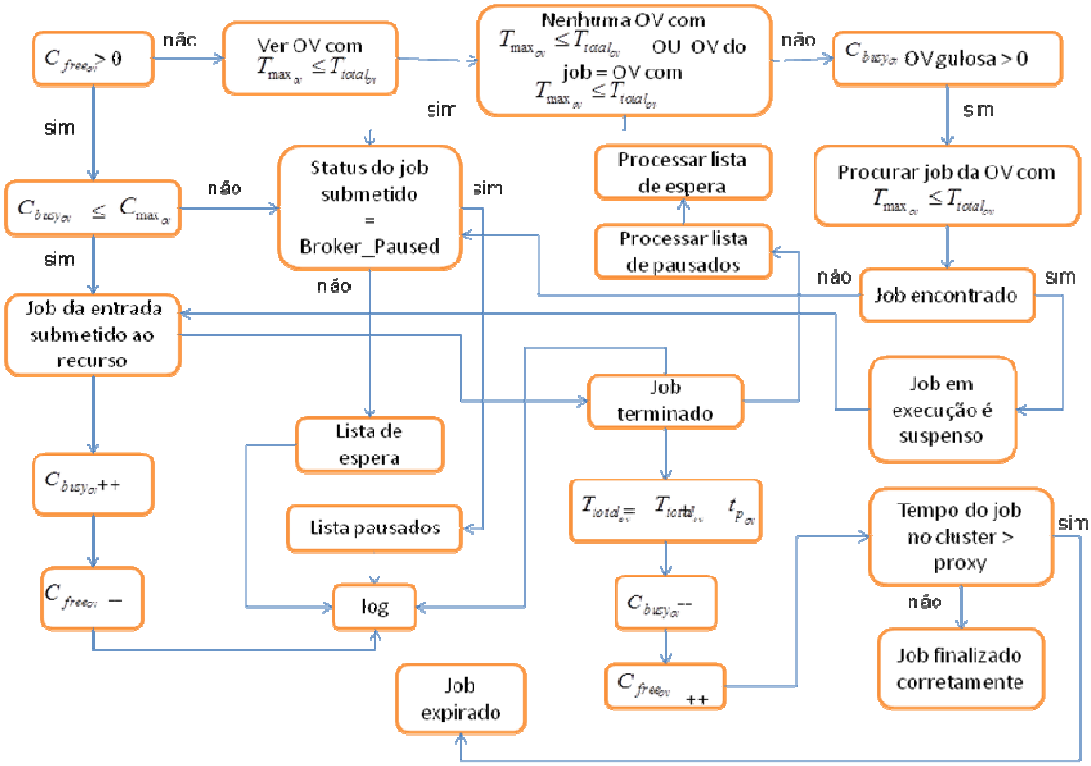

Figura 5 - Fluxograma do algoritmo de escalonamento

Observa-se que cada *job* J recebido pelo *broker* é encaminhado para este algoritmo e que durante sua permanência no *broker* (ou durante o seu regresso) ele pode ser enfileirado em uma lista de *jobs* em espera (ou seja, que ainda não tiveram sua primeira oportunidade de execução no *cluster*) ou em uma lista de *jobs* pausados (para onde são encaminhados os *jobs* que sofrem preempção para que a OV aos quais estão ligados libere recursos).

Os *jobs* que são pausados no *broker* têm seu status alterado para *BROKER_PAUSED.* Uma vez tendo sido pausado, ele fica marcado com esse status até ser processado. Os *jobs* são encaminhados para a fila de espera quando eles não podem ser processados e quando o status for diferente de *BROKER_PAUSED.* A fila de *jobs* pausados tem maior prioridade em relação à fila de espera. Segue abaixo o pseudocódigo do fluxograma da Figura 5.

```
IF ( C_free_ov > 0)
{
        IF ( C_busy_ov ≤ C_max_ov )
        {
                Submeter job para o recurso
                Jobsubmetido = TRUE
                C_busy_ov ++
                C_free_ov --
```

```
9           }
10          ELSE
11          {
12                IF ( J_status == BROKER_PAUSED)
13                      Adicionar job na lista de pausados
14                ELSE
15                      Adicionar job na lista de espera
16          }
17    }
18    ELSE   // nenhuma CPU está livre
19    {
20          V_lim= VO_QueMaisUltrapassouLimite( T_total )
21          IF ( (VO_UltrapassouLimite == FALSE) OR (V_lim==  J_vo))
22          {
23                IF ( J_status == BROKER_PAUSED)
24                      Adicionar job na lista de pausados
25                ELSE
26                      Adicionar job na lista de espera
27          }
28          ELSE
29          {
30                IF(C_busy [V_lim]  > 0)
31                {
32                      Procurar um job em execução da VO gulosa
33                      IF (JobEncontrado)
34                      {
35                            Cancelar JobEncontrado
36                            J_status = BROKER_PAUSED
37                            Adicionar  JobEncontrado na lista de pausados
38                            t_p  = GetJobProcessingTime()
39                            T_total  [V_lim] =  T_total  [V_lim] + t_p
40                            C_busy  [V_lim]--
41                            C_free ++
42                            Submeter job para o recurso
43                            Jobsubmetido = TRUE
44                            C_busy  [J_vo]++
45                            C_free --
46                      }
47                      ELSE // Job não foi encontrado
48                      {
49                            IF ( J_status == BROKER_PAUSED)
50                                  Adicionar job na lista de pausados
51                            ELSE
52                                  Adicionar job na lista de espera
53                      }
54                }
55          }
56    }
```

Na linha 1, a política verifica se existe alguma CPU livre no cluster. Se existir uma CPU livre no cluster, é averiguado na linha 3 se esta OV ultrapassou o limite de CPUs em uso estabelecido para ela. Se esta OV estiver utilizando menos CPUs que o máximo estabelecido, o *job* é submetido para o recurso na linha 5 e começa a ser processado. Nas linhas 6, 7 e 8 é feita a contabilidade das CPUs em uso. Caso tenha ultrapassado o limite para a OV, o *job* é direcionado para uma das filas dentro do *broker*.

A decisão de encaminhar o *job* para uma das filas vai depender do status atual do *job*. Se este *job* já foi pausado anteriormente (linha 12), ele será encaminhado para a fila de pausados (linha 13). Caso contrário, o *job* será encaminhado para a fila de espera (linha 15). Se não existirem CPUs livres no cluster (linha 18), a política verifica qual OV mais ultrapassou o limite de tempo de processamento estabelecido para cada OV (linha 20). Se alguma OV ultrapassou o limite e o *job* que aguarda ser executado não pertence a esta OV (linha 20), verifica-se se o número de *jobs* em execução da OV que ultrapassou o limite é maior que zero (linha 30). Se for maior que zero, procura-se um *job* da OV que ultrapassou o limite em execução (linha 32). Se o *job* da OV que mais ultrapassou o limite for encontrado (linha 33), ele é suspenso (linha 35), em seguida ele tem o status alterado para *BROKER_PAUSED* (linha 36), e encaminhado para a fila de pausados (linha 37). É obtido o tempo de processamento do *job* que foi pausado (linha 38) e este tempo é somada ao tempo total de processamento da OV à qual o *job* pertence (linha 39). O outro *job* passa a ser executado (linha 42). Nas linhas 40, 41, 44 e 45 é feita a contabilidade das CPUs em uso. Se não existir *jobs* da OV que mais ultrapassou o limite em execução (linha 47), o *job* que está passando pela política é direcionado para uma das filas do *broker* dependendo do valor de seu status.

Toda vez que um *job* acaba de ser processado no cluster, é verificado se existe algum *job* para ser processado na fila de pausados. Se a fila de pausados estiver vazia, então os *jobs* da fila de espera passam a ser processados. Se as duas filas estiverem vazias, o *broker* aguarda novos *jobs* vindos da grade para serem processados.

4.3 Contrato e Perfis

Para avaliarmos se a política implementada pelo algoritmo proposto consegue atender a contratos estabelecidos em situações diferentes, foram criados cenários diferentes. Estes cenários se diferem entre si nos requisitos de capacidade computacional e comprometimento com cada experimento. Primeiro, consideramos dois grupos de *sites* do LHC que chamaremos de grupo A e grupo B, de pequeno e de médio porte respectivamente. O grupo A é composto por clusters que atendem a 1% de cada experimento. O grupo B é composto por clusters que atendem a 5% de cada experimento. Para cada um destes 2 grupos, definimos 3 perfis de clusters físicos. Cada um dos 3 perfis de cluster possui a capacidade total de processamento expressa em HEPSPEC. O primeiro cluster do grupo A, é composto pela soma do HEPSPEC correspondente a 100% de cada experimento. O segundo cluster tem a capacidade correspondente a 75% do primeiro cluster e o terceiro corresponde a 50% do primeiro. O mesmo dimensionamento foi realizado para os clusters do grupo B. O dimensionamento dos clusters de cada grupo foi feito baseado no HEPSPEC comprometido com cada experimento no ano de 2012 [16].

Nas Tabelas 6 e 7 temos a capacidade de processamento para os clusters dos grupos A e B respectivamente.

Tabela 6 - Capacidade de processamento do grupo A em HEPSPEC

Cluster	CMS	ATLAS	ALICE	LHCb	Total
I	3150	2660	1450	430	7690
II	2362,5	1995	1087,5	322,5	5767,5
III	1575	1330	725	215	3845

Tabela 7 - Capacidade de processamento do grupo B em HEPSPEC

Cluster	CMS	ATLAS	ALICE	LHCb	Total
I	15750	13300	7250	2150	38450
II	11812,5	9975	5437,5	1612,5	28837,5
III	7875	6650	3625	1075	19225

Neste trabalho, a capacidade de processamento dos clusters dos dois grupos é dimensionada através de um HEPSPEC compartilhado, onde cada OV tem garantida sua exigência mínima que é definida em [16], mas que ao mesmo tempo é compartilhada entre as demais OVs.

A avaliação dos 3 clusters de cada grupo permite analisar como a política proposta se comporta em 3 cenários diferentes, sendo o primeiro correspondente à abundância de recursos (100% dos recursos necessários), o segundo com menos recursos (75% do necessário) e o terceiro com recursos escassos (50% do necessário).

Cada um desses cenários possui características específicas, como por exemplo, o número total de processadores, sistema operacional, OVs atendidas pelo *site* e etc. O conjunto de todas estas características forma o que chamaremos de perfil do *site*. Estabelecemos que o perfil do *site* é o conjunto das principais características de um *site*. O perfil do *site* é composto pelo seguinte conjunto de informações:

- Número total de CPUs no cluster - C_n;
- Número de OVs atendidas pelo cluster - V_n;
- Velocidade de processamento - PeRating;
- Sistema operacional presente no cluster – SO;
- SPEC ou HEPSPEC total presente no *site*;

Abaixo temos o exemplo do perfil de um *site*:

Perfil *Site*

- C_n= 823
- V_n = 4
- PeRating = 9270
- SO = Linux
- SPEC_total = 1922500
- HEPSPEC_total = 7690

Além de possuir um perfil com suas características, cada *site* possui um contrato para cada OV que ele atende. Nestes contratos são definidos como os recursos do *site* serão utilizados por cada OV. Nos contratos são especificados os limites máximos de utilização dos recursos de um *site* por uma determinada OV.

Nos contratos são estabelecidos os seguintes limites de utilização de recursos:

- Número máximo de CPUs ocupadas em caso de pico - C_{max};
- Tempo de processamento garantido sem preempção - T_{max};
- HEPSPEC da VO;

Abaixo, temos um exemplo de contrato que é definido para cada OV dentro do *site* cujos recursos são compartilhados entre si. Cada OV possui um contrato individual.

Contrato VO1

- C_{max} = 50
- T_{max} = 7200
- HEPSPEC_VO1 = 1200

O cálculo dos limites de utilização presentes no contrato foi baseado em dois parâmetros. Um deles foi o número de *jobs* submetidos para cada OV e o outro foi o cálculo da faixa do tempo de processamento dos *jobs* das 4 OVs.

A atribuição de valores do número de *jobs* submetidos para compor o perfil das OVs foi feita baseado nos valores obtidos em *sites* reais classificados em 3 categorias. A primeira é formada por *sites* de pequeno porte que atendem a um único experimento, que atendem entre 1% (CMS, ATLAS e ALICE) e 3% (LHCb) de seus experimentos. A segunda é formada por *sites* de médio porte que também atendem a um único experimento cuja participação varia entre 4% e 8%.

De acordo com [16] existem 5 federações no CMS, 6 federações no ATLAS e 3 federações no ALICE que atendem a 1% dos requisitos de CPU exigido por seus experimentos. Para o LHCb foram encontrados 3 sites que atendem a 3%, 4% e 5 % do exigido respectivamente. Cada federação possui apenas 1 *site*. Para cada *site* foi obtido o número de *jobs* submetidos por hora no intervalo de uma semana no período de 26/03/2012 à 03/04/2012. Esses valores foram encontrados somente

para os *sites* do CMS[30] e do ATLAS[31]. Foram encontrados gráficos de submissão para as outras OVs em [38], porém não foi possível extrair o as informações contidas nos gráficos para fazer o levantamento do número de *jobs* submetidos por hora. Os gráficos da Figura 5 à Figura 25 foram gerados a partir das planilhas de submissão obtidas em [30] e [31]. Para este trabalho, consideramos o valor no número de *jobs* submetidos para o ALICE sendo igual ao do CMS e o número de submissões do LHCb foi considerada igual ao do ATLAS.

A avaliação dos dados e gráficos obtidos em [16] permitiu confirmar que o *cluster* da maioria dos *sites* monitorados apresenta períodos de ociosidade. Assim sendo, confirmamos a motivação da proposta de uso compartilhado dos clusters por OVs distintas.

Nas Figuras 6, Figura 7, Figura 8, Figura 9 e Figura 10, temos os gráficos de submissão por hora dos *sites* presentes na Tabela 15. Eles correspondem aos *sites* que atendem a 5% do requerido pelo CMS.

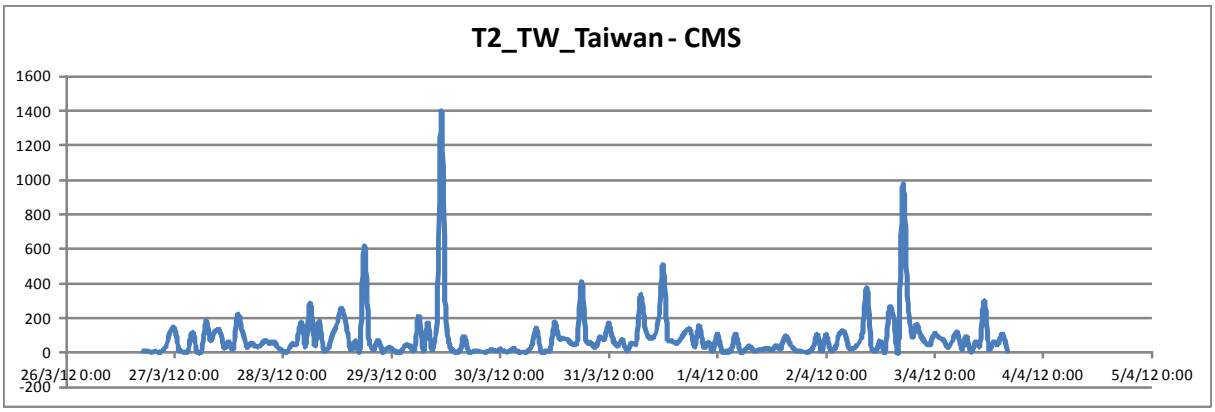

Figura 6 - *Jobs* processados na T2_TW_Taiwan

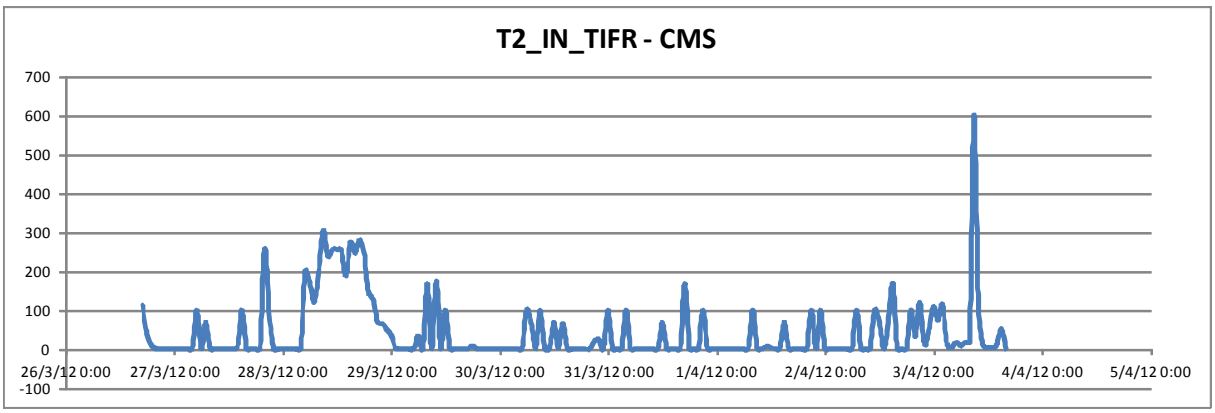

Figura 7 - *Jobs* processados na T2_IN_TIFR

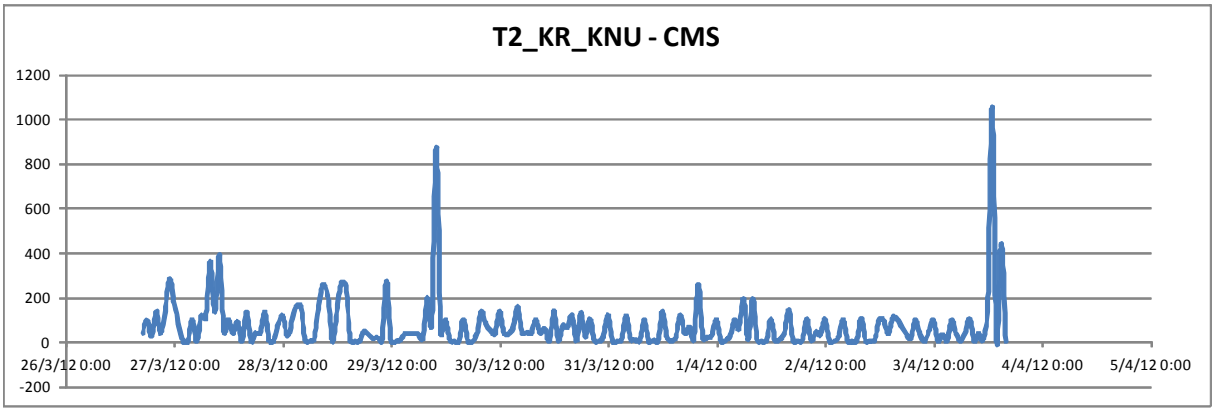

Figura 8 - *Jobs* processados na T2_KN_KNU

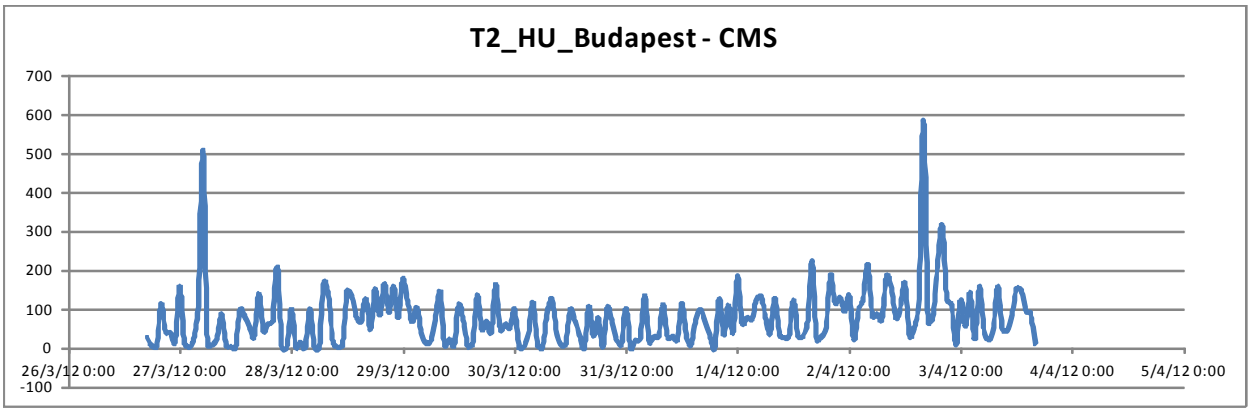

Figura 9 - *Jobs* processados na T2_HU_Budapest

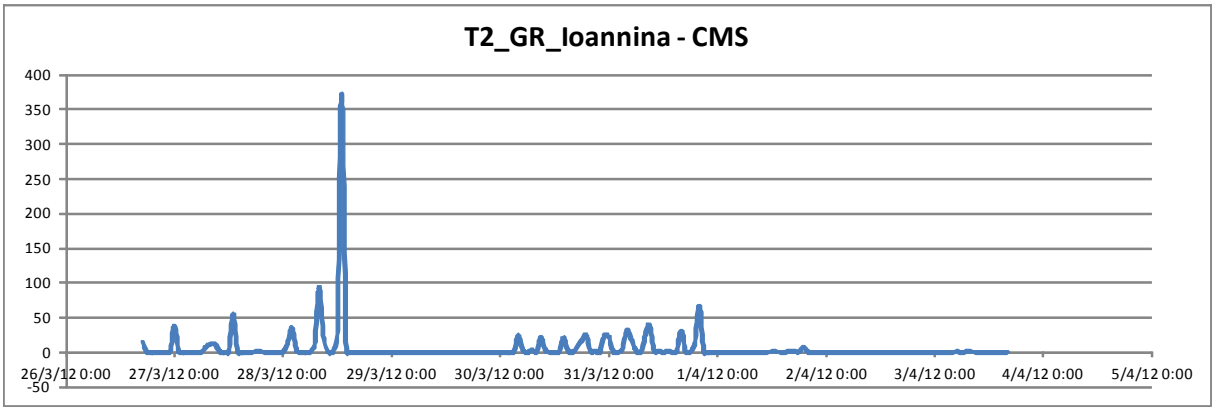

Figura 10 - *Jobs* processados na T2_GR_Ioannina

Nas Figuras 11, Figura 12, Figura 13, Figura 14 e Figura 15, temos os gráficos de submissão por hora dos *sites* presentes na Tabela 16. Eles correspondem aos *sites* que atendem a 5% do requerido pelo CMS.

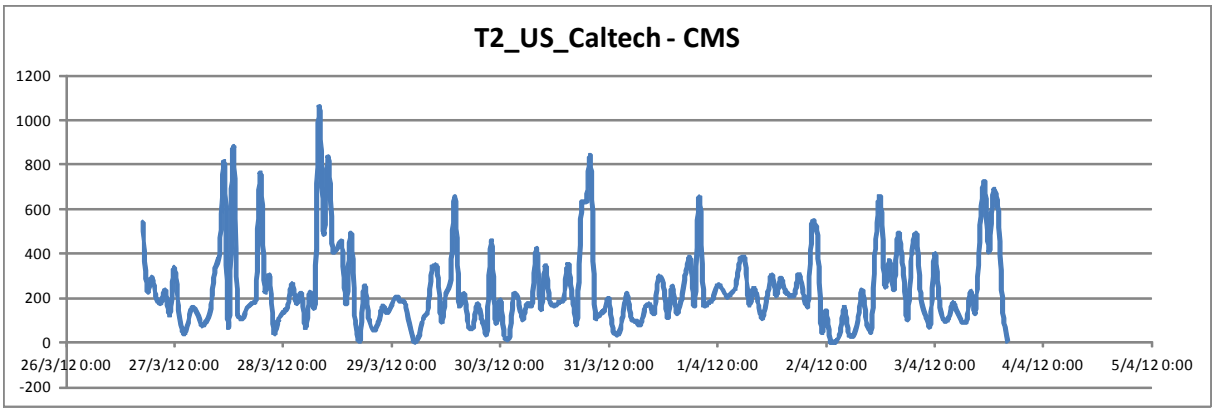

Figura 11 - *Jobs* processados na T2_US_Caltech

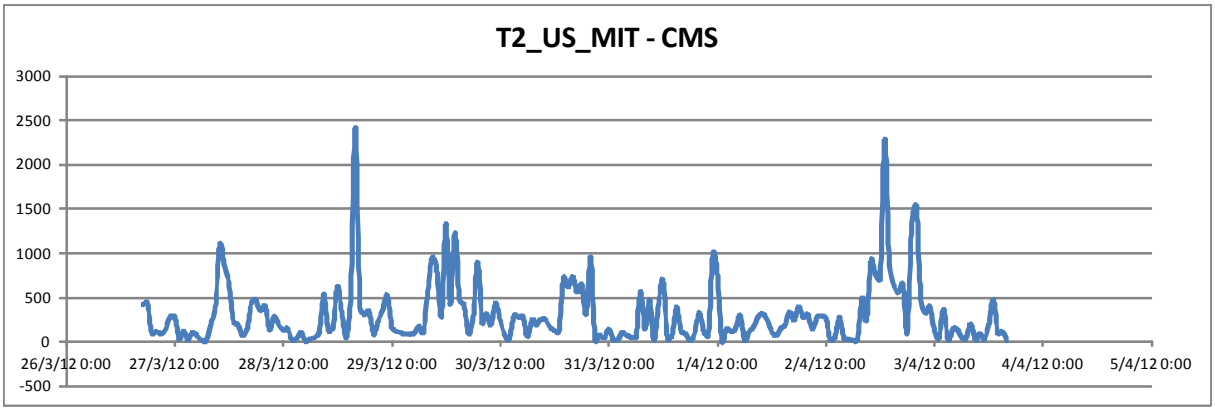

Figura 12 - *Jobs* processados na T2_US_MIT

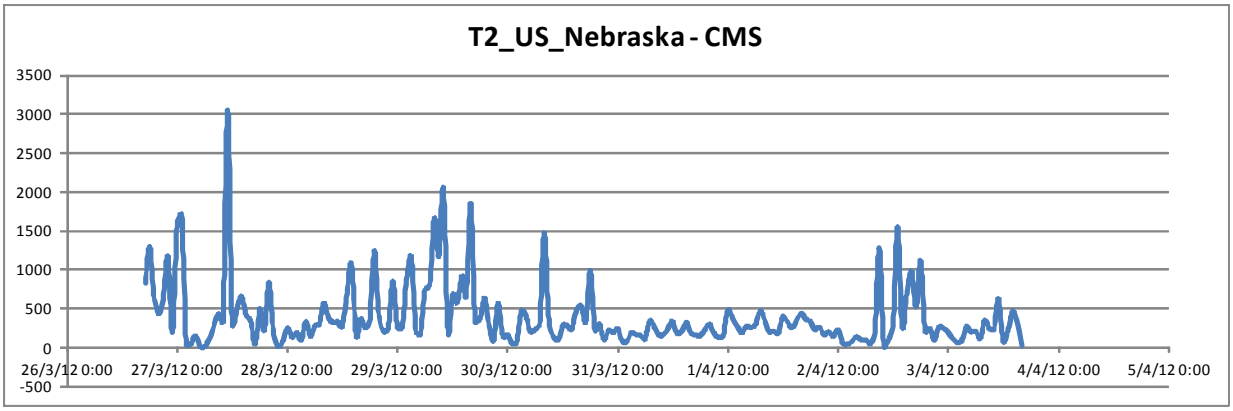

Figura 13 - *Jobs* processados na T2_US_Nebraska

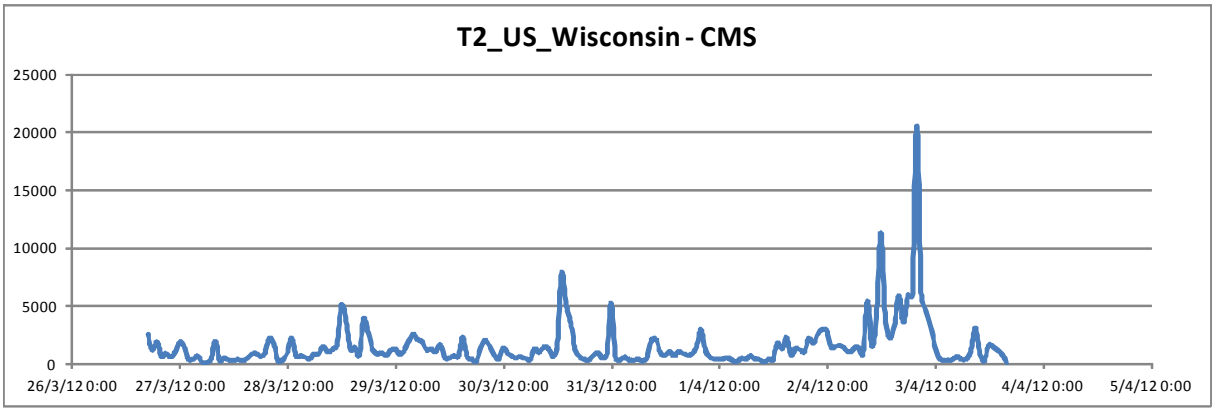

Figura 14 - *Jobs* processados na T2_US_Wisconsin

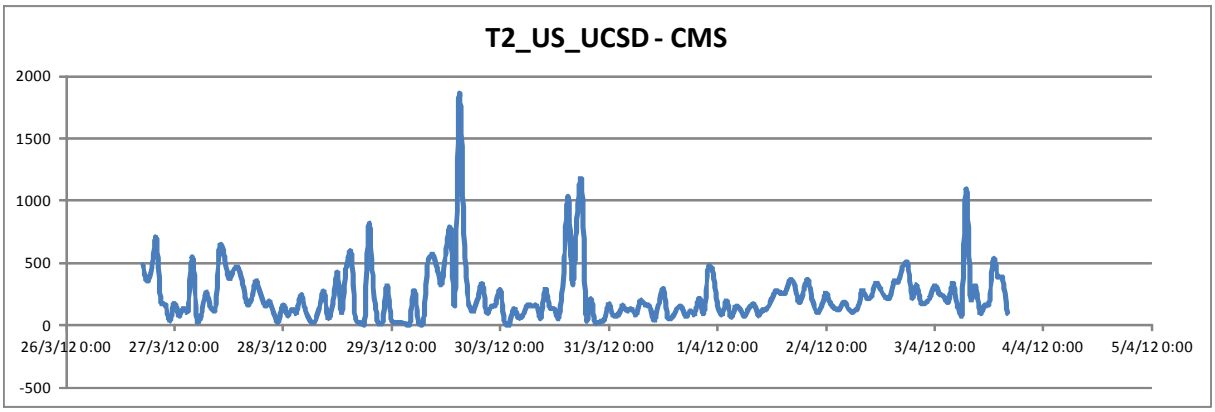

Figura 15 - *Jobs* processados na T2_US_UCSD

Nas Figuras 16, Figura 17, Figura 18, Figura 19, Figura 20 e Figura 21, temos os gráficos de submissão por hora dos *sites* presentes na Tabela 17. Eles correspondem aos *sites* que atendem a 1% do requerido pelo ATLAS.

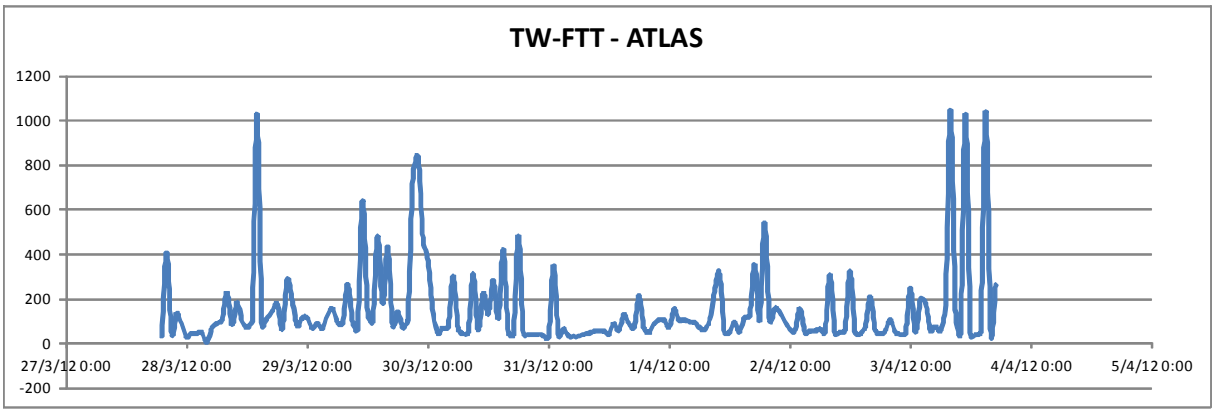

Figura 16 - *Jobs* processados na TW-FTT

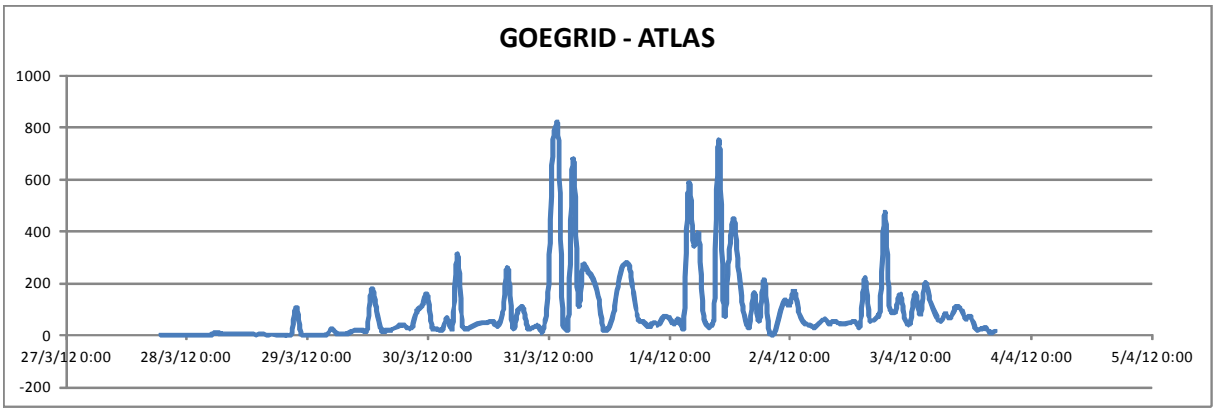

Figura 17 - *Jobs* processados na GOEGRID

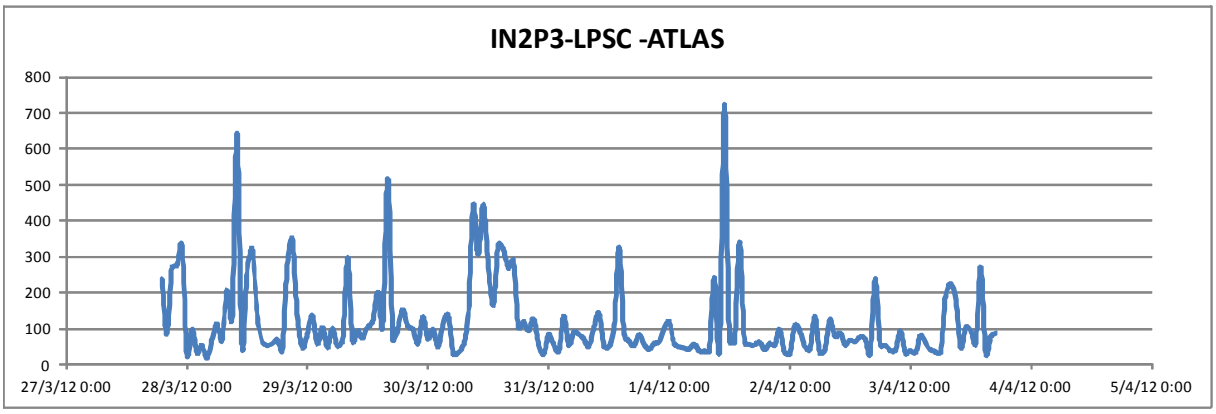

Figura 18 - *Jobs* processados na IN2P3-LPSC

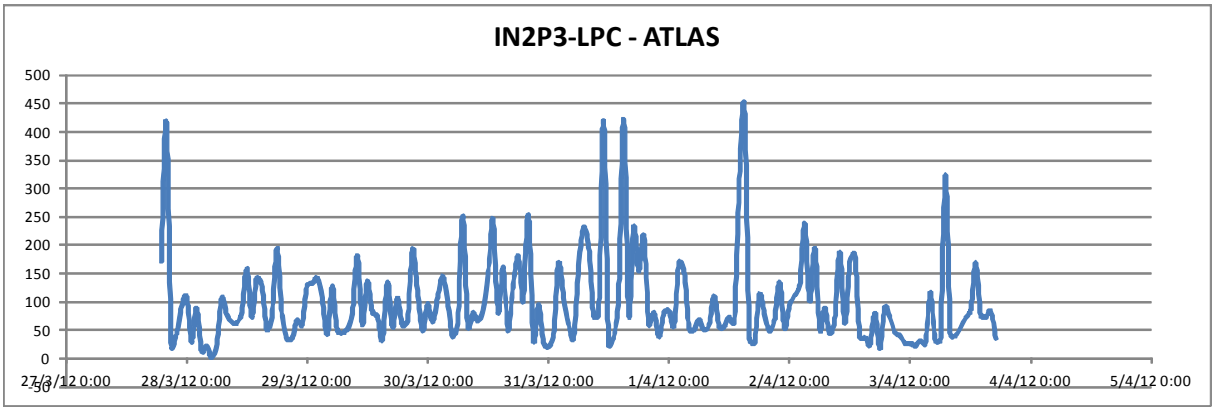

Figura 19 - *Jobs* processados na IN2P3-LPC

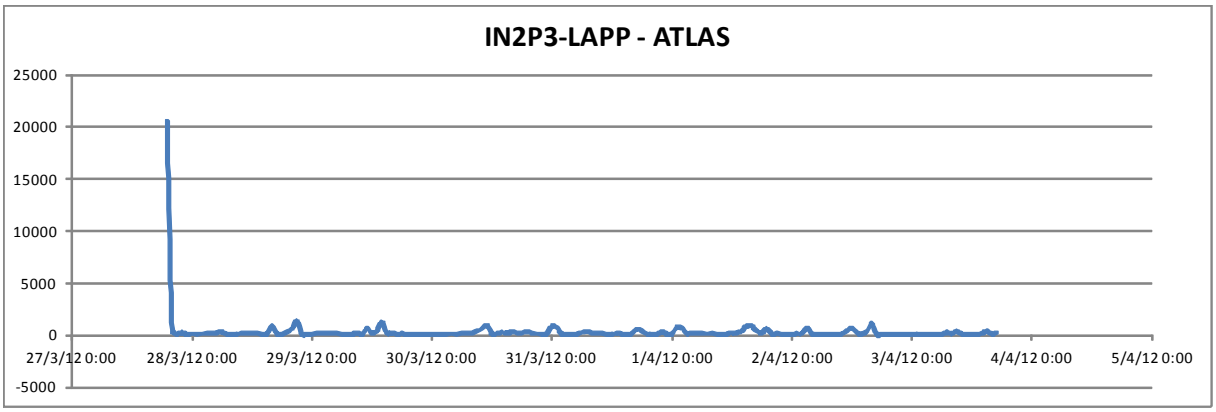

Figura 20 - *Jobs* processados na IN2P3-LAPP

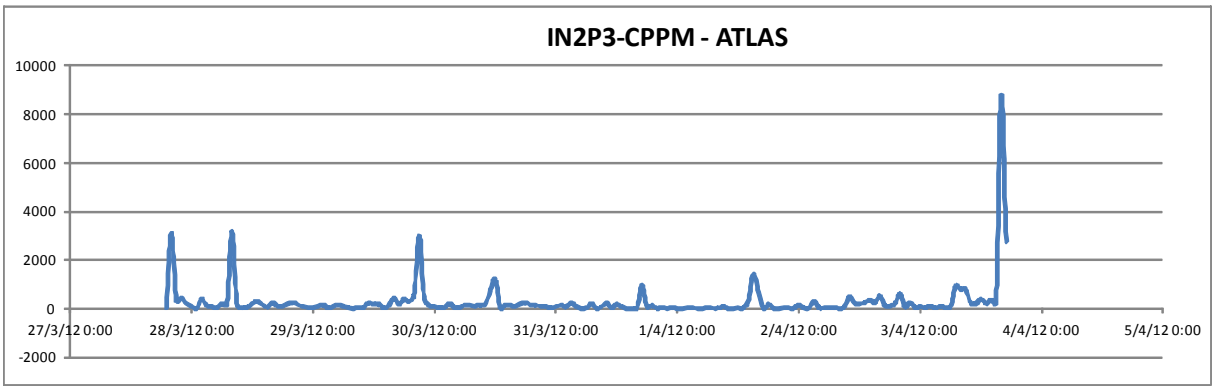

Figura 21 - *Jobs* processados na IN2P3-CPPM

Nas Figuras 22, Figura 23, Figura 24, Figura 25 e Figura 26, temos os gráficos de submissão por hora dos *sites* presentes na Tabela 18. Eles correspondem aos *sites* que atendem a 5% do requerido pelo ATLAS.

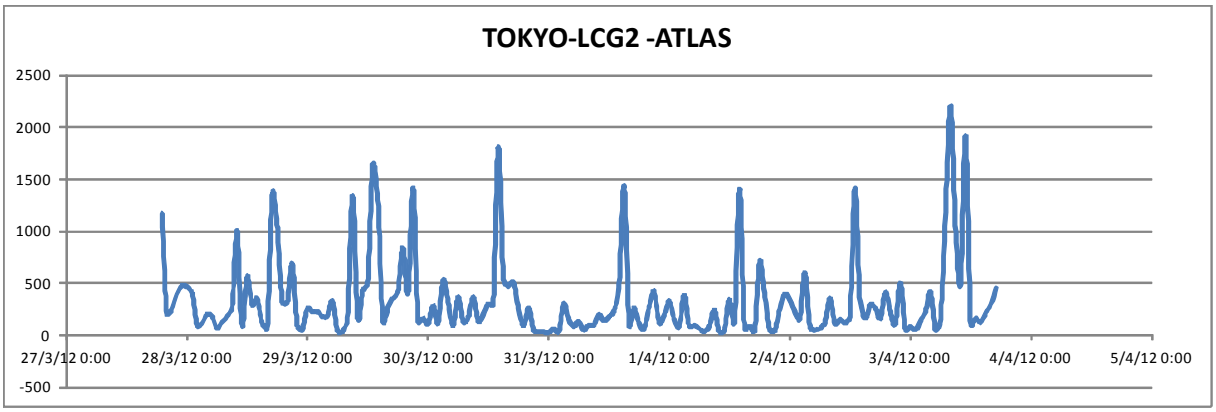

Figura 22 - *Jobs* processados na TOKYO-LCG2

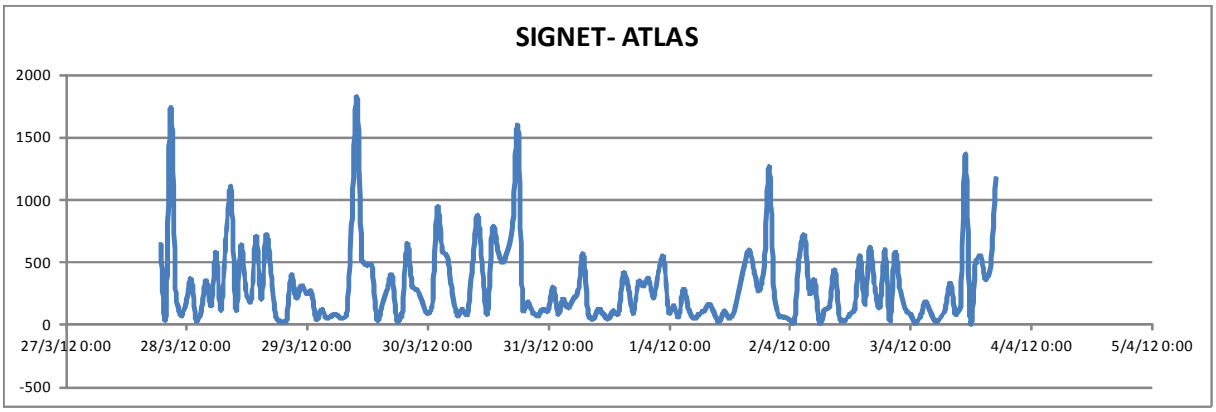

Figura 23 - *Jobs* processados na SIGNET

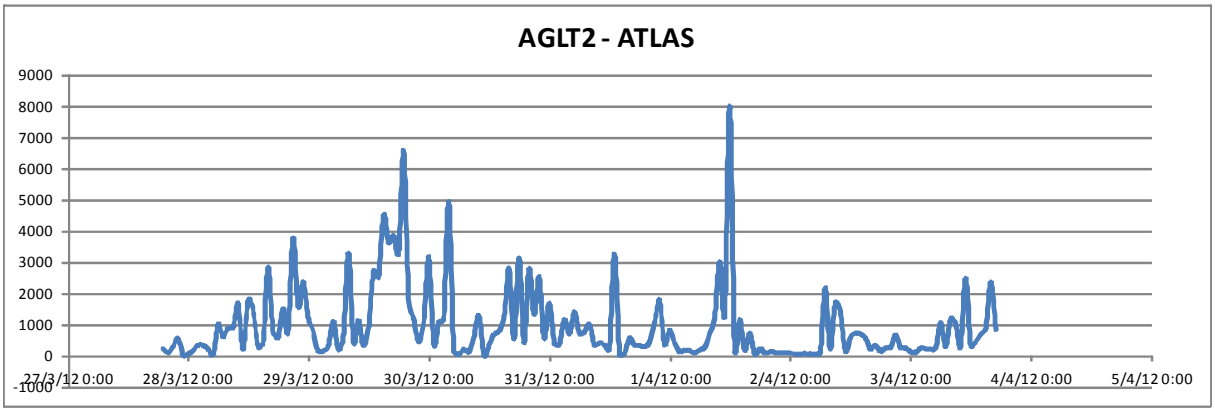

Figura 24 - *Jobs* processados na AGLT2

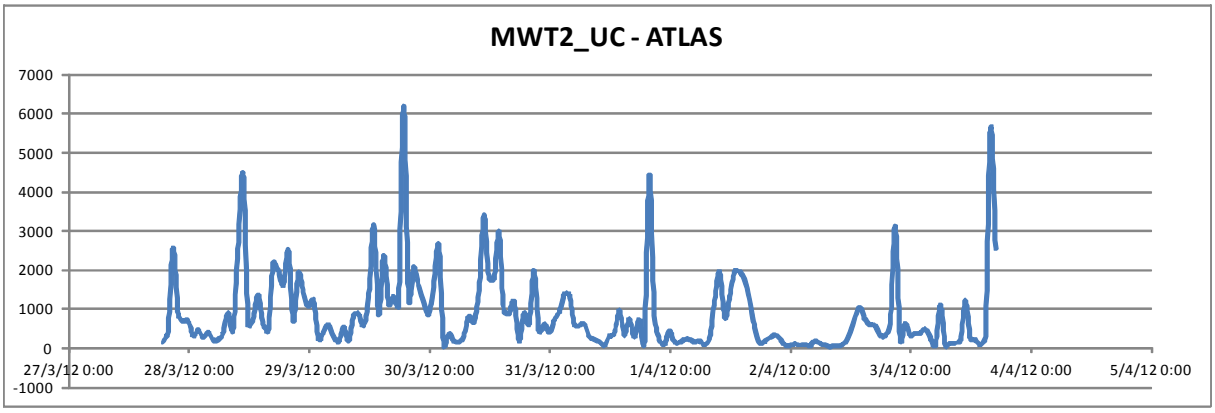

Figura 25 - *Jobs* processados na MWT2_UC

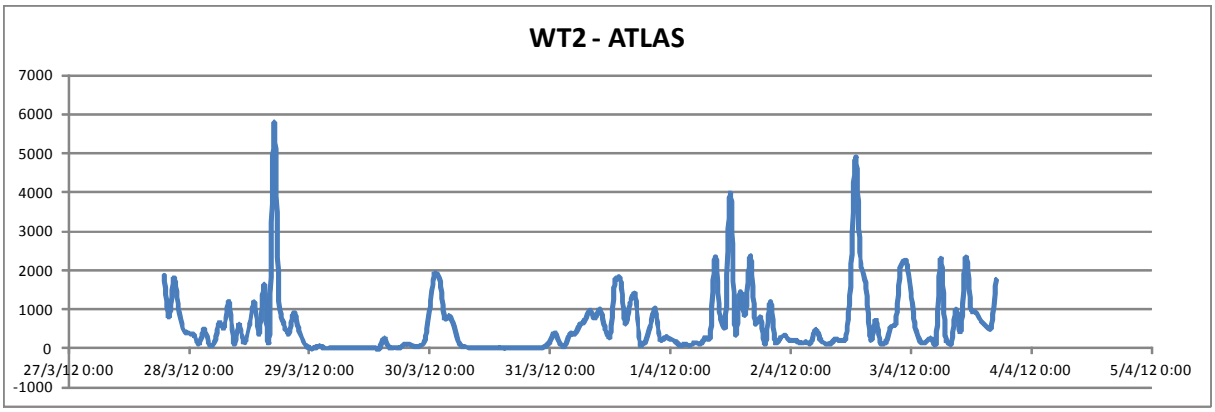

Figura 26 - *Jobs* processados na WT2

O número de *jobs* submetidos para cada OV foi dividido em faixas de frequências com amplitude de 50 unidades. As faixas de frequências, que representam o número de *jobs* submetidos, foram agrupadas de forma que correspondam a aproximadamente 91% de todos os *jobs*. Por exemplo, 91% da quantidade de *jobs* submetidos por hora do CMS ficou entre 0 e 350 para os *sites* que atendem a 1% dos requisitos do CMS. Este cálculo foi feito para os *sites* do CMS e do ATLAS que atendem a 1% e 5% dos requisitos de processamento dos experimentos. Esses valores estão na Tabela 12 na terceira coluna.

O cálculo da faixa do tempo de processamento dos *jobs* em segundos foi obtido em [27] a partir de um *log* de 11 dias que contém atividades de diversos nós da grade da LCG. Esses dados foram obtidos em novembro de 2005.

Observou-se uma grande variação de tempo de processamento dos *jobs* de cada OV dentro deste *log*, sendo que em alguns casos os *jobs* com grande duração correspondiam a uma pequena porcentagem. Devido a este fato, o tempo de processamento para cada OV foi dividido em faixas de frequências com amplitude de 200 unidades. As faixas de frequências, que representam o tempo de execução, foram agrupadas de forma que correspondam a aproximadamente 86% de todos os *jobs*. Por exemplo, 86% dos *jobs* do CMS contidos no log tem seu tempo de processamento entre 0 e 41600 segundos. Foi escolhida esta faixa de 86% pois os *jobs* acima desta faixa possuem o tempo de execução muito alto e não corresponde à realidade da maioria dos *jobs*, sendo apenas alguns casos isolados.

Na geração do *workload* de *jobs* para a simulação (Capítulo 5), quando um *job* do CMS é criado, por exemplo, o tempo de execução é sorteado e pode variar entre 0 e 41600 segundos. Visto que um *job* com tempo de processamento igual a

zero provavelmente representa um *job* com erro de processamento, consideramos o tempo mínimo de processamento igual a 10 segundos. Abaixo estão os valores mínimos e máximos de processamento para cada OV que correspondem a 86% dos *jobs* de cada OV.

Tabela 8 - Tempo de processamento de 86% dos *jobs*

VO	Tempo mínimo (s)	Tempo Máximo (s)
CMS	10	41600
ATLAS	10	14600
ALICE	10	11600
LHCB	10	400

O cálculo do número máximo de CPUs por VO em caso de pico foi feito baseado na duração máxima dos *jobs* de cada OV (DM) e no número máximo de *jobs* submetidos no intervalo de 1 hora (NM).

Cada *site* nos diversos cenários possui um determinado número de CPUs. O número total de CPUs em um determinado *site* corresponde a 100% do total para este *site*. Cada OV vai ter uma fatia desta porcentagem total em caso de pico. Para a avaliação realizada, o cálculo da fatia de cada OV foi feito de forma que a quantidade CPUs destinadas para uma OV fosse diretamente proporcional ao tempo de duração dos *jobs*, ou seja, OVs com *jobs* mais longos terão mais recursos disponíveis em caso de pico. A fatia de CPUs destinadas para cada OV é também diretamente proporcional ao volume de submissões de uma determinada OV, ou seja, OVs que submetem maior quantidade de *jobs* terão maior porcentagem de recursos disponíveis. É possível para o administrador configurar outros perfis para o *site* e para cada OV.

Para cada uma das OVs, multiplicamos o número máximo de *jobs* submetidos pela duração máxima do *job*. Na Tabela 9 temos o resultado desta multiplicação para cada uma das OVs. A partir deste valor, foi feita a conversão em termos percentuais do número de CPUs destinadas a cada OV. Na Tabela 10 temos o valor da conversão em termos percentuais.

Tabela 9 - Duração Máxima do *Job* X Número Máximo de *Jobs* (Grupo A)

OV	Duração máxima do *job* (DM) em segundos	Número máximo de *jobs* (NM)	DM x NM
CMS	41600	350	14560000
ATLAS	14600	700	10220000
ALICE	11600	350	4060000
LHCB	400	700	280000

A conversão foi feita da seguinte maneira. O valor da multiplicação obtido de cada uma das OVs foi somado. Foi feita a correspondência deste valor ao número total de CPUs em termos percentuais. Este valor obtido corresponde a 100% do número de CPUs. Foi feita uma regra de três para calcular a porcentagem de recursos disponíveis para cada uma das OVs. Por exemplo, de acordo com a Tabela 10, 100% dos recursos correspondem a 29120000, que é o valor da soma de todas as OVs. A OV ATLAS cujo valor da multiplicação foi igual a 10220000, pela regra de três vai ter 35% dos recursos disponíveis. O cálculo da quantidade de recursos destinados a cada OV foi feito de acordo com as Tabela 9 à Tabela 14.

Tabela 10 - Porcentagem do cluster comprometida com cada OV em caso de pico (Grupo A)

OV	DM x NM	Porcentagem
CMS	14560000	50%
ATLAS	10220000	35%
ALICE	4060000	14%
LHCB	280000	1%
TOTAL	29120000	100%

Baseado nos cálculos da Tabela 9 e na Tabela 10 foi feito o dimensionamento dos clusters do grupo A. O cálculo do número de CPUs em caso de pico para cada cluster do grupo A e o valor em termos percentuais está expresso na Tabela 11.

Tabela 11 - Núcleos comprometidos com cada OV em caso de pico (Grupo A)

OV	Porcentagem	Cluster I	Cluster II	Cluster III
CMS	50%	412	309	206
ATLAS	35%	288	216	144
ALICE	14%	115	87	58
LHCB	1%	8	6	4
TOTAL	100%	823	618	412

O mesmo cálculo feito para os clusters do grupo A se repetem para os clusters do grupo B e está na Tabela 12, Tabela 13 e Tabela 14.

Tabela 12 - Duração Máxima do *Job* X Número Máximo de *Jobs* (Grupo B)

OV	Duração máxima do *job* (DM) em segundos	Número máximo de *jobs* (NM)	DM x NM
CMS	41600	1500	62400000
ATLAS	14600	2500	36500000
ALICE	11600	1500	17400000
LHCB	400	2500	1000000

Tabela 13 - Porcentagem do cluster comprometida com cada OV em caso de pico (Grupo B)

OV	DM x NM	Porcentagem
CMS	62400000	53%
ATLAS	36500000	31%
ALICE	17400000	15%
LHCB	1000000	1%

Tabela 14 - Núcleos comprometidos com cada OV em caso de pico (Grupo B)

OV	Porcentagem	Cluster I	Cluster II	Cluster III
CMS	53%	2179	1634	1090
ATLAS	31%	1275	956	637
ALICE	15%	617	463	308
LHCB	1%	41	31	21
TOTAL	100%	4112	3084	2056

Este cálculo do número de CPUs para cada cluster faz com que o número máximo de CPUs alocadas seja diretamente proporcional ao número de *jobs* submetidos e à duração desses *jobs*. O mesmo acontece com as OVs que submetem mais *jobs* , essas OVs acabam tendo mais recursos disponíveis. Assim, o tamanho de cada cluster varia dinamicamente de acordo com a carga de *jobs* submetida. O objetivo deste dimensionamento é permitir a utilização dos recursos, pelas diversas OVs, de forma equilibrada. O número máximo de CPUs para cada OV é fixo apenas em casos de pico no cluster.

Nas Tabelas 15 até a Tabela 23, temos dados relativos à capacidade de processamento de diversos *sites* do LHC. Na primeira coluna temos o nome da federação. Uma federação pode ter vários *sites* sendo que optamos por escolher as federações formadas por um único *site*. Na segunda coluna temos o nome do *site*. Na terceira coluna temos o HEPSPEC total presente no *site*. Na quarta coluna temos o HEPSPEC comprometido com cada experimento que o *site* atende.

Na tabela 15 temos os *sites* que atendem a 1% do mínimo exigido pelo CMS. Observamos que esses *sites* apesar de se comprometerem com apenas 1 experimento, a maioria destes *sites* possui a capacidade de processamento superior à exigida pelo CMS.

Tabela 15 - *Sites* com 1 único cluster e que atendem 1% do CMS

Federação	***Sites* CMS**	**HEPSPEC presente**	**HEPSPEC do CMS**
Taiwan Analysis Facility Federation	TW-FTT	15.152	2.660
CHEP of KNU, Daegu	LCG_KNU	4.669	3.600
TIFR, Mumbai	INDIACMS-TIFR	2.582	3.000
HGCC Federation	BUDAPEST	5.350	2.800
HEP Laboratory, University of Ioannina	GR-07-UOI-HEPLAB	1.872	3.040

A Tabela 16 possui os *sites* que se comprometeram a atender a 4% do requerido pelo CMS. Semelhante à Tabela 15, observamos que esses *sites* possuem a capacidade de processamento superior à exigida pelo CMS.

Tabela 16 - *Sites* com 1 único cluster e que atendem 4% do CMS

Federação	***Sites* CMS**	**HEPSPEC presente**	**HEPSPEC do CMS**
Caltech CMS T2	CIT_CMS_T2	15.000	12.500
MIT CMS T2	MIT_CMS	15.000	12.500
Nebraska CMS T2	Nebraska	15.000	12.500
U. Wisconsin CMS T2	GLOW	15.000	12.500
UC San Diego CMS T2	UCSDT2	15.000	12.500

A Tabela 17 possui os *sites* que se comprometeram a atender a 1% do requerido pelo ATLAS. No entanto, alguns destes *sites* atendem a outros experimentos simultaneamente. Mesmo atendendo a outro experimento, observamos que esses *sites* possuem a capacidade de processamento superior à exigida pelos dois experimentos que eles atendem.

Tabela 17 - *Sites* com 1 único cluster e que atendem a 1% do ATLAS

Federação	***Sites* ATLAS**	**HEPSPEC presente**	**HEPSPEC do ATLAS**
Taiwan Analysis Facility Federation	TW-FTT *3	15.152	2.660
ATLAS Federation	GoeGrid	25.501	3.853
LPSC Grenoble	IN2P3-LPSC *1	6.217	2.322
LPC, Clermont-Ferrand	IN2P3-LPC *4	9.027	3.146
LAPP, Annecy	IN2P3-LAPP *4	7.811	3.200
CPPM, Marseille	IN2P3-CPPM *4	7.855	2.264

A Tabela 18 possui os *sites* que se comprometeram a atender a 5% do requerido pelo ATLAS. Observamos que esses *sites* apesar de se comprometerem com apenas 1 experimento, a maioria destes *sites* (exceto o segundo *site*) possui a capacidade de processamento superior à exigida pelo ATLAS.

Tabela 18 - *Sites* com 1 único cluster e que atendem a 5% do ATLAS

Federação	***Sites* ATLAS**	**HEPSPEC presente**	**HEPSPEC do ATLAS**
ICEPP, Tokyo	TOKYO-LCG2	16.531	12.000
SiGNET	SiGNET	11.388	12.000
Great Lakes ATLAS T2	AGLT2	37.439	12.500
Midwest ATLAS T2	MWT2	54.230	12.500
SLAC ATLAS T2	WT2	29.376	12,500

A Tabela 19 possui os *sites* que se comprometeram a atender a 1% do requerido pelo ALICE. No entanto, alguns destes *sites* atendem a outros experimentos simultaneamente. Mesmo atendendo a outro experimento, observamos que esses *sites* possuem a capacidade de processamento superior à exigida pelos experimentos que eles atendem.

Tabela 19 - *Sites* com 1 único cluster e que atendem a 1% do ALICE

Federação	***Site* ALICE**	**HEPSPEC presente**	**HEPSPEC do ALICE**
HGCC Federation	BUDAPEST *3	5.350	960
LPSC Grenoble	IN2P3-LPSC *2	6.217	1.900
LPC, Clermont-Ferrand	IN2P3-LPC *1 *2 atlas lhcb	9.027	3.146

A Tabela 20 possui os *sites* que se comprometeram a atender a 8% do requerido pelo ALICE. Observamos que esses *sites* apesar de se comprometerem com apenas 1 experimento, a maioria destes *sites* possui a capacidade de processamento superior à exigida pelo ALICE, exceto o primeiro *site*.

Tabela 20 - *Sites* com 1 único cluster e que atendem a 8% do ALICE

Federação	***Site* ALICE**	**HEPSPEC presente**	**HEPSPEC do ALICE**
LBNL ALICE Berkeley CA	NERSC-PDSF	9.500	12.000
LLNL ALICE, Livermore CA	LC-glcc	13.500	11.500
GRIF,Paris	GRIF	69.071	6.670

A Tabela 21 possui os *sites* que se comprometeram a atender a 3%, 4% e 5% do requerido pelo LHCB. No entanto, alguns destes *sites* atendem a outros experimentos simultaneamente. Mesmo atendendo a outro experimento, observamos que esses *sites* possuem a capacidade de processamento superior à exigida pelos experimentos que eles atendem.

Tabela 21 - *Sites* com 1 único cluster e que atendem a 3%, 4% e 5% do LHCb

Federação	***Site* LHCb**	**HEPSPEC presente**	**HEPSPEC do LHCb**
LPC, Clermont-Ferrand	IN2P3-LPC*1*2	9.027	1.303
LAPP, Annecy	IN2P3-LAPP *2	8.883	1.600
CPPM, Marseille	IN2P3-CPPM*2	7.855	2.000

A Tabela 22 possui os *sites* que se comprometeram a atender a 7% do requerido pelo LHCb. Observamos que esses *sites* apesar de se comprometerem com apenas 1 experimento, esses *sites* possuem a capacidade de processamento superior à exigida pelo LHCb.

Tabela 22 - *Sites* com 1 único cluster e que atendem a 7% do LHCb

Federação	***Site* LHCb**	**HEPSPEC presente**	**HEPSPEC do LHCb**
LHCb Federation DESY	DESY-ZN	13.690	3.200
LHCb Federation	UB-LCG2	3,649	2,800

A Tabela 23 possui os *sites* que se comprometeram a atender aos 4 experimentos simultaneamente. O primeiro *site* possui a capacidade de processamento ligeiramente menor que a exigida pelo conjunto dos 4 experimentos. Já o segundo *site* possui a capacidade de processamento muito maior que a exigida pelos 4 experimentos.

Tabela 23 - *Sites* com 1 único cluster e que atendem aos 4 experimentos

Federação	***Site* com 4 OVs**	**CPU HS06 presente**	**CPU HS06 CMS**	**CPU HS06 ATLAS**	**CPU HS06 ALICE**	**CPU HS06 LHCb**
CC-IN2P3 AF	IN2P3-CC-T2	23.785	6.600	9.750	2.300	5.200
GRIF, Paris	GRIF	69.642	9.324	10.100	6.685	4.064

*1 – Atendem ao ALICE

*2 – Atendem ao ATLAS

*3 – Atendem ao CMS

*4 – Atendem ao LHCb

A partir dos dados relativos à capacidade computacional de *sites* reais, podemos observar os cenários reais onde a política poderia ser implementada. Observamos que na maioria dos casos, os *sites* reais possuem mais recursos do que o necessário para o processamento. Apesar de possuírem recursos em abundância, veremos na análise dos resultados que a política pode ser eficiente nestes casos também. Veremos também que quando o *site* tiver menos recursos que o necessário até certo limite, a política pode ser eficiente.

5 SIMULAÇÃO

Para avaliar a arquitetura e a política propostas é possível trilhar o caminho da construção de um protótipo do *broker*. Para isso é necessário integrar os diversos mecanismos e sistemas de *middeware* de cada OV, bem como dispor de uma infraestrutura de máquinas virtuais, tal como a *CernVM* mencionada no Capítulo 3. Além disso, é necessário ter à disposição um cluster com recursos suficientes para permitir a instanciação de um número razoável de MVs para um teste consistente. A avaliação da arquitetura e da política propostas foi realizada através de simulações Um simulador permitiria a configuração de cenários onde o cluster poderia ter tamanhos arbitrariamente grandes.

Para a implementação da proposta apresentada neste trabalho, optamos por utilizar o simulador de grades GridSim [28]. A escolha deste simulador foi feita devida a sua utilização e aceitação no meio acadêmico e por atender aos requisitos específicos para simular a arquitetura proposta, como um sistema de eventos para representar a troca de mensagens entre os elementos da grade, a consideração sobre os aspectos de rede e a formação de clusters customizados.

5.1 O Simulador GridSim

O GridSim é um *toolkit* que propõe facilitar a simulação de diferentes classes de recursos heterogêneos, usuários, aplicações e escalonadores. Ele pode ser usado para simular escalonadores de tarefas, para sistemas distribuídos como *clusters* (um único domínio administrativo) e *grids* (múltiplos domínios administrativos).

No GridSim os recursos podem ser modelados de forma que sejam compartilhados no tempo e no espaço. Ou seja, os recursos computacionais podem ser sistemas monoprocessados, multiprocessados de memória compartilhada, ou multiprocessados de memória distribuída. Os dois primeiros empregam o conceito de compartilhamento no tempo, devido ao fato de o sistema operacional ser multitarefa. O último explora o conceito de compartilhamento no espaço, adotando um modelo de escalonamento interno distinto. No GridSim a entidade *AllocationPolicy* é responsável por alocar nós às tarefas dos usuários de acordo com a política de escalonamento a ser adotada. Neste trabalho utilizamos a política de compartilhamento no espaço.

A capacidade dos recursos é definida de acordo com o padrão estabelecido pelos *benchmarks* SPEC (*Standard Performance Evaluation Corporation*), isto é, em termos de MIPS (milhões de instruções por segundo).

O GridSim é implementado em linguagem Java, no topo da biblioteca de simulação SimJava, utilizando o paradigma de programação orientada a objetos. A ferramenta implementa diversas classes que modelam componentes de sua arquitetura, além de classes e métodos úteis na parte estatística (geração de números aleatórios, extração de estatísticas sobre o uso e estado dos recursos), que facilitam a modelagem dos dados de entrada e a interpretação dos resultados.

Cada componente do *toolkit* do GridSim é executado por uma *thread* distinta. Este mecanismo de execução *multithreaded* procura conferir uma maior escalabilidade ao simulador.

O GridSim suporta entidades para simulação de um único processador e múltiplos processadores, recursos heterogêneos que podem ter seu modo de operação configurado como tempo compartilhado ou espaço compartilhado. Ele suporta entidades que simulam redes utilizadas para comunicação entre os recursos. Durante a simulação, o GridSim cria múltiplas threads que executam concorrentemente. O comportamento dinâmico de cada entidade (recebimento e tratamento de eventos) é simulado dentro de seu método *body().* As principais entidades no GridSim são:

User – Cada instância desta entidade representa um usuário da grade, o qual é caracterizado pelo tipo de tarefa que cria (tempo de execução da tarefa, número máximo de réplicas, etc.), pela estratégia de otimização adotada no escalonamento (minimização do custo, do tempo, de ambos, etc.), taxa de atividade (quão frequentemente o usuário cria uma nova tarefa) e fuso-horário.

Broker – Cada usuário é conectado a uma instância de uma entidade *broker*. Toda tarefa de um usuário é primeiramente submetida ao seu *broker*, e este então escalona a tarefa de acordo com a política de escalonamento adotada pelo usuário.

Resource – Cada instância desta entidade representa um recurso da grade, e possui os seguintes atributos: número de processadores, custo de processamento, velocidade de processamento, política interna de

escalonamento (compartilhamento no tempo ou no espaço), fator de carga local, e fuso-horário.

Grid Information Service – Provê o serviço de registro dos recursos existentes na grade, mantendo uma lista atualizada dos recursos disponíveis. Os *brokers* consultam esta entidade para obter informações sobre a disponibilidade, configuração e estado dos recursos.

Input/Output – O fluxo de informações entre as entidades do GridSim ocorre através de suas entidades de Entrada e Saída. A Entrada e a Saída têm cada uma sua thread de execução com o método body() que manipula os eventos. O uso de entidades separadas para entrada e saída permite que as demais entidades modelem canais de comunicação *full-duplex* e multiusuários.

A modelagem das entidades participantes da grade e das tarefas que deverão ser submetidas é feita por meio da implementação, em Java, de um programa que faça uso da biblioteca provida pelo GridSim.

A criação e manipulação das tarefas ou *jobs* é feita através do pacote *Gridlet*, que modela um *job* (ou os vários processos de um *job*) e contém informações relacionadas ao tamanho computacional da tarefa, às operações de E/S, ao tamanho dos dados de entrada e de saída, e ao usuário criador da tarefa.

A interação entre as entidades da grade modelada por meio de eventos, que são gerados pelas entidades e podem significar uma requisição de serviço a outra entidade, um atendimento de serviço solicitado por outra entidade, ou uma atividade interna. Os eventos gerados podem ser classificados como síncronos ou assíncronos. Um evento é síncrono, quando a entidade fonte do evento espera até que a entidade destinatária efetue todas as ações associadas ao evento em questão. Se a entidade fonte continua com outras entidades após gerar o evento, então este evento e classificado como assíncrono. Eventos internos de uma entidade (fonte e destino são idênticos) são necessariamente assíncronos, para evitar a ocorrência de *deadlocks*. Eventos externos podem ser síncronos ou assíncronos. Uma entidade pode solicitar, por exemplo, informações ao GIS (*Grid Information Service*) sobre os recursos disponíveis na grade, criando um evento de requisição. Esse modelo de interação também é importante em atividades como escalonamento de tarefas pelo *broker*, escalonamento realizado internamente pelo recurso, entre outras [37].

Os recursos do sistema são modelados criando-se o que se denomina elementos de processamento (PEs), com diferentes velocidades. Um ou mais PEs podem ser combinados para criar uma máquina mono ou multiprocessada. Analogamente, uma ou mais máquinas podem ser combinadas para criar um recurso da grade. Assim, os recursos podem ser sistemas monoprocessados, multiprocessadores de memória compartilhada (SMP), ou cluster de computadores, em que a memória é distribuída. Os recursos monoprocessados e os multiprocessados de memória compartilhada são tipicamente gerenciados por um sistema operacional de tempo compartilhado, que usam alguma política de escalonamento para prover um ambiente multitarefas aos seus usuários. Os sistemas multiprocessados de memória distribuída são gerenciados por sistemas de filas chamados escalonadores de espaço compartilhado. Tais sistemas usam políticas de alocação de recursos como FCFS (*First-Come,First-Served*), SJFS (*Shortest-Job-First Served*),etc. [37]

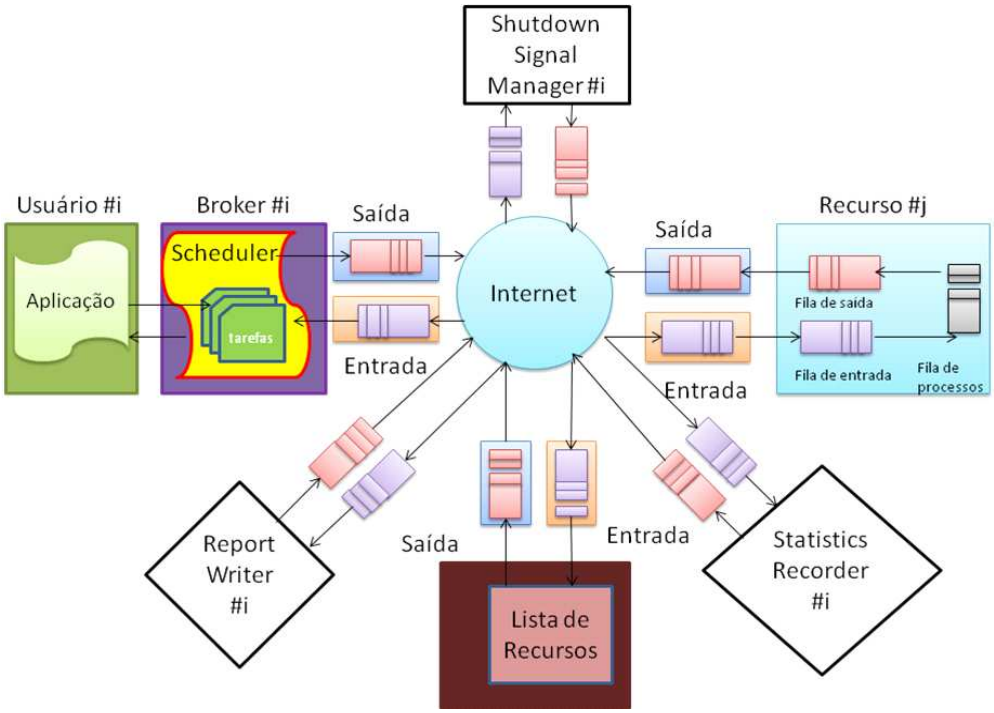

Figura 27 - Arquitetura do GridSim [28]

Quando o GridSim é inicializado, os recursos se registram na entidade GIS (*Grid Information Service*) através do envio de eventos. Dependendo da requisição da entidade usuário, a entidade *broker* envia um evento ao GIS para significar uma consulta para descoberta de recursos. A entidade GIS retorna uma lista de recursos

registrados. A entidade *broker* envia eventos para os recursos com o pedido pelas configurações dos recursos e propriedades. Os recursos respondem com informações dinâmicas como custo do recurso, capacidade, disponibilidade, carga e outros parâmetros de configuração. Esses eventos envolvendo o GIS são assíncronos. Dependendo da escolha do recurso e da estratégia de escalonamento, a entidade *broker* coloca eventos assíncronos para os recursos a fim de enviar *Gridlets* para execução. Quando o processamento de um *Gridlet* termina, a entidade recurso atualiza o status e o tempo de processamento do *Gridlet* e o envia de volta para o *broker* levantando um evento para significar sua conclusão.

5.2 Modelagem da arquitetura no simulador

Para a modelagem da arquitetura no simulador, foram criados 4 usuários cada um pertencendo a uma OV do LHC. Cada usuário submeteu uma rampa de *jobs*. Foi criada a entidade *broker*, que recebia os *jobs* dos usuários e os submetia a política de escalonamento proposta presente no *broker*. Foi criado um cluster para cada simulação de acordo com a configuração presente na Tabela 26 e na Tabela 30. A Figura 28 apresenta o esquema da modelagem proposta.

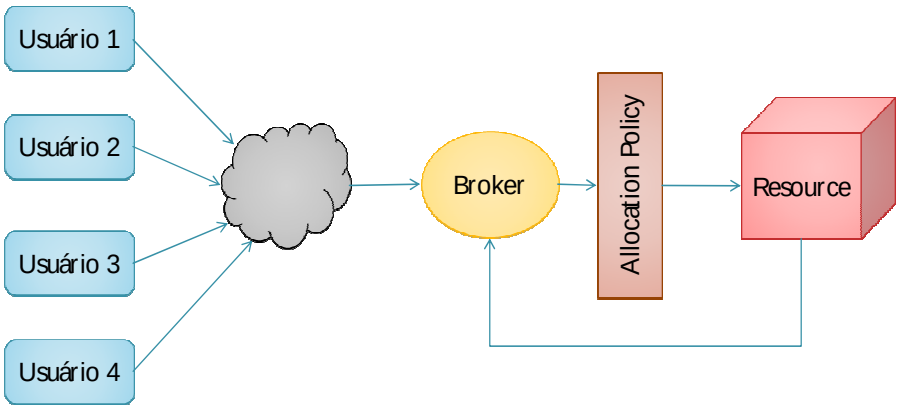

Figura 28 - Modelagem proposta

No GridSim o cluster é representado pela entidade *GridResource* que contém os recursos computacionais do cluster. Cada *GridResource* possui um conjunto de máquinas sendo cada uma representada pela entidade *Machine*. Cada máquina tem um número de processadores que são representados pela entidade PE (*Processing Element*). Para a simulação foram criadas máquinas com 1 processador cada. A velocidade de processamento de cada processador foi configurada de acordo com os parâmetros da Tabela 26 e Tabela 30.

Um cluster no GridSim está associado à uma política de alocação de recursos chamada *NewPolicy*. Ele é uma extensão da classe *AllocPolicy*. Quando o *GridResource* é criado, ele é associado a um escalonador de tarefas do tipo *AllocPolicy* que tem acesso direto aos PEs do *GridResource* e determina quando e quais tarefas serão executadas, canceladas ou pausadas. Neste trabalho, a decisão de quais tarefas seriam executadas, canceladas ou pausadas não foi executada pelo *NewPolicy*. A tomada de decisão foi toda executada dentro do *broker*, porém o *NewPolicy* recebia as mensagens vindas do *broker* e executava no *GridResource* as decisões tomadas pelo *broker*. Para o *NewPolicy* utilizamos o escalonador default *Space Shared* . De acordo com [28] o Gridsim utiliza a política de alocação *Space Shared* em recursos com múltiplos PEs. No Gridsim a classe *Space Shared* se comporta de maneira semelhante à política *First Come First Serve* (FCFS) onde cada job é processado em um PE. No Gridsim um único PE é gerenciado pela política *Time Shared* que utiliza *Round-Robin* para múltiplas tarefas.

O *Broker*, neste trabalho, é uma entidade localizada dentro do *site* e que recebe os *jobs* oriundos de diversas OVs distintas. Ele é quem realiza o controle dos recursos utilizados por cada OV através da política de alocação de recursos proposta.

Os aspectos de virtualização foram emulados aproveitando-se o fato que no GridSim, o ciclo de vida de um *Gridlet* pode ser gerenciado. Por exemplo, é possível pausar ou cancelar um *Gridlet*. Assim, criamos extensões no *Gridlet* para introduzir um novo estado, chamado BROKER_PAUSED (Seção 4.2), e criamos no *broker* um procedimento para identificar um *Gridlet* de determinada VO no cluster e, através da classe *NewPolicy*, atomicamente pausamos o mesmo, colocamos este na fila de pausados e em seguida retiramos o mesmo do cluster. Isso seria equivalente a uma pausa da VM onde o *job* está executado, combinada com a migração desta VM para uma fila (sem considerar possíveis otimizações).

5.3 Workload em Rampa

Para simular a carga de submissão de *jobs* dos 4 experimentos foi desenvolvida a classe *WorkloadRamp* que gera rampas de *jobs* para cada usuário (cada OV, no caso). Cada rampa contém uma quantidade de rajada de *jobs* (ou passos) [43]. Para cada rampa foram definidas a rajada mínima e a rajada máxima de *jobs*. Cada *job* gerado nas rajadas pode ter uma duração mínima de tempo e

uma duração máxima. O tempo de cada *job* é sorteado pelo gerador de números aleatórios do GridSim dentro deste intervalo, que é definido para cada OV de acordo com a Tabela 8. Estes atributos associados aos *jobs* são gerados aleatoriamente, por meio da classe *GridSimRandom*.

Cada *job* é associado, no simulador, a um arquivo de entrada. Este arquivo pode ter um tamanho mínimo e um máximo. O tamanho deste arquivo é sorteado pelo gerador de números aleatórios do GridSim. Foi definido para todas as OVs da simulação o mesmo tamanho mínimo e máximo que corresponde à 100Kbytes e 1000Kbytes respectivamente.

A duração da rampa, o número de rajadas (os passos da rampa), o valor do número de *jobs* mínimo e máximo (no pico da rampa) são definidos por parâmetros de entrada no construtor da classe. Adiante temos o exemplo da chamada implementada no simulador para criação de uma rampa para um usuário que faz parte da OV CMS.

```
WorkloadRamp model1 = new WorkloadRamp(
  SimulationParameters.MAXJobs1HourCMS,  //int maxBurst, rajada máxima
  SimulationParameters.MINJobs1HourCMS,  //int minBurst, rajada mínima
  0,                 //long initialTime,
  3*HOUR,            //long totalInterval, total do workload
  steps,             //int steps, passos, em cada passo uma rajada
  1,                 //long minDuration, tempo mínimo de execução do job
  41600,             //long maxDuration, tempo máximo de execução do job
  100,                   //int minSize,   o mesmo para o tamanho ocupado ou
arquivo
  100 * 10,          //int maxSize,
  1,                 //int numProc, número de processadores por nó
  Gridlet.CMS,       //int vo, identificação da VO relacionada ao job
  0,                 //int user_id,
  rating             //int rating, classificação do processador típico
);
```

Na Figura 29 temos o diagrama com as classes *RunBrokerRamp*, *HEPGridPolicy*, *RampWorkload* e *WorkloadRamp* que implementam a geração de rampas de *jobs*. A classe *RunBrokerRamp* é responsável por criar o recurso e instanciar as 4 rampas utilizando as classes *RampWorkload* e *WorkloadRamp*. A classe *RampWorkload* é responsável por submeter os *Gridlets* para o recurso e a classe *WorkloadRamp* por gerar os *Gridlets*.

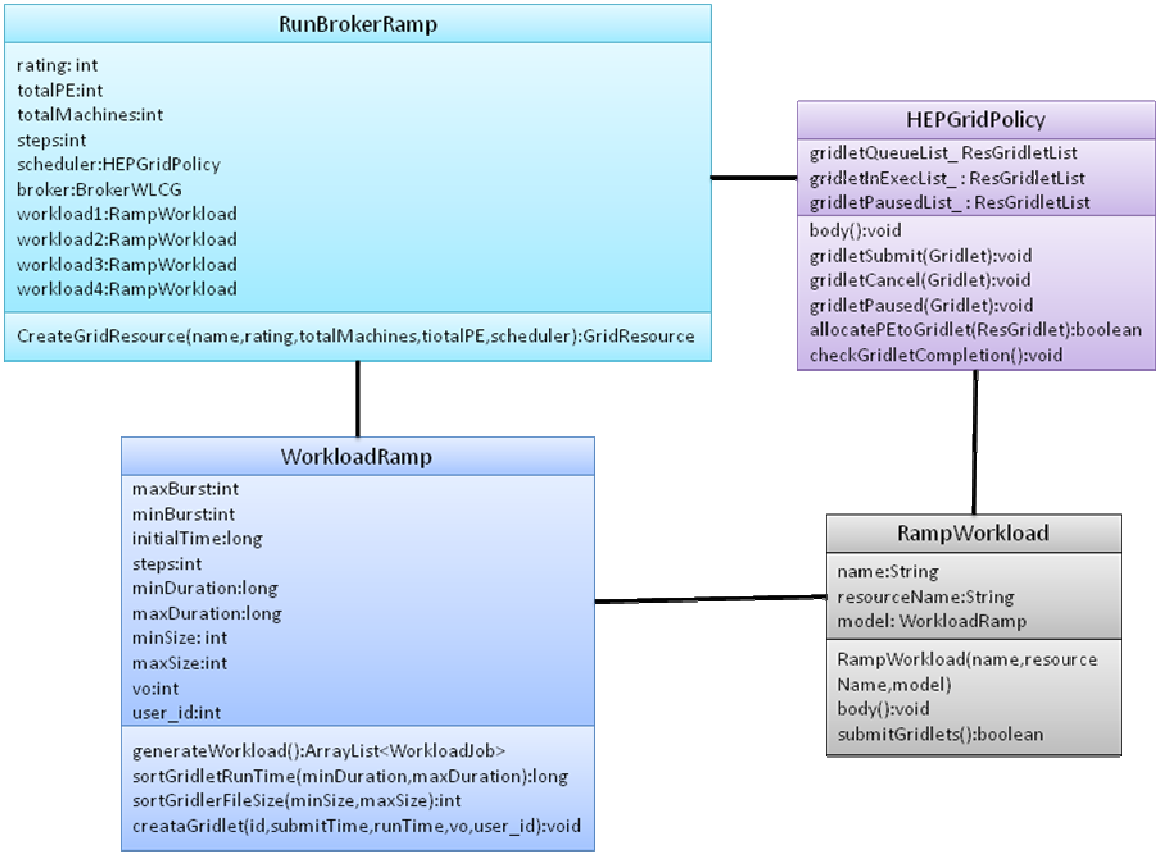

Figura 29 - Diagrama das classes *RunBrokerRamp*, *HEPGridPolicy*, *RampWorkload* e *WorkloadRamp*

Os gráficos da Figura 30 apresentam um exemplo da rampa gerada pela classe *WorkloadRamp.* O eixo de Tempo utiliza uma métrica abstrata, que podemos definir como segundos. Assim, no exemplo a última rajada (12) termina 12 horas depois da primeira (0). Observamos nos gráficos a variação no tamanho dos *jobs* de cada rajada (o que define quanto cada *job* precisa ocupar de tempo de CPU no cluster) e o seu tamanho em Kbytes (o que define o espaço em memória necessário e, de certa forma a banda de rede necessária para enviar o *job* pelos enlaces de rede).

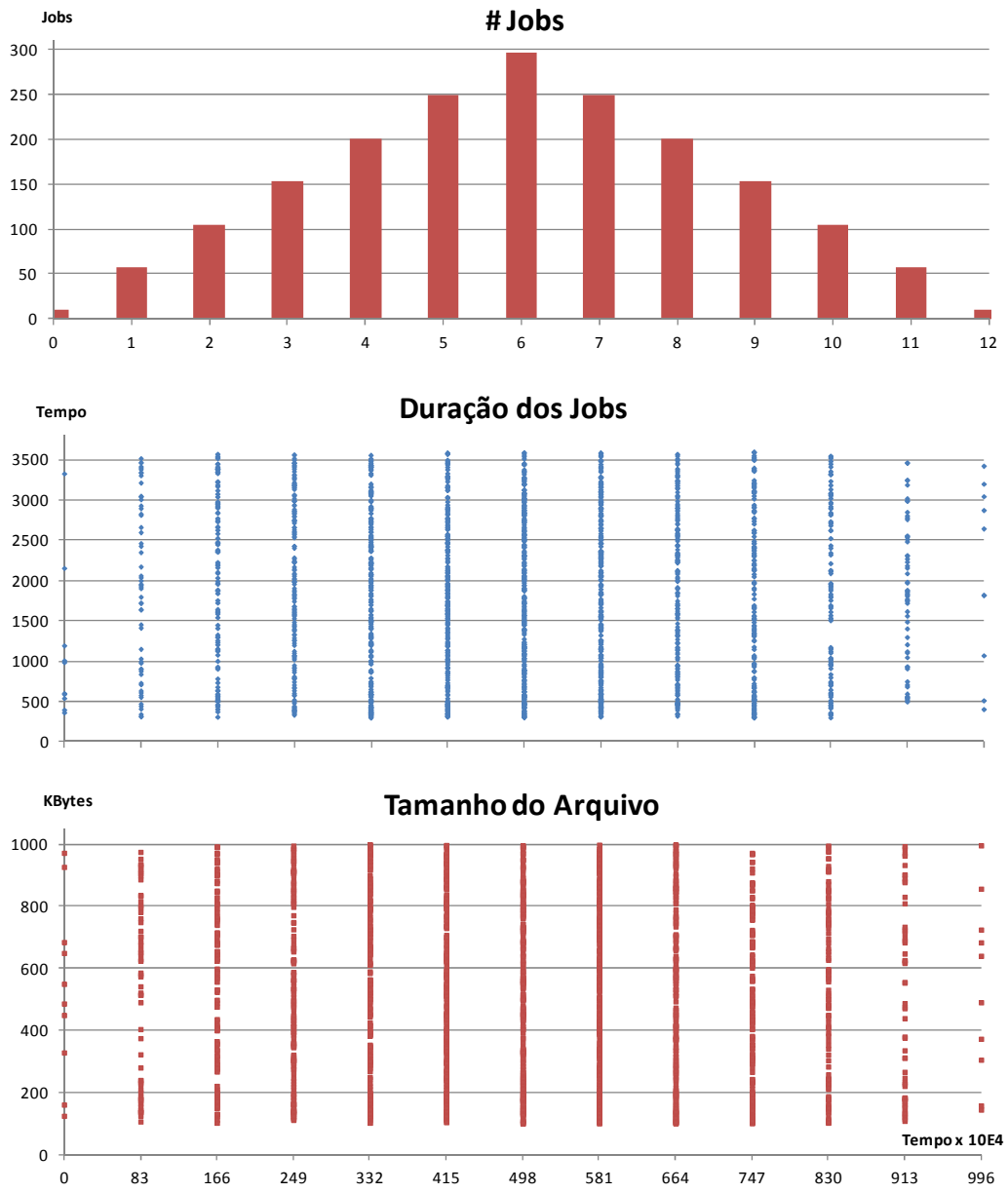

Figura 30 - Rampa de *jobs* submetidos

Os *jobs* criados por usuários precisam ter sua OV identificada para a aplicação das políticas de uso de cada *site* dentro do cluster. Para isso, foi adicionado à classe *Gridlet* o campo OV que identifica a OV de cada *job*.

Para cada cluster foram submetidas 3 rampas de *jobs* sendo cada rampa corresponde a *jobs* de uma OV. Para cada cluster de cada grupo foram feitas 2 simulações uma sem política de alocação e outra com a política proposta. Em ambas as simulações foram submetidas a mesma rampa com 3 rajada de *jobs*. Ao todo, foram simulados 6 clusters para a avaliação da política proposta

5.4 Perfil do Site

A configuração das máquinas da simulação corresponde à configuração dos servidores Dell PowerEdge 2950 que possuem sistema operacional Scientific Linux 4.6, arquitetura de 64 bits, com 2 processadores Dual-Core Intel Xeon série 5160, 3.0GHz, 16GB de memória cujo HEPSPEC por núcleo é igual a 9,35 [24]. O resultado deste benchmark varia de acordo com a arquitetura, tipo do processador, número de núcleos e o sistema operacional instalado da máquina hospedeira. Este resultado do HEPSPEC foi obtido para uma máquina com as configurações de hardware e software citadas acima e utilizamos este resultado para simular todos os clusters no GridSim. Baseada nesta configuração, os clusters dos grupos A e B foram formados com o número de núcleos de processamento conforme a Tabela 24 e a Tabela 25.

O cálculo do número de núcleos de processamento em cada cluster foi feito dividindo-se o HEPSPEC total do cluster pelo HEPSPEC de um único núcleo com as configurações descritas acima. Visto que os resultados não foram números inteiros, os valores foram aproximados.

Tabela 24 - Número de núcleos por OV do grupo A

Cluster	CMS	ATLAS	ALICE	LHCB	TOTAL
I	337	285	155	46	823
II	253	213	116	35	617
III	169	142	78	23	412

Tabela 25 - Número de núcleos por OV do grupo B

Cluster	CMS	ATLAS	ALICE	LHCB	TOTAL
I	1684	1423	775	230	4112
II	1263	1067	582	172	3084
III	842	711	388	115	2056

A partir de agora vamos detalhar os perfis e os contratos para cada cluster da simulação. No grupo A temos 3 clusters, cada cluster com um perfil e uma política especifica para cada OV.

Tabela 26 - Grupo A - Perfil dos *Sites*

Site	**I**	**II**	**III**
C_n	823	617	412
V_n	4	4	4
PeRating (SPEC)	2336	2336	2336
SO	SL4	SL4	SL4
BaundRate (bits/s)	2Gb	2Gb	2Gb
SPEC_total	1922500	1441875	961250
HEPSPEC_total	7690	5767,5	3845

Tabela 27 - Grupo A - Contrato *Site* I

OV	**CMS**	**ATLAS**	**ALICE**	**LHCB**
C_{max}	337	285	155	46
T_{max} (s)	3600 (1h)	3600 (1h)	3600 (1h)	3600 (1h)
HEPSPEC	3150	2660	1450	430

Tabela 28 - Grupo A - Contrato *Site* II

OV	**CMS**	**ATLAS**	**ALICE**	**LHCB**
C_{max}	253	213	116	35
T_{max} (s)	3600 (1h)	3600 (1h)	3600 (1h)	3600 (1h)
HEPSPEC	2362,5	1995	1087,5	322,5

Tabela 29 - Grupo A - Contrato *Site* III

OV	CMS	ATLAS	ALICE	LHCB
C_{max}	169	142	78	23
T_{max} (s)	3600 (1h)	3600 (1h)	3600 (1h)	3600 (1h)
HEPSPEC	1575	1330	725	215

No grupo B temos 3 clusters cada cluster com um perfil e uma política especifica para cada OV, semelhantemente ao cluster A.

Tabela 30 - Grupo B - Perfil dos *Sites*

Site	**I**	**II**	**III**
C_n	4112	3084	2056
V_n	4	4	4
PeRating (SPEC)	2337,670	2337,670	2337,670
SO	SL4	SL4	SL4
BaundRate (bits/s)	10Gb	10Gb	10Gb
SPEC_total	9612500	7209375	4806250
HEPSPEC_total	38450	28837,5	19225

Tabela 31 - Grupo B - Contrato *Site* I

OV	CMS	ATLAS	ALICE	LHCB
C_{max}	1684	1423	775	230
T_{max} (s)	3600 (1h)	3600 (1h)	3600 (1h)	3600 (1h)
HEPSPEC	15750	13300	7250	2150

Tabela 32 - Grupo B- Contrato *Site* II

OV	CMS	ATLAS	ALICE	LHCB
C_{max}	1263	1067	582	172
T_{max} (s)	3600 (1h)	3600 (1h)	3600 (1h)	3600 (1h)
HEPSPEC	11812,5	9975	5437,5	1612,5

Tabela 33 - Grupo B - Contrato *Site* III

OV	CMS	ATLAS	ALICE	LHCB
C_{max}	842	711	388	115
T_{max} (s)	3600 (1h)	3600 (1h)	3600 (1h)	3600 (1h)
HEPSPEC	7875	6650	3625	1075

Observamos que mesmo reduzindo à metade a capacidade de processamento do cluster em HEPSPEC, o total obtido é um pouco maior que a capacidade de processamento necessária para a OV que necessita de mais recursos. Na análise dos resultados, veremos como a política se comporta em cada um destes cenários.

A seguir, temos a configuração do arquivo com os parâmetros da simulação. O contrato das OVs com os limites máximos de utilização estão definidos aqui. Foi definido o número máximo de processadores que podem ser utilizados por cada OV em caso de pico e o tempo máximo de processamento total de cada OV sem que haja preempção. Neste exemplo, temos um contrato que atende ao Cluster I do Grupo A. Aqui também são definidas algumas características do perfil do *site*, como por exemplo, o número de processadores e a velocidade de cada processador. Foram colocados neste arquivo alguns parâmetros utilizados na criação do *WorkloadRamp*, como por exemplo, o número de passos da rampa e a duração mínima e máxima dos *jobs*.

```
package broker_new;
import java.math.BigInteger;
public class SimulationParameters {
      public static final long SEC = 1;
      public static final long MIN = 60 * SEC;
      public static final long HOUR = 60 * MIN;
      public static final long DAY = 24 * HOUR;
      public static final int NumberOfUsers = 5000;
      public static final int PeRating= 2336;
      public static final int TotalPe = 1;
      public static final int TotalMachines = 823;
      public static final int TotalPesInUse = 823
      public static final double BaundRate = 2000000000.0;
      public static final long Seed = 22L*26*34*38*46+2;
      public static final int MaxJobsVoCMS= 337;
      public static final int MaxJobsVoATLAS= 285;
      public static final int MaxJobsVoALICE= 155;
      public static final int MaxJobsVoLHCB= 46;
      public static final double  MaxCpuTimeDayVoCMS= HOUR;
      public static final double MaxCpuTimeDayVoATLAS= HOUR;
      public static final double MaxCpuTimeDayVoALICE= HOUR;
      public static final double MaxCpuTimeDayVoLHCB= HOUR;
      public static final String slog = “brokerWLCG.txt";
      public static final String rlog = “genRamp.txt";
      public static final String glog = “gridRet.txt";
      public static final int NumberOfVOs= 4;
      public static final int MINJobs1HourCMS = 50;
      public static final int MINJobs1HourATLAS = 50;
      public static final int MINJobs1HourALICE = 50;
      public static final int MINJobs1HourLHCB = 50;
      public static final int MAXJobs1HourCMS = 1500;
      public static final int MAXJobs1HourATLAS = 2500;
      public static final int MAXJobs1HourALICE = 1500;
      public static final int MAXJobs1HourLHCB = 2500;
      public static final int rampStepsNumber = 3;
      public static final double TempoMaximoTotalNoCluster = 43200.0;
}
```

5.5 A política

O *broker* foi implementado no GridSim através da criação de 3 classes. A primeira classe *Broker* é responsável pela comunicação entre os *jobs* gerados pelo usuário e o recurso. No GridSim a comunicação entre as entidades é feita através do envio e recebimento de eventos. Cada entidade é configurada para receber eventos vindos de outra entidade. Quando um *job* é submetido pelo usuário, um evento de submissão é enviado para o *broker*. Quando um *job* termina no recurso, um evento de término é enviado do recurso para o *broker*. A classe *Broker* é responsável pela comunicação descrita acima.

A segunda classe, *BrokerNoPolicy,* é uma extensão da classe *Broker*. Esta classe não possui política de alocação de tarefas. Ela apenas coleta as informações de submissão dos *jobs*, tempo de execução e etc. para a geração dos logs de um

site. Esta classe serviu para simular os *sites* que não possuem política de alocação de *jobs.*

A terceira classe, *BrokerWLCG,* é onde a política foi implementada. Nesta classe ocorre o gerenciamento das filas de *jobs* pausados e em espera.

Cada submissão e término de *job* representam um evento que é processado no *broker*. O esvaziamento e o povoamento das filas no *broker* são feitos em função dos eventos recebidos pelo *broker*. Por exemplo, se um *job* termina, ele envia um evento de término para o *broker*. A partir deste evento, um *job* de uma das filas é liberado para execução. Quando o *broker* recebe o evento de submissão e não existem CPUs livres, o *job* é colocado na fila de espera ou um *job* em execução é colocado na fila de pausados. Na Figura 31 temos o diagrama de classes do *broker*.

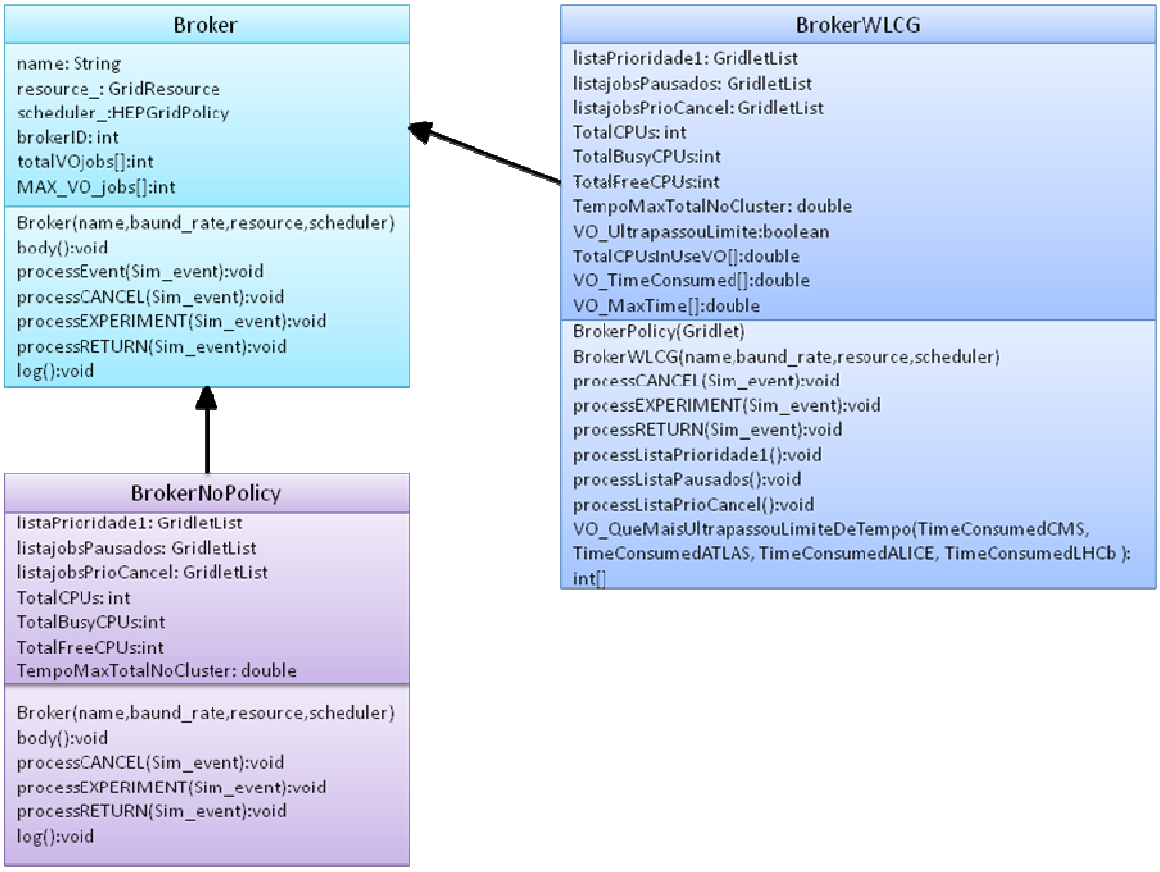

Figura 31 - Diagrama das classes *Broker*, *BrokerNoPolicy* e *BrokerWLCG*

Para introduzir os elementos de suporte necessários à representação dos *jobs* no simulador, optou-se por modificar diretamente a classe *Gridlet* original. Foi adicionado na classe *Gridlet* o parâmetro de identificação da OV de cada *job*. Foi

adicionada a esta classe uma nova opção de status para o *Gridlet* chamada *Broker_Paused*. Este status representa a solicitação de pausa no processamento de um *job* no cluster originada do *broker*.

Na Figura 32 é apresentada a evolução de elementos importantes da modelagem da arquitetura no GridSim e o efeito da política ao longo de 50000 segundos. Podem ser observados os comportamentos das filas de espera e de suspensos e como as CPUs do cluster são usadas. O gráfico do número de *jobs* submetidos de cada OV, acumulados, pode ser confrontado com o uso de CPU que cada OV utiliza ao longo do tempo. Observa-se em torno do tempo de 10000 a atuação da política, quando *jobs* do CMS e do ATLAS são retirados de execução (preempção) para dar lugar aos *jobs* do LHCB e do ALICE (observado pela fila de suspensos e pelo número de CPUs alocados para cada OV). Esta preempção pode ser observada no gráfico da fila de *jobs* suspensos, que começa a crescer no tempo 10000.

Os degraus no número de *jobs* submetido são as rajadas conforme está descrito na seção 5.3. O CMS sofre preempção, mas o mais “punido” é a OV ATLAS que de acordo com o gráfico sofre uma grande queda no número de execuções no tempo 1000. Máximo de CPUs alocado para cada OV é diferente. Devido à alta taxa de utilização do cluster, o número de *jobs* em execução para cada OV coincidiu com o limite máximo estabelecido para execução em caso de pico. Este fato foi observado durante toda a duração da simulação. Observamos também que, na medida em que os *jobs* em execução do CMS diminuem, a fila de *jobs* na fila de espera também diminui. Esses *jobs* devem pertencer as OVs ATLAS, ALICE e LHCB, pois os gráficos de execução permanecem constantes para essas OVs, mesmo com a diminuição da fila de espera.

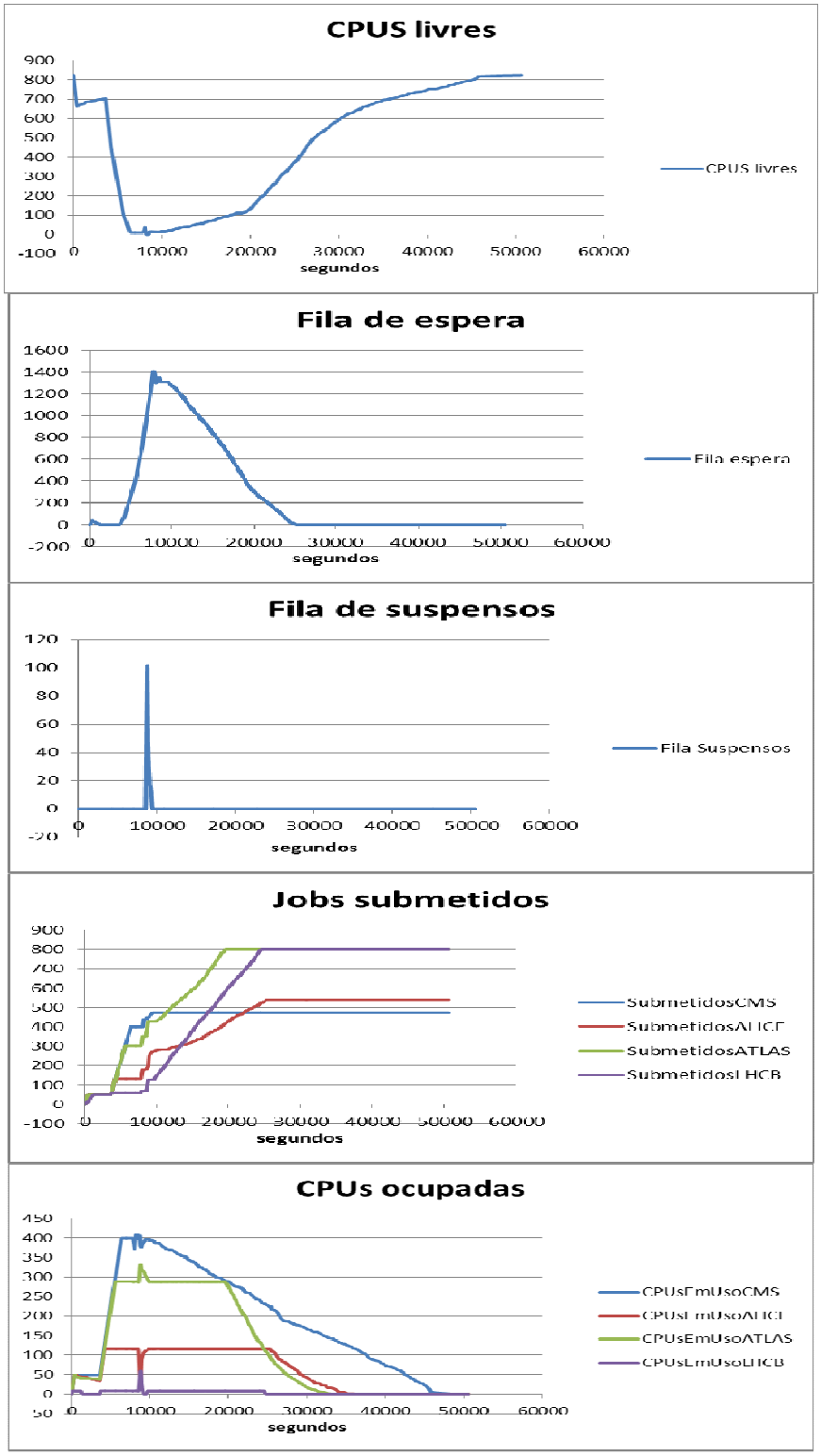

Figura 32 - Evolução dos elementos da arquitetura de acordo c/ a política ao longo de 50000 s

5.6 Logs

Para averiguar o comportamento da política, foi criado um sistema de logs customizado para expor os detalhes das 4 OVs, separando os eventos de cada uma delas e discriminando eventos específicos. Abaixo temos uma chamada ao sistema de *logs*. Esta chamada era feita em qualquer alteração de status de um *job*. No log foi registrado para cada evento, o tipo do evento (se estava entrando ou saindo de uma fila, se era uma submissão e etc.), o número de CPUs livres e em uso por cada OV, a quantidade de *jobs* em cada fila e o tempo em que o evento ocorreu.

```
1     cLog (
2           clock(),                    // tempo do evento
3           "in-fila1",                 // tipo do evento
4           TotalFreeCPUs,              // quantidade de CPUS livres
5           TotalCPUsInUseVO,           // CPUs sendo usadas por cada VO
6           listaPrioridade1.size(),   // na fila prio1
7           listajobsPausados.size(), // na fila suspensos
8           gl.getGridletVO(),          // ID da VO
9           false,                      // é uma submissão?
10          false,                      // é término
11          0                           // tempo de processamento se terminou
12    );
```

Na linha 2 é feita a chamada à função *clock()* que obtém o tempo, em segundos, em que o evento é gerado. Na linha 3 é descrito o tipo do evento que está sendo logado. Se o evento é uma submissão, se é uma pausa, se o *job* está sendo encaminhado ou retirado de alguma das filas ou se é um término de execução. Na linha 4 é logado o número total de CPUs livres no cluster. Na linha 5 é registrado o número total de CPUs em uso por cada OV. Na linha 6 é logado o número de *jobs* presentes na fila de espera. Na linha 7 é registrado o número de *jobs* presentes na fila de pausados. Na linha 8 é logado a OV do *job* cujo log está sendo registrado. Na linha 8 é registrado se o evento se trata de uma submissão nova. Um *job* pode estar sendo submetido para o cluster sendo que ele anteriormente estava pausado. Este parâmetro serve para evitar que *jobs* pausados sejam contabilizados como uma nova submissão. Na linha 10 é registrado se o evento se trata de um término de *job*.

Além deste log, foram utilizadas funções presentes no GridSim para o cálculo do tempo de processamento e do tempo de submissão para cada OV.

6 RESULTADOS

Neste capítulo, são analisados os resultados obtidos através da simulação da política proposta. Inicialmente, foram previstos diversos parâmetros para avaliar a qualidade da política proposta entre eles o tempo de submissão do *job*, o tempo de espera para execução, tempo de início de execução, tempo de término, o tempo de permanência total no cluster e o número de *jobs* expirados. Abaixo está a definição de cada um destes parâmetros:

- Tempo de Submissão: que representa o momento exato em que o *job* foi submetido à política do *broker*.
- Tempo de espera para execução: é a diferença entre o tempo de submissão e o tempo de início de execução. Representa quanto tempo o *job* aguardou para ser executado desde quando ele chegou à política do *broker*.
- Tempo de início de execução: momento em que o *job* chega ao nó do cluster e começa a ser processado.
- Tempo de Processamento: tempo que o *job* leva para ser processado dentro do nó.
- Tempo de término: tempo em que o *job* termina de ser processado dentro do nó.
- Tempo total no cluster: representa a diferença entre o tempo de término e o tempo em que ele foi submetido ao *broker*.
- *Jobs* expirados: representa o número de *jobs* que não completaram com sucesso devido ao *proxy* ter expirado antes do término da tarefa.

Entre os parâmetros observados, destacamos os três mais significativos para a avaliação do desempenho da política proposta: o tempo de espera para execução, o tempo total no cluster e o número de *jobs* expirados.

Para cada um dos parâmetros destacados temos o resultado nos 3 cenários. Para melhorar a qualidade dos dados coletados, foram feitas 5 rodadas de simulações com sementes para geração de números aleatórios diferentes. Foram

executadas 5 rodadas de simulação com a política e 5 rodadas sem a política para cada um dos cenários. A partir dos resultados obtidos, os gráficos com a média de cada parâmetro para cada experimento foram gerados. Para cada média, foi calculado o intervalo de confiança de 90%, considerando uma distribuição *t de Student*, ainda que pelo número de repetições pudesse ser usada a distribuição normal. O intervalo de confiança das médias está representado no gráfico pela barra de erros.

Para cada um dos parâmetros foram obtidos os resultados com e sem a política. Além das médias totais, foram obtidas as médias para cada experimento para cada um dos parâmetros em cada cenário. A avaliação da política para cada cenário foi feito em termos percentuais. Através dos resultados, analisamos como a política se comportou individualmente para cada OV e em conjunto nos diversos cenários. Os gráficos desta seção mostram o resultado das simulações para o cenário dos grupos A (clusters I, III e III) e B (clusters I, II e III).

Nas Tabelas 34, 35 e 36 temos os resultados para cada um dos parâmetros analisados para a avaliação da política, que são respectivamente o tempo de espera para execução, o tempo total no cluster e os *jobs* com *proxy* expirados. A última coluna das Tabelas 34, 35 e 36 tem o desempenho da política em termos percentuais. Os valores positivos representam o aumento de tempo em termos percentuais com a utilização da política. Os valores negativos representam a redução do tempo em termos percentuais com a utilização da política.

6.1 Tempo de Espera de Execução

No gráfico da Figura 33 vemos um aumento considerável do tempo de espera para execução dos *jobs* do ALICE e do LHCb, cerca de 184,48% e 86,19% respectivamente. Os *jobs* do ATLAS tiveram um pequeno aumento de 2,95%. Os *jobs* do CMS tiveram uma redução de 55,8% no tempo de espera.

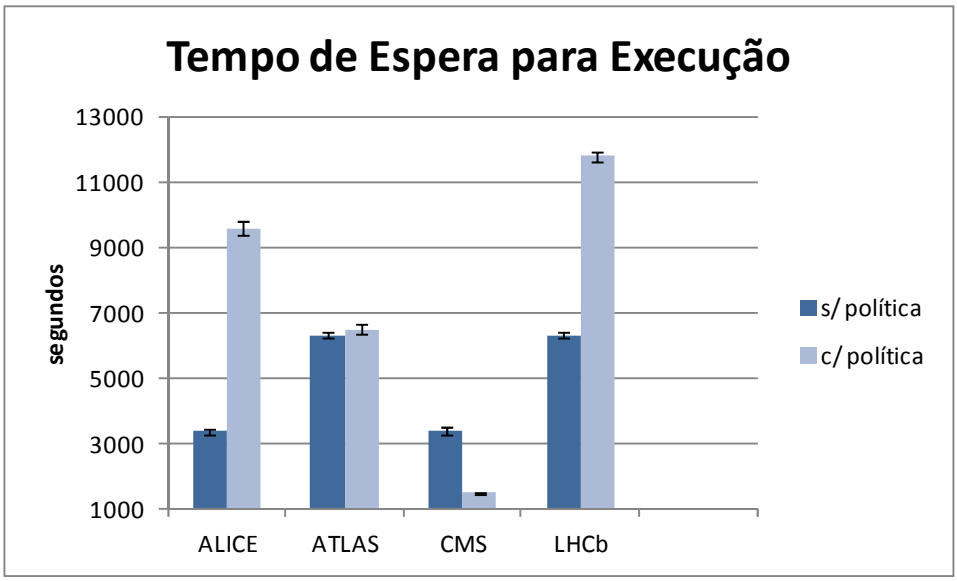

Figura 33 - Tempo de Espera para Execução – *Site* A Cluster I

Vemos na Figura 33 que os *jobs* no cluster II gastaram mais tempo para começar a ser processados em relação aos *jobs* no cluster I. Em termos relativos, vemos um aumento considerável do tempo de espera para execução dos *jobs* do ALICE e do LHCb, cerca de 140,02% e 57,22% respectivamente. No entanto, este aumento é menor, em termos percentuais, que o aumento observado no cluster I. Os *jobs* do ATLAS tiveram um aumento de 21,38%. Os *jobs* do CMS tiveram uma redução de 19,37% no tempo de espera, redução menor que a observada no cluster I.

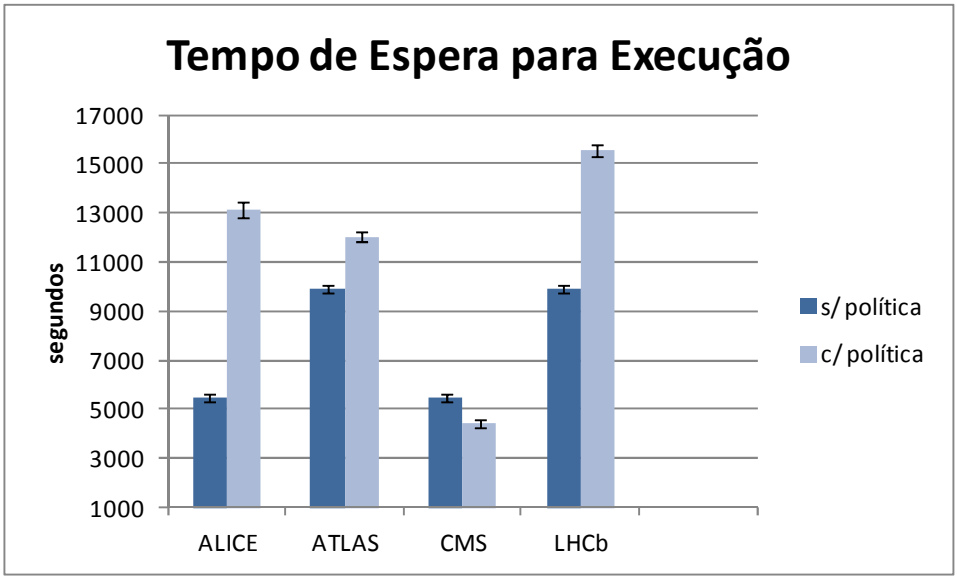

Figura 34 - Tempo de Espera para Execução – *Site* A Cluster II

Na Figura 35 vemos que os *jobs* no cluster III gastaram mais tempo para começar a ser processados em relação aos *jobs* no cluster II. Em termos relativos, vemos um aumento considerável do tempo de espera para execução dos *jobs* do ALICE, cerca de 96,99%. Os *jobs* do CMS do ATLAS e do LHCb, tiveram o aumento de 6,65%, 8,47% e 19,70% respectivamente.

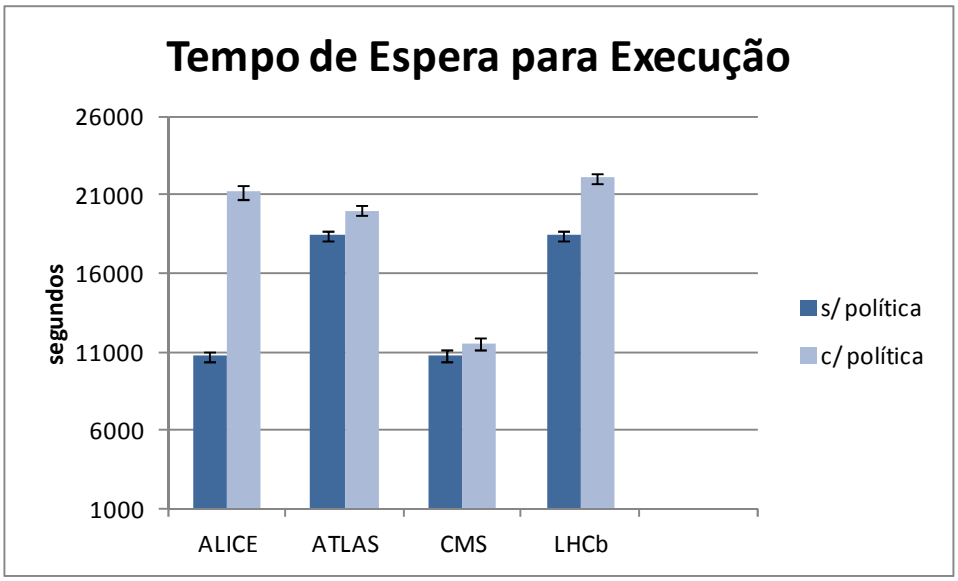

Figura 35 - Tempo de Espera para Execução – *Site* A Cluster III

No gráfico da Figura 36 vemos um aumento considerável do tempo de espera para execução dos *jobs* do LHCb, cerca de 92,42%. Os *jobs* do ALICE tiveram um aumento de 41,22%. Os *jobs* do ATLAS e do CMS tiveram uma redução no tempo de espera de 14,95% e 32,08% respectivamente.

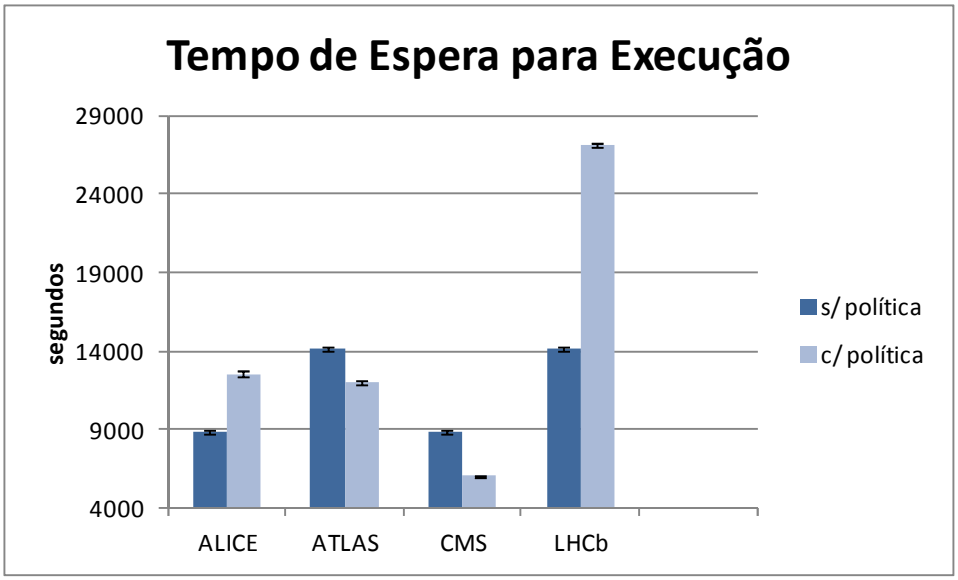

Figura 36 - Tempo de Espera para Execução – *Site* B Cluster I

Vemos na Figura 37 que os *jobs* no cluster II gastaram mais tempo para começar a ser processados em relação aos *jobs* no cluster I. Em termos relativos, vemos um aumento do tempo de espera para execução dos *jobs* do LHCb e do ALICE, cerca de 67,13% e 36,34% respectivamente. No entanto, este aumento é menor, em termos percentuais, que o aumento observado no cluster I. Os *jobs* do ATLAS e do CMS tiveram uma redução no tempo de espera de 14,18% e 46,47% respectivamente. Em termos percentuais a redução no tempo de espera dos *jobs* do CMS foi maior que a observada no cluster I Os *jobs* do ATLAS tiveram desempenho semelhante no cluster I e no cluster II.

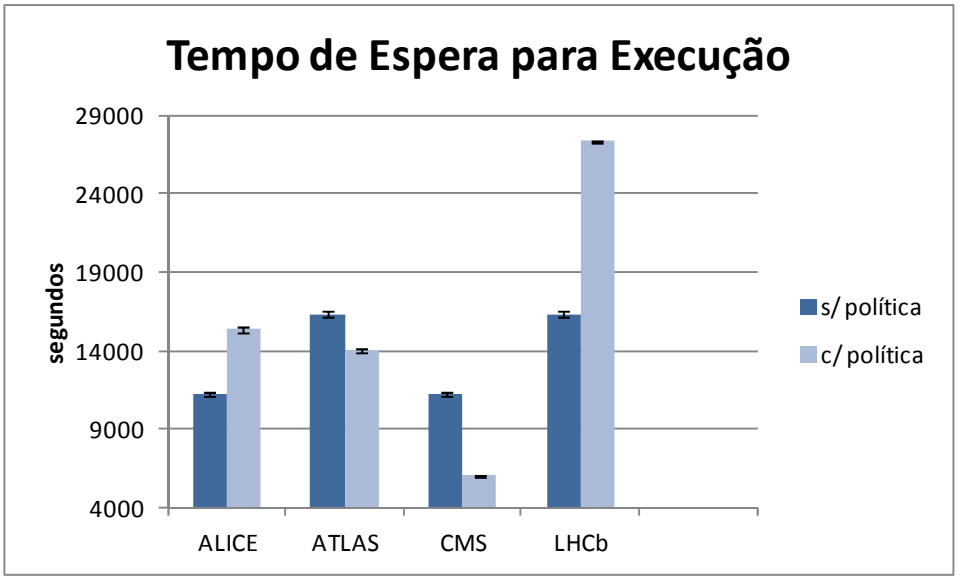

Figura 37 - Tempo de Espera para Execução – *Site* B Cluster II

Na Figura 38 vemos que os *jobs* no cluster III gastaram mais tempo para começar a ser processados em relação aos *jobs* no cluster II. Em termos relativos, vemos um aumento do tempo de espera para execução dos *jobs* do LHCb do, ALICE e do ATLAS de cerca de 40,85%, 54,69% e 10,3% respectivamente. Os *jobs* do do CMS tiveram uma redução no tempo de espera de 35,36%.

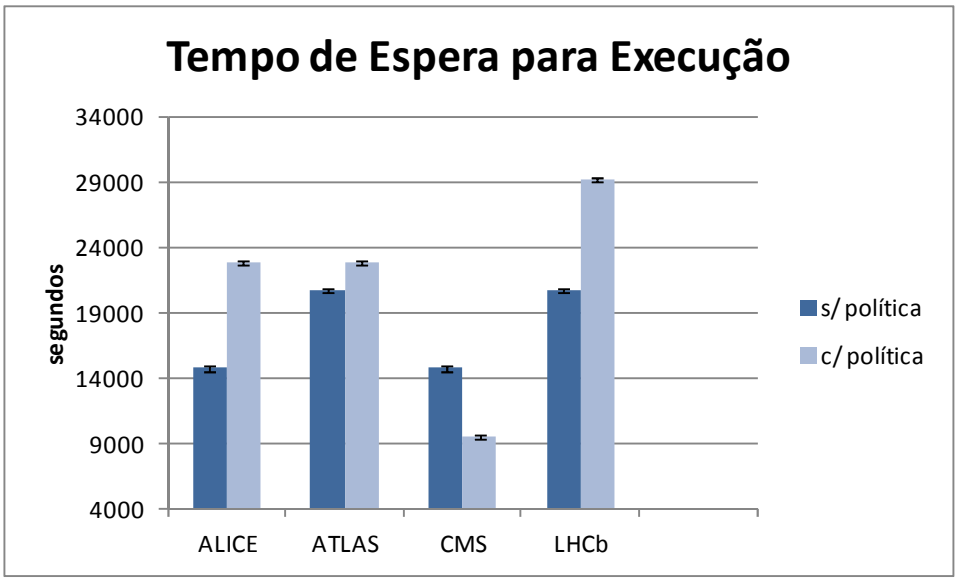

Figura 38 - Tempo de Espera para Execução – *Site* B Cluster III

Na Tabela 34 temos as médias do tempo de espera para a execução de cada experimento em cada um dos cenários. Em todos os *sites* observamos que com a política a média total do tempo de espera para a execução foi maior. O percentual de aumento para os *sites* do grupo A ficou entre 23,38% e 48,99%. Nos *sites* do grupo B o percentual de aumento ficou entre 17,06% e 29,17%. No grupo A o cluster com a menor porcentagem de aumento foi o cluster III. No grupo B o cluster com a menor porcentagem de aumento foi o cluster II.

Apesar de ter sido maior em tempos percentuais para os três clusters, observamos que no cluster I o tempo de espera em termos percentuais foi maior que no cluster III. Neste caso, verificamos que o tempo de espera para execução melhorou em tempos percentuais nos casos onde se tinha menos recursos computacionais disponíveis.

Apesar do aumento do tempo de execução total, foi observado através dos resultados que o resultado individual para cada OV oscilou, sendo que algumas tiveram o desempenho melhor e outras tiveram desempenho pior. Observamos que a melhora ou piora dos resultados de uma determinada OV está diretamente relacionado com as características de cada OV. Por exemplo, as OVs cujos *jobs* são mais longos tendem a esperar menos tempo para iniciar a execução.

As OVs cujo volume de submissões é maior também tendem a esperar

menos que as outras. No geral, as OVs que tiveram melhor desempenho no grupo A foram primeiro o CMS seguido do ATLAS, do LHCb e por último do ALICE. No grupo B o melhor desempenho foi do CMS seguido do ATLAS, do ALICE (nos clusters I e II) e depois do LHCb (nos clusters I e II).

Em todos os casos, com exceção do grupo A (cluster III), observou-se uma redução do tempo de espera para a execução dos *jobs* do CMS. A explicação para este fato é que os *jobs* do CMS foram privilegiados pela política por terem o tempo de execução maior. Percebeu-se que os *jobs* das OVs que tiveram menos recursos alocados demoraram mais tempo para começar a ser processados. Neste caso, a política privilegia a execução dos *jobs* das OVs mais gulosas. Nos *sites* I e II do grupo B observou-se uma redução to tempo de espera dos *jobs* do ATLAS.

Tabela 34 - Tempo de espera para execução

Tempo de Espera para Execução (s)					
GRUPO	CLUSTER	VO	Sem política	Com política	%
A	I	ALICE	3363,68	9569	184,48
		ATLAS	6319,40	6506	2,95
		CMS	3384,21	1496	-55,8
		LHCb	6319,99	11767	86,19
		TOTAL	5259,23	7836	49
	II	ALICE	5474,94	13141,17	140,02
		ATLAS	9898,1	12014,14	21,38
		CMS	5498,4	4433,24	-19,37
		LHCb	9901,7	15567,71	57,22
		TOTAL	8310,44	11986,7	44,24
	III	ALICE	10748	21173	96,99
		ATLAS	18431	19992	8,47
		CMS	10789	11506	6,65
		LHCb	18430	22061	19,70
		TOTAL	15672	19337	23,39
B	I	ALICE	8884	12546	41,22
		ATLAS	14092	11985	-14,95
		CMS	8893	6040	-32,08
		LHCb	14090	27112	92,42
		TOTAL	12108	15640	29,17
	II	ALICE	11274	15371	36,34
		ATLAS	16352	14033	-14,18
		CMS	11283	6040	-46,47
		LHCb	16353	27330	67,13
		TOTAL	14420,79	16881	17,06
	III	ALICE	14760	22832	54,69
		ATLAS	20687	22817	10,3
		CMS	14764	9543	-35,36
		LHCb	20690	29142	40,85
		TOTAL	18430	22248	20,72

Em relação ao tempo total no cluster, observamos que os *jobs* de algumas OVs ficaram menos tempo no cluster. Outras OVs ficaram mais tempo no cluster. Os *jobs* de todas as OVs em conjunto, ficaram mais tempo dentro do cluster. Os *jobs* que tiveram melhor desempenho foram os do CMS e do ATLAS. Os *jobs* do LHCb tiveram o desempenho prejudicado pois são os mais curtos e de acordo com a política proposta, essa OV teve menos recursos disponíveis em caso de pico. No grupo A o cluster com a menor porcentagem de aumento foi o cluster III. No grupo B o cluster com a menor porcentagem de aumento foi o cluster II.

6.2 Tempo Total no Cluster

No gráfico da Figura 39 vemos um aumento do tempo total no cluster dos *jobs* do ALICE e do LHCb, cerca de 68,25% e 83,56% respectivamente. Os *jobs* do ATLAS tiveram um pequeno aumento de 1,55%. Os *jobs* do CMS tiveram uma redução de 11,18% no tempo total no cluster.

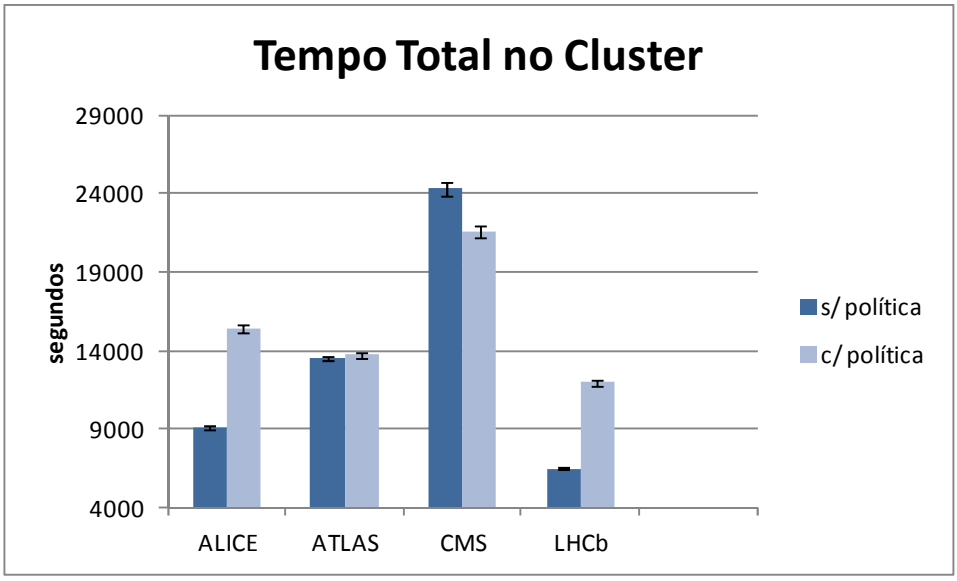

Figura 39 - Tempo Total no Cluster – *Site* A Cluster I

Vemos na Figura 40 que os *jobs* no cluster II gastaram mais tempo no cluster em relação aos *jobs* no cluster I. Em termos relativos, vemos um aumento do tempo total no cluster dos *jobs* do ALICE, do ATLAS e do LHCb, cerca de 67,92%, 11,82% e 56,12% respectivamente. Os *jobs* do CMS tiveram uma redução de 2,87% no tempo total no cluster.

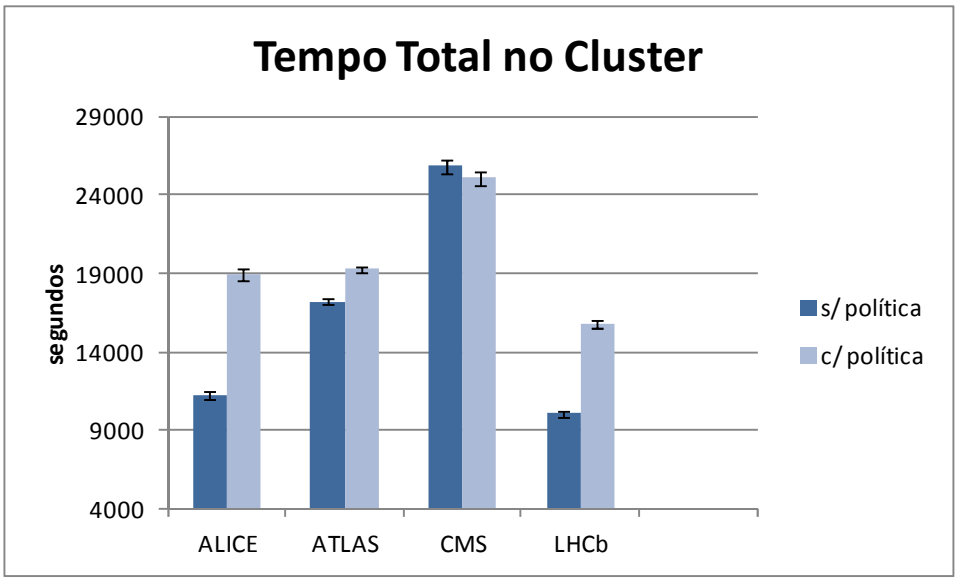

Figura 40 - Tempo Total no Cluster – *Site* A Cluster II

Na Figura 41 vemos que os *jobs* no cluster III gastaram mais tempo no cluster em relação aos *jobs* no cluster II. Os *jobs* do ALICE, ATLAS, CMS e do LHCb, tiveram o aumento de 63,48%, 5,66%, 2,4% e 19,5% respectivamente.

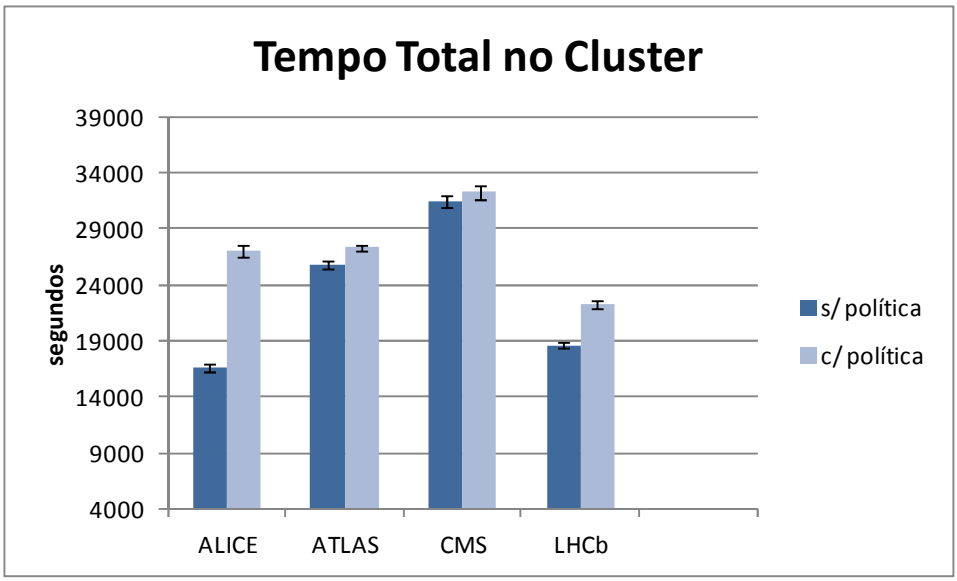

Figura 41 - Tempo Total no Cluster – *Site* A Cluster III

No gráfico da Figura 42 vemos um aumento considerável do tempo total no cluster dos *jobs* do LHCb, cerca de 91,11%. Os *jobs* do ALICE tiveram um aumento de 24,37%. Os *jobs* do ATLAS e do CMS tiveram uma redução no tempo total no cluster de 10,11% e 9,22% respectivamente.

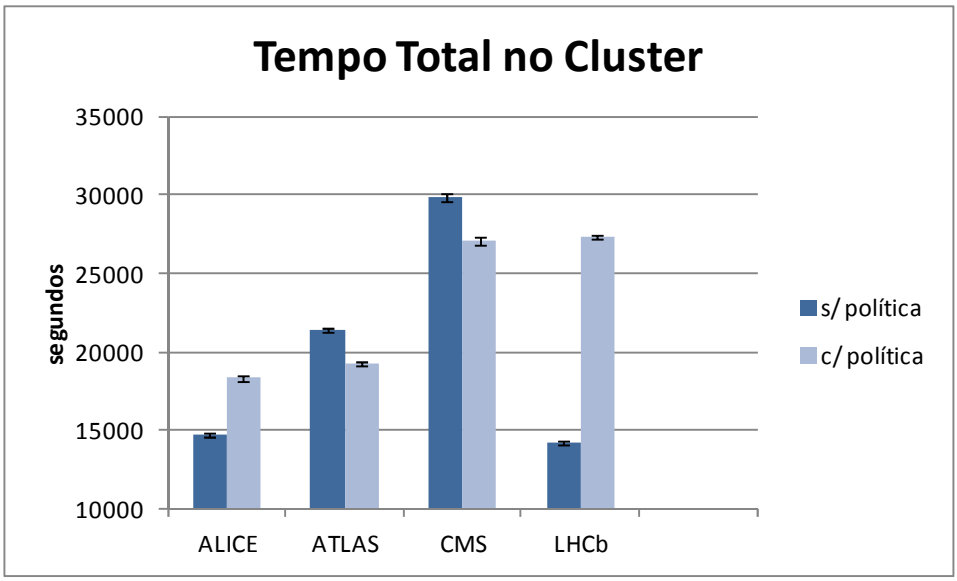

Figura 42 - Tempo Total no Cluster – *Site* B Cluster I

Vemos na Figura 43 que os *jobs* no cluster II gastaram mais tempo no cluster em relação aos *jobs* no cluster I. Em termos relativos, vemos um aumento do tempo total no cluster dos *jobs* do ALICE e do LHCb, cerca de 24,32% e 66,31% respectivamente. Os *jobs* do ATLAS e do CMS tiveram uma redução de 9,96% e 15,32% no tempo total no cluster respectivamente.

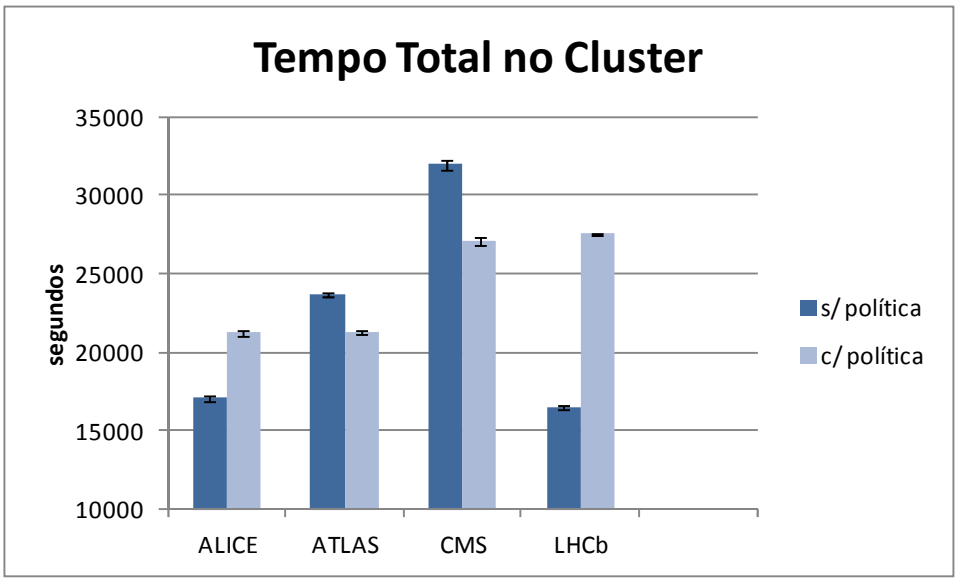

Figura 43 - Tempo Total no Cluster – *Site* B Cluster II

Na Figura 44 vemos que os *jobs* no cluster III gastaram mais tempo no cluster em relação aos *jobs* no cluster II. Em termos relativos, vemos um aumento do tempo total no cluster dos *jobs* do ALICE, do LHCb e do ATLAS, cerca de 38,99%, 40,45%

e 7,72% respectivamente. Os *jobs* do CMS tiveram uma redução de 14,9% no tempo total no cluster.

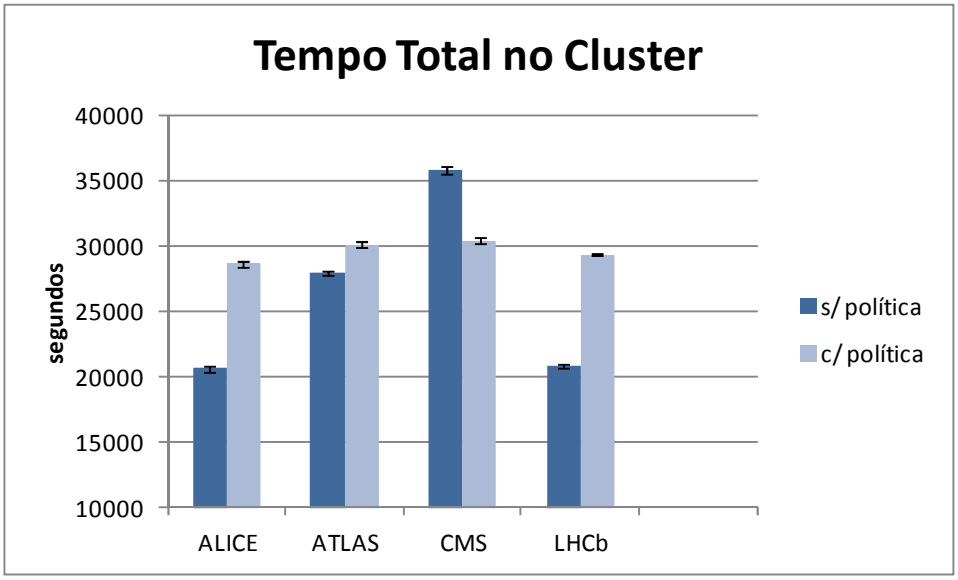

Figura 44 - Tempo Total no Cluster – *Site* B Cluster III

Na Tabela 35 temos as médias do tempo total no cluster de cada experimento em cada um dos cenários. Em todos os *sites* observamos que com a política a média total do tempo total no cluster foi maior. O percentual de aumento para os *sites* do grupo A ficou entre 15,97% e 24,03%. Nos *sites* do grupo B o percentual de aumento ficou entre 11,59% e 18,00%. No grupo A o cluster com a menor porcentagem de aumento foi o cluster III. No grupo B o cluster com a menor porcentagem de aumento foi o cluster II.

Observamos que no cluster I o tempo total no cluster em termos percentuais foi maior que no cluster III. Verificamos que o tempo de espera para execução melhorou em tempos percentuais nos casos onde se tinha menos recursos computacionais disponíveis. Este resultado foi semelhante ao do tempo de espera para execução.

Apesar do aumento do tempo total no cluster, foi observado através dos resultados que o resultado individual para cada OV oscilou, sendo que algumas tiveram o desempenho melhor e outras tiveram desempenho pior.

As OVs cujo volume de submissões é maior também tendem a esperar menos que as outras. No geral, as OVs que tiveram melhor desempenho no grupo A

foram primeiro o CMS seguido do ATLAS, do LHCb (com exceção do cluster I) e por último do ALICE (com exceção do cluster I). No grupo B o melhor desempenho foi do CMS (com exceção do cluster I) seguido do ATLAS (com exceção do cluster I), do ALICE e depois do LHCb.

Em todos os casos, com exceção do cluster III do grupo A, observou-se uma redução do tempo total no cluster dos *jobs* do CMS. Também observamos a redução do tempo total no cluster para os *jobs* do ATLAS processados nos clusters I e II do grupo B.

Tabela 35 - Tempo total no cluster

Tempo Total no Cluster (s)					
GRUPO	CLUSTER	VO	Sem política	Com politica	%
A	I	ALICE	9132,97	15366	68,25
		ATLAS	13551,32	13762	1,55
		CMS	24292,82	21577,03	-11,18
		LHCb	6520,49	11969	83,56
		TOTAL	12442,05	14951	20,17
	II	ALICE	11273,57	18930,97	67,92
		ATLAS	17223,26	19259,88	11,82
		CMS	25821,29	25080,14	-2,87
		LHCb	10098,84	15765,91	56,12
		TOTAL	15425,58	19132,45	24,03
	III	ALICE	16545	27048	63,48
		ATLAS	25837	27299	5,66
		CMS	31495	32254	2,41
		LHCb	18629	22262	19,50
		TOTAL	22880	26535	15,97
B	I	ALICE	14736	18327	24,37
		ATLAS	21414	19249	-10,11
		CMS	29837	27085	-9,22
		LHCb	14291	27312	91,11
		TOTAL	19543	23062	18,01
	II	ALICE	17086	21242	24,32
		ATLAS	23675	21316	-9,96
		CMS	31942	27049	-15,32
		LHCb	16552	27528	66,31
		TOTAL	21791,11	24317	11,59
	III	ALICE	20610	28646	38,99
		ATLAS	27939	30097	7,72
		CMS	35732	30408	-14,9
		LHCb	20890	29340	40,45
		TOTAL	25847	29646	14,7

6.3 *Jobs* expirados

Em relação ao número de *jobs* expirados, obtivemos uma melhora significativa com a política proposta nos clusters I e II de ambos os grupos. Na Tabela 36 temos as médias do número de *jobs* com *proxy* expirado de cada experimento em cada um dos cenários. No grupo A o cluster I teve o desempenho 78,68% melhor com a política proposta. No grupo B o cluster I teve desempenho 36,98% melhor com a política. Observou-se que os *jobs* do CMS têm maior tendência para ter o *proxy* expirado, pois podem ter o tempo de processamento longo. Devido a este fato, observamos que nos clusters I e II de ambos os grupos, apenas os *jobs* da OV CMS tiveram o *proxy* expirado. Em ambos os grupos no cluster III a média de *jobs* com *proxy* expirado com a política foi maior. No grupo III as OVs que tiveram *jobs* com *proxy* expirado sem a implementação da política foram a OV CMS e a OV ATLAS. Com a política, todas as OVs tiveram *jobs* com *proxy* expirados.

No gráfico da Figura 45 vemos uma diminuição considerável de 78,69% de *jobs* expirados do CMS. As outras OVs não tiveram *jobs* expirados.

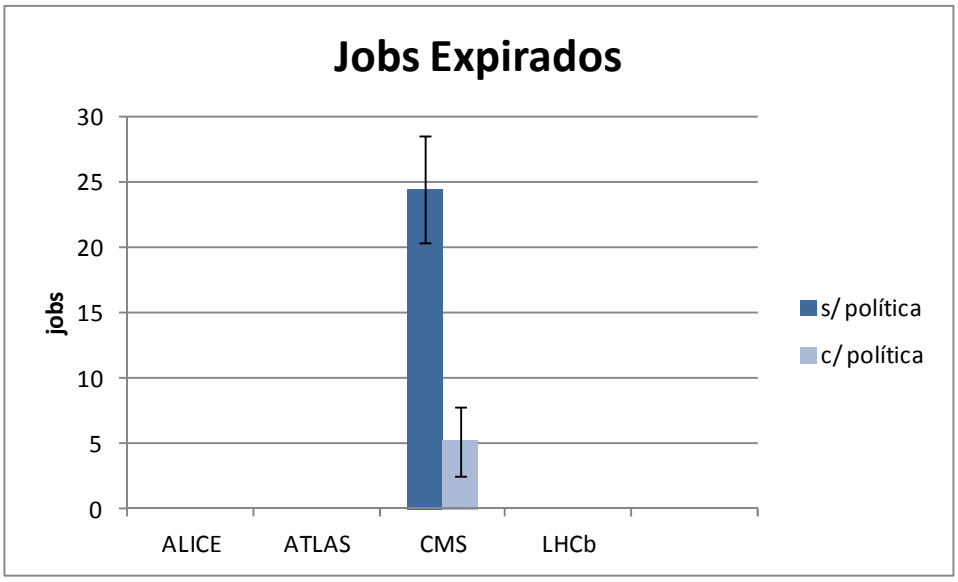

Figura 45 - *Jobs* Expirados – *Site* A Cluster I

No gráfico da Figura 46 vemos uma diminuição de 13,7% de *jobs* expirados do CMS. As outras OVs não tiveram *jobs* expirados.

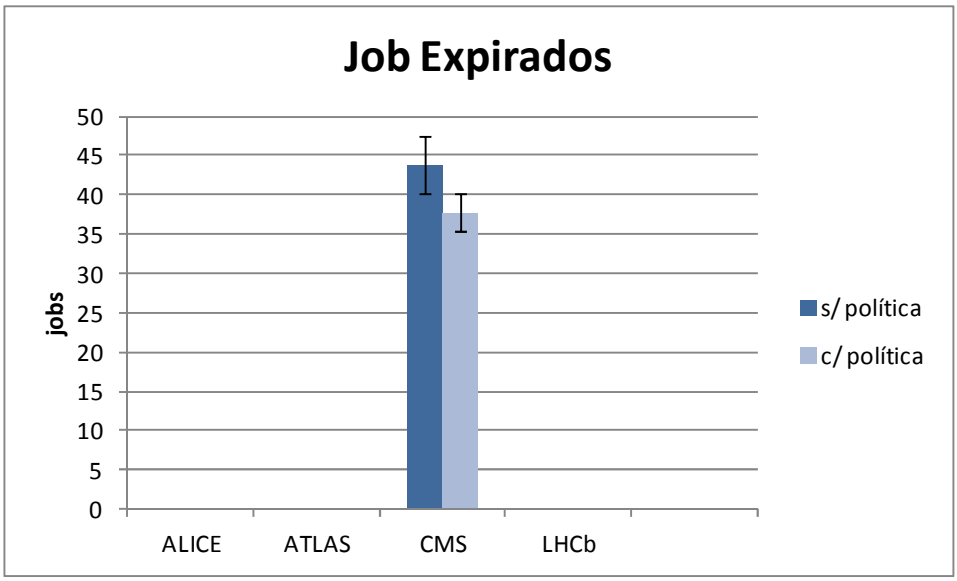

Figura 46 - *Jobs* Expirados – *Site* A Cluster II

No gráfico da Figura 47 vemos um aumento do número de *jobs* expirados do ATLAS e do CMS de cerca de 135% e 3,77% respectivamente. As outras OVs tiveram *jobs* expirados após a implantação da política.

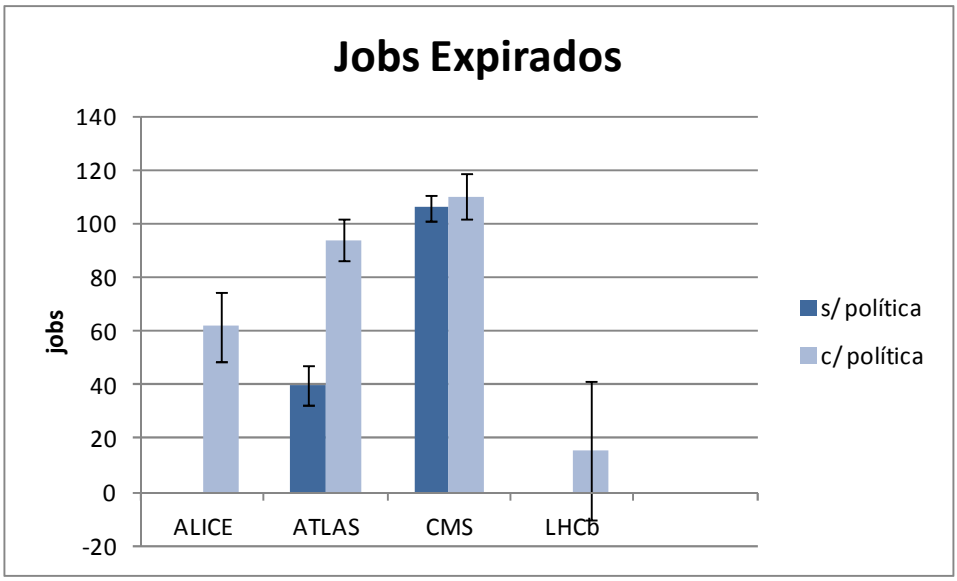

Figura 47 - *Jobs* Expirados – *Site* A Cluster III

No gráfico da Figura 48 vemos uma diminuição de 36,99% de *jobs* expirados do CMS. As outras OVs não tiveram *jobs* expirados.

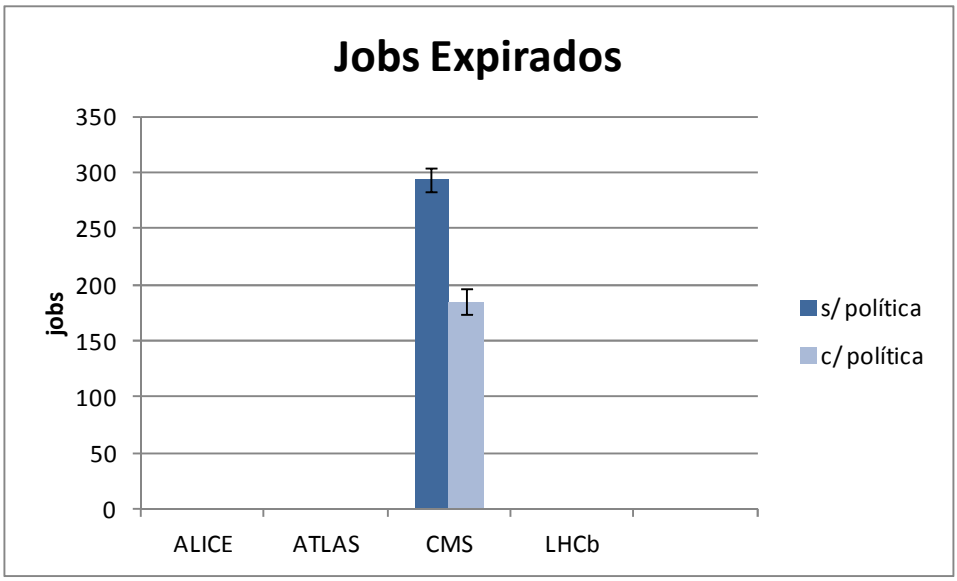

Figura 48 - *Jobs* Expirados – *Site* B Cluster I

No gráfico da Figura 49 vemos uma diminuição de 51,97% de *jobs* expirados do CMS. As outras OVs não tiveram *jobs* expirados.

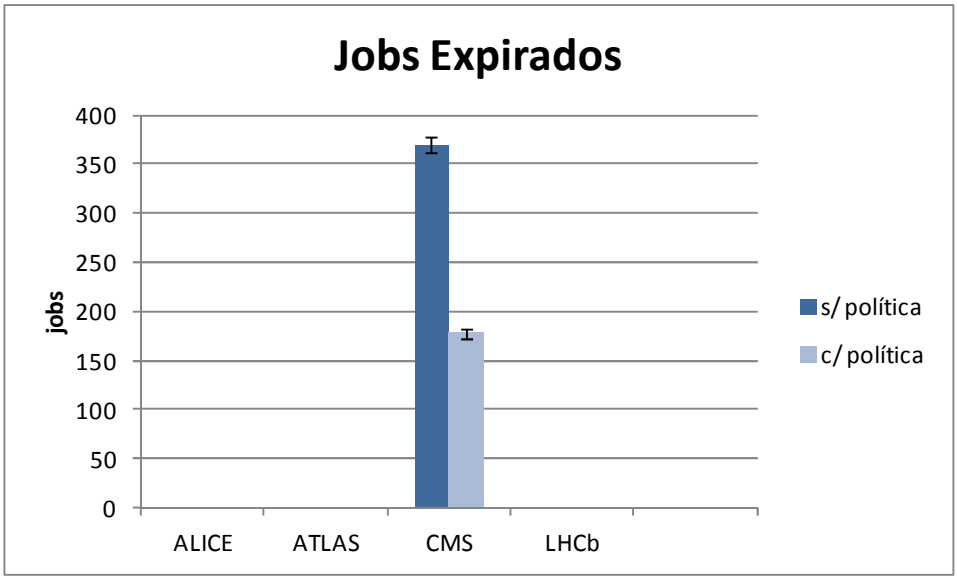

Figura 49 - *Jobs* Expirados – *Site* B Cluster II

No gráfico da Figura 50 vemos um aumento do número de *jobs* com *proxy* expirado do ATLAS de cerca de 222,61%. Os *jobs* do CMS tiveram uma redução de 51,97% As outras OVs tiveram *jobs* expirados após a implantação da política.

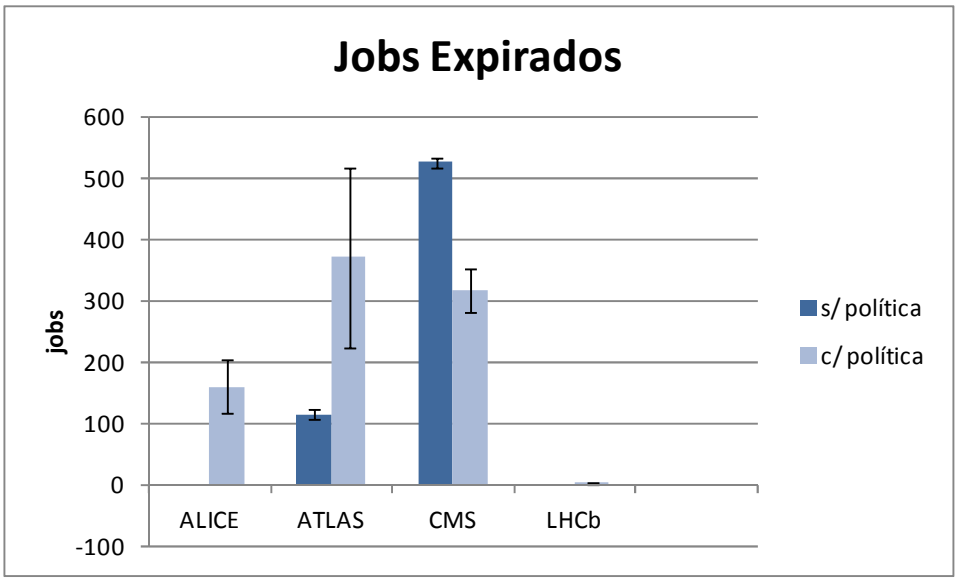

Figura 50 - *Jobs* Expirados – *Site* B Cluster III

Na Tabela 36 temos as médias do número de *jobs* com *proxy* expirado de cada experimento em cada um dos cenários. Observamos que no cluster I o tempo total no cluster em termos percentuais foi maior que no cluster III. Verificamos que o tempo de espera para execução melhorou em tempos percentuais nos casos onde se tinha menos recursos computacionais disponíveis. Este resultado foi semelhante ao do tempo de espera para execução.

Tabela 36 - *Jobs* com *proxy* expirado

Jobs com *proxy* expirado					
GRUPO	CLUSTER	VO	Sem política	Com politica	%
A	I	ALICE	0	0	0
		ATLAS	0	0	0
		CMS	24,4	5,2	-78,69
		LHCb	0	0	0
		TOTAL	24,4	5,2	-78,69
	II	ALICE	0	0	0
		ATLAS	0	0	0
		CMS	43,8	37,8	-13,7
		LHCb	0	0	0
		TOTAL	43,8	37,8	-13,7
	III	ALICE	0	61,6	0
		ATLAS	40	94	135
		CMS	106	110	3,77
		LHCb	0	15,6	0
		TOTAL	146	281	92,47
B	I	ALICE	0	0	0
		ATLAS	0	0	0
		CMS	293,6	185	-36,99
		LHCb	0	0	0
		TOTAL	293,6	185	-36,99
	II	ALICE	0	0	0
		ATLAS	0	0	0
		CMS	370,2	177,8	-51,97
		LHCb	0	0	0
		TOTAL	370,2	177,8	-51,97
	III	ALICE	0	160	-
		ATLAS	115	371	222,61
		CMS	525,2	316,6	-39,72
		LHCb	0	1,2	-
		TOTAL	640,2	848,8	32,58

A partir dos resultados obtidos, concluímos que a política proposta é aplicável para se reduzir o número de *jobs* com *proxy* expirados (ou seja, menos *jobs* são abortados), nos cenários com um cluster dimensionado com 100% e 75% de recursos do total necessário para cada VO. O preço da aplicação da política é o aumento do tempo de permanência no cluster e o tempo de espera para iniciar a execução.

6.4 Comentários

Para os cenários com clusters com número reduzido de máquinas, tipicamente 50% do total, a política não se mostrou eficiente. Houve um aumento do número de *jobs* com *proxy* expirados. Estes resultados facilitam a tomada de decisão pelo cluster que atende a 75% de recursos do total, caso haja menos recursos computacionais disponíveis visto que os resultados obtidos neste cenário foram satisfatórios. A melhor opção continua sendo ter o cluster dimensionado com 100% de recursos do total necessário, pois este cenário obteve os melhores resultados com a política.

Observamos que a melhora ou piora dos resultados de uma determinada OV está diretamente relacionado com as características de cada OV e às decisões empregadas na política proposta. Por exemplo, os *jobs* da OV CMS, tendem a esperar menos tempo para iniciar a execução. Este resultado possivelmente está associado ao fato de se dedicar mais processadores ao CMS em situações de pico. Devido ao critério de dimensionamento adotado, observou-se que a OV CMS foi, de certa forma, privilegiada. O critério de dimensionamento, ou seja, a política adotada, poderia ter sido outro e outros critérios podem ser utilizados em trabalhos futuros.

7 CONCLUSÃO

Na primeira etapa da pesquisa foi feito o levantamento de informações relativas à capacidade computacional dos *sites* da WLCG que atendiam entre 1% e 5% de cada experimento. São os perfis de *sites* em maior número. Foram escolhidos preferencialmente os *sites* que atendiam a um único experimento com exceção do LHCb, pois não foi encontrado um *site* que atende exclusivamente a este experimento. Verificou-se que cada experimento possui exigências mínimas em relação à capacidade computacional a serem satisfeitas. Foram averiguados também os requisitos mínimos para um *site* se tornar uma *Tier-2*. Estas informações foram usadas como ponto de partida para se propor uma política para uso compartilhado pelos quatro experimentos e, posteriormente, como parâmetro de avaliação da proposta.

A partir do levantamento da capacidade computacional de alguns *sites*, percebeu-se que alguns deles possuem uma capacidade computacional maior que a comprometida com o LHC. Esses recursos extras poderiam estar ociosos nestes casos. Verificamos nos gráficos de submissões de alguns destes *sites* em [38] que em alguns períodos aleatórios, o número de submissões era inferior à capacidade computacional do *site*. A partir da informação contida em [15] verificou-se a possibilidade de utilização dos recursos ociosos de um *site* por outra OV.

A partir desta motivação, iniciou-se a pesquisa por uma política que pudesse otimizar a utilização dos recursos computacionais de uma *Tier-2*. O objetivo era permitir a utilização dos recursos excedentes de um *site*, e os recursos eventualmente ociosos. Pensou-se também na possibilidade de um *site* com poucos recursos poder participar de todos os experimentos do LHC mantendo a qualidade do serviço para todos se a política de uso permitisse um compartilhamento eficiente de recursos ao longo do tempo. A política de utilização de recursos deveria ser capaz de atender *sites* com poucos recursos disponíveis e ao mesmo tempo atender *sites* com recursos extras.

Na abordagem adotada utilizamos o conceito de contratos, onde são definidos os requisitos e as restrições de cada experimento com relação à utilização dos recursos. Para cada *site* foram descritos perfis com suas características. Uma política de alocação de recursos foi definida através de um algoritmo de escalonamento dos *jobs*, que utiliza mecanismos de preempção para controlar a

alocação de nós do cluster do *site* dependendo do uso corrente e do uso acumulado de recursos por cada OV. Este algoritmo é executado em um dos elementos da arquitetura batizado de *broker*, dado que o mesmo intermedeia a chegada de novos *jobs* e a alocação dos nós do cluster, e tem com objetivo manter o controle dos contratos de cada OV. A simulação da arquitetura proposta foi feita no simulador de grades GridSim e os resultados obtidos foram avaliados.

Foram simulados diversos cenários para averiguar o comportamento da política em clusters com abundância de recursos e também em clusters com poucos recursos. Foram criados dois grupos de cluster chamados grupo A e grupo B. Cada um destes grupos atendiam a 1% e 4% de cada OV do LHC respectivamente. Dentro de cada grupo foram criados 3 clusters . O primeiro tinha 100% dos recursos necessários para cada OV, o segundo tinha 75% dos recursos e o terceiro 50%.

Os resultados obtidos das simulações mostraram que a política proposta é aplicável para se reduzir o número de *jobs* com *proxy* expirados (ou seja, menos *jobs* são abortados), nos cenários com um cluster dimensionado com 100% e 75% de recursos do total necessário para cada VO. O preço da aplicação da política é o aumento do tempo de permanência no cluster e o tempo de espera para iniciar a execução. Para os cenários com clusters com número reduzido de máquinas, tipicamente 50% do total, a política proposta não se mostrou tão eficiente. Houve um aumento do número de *jobs* com *proxy* expirados.

Estes resultados facilitam a tomada de decisão pelo cluster que atende a 75% de recursos do total, caso haja menos recursos computacionais disponíveis visto que os resultados obtidos neste cenário foram satisfatórios. A melhor opção continua sendo ter o cluster dimensionado com 100% de recursos do total necessário, pois este cenário obteve os melhores resultados com a política. Embora seja um cenário que considera um *cluster* grande, em um *site* com recursos abundantes, os resultados indicam que a proposta permite otimizar o uso do mesmo.

Cada OV tinha um limite de utilização dos recursos que foi calculado em função do tempo de execução dos *jobs* e da quantidade de submissões para cada OV. Na abordagem utilizada, o número de CPUs destinadas a uma OV é diretamente proporcional ao tempo de duração do *job* e ao volume de submissões desta OV. Esta foi uma decisão de implementação. Outras abordagens para o cálculo do número de CPUs para cada OV devem ser avaliadas, isto é, políticas

diferentes podem ser definidas. Uma possibilidade é verificar o comportamento do algoritmo de escalonamento utilizando outros parâmetros de entrada. Ou dividir os recursos do cluster igualmente entre as OVs independentemente das características de submissão de *Jobs* de cada uma.

O *workload* de *jobs* submetido para a avaliação da política é composto de uma rampa com 3 degraus cada. A ideia inicial era simular uma rampa com um cenário mais próximo ao real, ou seja, uma rampa com 24 degraus representando cada degrau representando 1 hora do dia. Entretanto, os recursos computacionais disponíveis não permitiram tal teste. Ainda assim, a rampa continha os mínimos e máximos em relação a picos de submissão simultânea de *jobs*, que é um dos cenários de pior caso.

Alguns caminhos podem ser explorados para aprimorar a proposta. Como mencionado, outras políticas podem ser propostas e avaliadas, alterando-se os parâmetros de mínimos e máximos que fazem, por exemplo, que *jobs* de uma determina OV sofram ou não preempção.

Aspectos relacionados a capacidade de armazenamento e de transferência de arquivos devem ser consideradas em uma avaliação mais abrangente. A sobrecarga das possíveis transferências de arquivo com dados de análise ou bibliotecas que devem ser obtidas nas *Tier-1* ou *Tier-0* não foram consideradas. Além disso, não forma avaliados aspectos da sobrecarga de máquinas virtuais. Nos trabalhos [44] esta sobrecarga foi avaliada em um cluster de máquinas virtuais interconectados por rede GigaBit Ethernet. A sobrecarga para se parar ou transferir uma máquinas virtual, em um cluster bem sintonizado é da ordem de 3,4 segundos, compatível com o tamanho dos *jobs*.

Em nossa proposta original havia uma expectativa de que uma política dinâmica pudesse ser avaliada. Tal política poderia mudar seus limites de atuação de acordo com o estado corrente do cluster e das submissões de *jobs*, ou até mesmo de acordo com o uso prioritário que a administração do *site* quisesse impor. Esta abordagem pode ser investigada futuramente.

Um outro possível trabalho futuro consiste na implementação da política proposta em um cluster para ver o comportamento da política em uma situação real onde ocorre a virtualização. Depois de avaliada a política em um cenário real, seria interessante aplicar a política nos *sites* do LHC que possuem mais recursos disponíveis que o necessário para honrar os compromissos com o LHC. Neste

caso, pode-se diminuir consideravelmente o número de resubmissões de *jobs* devido à expiração do *proxy*.

REFERÊNCIAS

[1] KEAHEY, Katarzyna; FOSTER, Ian; FREEMAN, Timothy; ZHANG, Xuehai; GALRON. Daniel. Virtual Workspaces in the Grid. In: INTERNATIONAL EURO-PAR CONFERENCE, 11., 2005, Lisboa.

[2] BUNCIC, Predrag; SANCHEZ, Carlos Aguado; BLOMER, Jakob; FRANCO, Leandro; KLEMER, Steffen; MATO, Pere. *CernVM*: A virtual Appliance for LHC Applications. In: INTERNATIONAL WORKSHOP ON ADVANCED COMPUTING AND ANALYSIS TECHNIQUES IN PHYSICS RESEARCH (ACAT08), 12., 2008, Genebra.

[3] SEGAL, Ben; BUNCIC, Predrag; QUINTAS, David Garcia; SANCHEZ, Carlos Aguado; BLOMER, Jakob; MATO, Pere; HARUTYUNYAN, Artem; RANTALA, Jarno; WEIR, David J.; YAO, Yushu. LHC Cloud Computing with CernVM. In: INTERNATIONAL WORKSHOP ON ADVANCED COMPUTING AND ANALYSIS TECHNIQUES IN PHYSICS RESEARCH (ACAT10), 13., 2010, Jaipur.

[4] MURPHY, Michael A.; FENN, Michael; GOASGUEN, Sebastien. Virtual Organization Clusters. In: EUROMICRO INTERNATIONAL CONFERENCE ON PARALLEL, DISTRIBUTED AND NETWORK-BASED PROCESSING (PDP '09), 17., 2009, Weimar.

[5] MURPHY, Michael A.; KAGEY, Brandon; FENN, Michael; GOASGUEN, Sebastien. Dynamic Provisioning of Virtual Organization Clusters. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID (CCGRID '09), 9., 2009, Xangai.

[6] GILBERT, Laura; TSENG, Jeff; NEWMAN, Rhys; IQBAL, Saeed; PEPPER, Ronald; CELEBIOGLU, Onur; HSIEH, Jenwei; MASHAYEKHI, Victor; COBBAN, Mark. Implications of Virtualization on Grids for High Energy Physics Applications. In: International Parallel and Distributed Processing Symposium(IPDPS 2005), 19., Denver, 2005.

[7] HARUTYUNYAN, A.; BUNCIC, P.; FREEMAN, T.; KEAHEY, K. . Dynamic Virtual AliEn Grid Sites on Nimbus with CernVM. In: INTERNATIONAL CONFERENCE ON COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS (CHEP09), 17., 2009, Praga.

[8] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *The Large Hadron Collider*. Disponível em: <http://public.web.cern.ch/public/en/LHC/LHC-en.html>. Acesso em: 20 abr. 2011.

[9] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *CMS*. Disponível em: <http://public.web.cern.ch/public/en/LHC/CMS-en.html>. Acesso em: 20 abr. 2011.

[10] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *ATLAS*. Disponível em: <http://public.web.cern.ch/public/en/LHC/ATLAS-en.html>. Acesso em: 20 abr. 2011.

[11] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *ALICE*. Disponível em: <http://public.web.cern.ch/public/en/LHC/ALICE-en.html> Acesso em: 20 abr. 2011.

[12] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *LHCb*. Disponível em: <http://public.web.cern.ch/public/en/LHC/LHCb-en.html> Acesso em: 20 abr. 2011.

[13] SULISTIO, Anthony; BUYYA, Rajkumar. A Grid Simulation Infrastructure Supporting Advance Reservation. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS (PDCS'04). 16., 2004, MIT Cambridge.

[14] CABALLERO, González. Operational Experience with CMS Tier-2 Sites. In: INTERNATIONAL CONFERENCE ON COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS (CHEP09), 17., 2009, Praga.

[15] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH, Memorando de Entendimento para Colaboração na Implantação e Utilização do Grid Computacional Mundial do LHC. Disponível em: <https://espace.cern.ch/WLCG-document-repository/MoU/countries/Brazil/MoU-CERN-Brazil-20Apr09_PortugueseVersion.pdf> Acesso em: 31 maio 2011.

[16] WLCG RESOURCE, BALANCE AND USAGE (WLCG REBUS). VO Requirements. Disponível em: <http://wlcg-rebus.cern.ch/apps/pledges/requirements/>. Acesso em: 16 nov. 2011.

[17] ATLAS COMPUTING GROUP. ATLAS Computing Technical Design Report. Disponível em: <http://cds.cern.ch/record/837738/files/lhcc-2005-022.pdf>. Acesso em: 20 jun. 2011.

[18] BRAGA, Antonio Rafael; DA SILVA, Paulo Renato Xavier; SOARES, José Marques; GOMES, Danielo G.. *Ócio Produtivo*: Explorando a Ociosidade de Clusters Virtuais para Execução de Aplicações do Tipo Saco de Tarefas. In: X WORKSHOP EM CLOUDS, GRIDS E APLICAÇÕES (WCGA 2012), 10., 2012, Ouro Preto.

[19] KIM, Kyong Hoon; BUYYA, Rajkumar. Policy-Based Resource Allocation in Hierarchical Virtual Organizations for Global Grids. In: COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING (SBAC-PAD '06),18., 2006, Ouro Preto.

[20] DUMITRESCU, Catalin L.; FOSTER, Ian. *GRUBER*: A Grid Resource Usage SLA Broker. In: INTERNATIONAL EUROPEAN PARALLEL COMPUTING CONFERENCE (EUROPAR), 11., 2005, Lisboa.

[21] ELMROTH, Erik ; GARDFJÄLL, Peter. Design and Evaluation of a Decentralized System for Grid-Wide Fairshare Scheduling. In: FIRST INTERNATIONA CONFERENCE ON E-SCIENCE AND GRID COMPUTING (E-SCIENCE'05), 1., 2005, Melbourne.

[22] BÜGE, Volker; HESSLING, Hermann; KEMP, Yves; KUNZE, Marcel; OBERST, Oliver; QUAST, Günter; SCHEURER, Armin; SYNGE, Owen. Integration of Virtualized Worker Nodes in Standard Batch Systems. In: INTERNATIONAL CONFERENCE ON COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS (CHEP09),17., 2009.

[23] STANDARD PERFORMANCE EVALUATION CORPORATION (SPEC). *CINT200 Result: Dell PowerEdge 2950 – Intel Xeon processor 5160 3.00GHz.* Disponível em: <http://www.spec.org/osg/cpu2000/results/res2006q2/cpu2000-20060501-05940.html> Acesso: em: 24 abr.2012.

[24] GRIDPP – UK COMPUTING FOR PARTICLES PHYSICS. *HEPSPEC06*. Disponível em: <https://www.gridpp.ac.uk/wiki/HEPSPEC06>. Acesso em: 24 abr.2012

[25] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *Running HEP-SPEC* Disponível em: <https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiBenchHEPSPEC>. Acesso em: 24 abr. 2012.

[26] Gonzalo MERINO, Manfred ALEF, Helge MEINHARD, Miche MICHELOTTO, Alex IRIBARREN, Peter HRISTOV, Alessandro DE SALVO, Franco BRASOLIN, Marie-Christine SAWLEY, Gabriele BENELLI, Hubert DEGAUDENZI. Transition to a New CPU Benchmarking Unit for the WLCG. Disponível em: <http://indico.cern.ch/getFile.py/access?contribId=3&sessionId=0&resId=0&materialId=0&confId=49388 >. Acesso em: 24 abr 2012.

[27] THE HEBREW UNIVERSITY OF JERUSALEM. *Parallel Workloads Archive: LCG Grid – The LCG Grid Log*. Disponível em: <http://www.cs.huji.ac.il/labs/parallel/workload/l_lcg/index.html>. Acesso em: 24 abr. 2012.

[28] BUYYA, Rajkumar; MURSHED, Manzur. *GridSim*: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. Concurrency and Computation: Practice and Experience.14. ed. Nova Yorque: Ed. Wiley Press, 2002. 1593 p.

[29] FOSTER, Ian; DUMITRESCU, Catalin. Usage Policy-Based CPU Sharing in Virtual Organizations. In: IEEE/ACM INTERNATIONAL WORKSHOP ON GRID COMPUTING (GRID'04), 5., 2004, Pittsburgo.

[30] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH . *CMS Dashboard:Daily Job Summary*. Disponível em: <http://dashb-cms-*jobs*mry.cern.ch/dashboard/request.py/dailysummary>. Acesso em: 17 maio 2012.

[31] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *ATLAS Dashboard:Daily Job Summary*. Disponível em: <http://dashb-atlas-*job*.cern.ch/dashboard/request.py/dailysummary> Acesso em: 17 maio 2012.

[32] EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH. *Proxy Certificates*. Disponível em: <http://egee-uig.web.cern.ch/egee-uig/production_pages/*Proxy*Certs.html>. Acesso em: 10 jul. 2012.

[33] OPEN SCIENCE GRID. *The Open Science Grid*. Disponível em: <https://www.opensciencegrid.org/bin/view>. Acesso em: 10 ago. 2012.

[34] SHIERS, Jamie. *The Worldwide LHC Computing Grid:* Worldwide LCG. Computer Physics Communications, 2007, v. 177, p. 219-223.

[35] FOSTER, Ian; KESSELMAN, Carl; TUECKE, Steven. *The Anatomy of the Grid*: Enabling
Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 2001, v. 15, p. 200-222.

[36] CZARNECKI, Krysztof; EISENECKER, Ulrich*. Generative Programming*: Methods, Tools, and Applications, Nova Iorque: ACM Press/Addison-Wesley Publishing Co., 2000.

[37] DE OLIVEIRA, Luigi Jacometti. *Comparação de Ferramentas de Simulação de Grades Computacionais*. 2007. 65 f. Monografia (Pós-Graduação em Ciência da Computação) – Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista, São José do Rio Preto, 2007.

[38] *LCG Gstat.* Disponível em: <http://gstat2.grid.sinica.edu.tw/gstat/vo/>. Acesso em: 18 fev. 2012.

[39] STANDARD PERFORMANCE EVALUATION CORPORATION (SPEC). *SPEC FAQ* Disponível em: <http://www.spec.org/spec/faq/>. Acesso em: 14 set. 2012.

[40] ABBAS, Ahmar. *Grid Computing:* A Practical Guide to Technology and Applications, Charles River Media, 2004.

[41] RODRIGUES, André Luiz Barbosa; SZTAJNBERG, Alexandre; LOQUES, Orlando. Auto-Adaptação de Requisitos de Tolerância a Faltas através de Contratos. SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 26., 2008, Rio de Janeiro.

[42] KICZALES, Gregor; LAMPING, John; MENDHEKAR, Anurag; MAEDA, Chris; LOPES, Cristina Videira; LOINGTIER, Jean-Marc; IRWIN, John. Aspect-Oriented Programming. EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING (ECOOP), 11., 1997, Jyväskylä.

[43] IOSUP, Alexandru; JAN, Mathieu; SONMEZ, Ozan; EPEMA, Dick. The Characteristics and Performance of Groups of Jobs in Grids. INTERNATIONAL

EURO-PAR CONFERENCE ON PARALLEL PROCESSING (Euro-Par ’07), 13., 2007, Rennes.

[44] AZEVEDO, Marcus Vinicius. *Otimização de Recursos e Economia de Energia em Clusters Usando Virtualização*. 2010. 76 f. Dissertação (Mestrado em Ciências em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro 2010.

[45] PITANGA, Marcos. *Computação em Cluster*: O Estado da Arte da Computação. Rio de Janeiro: Brasport Livros e Multimídia Ltda, 2003.