COPPE/UFRJ

CONTROLE DISTRIBUÍDO DE *WORKFLOWS* EM MALHAS COMPUTACIONAIS

Diego Moreira de Araujo Carvalho

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Felipe Maia Galvão França

Rio de Janeiro
Março de 2010

CONTROLE DISTRIBUÍDO DE *WORKFLOWS* EM MALHAS COMPUTACIONAIS

Diego Moreira de Araujo Carvalho

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Felipe Maia Galvão França, PhD.

Prof. Valmir Carneiro Barbosa, PhD.

Prof. Bernard Marie Marechal, Ph.D.

Prof. Francisco Vilar Brasileiro, Ph.D.

Prof. Inês de Castro Dutra, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2010

Carvalho, Diego Moreira de Araujo

Controle Distribuído de *Workflows* em Malhas Computacionais/Diego Moreira de Araujo Carvalho. – Rio de Janeiro: UFRJ/COPPE, 2010.

XVI, 99 p.: il.; 29, 7cm.

Orientador: Felipe Maia Galvão França

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2010.

Referências Bibliográficas: p. 70 – 88.

1. fluxo de processo científico. 2. escalonamento. 3. computação em malha. I. França, Felipe Maia Galvão. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A minha filha Sofia.

Agradecimentos

À Luciana, minha esposa, que, durante esses anos todos, me encheu de carinho, consideração e compressão, confiando em meus julgamentos e me dedicando um tempo significativo para que eu pudesse findar esse trabalho.

Aos meus pais, Jorge Guilherme e Heloisa Helena, que sempre me confortaram com um carinho intenso em todos os momentos da minha vida.

Ao amigo e auto-intitulado "desorientador" Felipe e sua esposa Priscila, que sempre compreenderam as dificuldades que eu encontrei durante o caminho que trilhei nesse trabalho.

Ao amigo Rafael Barbastefano, que sempre foi muito objetivo no incentivo e sempre me ajudou a compreender qual era a meta final.

Aos amigos do *EELA-2 Operations Centre*, Ramon Diacovo, Frederico de Oliveira e Vinícius Ferrao que sempre me ajudaram a resolver os outros pepinos e manter o desempenho da malha do EELA-2.

Ao amigo Felipe Canto, em quem eu sempre encontrei um ponto de equilíbrio e exemplo de vida profissional.

Aos amigos Pedro Henrique e Marcelo Spindola Bacha que sempre estiveram lá desde o início...

Ao amigo Bernard, com quem viajei a metade do mundo acreditando no sonho de se construir uma *e-Infrastructure*.

Aos membros da banca pelas importantes sugestões e comentários que fizeram sobre esse texto.

A Solange que sempre me ajudou com os pepinos burocráticos do PESC e da COPPE.

A todos que de alguma forma contribuíram para esse trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

CONTROLE DISTRIBUÍDO DE *WORKFLOWS* EM MALHAS COMPUTACIONAIS

Diego Moreira de Araujo Carvalho

Março/2010

Orientador: Felipe Maia Galvão França

Programa: Engenharia de Sistemas e Computação

Diversos sistemas surgiram na literatura para efetuar o gerenciamento do fluxo de processos em malhas computacionais, contudo, com a evolução e o aumento da adoção das malhas computacionais, novas dificuldades foram acrescentadas, criando a necessidade de novas soluções para o problema do gerenciamento de fluxo de processos. Este trabalho apresenta uma revisão das *e-Infrastructures* de serviço em produção, do *middleware* que é empregado na construção mas mesmas, do estado da arte do uso de fluxos de processos na *e-Science* e faz uma apresentação dos problemas relacionados às falhas que ocorrem nessas infraestruturas, junto com as imperfeições existentes nos sistemas de informação que mantêm o estado da malha. Além disso, esse trabalho apresenta o WAS (*Workflow Agile Scheduler*), um Sistema de Gerenciamento de Fluxo de Processos (SGFP) que tem por objetivo principal endereçar os problemas de falhas e imperfeição da informação sobre o estado da malha computacional. O WAS apresentou um *speed up* de 1,6 quando comparado com uma solução de plena aceitação na comunidade de *grid*. A sua principal contribuição é o uso do conceito de infraestrutura virtual, construída com *pilot jobs*, pois o seu uso impacta diretamente nos problemas de imperfeição da informação existentes no momento do sequenciamento. Apesar de não atacar o problema de falhas diretamente, é mostrado os potenciais de se contornar o problema, tirando proveito de maneira mais eficiente da malha, não importando qual o seu estado corrente. Outro fato relevante é que uma infraestrutura virtual esconde os atrasos gerados pela a influência do sistemas de gerenciamento de recursos existentes nos RCs constituintes da malha. Isso pode ser visto como uma movimentação do momento de tomada de decisão do escalonamento, que é levado do momento da submissão da tarefa para o momento em que o recurso está realmente disponível para se usado pelo SGFP.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DISTRIBUTED SCHEDULING OF WORKFLOWS ON COMPUTATIONAL GRIDS

Diego Moreira de Araujo Carvalho

March/2010

Advisor: Felipe Maia Galvão França

Department: Systems Engineering and Computer Science

Several systems have appeared in literature to control scientific workflows in computational grids. However, with the evolution and the increase adoption of grids or *e-Infrastructures*, new challenges have emerged, creating the need of new solutions for the scientific workflow scheduling problem. This work presents a revision of service *e-Infrastructures* currently in production around the globe and related *middlewares*. Moreover, it reviews the scheduling of workflows in computational grids and introduces a list of problems related to the imperfections that affect the quality of the grid state information available on the information systems. Besides, we introduce the WAS (Workflow Agile Scheduler), a workflow control system that addresses these problems related to the quality of the state information. On a real *e-Infrastructure*, the WAS scheduler achieved a speed up of 1.6, when compared with a competitor well accepted in the grid community. It makes use of the concept of virtual infrastructure (VI) built using pilot jobs. The VI allows one to manage all information imperfections created by the existence of the local resource management system on every grid site, moving the job dispatch decision time from the job submission time to the time when the resource is really available.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AJO	*Abstract Job Object*, p. 24
ALICE	*A Large Ion Collider Experiment*, p. 12
ARPANET	*Advanced Research Projects Agency Network*, p. 6
ATLAS	*A Toroidal LHC Apparatus*, p. 12
Babar	*the B and B-bar experiment*, p. 12
CAR	Conjuntos de Atividades Relacionadas, p. 59
CDF	*Collider Detector at Fermilab*, p. 12
CDSS	*Clinical Decisions Support System*, p. 13
CERN-PROD	*CERN - Tier 0 Production Site* , p. 28
CERN	*European Organization for Nuclear Research*, p. 6
CE	*Computing Element*, p. 9
CMS	*the Compact Muon Solenoid*, p. 12
D0	*the D Zero experiment*, p. 12
DAG	grafo acíclico, em inglês *Directed Acyclic Graph*, p. 39
DEISA	*Distributed European Infrastructure for Supercomputing Applications*, p. 14
DESY	*Deutsches Elektronen-Synchrotron*, p. 12
DIM	*Distributed Information Management*, p. 58
EELA-2	*E-science grid facility for Europe and Latin America*, p. 13
EELA	*E-infrastructure shared between Europe and Latin America*, p. 13

EGEE *Enabling Grids for E-sciencE*, p. 8

EGI *European Grid Initiative*, p. 14

EOC EELA-2 Operations Centre, p. 62

GASS *Globus Access to Secondary Storage*, p. 16–18

GATE *Geant4 Application for Tomographic Emission*, p. 13

GFP *Global File Prefetcher*, p. 58, 60

GIIS *Grid Information Index Service*, p. 18

GPS@ *Grid protein Sequence @nalysis*, p. 13

GPS *Global Positioning System*, p. 28

GRAM *Globus Resource Allocator Manager*, p. 16

GRIS *Grid Resource Information Service*, p. 18

GSI *Grid Security Infrastructure*, p. 16

GridFTP *Grid File Transfer Protocol*, p. 19

HTTP *Hypertext Transfer Protocol*, p. 18

ICP Infraestrutura de Chaves Públicas, p. 11

IDB *Incarnation Data Base*, p. 24

IDL *interface description language*, p. 22

IGTF *International Grid Trust Federation*, p. 11

IN2P3-CC *Centre de Calcul de l'IN2P3 - French Tier 1*, p. 28

JDL *Job Description Language*, p. 9

JMC *Job Monitor Component*, p. 24

JPA *Job Preparation Agent*, p. 24

LB *Logging and Bookeeping System*, p. 22

LDAP *Lightweight Directory Access Protocol*, p. 18

LFN *Logical File Name*, p. 21

LHC	*Large Hardron Collider*, p. 6
LHCb	*the Large Hardron Collider beauty*, p. 12
LIGO	*Laser Interferometer Gravitational Wave Observatory*, p. 33
LRMS	*Local Resource Management System*, p. 9, 18
LSF	*Load Sharing Facility*, p. 18
MDS	*Metacomputing Directory Service*, p. 16, 18
MIP	mix integer-programming, p. 90
MTBF	*Mean Time Between Failures*, p. 39
NJS	*Network Job Supervisor*, p. 24
NRENs	*National Research and Education Networks*, p. 9
OGSA	*Open Grid Service Architecture*, p. 19
OSG	*Open Science Grid*, p. 13
PBS	*Portable Batch System*, p. 18
PF	*Pilot Factory*, p. 55
PI	Pilot Inventory, p. 57
PJ	*Pilot Job*, p. 55
PKI	Public Key Infrastructure, p. 11
RAL-LCG2	*UK Tier 1 Centre*, p. 28
RC	*Resource Centre*, p. 9, 10
RLM	*Replica Location Management*, p. 19
ROC	*Regional Operating Centre*, p. 11
RSL	*Globus Resource Specification Language*, p. 18
SC	*Submission Controller*, p. 55
SECT	*Schedule Enactor*, p. 57
SE	*Storage Element*, p. 10

SGFP	Sistemas de Gerenciamento de Fluxo de Processo, p. 4
SOA	*Service Oriented Architecture*, p. 19
SSL	*Security Socket Layer*, p. 16
SURL	*Site Universal Resource Locators*, p. 21
SeeGrid 2	*South-East Europe Grid*, p. 15
TCP/IP	*Transmission Control Protocol*, p. 16
TSI	*Target System Interface*, p. 24
UI	*User Interface*, p. 9
UUDB	*Unicore User Data Base*, p. 24
VO	*Virtual Organization*, p. 9
WAN	Wide Area Network, p. 3
WAS	*Workflow Agile Scheduler*, p. 5, 54
WCS	*Wide-area computing systems*, p. 3
WLCG	*Worldwide LHC Computing Grid*, p. 14
WMS	*Workload Management System*, p. 9
WN	*Worker Node*, p. 9
iSGTW	International Science Grid This Week, p. 62

Capítulo 1

Introdução

Sistemas distribuídos têm atraído uma grande atenção das ciências da computação nas últimas décadas [5]. Nesses anos, principalmente o desenvolvimento das redes avançadas de computadores, conjugado com a redução dos custos dessas novas tecnologias de comunicação, teve um papel importante no crescimento da importância e do uso dos sistemas distribuídos nas diversas áreas das ciências, não só das exatas e tecnológicas, como das biológicas e da saúde.

Essa disseminação do uso dos sistemas distribuídos em várias áreas do conhecimento é traduzido em diversas implementações de aplicações em plataformas diferentes, indo desde computadores paralelos *ad hoc*, até sistemas fisicamente distribuídos geograficamente de maneira global. Como um primeiro exemplo, podemos citar o PlanetLab[1] que é um sistema distribuído baseado em uma rede de instituições de pesquisa que fornecem máquinas virtuais em todas as partes do planeta. Nessa infraestrutura se desenvolve pesquisas nas áreas de armazenamento distribuído, sistemas ponto-a-ponto, base de dados distribuídas, etc.

Por causa da diversidade de plataformas, um erro comum [6] é o de se considerar um sistema distribuído como uma simples rede de computadores, como por exemplo a Internet. Na realidade, um sistema distribuído pode também ser construído sobre uma rede de computadores, com a característica distinta de se apresentar como um único sistema autônomo, mascarando a existência de seus componentes variados e heterogêneos. Dessa forma, a execução de uma aplicação pelo usuário no sistema se faz através da troca de mensagens entre os computadores constituintes, utilizando o substrato de rede através de protocolos convencionais de rede como por exemplo o TCP/IP [7].

De acordo com os objetivos e a forma que os recursos são agregados, os sistemas distribuídos podem ser classificados de uma maneira livre em: *Clusters* [8], redes de comunicação ponto-a-ponto [9] e os sistemas de computação em malha [10]. Um

[1] `http://www.planet-lab.org`

cluster é um grupo dedicado de computadores interconectados por uma rede de alto desempenho que se apresenta como um único computador de alto desempenho com memória distribuída, ideal para a execução de aplicações paralelas e geralmente usado nas áreas avançadas de engenharia e ciências exatas, além de existirem diversos casos de uso na indústria e na área financeira.

As redes de comunicação ponto-a-ponto são sistemas distribuídos descentralizados, que fornecem o substrato para aplicações de compartilhamento de recursos, transferência de arquivos, mensagem instantânea e distribuição de conteúdo público etc. Além disso, sistemas de armazenamento distribuído como CODA [11] permitem ao usuário o acesso unificado ao sistema de arquivamento, independente de onde a informação esteja armazenada ou de onde ela é acessada.

Os sistemas de computação em malha, sistemas de computação em grades, ou simplesmente *grids*[2], são sistemas que permitem a agregação ou federação coordenada de recursos computacionais heterogêneos (processadores, sistemas de armazenamento, bases de dados) e até instrumentos científicos (detetores de partículas de altas ou baixas energias, radio-telescópios, etc). Os *grids* são de fundamental importância para as áreas emergentes de *e-Science* e *e-Negócios*.

Jim Gray disse que *e-Science* é onde a "tecnologia da informação se junta aos cientistas"[12]. Em particular, *e-Science* se refere as novas oportunidades científicas que necessitam de colaborações distribuídas e proporcionadas pelas novas tecnologias de rede. Essas tecnologias incluem computação distribuída, gerenciamento de dados distribuídos, ferramentas de colaboração, etc. Muitas dessas tecnologias estão em estado de rápido desenvolvimento e padrões ainda não foram estabelecidos. Essas oportunidades estão relacionadas com extensão de fronteiras da ciência ou, até mesmo, na criação de novas, por causa da fusão entre a teoria, experimentação e simulação. Além disso, observamos que a *e-Science* normalmente está envolvida com um grande volume de dados, que são gerados através dos instrumentos de pesquisa ou provenientes da simulação dos próprios aparatos experimentais. Para tornar factível a manipulação do conhecimento gerado, o uso massivo de computadores se torna necessário, transformando os sistemas distribuídos, redes avançadas de computadores e programas de análise de dados no que conhecemos por *malhas computacionais*.

1.1 Malhas computacionais

Sistemas distribuídos similares aos *grids* já tinham surgido na literatura, como apresentado no trabalho sobre a federação de recursos em uma Wide Area Network

[2]nesse trabalho, os termos sistemas de computação em malha ou grade e *grids* serão usados de forma indistinta.

(WAN) [13], onde é discutido o sequenciamento de trabalhos em uma coleção de sistemas espalhados em diversas universidades, interconectados pela primeira infra-estrutura de rede *Gigabit*. Essas novas infra-estruturas de redes logo se espalharam pelos continentes e países (*Internet2* nos EUA, *Geánt* na Europa e *RedeIpê* no Brasil), servindo de base para a criação de redes de computadores que tinham como o principal resultado a mudança do modelo vigente: sistemas distribuídos passaram a ser construídos com a capacidade das redes e não somente nas distâncias geográficas[13]. Essa "redução das distâncias" surge pois a capacidade dos enlaces entre as instituições se tornaram superiores aos enlaces locais, permitindo então a consideração de sistemas maiores.

Os novos sistemas foram chamados de *Wide-area computing systems* (WCS). WCS não eram inteiramente novos pois aplicações que usavam a rede, como o e-mail, já existiam há muito tempo, contudo, as novas redes criaram a possibilidade de se construir sistemas de alto desempenho com o uso dos recursos distribuídos geograficamente, reduzindo os tempos e custos de computação, já que os custos variáveis de manutenção e operação passaram a ser distribuídos entre um maior número de membros do sistema.

Apesar de diversos trabalhos no final da década de 80 e da década de 90, somente depois do trabalho de Ian Foster e Kesselman[10], o termo computação em malha, grade ou simplesmente *grid*, se tornou comum. Na própria definição, revisada três vezes pelos próprios autores [10, 14], um sistema de computação em malha deve mostrar as seguintes características:

- coordenação de recursos que não estão sujeitos à uma entidade central – o sistema em malha integra e coordena o uso de recursos que estão em áreas administrativas distintas, podendo pertencer a departamentos distintos de uma mesma empresa, a instituições diferentes no mesmo país, ou até mesmo a instituições espalhadas geograficamente no globo;

- uso de interfaces e protocolos padronizados e abertos – o sistema em grade deve utilizar protocolos de comunicação de propósito geral e interfaces-padrão responsáveis pela autenticação, autorização, descoberta e acesso de/a recursos;

- apresenta um serviço que pode fornecer recursos não triviais – o *grid* permite que os seus recursos constituintes possam ser usados de maneira coordenada para fornecer diversos níveis de qualidade de serviço, além de fornecer poder computacional não trivial. Fornecendo a idéia de que o sistema combinado é superior a soma das partes.

Além das características apresentadas, podemos sintetizar algumas motivações fundamentais para o uso de sistemas de computação em malha, como por exemplo,

os apresentados em [15, 16]:

- custos significativos como manutenção, operação e atualização são distribuídos de maneira mais efetiva quando o ambiente computacional é distribuído geograficamente entre os parceiros, que são aptos a encontrar financiamento local para esses custos, contribuindo para um objetivo global da colaboração;
- sistemas de computação em malha podem contribuir para a criação de sistemas computacionais sem pontos críticos a falha. Diversas cópias dos dados podem ser distribuídas no *grid*, além de se poder movimentar os trabalhos a serem executados para centro de recursos diverso;
- aumentar a utilização dos recursos, já que o tempo livre poderia ser utilizado por outro participante da malha.

1.2 Fluxos de processo

Deve ser percebido que a execução das aplicações dos usuários e cientistas se aproxima de uma produção fabril, onde fluxo de processo, em inglês *workflow*, define as tarefas e recursos que serão utilizados para a produção dos resultados finais. Podemos observar que com a quantidade massiva de recursos disponíveis no *grid*, os cientistas e engenheiros estão construindo aplicações cada vez mais complexas[2]. Nesse cenário, os fluxos de processo emergiram como um paradigma para a representação (Figura 1.1) e o gerenciamento de complexas computações distribuídas e são ferramentas básicas para a o progresso das comunidades científicas [17].

Um fluxo de processo pode ser considerado como a automação de procedimentos onde arquivos e dados são transacionados entre aplicações participantes de acordo com um conjunto definido de regras para se atingir um objetivo final [2]. Um Sistemas de Gerenciamento de Fluxo de Processo (SGFP) tem como principais objetivos: o desenho, a coordenação e a administração da execução de um fluxo em recursos computacionais.

No início dessa década, diversos SGFPs para sistemas de computação em malha foram propostos para atender especificamente fluxos científicos. O trabalho [18] apresenta uma taxonomia que agrega os estilos arquiteturais e identifica as diferenças de desenho e engenharia de cada um. Existem outras taxonomias para sistemas heterogêneos e distribuídos genéricos que contribuíram como base para os SGFPs, como [19–22]. Além disso, diversas tarefas envolvidas com a confecção e execução de um fluxo de processo são extremamente difíceis para o cientista sem uma proficiência em computação[23], principalmente as atividades de otimização do fluxo.

Nesse trabalho focamos o nosso estudo em infraestruturas reais e em produção, pois acreditamos que as malhas computacionais já atingiram um determinado nível

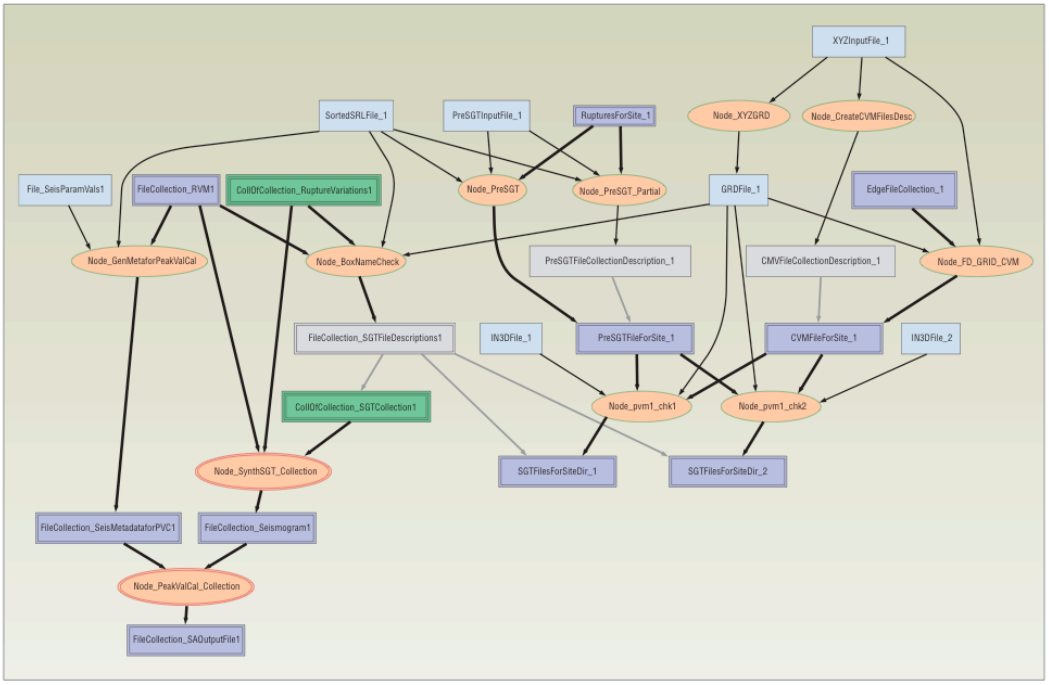


Figura 1.1: Representação visual de um fluxo de processo para o processamento de dados para estudo de terremotos.

de maturidade que as confere um estado que não é mais transiente. No próximo capítulo, apresentamos uma revisão dessas infraestruturas e do *middleware* que as compõe. Além disso, apresentaremos alguns detalhes da parte organizacional que influenciam diretamente na operação das infraestruturas e de como elas se apresentam aos seus usuários. No Capítulo 3 é apresentada uma revisão dos fluxos de processos e dos sistemas construídos para a gestão dos mesmos. Já no Capítulo 4 apresentamos o problema do controle distribuído dos fluxos de processo e introduzimos o *Workflow Agile Scheduler* (WAS), um sistema de controle para o gerenciamento dos fluxos. Encerramos esse trabalho com um capítulo apresentando as conclusões.

Capítulo 2

e-Infrastructure e *middleware*

A *e-Science* apresentada no capítulo anterior necessita de um novo ferramental onde os cientistas possam acessar informação de lugares diferentes, federar e controlar recursos computacionais, analisar informação e dados, gerar simulações, validar teorias, inspecionar experimentos remotos, etc. Esse novo ferramental ou infraestrutura é bem próxima da visão apresentada por J. C. R. Licklider, quando foi desenhado um projeto de pesquisa chamado *Advanced Research Projects Agency Network* (ARPANET) no Departamento de Defesa dos Estados Unidos da América [24].

2.1 *e-Infrastructure*

A necessidade dessa infraestrutura especial ou *e-Infrastructure*[1] pela *e-Science* levou à criação de importantes projetos de pesquisa no primeiro mundo que impulsionaram a formação de verdadeiras colaborações científicas mundiais, fato antes só existente em um número reduzido de áreas do conhecimento. Não é coincidência que um dos motores de criação se encontra no *European Organization for Nuclear Research* (CERN) com base num dos maiores aparados experimentais[24]: o *Large Hardron Collider* (LHC)[25]. Com mais de 60 anos de experiência em colaborações internacionais, os físicos do CERN trabalham na montagem de uma infraestrutura para processar os dados gerados pelo LHC, hoje na ordem de 15 Petabytes por ano.

Hoje, podemos dizer que *e-Infrastructure* (Figura 2.1) se refere ao ambiente onde pesquisadores e cientistas têm acesso às instalações de tecnologia de informação (incluindo dados, instrumentos, ciclos de processamento e comunicações) independente da sua localização geográfica [24]. Além disso, esse acesso permite o trabalho em diversos contextos que podem ser dentro de suas instituições de trabalho, iniciativas nacionais, ou até mesmo internacionais.

[1] *e-Infrastructure* é o termo normalmente utilizado na Europa e na Austrália. Nos Estados Unidos, normalmente se usa o termo *Cyberinfrastructure*

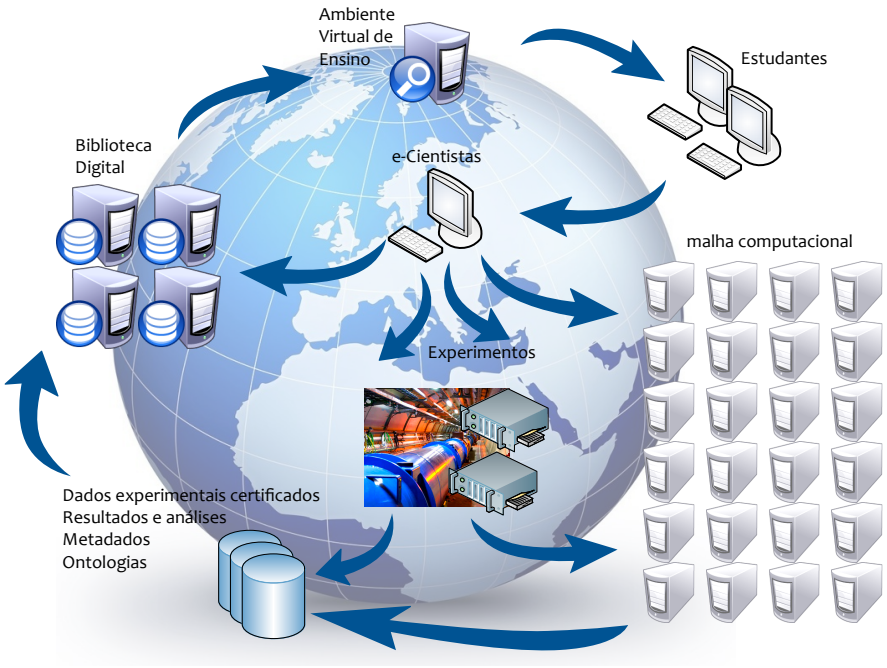


Figura 2.1: Ambiente onde *e-Infrastructure* é usada para o desenvolvimento da *e-Science* (adaptado de [24]).

Uma outra maneira de se definir uma *e-Infrastructure* é citando a visão apresentada por Ian Foster [14, 26] quando ele define a computação em malha (*grid computing*), mas nesse capítulo vamos nos deter a apresentar os fatores organizacionais que podem afetar a confiabilidade na execução de fluxos de processo e os elementos arquiteturais do *middleware* que são utilizados na construção das *e-Infrastructure* nos projetos existentes.

2.1.1 Organizacional

A Figura 2.2 apresenta a distribuição geográfica dos principais projetos de *e-Infrastructure* ligados a computação em malha. Normalmente, esses projetos têm objetivos além da operação da infraestrutura e se apresentam divididos em diferentes pacotes de trabalho. Dentre os mais importantes, podemos citar:

- gestão do projeto – cordenação e atividade financeira;
- disseminação – divulgação técnica e científica dos resultados e organização de conferências;
- treinamento – coordenação dos programas de treinamento dos cientistas ou usuários no uso do *middleware*;
- aplicações – adaptação das ferramentas e aplicações à *e-Infrastructure*;

Figura 2.2: Principais projetos de computação em malha no mundo.

- operação de sítios – monitoramento e operação da *e-Infrastructure*;
- operação de rede – operação dos diversos segmentos (local, nacional e internacional) que compõem a rede;
- pesquisa e desenvolvimento – desenvolvimento de novos serviços do *middleware*.

Dentre os apresentados na Figura 2.2, o EGEE e o EELA são divididos como na estrutura apresentada acima e serão apresentados de maneira mais detalha nas próximas seções pois os experimentos apresentados nesse trabalho foram desenvolvidos e executados nas malhas computacionais suportadas por esses dois projetos.

EGEE

O projeto *Enabling Grids for E-sciencE* (EGEE)[27] oferece uma plataforma de *e-Science* à comunidade de pesquisa européia e aos seus colaboradores internacionais, contabilizando mais de 17.000 usuários distribuídos em mais de 160 diferentes grupos científicos. Esse projeto, com raízes há mais de uma década (através dos projetos: EDG - European Data Grid, EGEE-1, EGEE-2 e EGEE-3), é custeado pela Comissão Européia e tem um orçamento em torno de 32 milhões de Euros para um período de 24 meses. Esses fundos focam na implementação de uma e-Infraestrutura de computação distribuída para dar suporte aos pesquisadores de diversos domínios

científicos, dentre os quais: astrofísica, biomedicina, química computacional, ciências da terra, física de altas energias, fusão, geofísica, análise financeira e multimedia.

A plataforma oferecida pelo EGEE é composta por uma infraestrutura de computação em malha, que normalmente é construída com sistemas heterogêneos em *hardware* e *software* interligados por uma rede de alto desempenho[2]. Os recursos computacionais e de armazenamento da malha computacional são organizados em *organizações virtuais*, do inglês *Virtual Organization* (VO). Além desses recursos, aplicações, ferramentas e indivíduos completam uma VO. O acesso a esses recursos pelos usuários do EGEE é feito através do uso do *middleware* gLite, respeitando as políticas das VOs que são definidas pelos operadores da infraestrutura, provedores de recursos e usuários[28]. O gLite proporciona funcionalidades de programação de alto nível e ferramentas para o usuário final efetuar a manipulação de arquivos, réplicas, tarefas e credenciais.

No EGEE, onde existem mais de 210 VOs[3], usuários devidamente registrados numa VO específica podem obter credenciais que os habilitam a ter acesso ao conjunto completo de recursos compartilhados pela VO a qual ele pertence, não importando a localização geográfica corrente do recurso.

Para usar efetivamente a malha, os usuários têm acesso a computadores chamados de *User Interface* (UI), onde efetuam a submissão de tarefas na malha (Figura 2.3). Esses computadores também são conhecidos por *submission machines* e permitem monitorar o estado das tarefas submetidas, obter os arquivos de resultados e transferir arquivos para dentro e para fora da malha computacional.

De uma maneira geral, uma tarefa ou *job* na malha é composta por um conjunto de arquivos de entrada (*input sandbox*), um arquivo executável e um arquivo composto com a descrição dos recursos necessários a sua execução escrito em *Job Description Language* (JDL)[29, 30]. As informações fornecidas em JDL são utilizadas por um componente do *middleware*, o *Workload Management System* (WMS), para determinar em qual sítio ou *Resource Centre* (RC) a tarefa deverá ser direcionada para execução, pois a tarefa pode ter necessidades específicas relacionadas com capacidade computacional, tamanho de memória, proximidade com arquivos e disponibilidade de *software* das aplicações ou ferramentas específicas.

Depois da determinação do RC pelo WMS, a tarefa é devidamente submetida ao *Computing Element* (CE) que é um serviço fornecido pelos RCs para expor o *Local Resource Management System* (LRMS) que normalmente é um sistema simples de filas que permite a coexistência de tarefas oriundas da malha com outras locais. No RC, computadores chamados de "nó de trabalho", em inglês *Worker Node* (WN), são responsáveis pela efetiva execução da tarefa. Quando o LRMS detecta um WN

[2]as redes são normalmente formadas pelas *National Research and Education Networks* (NRENs)
[3]Fonte: CIC Portal - https://cic.gridops.org/

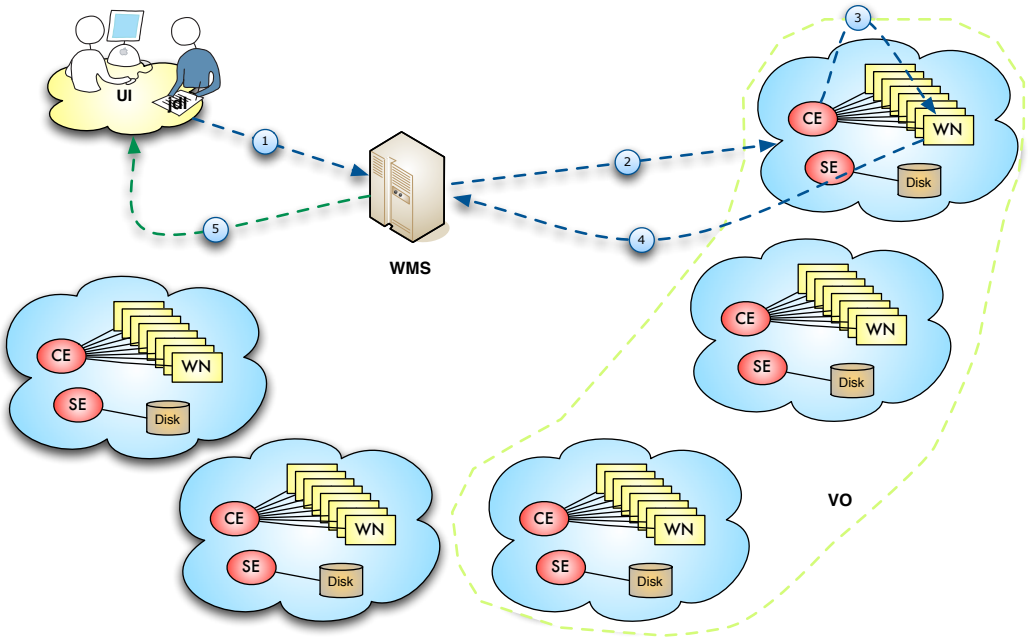


Figura 2.3: Submissão de uma tarefa no EGEE.

vago, a tarefa de maior prioridade é transferida e sua execução se inicia. Durante a execução, a tarefa efetua todas as transferências necessárias, contactando o *Storage Element* (SE) que é responsável por áreas de armazenamento de arquivos.

Tabela 2.1: Principais serviços do gLite.

Serviço	Nome	Função
CE	*Computing Element*	interface entre o *grid* e o recursos computacionais locais
SE	*Storage Element*	área de armazenamento de arquivos brutos
LFC	*Logical File Catalogue*	responsável pelo gerenciamento de meta-dados dos arquivos
WMS	*Workload Management System*	gerenciamento de despacho e escalonamento
UI	*User Interface*	computador de acesso do usuário
LB	*Logging and Bookkeeping*	responsável pelo armazenamento dos logs das tarefas

Um detalhe importante é que nem todo *Resource Centre* (RC) apresenta um SE, mas é aconselhado que se tenha um instalado pois o *middleware* apresenta um sistema de réplica de arquivos. Com esse sistema, a tarefa pode acessar o arquivo através de seu nome lógico e o *middleware* se encarrega de verificar se existe uma réplica no SE instalado no RC onde a tarefa se encontra em execução. Em seu

término de execução, a tarefa transfere a informação de estado para o WMS, de onde poderá ser recuperada pelo usuário em uma UI qualquer. A Tabela 2.1 apresenta um resumo dos serviços existentes no gLite.

Para um gerenciamento mais eficiente, o EGEE é dividido em federações/regiões e em cada federação se encontra um *Regional Operating Centre* (ROC), que é responsável pelo suporte e monitoramento de cada sítio ou RC que compõe a federação.

Além disso, é importante citar que a autenticação dos usuários é feita com o uso de uma Infraestrutura de Chaves Públicas (ICP), em inglês Public Key Infrastructure (PKI) formada pela *International Grid Trust Federation* (IGTF)[4]. Essa ICP é responsável pela emissão e controle de certificados X.509[31] em todos os continentes do planeta.

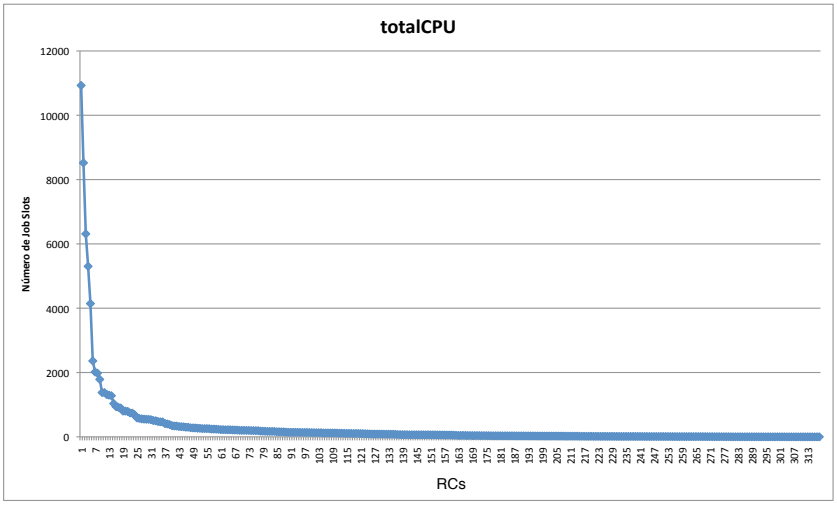


Figura 2.4: Número de COREs de processamento nos sites do EGEE-3.

Considerada a maior infraestrutura de computação em malha do planeta, em números, o EGEE apresenta mais de 300 RCs, totalizando mais de 80.000 processadores e mais de 110 PBytes de armazenamento. A Figura 2.4 mostra o tamanho em processadores de cada RC do EGEE, ordenados do maior para o menor. Podemos observar que os 10 maiores sítios são responsáveis por 51,81% dos processadores disponíveis e na Figura 2.5, que apresenta a proporção da contribuição acumulada pelos sítios, podemos observar que 80% da infraestrutura é formada por somente 50 RCs.

[4]IGTF - (http://www.igtf.org)

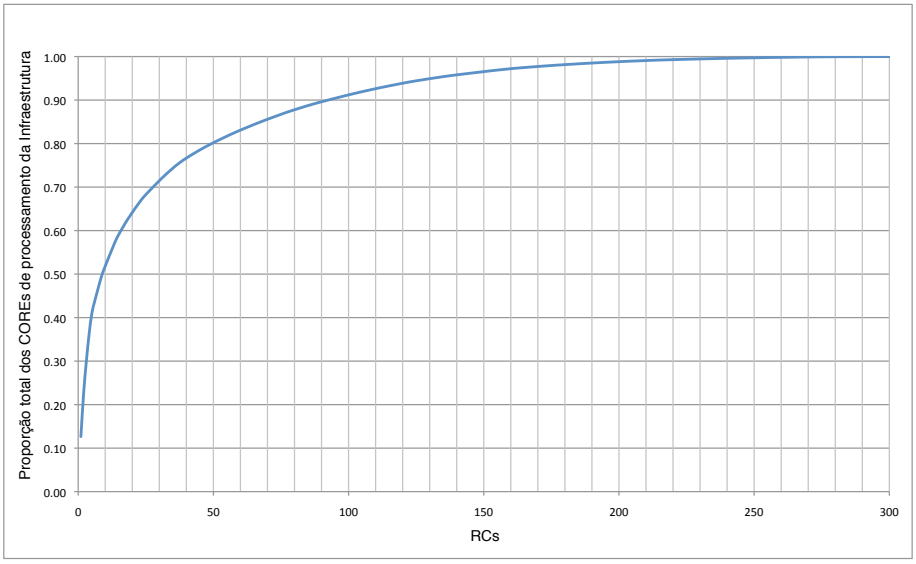


Figura 2.5: Proporção acumulada da contribuição dos sítios no EGEE-3.

Como apresentado anteriormente, os projetos de computação em malha são mais que a atividade de operação da infraestrutura. No EGEE, times são responsáveis por diversas atividades de pesquisa e desenvolvimento de aplicações, software componentes e *middleware*. Além disso, uma outra função importante do EGEE é de servir de fórum para diversas iniciativas nacionais européias como: GridPP [32], NorduGrid [33], INFNGrid [34], etc.

A comunidade de usuários do EGEE já portou para a malha mais de 250[5] aplicações distintas, dentre as quais, destacamos algumas das mais importantes:

- Física de Altas Energias – diversas aplicações específicas para o LHC, acelerador de partículas que está funcionando no CERN. O LHC é considerado um dos maiores instrumentos de pesquisa no mundo e vai produzir anualmente um volume de dados de 15 PBytes de informação. Além dos quatro experimentos do LHC — *A Large Ion Collider Experiment* (ALICE)[35], *A Toroidal LHC Apparatus* (ATLAS)[36], *the Compact Muon Solenoid* (CMS)[37] e *the Large Hardron Collider beauty* (LHCb)[38]), a *e-Infrastructure* do EGEE é utilizada por outras colaborações como *the B and B-bar experiment* (Babar)[39], *Collider Detector at Fermilab* (CDF)[40], *the D Zero experiment* (D0)[41] e Zeus [42], do laboratório alemão *Deutsches Elektronen-Synchrotron* (DESY).

- Biomédicas – Nove aplicações são providas pelo EGEE nos serviços de pro-

[5]App Registry - `https://na4rs.marie.hellasgrid.gr/na4-EGEE/index.php`

dução: *Geant4 Application for Tomographic Emission* (GATE)[43], *Grid protein Sequence @nalysis* (GPS@)[44], *Clinical Decisions Support System* (CDSS)[45], *Docking Platform for Tropical Diseases* [46], gPTM3D (análise tridimensional de imagens de órgãos humanos) [47], Xmipp MLrefine (análise de imagens de microscópio eletrônico) [48], GridGRAMM (análise de reações moleculares) [49] e GROCK (*Grid Dock* - simulação e observação de interações moleculares) [50].

- Outras áreas – além das duas áreas citadas acima, o EEGE apresenta aplicações nas seguintes outras áreas: bibliotecas digitais, química computacional, ciências da terra, geofísica, astrofísica, fusão nuclear e aplicações financeiras.

EELA

O projeto *E-infrastructure shared between Europe and Latin America* (EELA)[51, 52] e seu sucessor *E-science grid facility for Europe and Latin America* (EELA-2) tem três objetivos principais: i) a criação de uma colaboração entre a Europa e a América Latina na área de computação em malha; ii) a criação de uma *e-Infrastructure* na América Latina, compatível com a iniciativa EGEE [27] na Europa, permitindo o uso de aplicações de computação em malha nos dois continentes; iii) a criação das bases de colaboração e o ferramental necessário para o desenvolvimento da *e-Science* nos dois continentes.

A *e-Infrastructure* do EELA é composta por 25 RCs espalhados pela a Europa e América Latina e o Caribe (Figura 2.6), totalizando aproximadamente 10.000 processadores e 16 PBytes de armazenamento. Essa necessidade de compatibilidade levou ao uso do *middleware* gLite no projeto EELA e de diversas outras similaridades. Na parte de operação da *e-Infrastructure*, diversas ferramentas e protocolos de operação foram desenvolvidos[53] na tentativa de se obter uma infraestrutura com qualidade de produção. Além disso, alguns desses RCs são compartilhados também com o EGEE.

Outras iniciativas

Existem diversas outras iniciativas de *e-Infrastructure* e computação em malha no mundo, dentre as mais importantes, podemos destacar:

Open Science Grid (OSG) - *e-Infrastructure* [54] criada nos Estados Unidos[6] e *middleware* baseado no *globus*;

[6]http://www.opensciencegrid.org/

Figura 2.6: RCs do EELA.

Worldwide LHC Computing Grid (WLCG) - *e-Infrastructure* [55] composta por sítios do EGEE, OSG e outros com objetivo de fornecer recursos computacionais às VOs relacionadas com os experimentos de física de altas energias do LHC;

European Grid Initiative (EGI) - projeto europeu co-financiado pela *European Comission* que vem para substituir o EGEE-3. Baseado na formação das iniciativas nacionais de malha computacional na Europa;

Distributed European Infrastructure for Supercomputing Applications (DEISA) - projeto europeu para criação de uma malha computacional de supercomputação[7];

TeraGrid - projeto americano para a criação de uma malha computacional formada por supercomputadores nos EUA[56];

Naregi - projeto japonês[57] com o principal objetivo de desenvolvimento e pesquisa de *middleware*, utilizando os padrões mundiais disponíveis nos organismos de normalização;

Garuda - [58] é uma colaboração científica na Índia para a criação de uma *e-Infrastructure*, integrando computadores, sistemas de armazenamento e instrumentos científicos. Além de desenvolver pesquisa de *middleware*, o projeto

[7]http://www.deisa.eu/

Garuda vem desenvolvendo colaborações internacionais, principalmente com projetos europeus;

South-East Europe Grid (SeeGrid 2) - [59] tem por objetivo principal a criação das condições necessárias para o suporte sustentável da *e-Infrastructure* na região dos Bálcãs;

EuMEDGrid - [60] tem por objetivo implantar uma *e-Infrastructure* na região do Mediterrâneo;

BalticGrid - [61] tem por objetivo a integração de instituições dos países bálticos (Lituânia, Letônia e Estônia) na comunidade européia de computação em malha e o desenvolvimento das infra-estruturas eletrônicas nesses países;

EuChinaGrid - [62] tem como objetivo principal a conexão das *e-Infrastructure*s EGEE (européia) e CNGrid, habilitando a inter-operação das *e-Infrastructure*s, estudar a aplicabilidade do IPv6, além de desenvolver as aplicações de malha existentes, facilitando a cooperação entre as duas regiões.

No Brasil, gostaríamos de destacar o projeto *OurGrid*[63]. Esse projeto apresenta uma *e-Infrastructure* ponto-a-ponto que entrou em produção em 2004 e é desenvolvido pela Universidade Federal de Campina Grande. A principal diferença entre ele e os outros projetos é que sua infraestrutura é oportunista, ou seja, não se baseia em uma infraestrutura dedicada ou de serviço. Além disso, os recursos são compartilhados de acordo com um *modelo de rede de favores*, onde cada parte participante prioriza os os pares baseado nos créditos relativos às interações passadas.

2.2 Arquitetura e *middleware*

Nesta seção vamos apresentar algumas arquiteturas de *middleware* utilizadas na construção das malhas computacionais usadas para *e-Science*. Daremos especial importância ao globus pois ele fornece partes cruciais a diversas outras implementações de *middleware*, como por exemplo o *sistema de informação*. Depois apresentaremos o gLite que é o *middleware* utilizado tanto no EGEE quanto no EELA, pois foi nele em que as soluções desse trabalho foram desenvolvidas.

2.2.1 globus

O projeto Globus[64] criou um conjunto de ferramentas abertas de software livre que pode ser usado para a criação de infra-estruturas e aplicações distribuídas de malha computacional. Esse conjunto permite o compartilhamento de potência computacional, base de dados e outras ferramentas de maneira segura. Esse compartilhamento

pode ser estendido através de fronteiras institucionais e geográficas, sem o sacrifício da autonomia dos participantes.

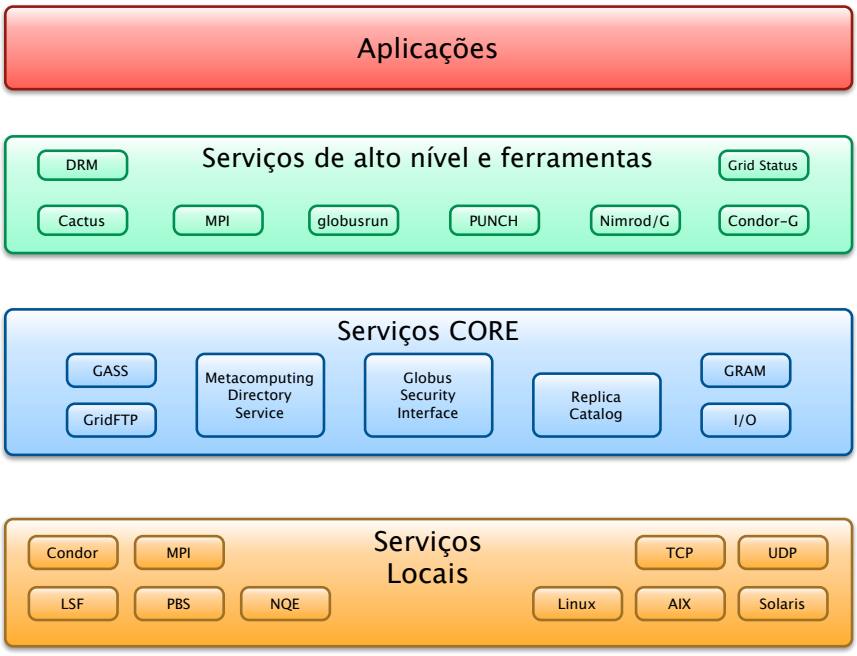


Figura 2.7: Arquitetura do Globus.

A arquitetura desse *middleware* é apresentada na Figura 2.7 e tem três grupos principais de serviços (*Core Services*) que são acessíveis através de uma camada de segurança (*Globus Security Interface*). Esses grupos são o gerenciamento de recursos — *Globus Resource Allocator Manager* (GRAM) e *Globus Access to Secondary Storage* (GASS), o gerenciamento de dados (GridFTP e catálogo de réplica) e o de sistema de informação — *Metacomputing Directory Service* (MDS).

A camada de serviços locais apresenta os serviços providos pelo sistema operacional local, como serviços de rede *Transmission Control Protocol* (TCP/IP), gerenciadores de filas de execução, gerenciamento de programas em paralelo, etc. As camadas superiores apresentam modelos que permitem a apresentação dos serviços de maneira uniforme permitindo a interoperação entre sistemas não homogêneos. Os serviços principais (camada intermediária) contêm todos os serviços necessários para o gerenciamento da segurança, submissão de trabalhos, gerenciamento de dados, sistemas de informação e outros.

GSI – camada de segurança

O *Grid Security Infrastructure* (GSI)[65] apresenta os métodos de autenticação dos usuários da malha e serviços de criptografia. Essa camada é baseada em *Security Socket Layer* (SSL), PKI e arquitetura de certificados X.509. A GSI fornece serviços,

protocolo e bibliotecas para os seguintes objetivos:

- Assinaturas para autenticação de usuários com o uso de certificados digitais;
- Autenticação de serviços através de certificados de máquina;
- Criptografia de dados;
- Autorização;
- Delegação de autoridade e *trust* com o uso de certificados *proxy* e correntes de confiança de unidades certificadoras (PKI).

As unidades certificadoras são também partes ativas na noção de organizações virtuais [66]. As organizações virtuais são colaborações verticais e o usuário que tem um certificado assinado por uma unidade certificadora aceita pela organização virtual tem acesso ao recurso autenticado pela mesma unidade certificadora. Esse conceito pode ser estendido para o caso onde existem grupos de unidades certificadoras que são aceitas por organizações virtuais. Dessa maneira, os usuários devidamente identificados por suas credenciais têm acesso aos recursos computacionais de uma organização virtual sem existir a necessidade de contato com os administradores locais dos mesmos recursos.

Outros serviços se baseiam na autenticação mútua antes de efetivamente serem servidos. Esse fato é importante pois garante o efeito de não repudiação, ou seja, usuário e servidor não podem repudiar a identidade de ambos os pares depois de efetuada a autenticação.

Efetivamente, do ponto de vista do computador local, cada usuário identificado por um certificado digital válido é mapeado, através de um arquivo gerenciado pelo administrador local, em usuários locais. Essa correspondência pode ser de "um para um" ou de "muitos para um". O mapeamento de "muitos para um" é encarado como um problema de segurança pois desde que dois usuários sejam mapeados no mesmo usuário local, eles podem ter acesso aos dados confidenciais de um e de outro.

Gerenciamento de recursos

Os serviços de gerenciamento de recursos permitem a alocação de recursos através da submissão de trabalhos. Além disso, esses serviços são responsáveis pela cópia dos arquivos executáveis, monitoramento de trabalhos e transferência de resultados. Os componentes do Globus para esses serviços são: GRAM (*Globus Resource Allocation Manager*) e *Globus Access to Secondary Storage* (GASS).

O GRAM [67] é responsável pela execução remota de trabalhos e relatório de estados. Com o uso do GRAM, o usuário demanda a submissão de um trabalho ao

processo *gatekeeper* em um computador remoto. O *gatekeeper* verifica se o cliente é autorizado (i.e., o certificado do cliente é mapeado no arquivo de mapeamento local em uma conta existente no computador remoto). Assim que o cliente ganha o acesso, o *gatekeeper* inicia um gerente de tarefas que inicia o trabalho e começa a sua monitorização. Os gerentes de tarefa são criados dependendo do sistema local de gerenciamento de filas — *Local Resource Management System* (LRMS) que pode ser o *Portable Batch System* (PBS), o *Load Sharing Facility* (LSF), etc.

Os detalhes do trabalho são especificados através do *Globus Resource Specification Language* (RSL), que é parte componente do GRAM. O RSL fornece uma sintaxe de pares de atributos e valores para a descrição de recursos necessários para o trabalho, incluindo o número de processadores e memória.

O *Globus Access to Secondary Storage* (GASS)[68] é um mecanismo de acesso a arquivos que permite à aplicação trazer antecipadamente arquivos para a máquina local e transferir os arquivos modificados de volta para o computador original. O GASS é usado para a transferência dos arquivos de entrada e executáveis necessários a execução dos trabalhos e para o envio dos arquivos gerados durante a execução. O GASS é baseado no uso de *streams Hypertext Transfer Protocol* (HTTP) seguro[69] e tem funções do GSI para garantir as permissões no acesso dos dados.

Serviço de informação

O serviço de informações do Globus provê informação estática e dinâmica dos nós conectados à malha computacional. Eesse pacote é chamado *Metacomputing Directory Service* (MDS)[70].

O MDS suporta a publicação e a demanda de informações sobre os recursos. Dentro do MDS, um esquema define as classes que representam diversas propriedades do sistema. O MDS apresenta uma estrutura dividida em três partes. Na base existem os provedores de informação que são responsáveis pela informação de estado dos recursos computacionais e efetuam a tradução para o formato de classes de objetos. O serviço *Grid Resource Information Service* (GRIS) efetua o serviço da segunda parte e é um processo que faz o levantamento de cada provedor de informação relevante, armazena versões sobre o estado e permite que essa informação seja interrogada pela terceira camada. A terceira camada é o *Grid Information Index Service* (GIIS) que cataloga a informação fornecida pelos GRISs e por outros GIIS, servindo de único ponto de acesso para toda a informação. Tanto o GRIS, quanto o GIIS são implementados em cima do *Lightweight Directory Access Protocol* (LDAP).

Gerenciamento de dados

O pacote de gerenciamento de dados fornece utilitários e bibliotecas para a transmissão, armazenamento e gerenciamento de grande massa de dados na malha que normalmente estão presentes nas aplicações [71]. Os elementos desse pacote são: *GridFTP* e *Replica Location Management*.

O *Grid File Transfer Protocol* (GridFTP) é uma extensão do popular protocolo de transmissão de arquivos FTP que implementa segurança e confiabilidade na movimentação massiva de dados na malha.

O *Replica Location Management* (RLM) suporta a distribuição dos dados (arquivos) em diversos lugares através de réplicas. Cada réplica deve ser registrada no serviço específico que permite o acesso ao arquivo mais próximo através do metadado do arquivo.

2.2.2 gLite

O gLite [72] segue um modelo *Service Oriented Architecture* (SOA) que facilita a interoperabilidade com outras iniciativas de computação em malha e permite a conformidade com os padrões, como *Open Grid Service Architecture* (OGSA) [73].

A arquitetura do gLite é constituída por um conjunto de serviços independentes das implementações existentes. Para atender as necessidades do usuário, esses serviços devem funcionar em conjunto, de maneira coordenada, contudo os serviços podem ser usados de maneira independente, que facilita o uso em diferentes contextos e a verificação de corretude.

A Figura 2.8 apresenta os serviços de alto nível que podem ser agrupados em cinco grupos temáticos.

Os serviços de segurança (*security services*) são responsáveis pela autenticação e serviços de auditoria, que são preparados para a identificação de entidades (usuários, computadores e outros serviços), permitindo ou negando o acesso aos serviços e recursos, além de fornecer dados *post-mortem* para análise de trabalhos. Além disso, esses serviços fornecem um ferramental para confidencialidade de dados e *proxies*.

A interface de programação e serviços de acesso à malha (*Application Programming Interface e Grid Access Services*) provêm um ferramental comum pelo qual o usuário é capaz de acessar os serviços de malha. O serviço de acesso gerencia o ciclo de vida dos serviços disponíveis ao usuário de acordo com seus privilégios. Ele também proporciona interfaces complexas para o acesso à malha como os portais *Genius* [74]. Além dessas ferramentas, existem bibliotecas para as linguagens de programação correntes para que o usuário possa acessar a infra-estrutura diretamente de seus programas.

Sistemas de informação e monitoramento são responsáveis pelos mecanismos de

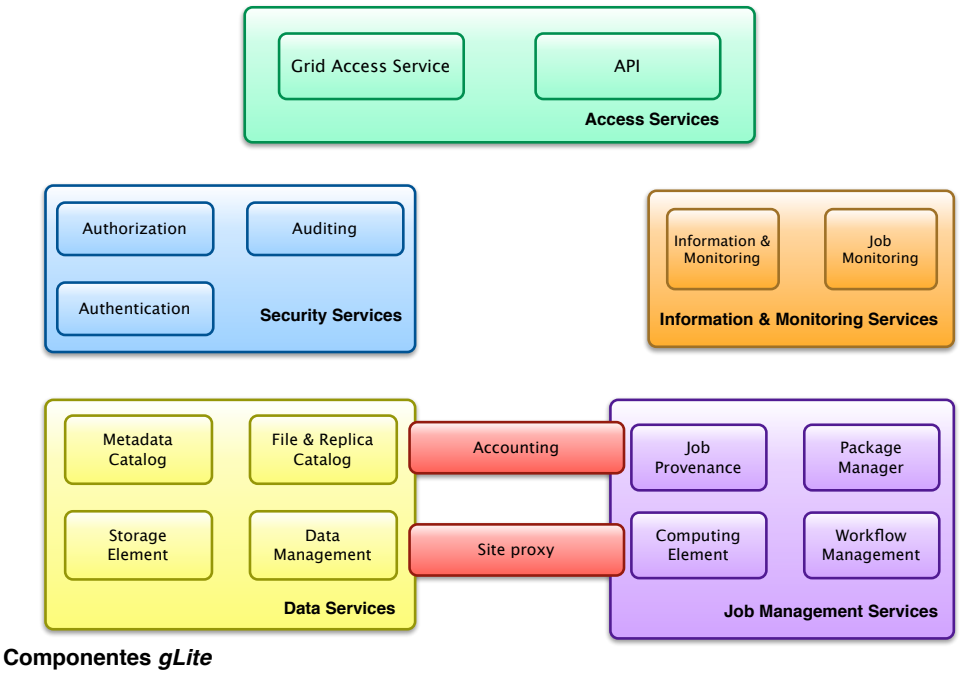


Figura 2.8: Arquitetura do gLite.

publicação e consumo de informação e seu uso para propósito de monitoramento. Esses sistemas podem publicar a informação explícita sobre o estado dos recursos da malha, ou através de serviços mais especializados como o sistema de monitoramento de trabalhos que pode agregar a informação dinâmica sobre a execução de um trabalho específico.

Toda a informação publicada apresenta a data e hora de última atualização, o que permite que esses dados também possam ser utilizados para propósitos de contabilidade, além de controles baseados em regras que regulam o acesso dessas informações pelos usuários.

Os serviços de gerenciamento de trabalhos (*Job Management Services*) apresentam interfaces para o gerenciamento e execução de trabalhos em elementos de computação, além de fornecer acesso ao gerenciamento de carga, contabilidade, histórico e gerenciamento de pacotes de aplicações. Nesse nível, a contabilidade agrega não só informações sobre os trabalhos, como um pacote completo de informações sobre o uso de recursos de armazenamento, use de rede, etc. Todo esse trabalho é feito através da comunicação entre os serviços de gerenciamento de trabalhos e os trabalhos propriamente ditos (com o uso de *job wrappers*), tentando manter a consistência e validade da informação existente no sistema.

Os serviços de dados (*Data Services*) são providos pelos seguintes componentes: elementos de armazenamento, serviços de catálogo (ou gerenciamento de metadados) e gerenciamento de dados. Todos esses serviços são sempre acessados através dos serviços de segurança que garantem o correto acesso somente a usuários devida-

mente autorizados.

No gLite existe a separação explícita do dado e meta-dado referente aos arquivos armazenados na malha. O usuário identifica um arquivo usando um nome lógico *Logical File Name* (LFN). Os LFNs são a chave para os usuários identificarem os arquivos e sua localização na malha. Um LFN, além de apontar para o arquivo original, pode indicar réplicas que são identificadas por *Site Universal Resource Locators* (SURL). Cada réplica tem o seu próprio SURL. Vale lembrar que os usuários não são expostos à complexidade do gerenciamento dos SURLs, mas somente ao LFN de um determinado arquivo. Os serviços de catálogo são os responsáveis pela manutenção da consistência entre os diversos LFNs e SURLs correspondentes.

Serviços de gerenciamento de dados (*Data Management System*) apresentam ao usuário as interfaces existentes para a colocação de dados que consiste num ambiente de movimentação de dados equivalente ao sistema de submissão de trabalhos, onde o usuário indica uma determinada movimentação de dados ao sistema e ele se encarrega de encontrar o melhor momento para efetuá-la. Esses serviços são de responsabilidade do *Data Scheduler*, *Transfer Fetcher*, *File Placement Service* e do *File Transfer Library*.

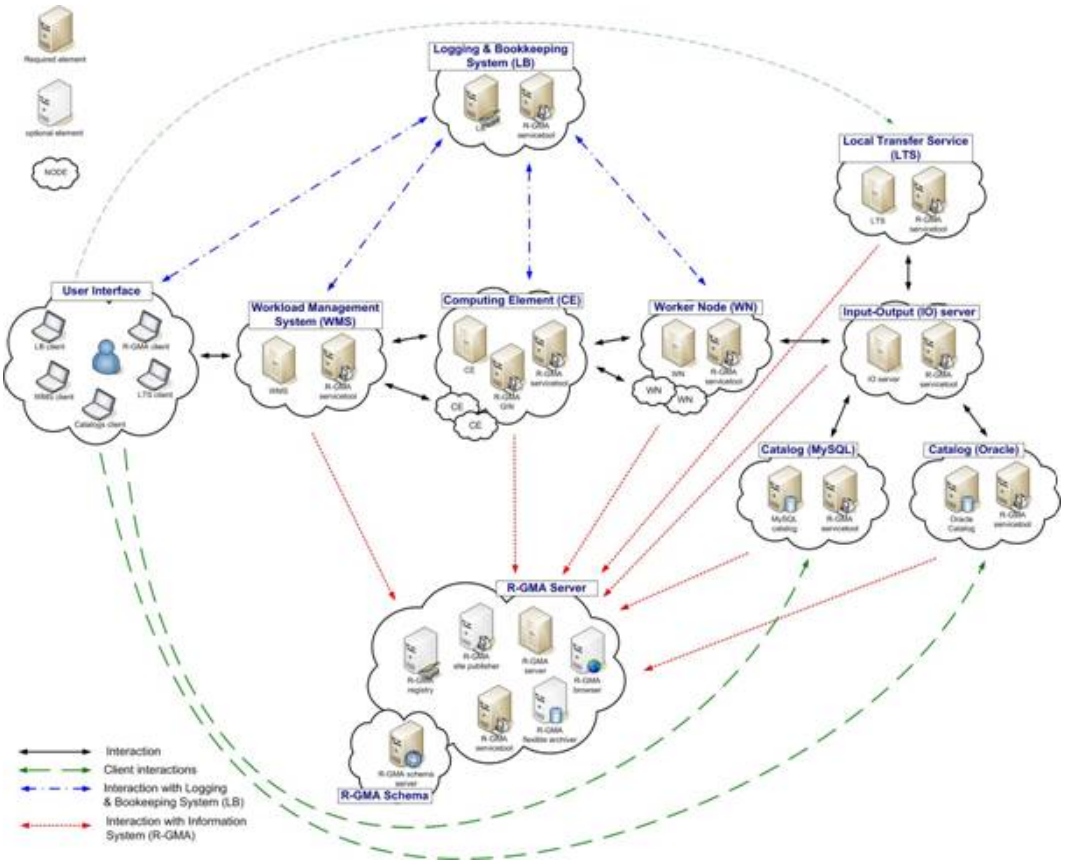


Figura 2.9: Macro componentes do gLite.

A Figura 2.9 apresenta os macro componentes do gLite, ilustrando a interação

entre os principais agentes do *middleware*, onde um trabalho (submetido de uma interface do usuário ou *User Interface*) é transferido para um WMS que avalia quais elementos de computação (CEs) são viáveis para a execução do trabalho. Entre os viáveis, o WMS escolhe um através de regras de *ranking* de recursos e efetua a transferência.

No elemento de computação escolhido, o trabalho é submetido ao sistema local de balanceamento de carga e de acordo com a disponibilidade é executado em um nó de trabalho WN. Quando o trabalho finaliza, os resultados imediatos são transferidos todo o caminho de volta até o WMS, onde ficam disponíveis para o usuário da malha.

Cada passo dado pelo trabalho é registrado no sistema de registro *Logging and Bookeeping System* (LB). Durante a execução do trabalho, a aplicação conta com todos os serviços apresentados pelos componentes adicionais, tais como base de dados, catálogos, servidores de entrada e saída, etc.

2.2.3 Outros *middlewares*

Legion

O Legion [75] é um *middleware* que combina um grande número de computadores, sistemas de arquivamento, base de dados e objetos heterogêneos e distribuídos de administração independente. Ele proporciona mecanismos para agregar recursos computacionais em um único *metacomputador* que apresenta um alto grau de flexibilidade e autonomia.

A Figura 2.10 mostra a arquitetura do *middleware* do Legion, que é constituído de um sistema de objetos distribuídos, onde processos ativos que se comunicam com o uso de um serviço de invocação remoto uniforme. Todo o hardware e software do sistema é descrito e representado por um objeto Legion. Os objetos fundamentais do Legion são descritos usando uma *interface description language* (IDL) e são compilados na linguagem alvo do usuário. Esse método permite que os componentes operem independente da linguagem escolhida, plataforma ou localização geográfica.

O Legion define um conjunto primordial de objetos que suportam os serviços básicos da malha computacional que são responsáveis por localização, criação, ativação, desativação e deleção. Além disso, eles provêem os mecanismos necessários para a implementação de políticas para as instâncias dos objetos. O Legion permite ao seu usuário a definição de suas classes baseadas nas classes existentes. Alguns dos objetos principais são:

- *host* – representam os processadores;
- *vault* – representam os sistemas de armazenamento persistente;
- *context* – efetuam o mapeamento entre os nomes lógicos e objetos;

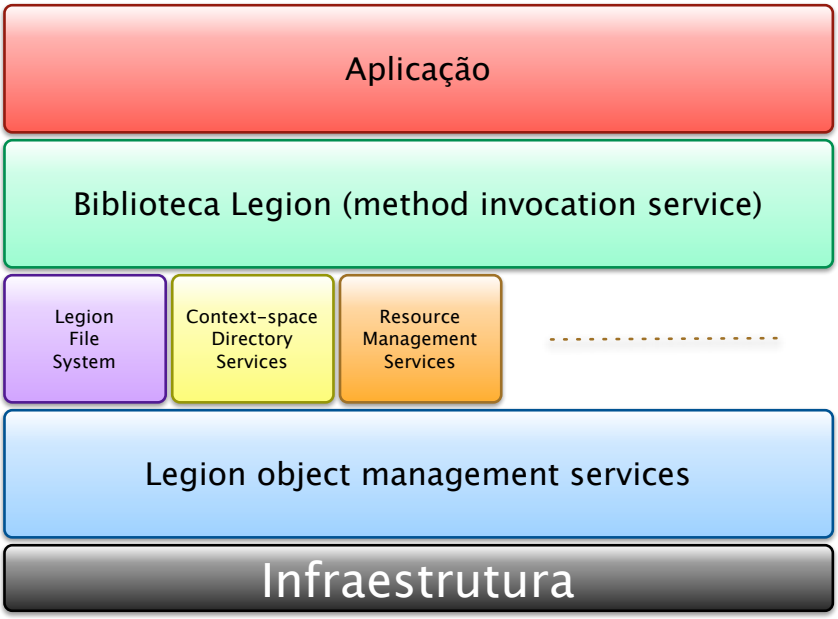


Figura 2.10: Arquitetura do Legion.

- *binding* – responsáveis pelo endereçamento;
- *implementation* – armazenam o arquivo executável que um *host* possa executar.

O sistema de mapeamento de nomes é formado por um esquema de três níveis: os nomes usados pelos usuários são chamados de *context names* que têm forma livre e facilitam a compreensão. Os *context names* são mapeados em objetos através de descritores de objeto do Legion, que são suficientes para o sistema de comunicação. Esses descritores de objeto são mapeados localmente em endereços locais de objeto, ou endereços físicos.

Unicore

O Unicore[76]é um ambiente de computação em malha vertical que proporciona:

- acesso de maneira transparente a recursos distribuídos pelos usuários;
- mecanismo de autenticação robusto e integrado nos procedimentos de administração;
- fácil re-alocação de trabalhos entre plataformas diferentes.

O Unicore segue uma arquitetura que é mostrada na Figura 2.11. Ela consiste de um cliente executando uma máquina virtual Java e diversas instâncias dos *Network*

Job Supervisor (NJS) que executam os servidores configurados e várias instâncias do *Target System Interface* (TSI), que executando em nós diferentes proporcionam acesso aos gerenciadores de recursos locais dos computadores.

Do ponto de vista do usuário podemos ver o sistema como:

- usuário – representa o usuário que está executando um cliente Unicore em seu computador pessoal;
- servidor – representa um centro de computação onde os clientes podem se conectar;
- sistema alvo – oferece o acesso aos recursos computacionais. São organizados em sítios virtuais e representam os recursos físicos nos centros computacionais.

A interface do cliente Unicore consiste de dois componentes principais: *Job Preparation Agent* (JPA) e *Job Monitor Component* (JMC). Trabalhos são montados com o uso dos JPAs e os resultados e status podem ser obtidos com o uso dos JMC. Os trabalhos e estados são formulados de maneira abstrata utilizando a classe Java *Abstract Job Object* (AJO). O cliente se conecta a um sítio virtual Unicore e através de um portal submete trabalhos utilizando os AJOs. O portal é simplesmente um ponto de acesso conhecido pelos clientes que fornece através de um endereço Internet e identificador de porta os serviços criptografados para a criação de tarefas.

Um sítio virtual Unicore é feito de até dois componentes: NJS e o TSI. O NJS efetua a gerência de todos os trabalhos criados e faz o processo de autorização de usuários utilizando a base de dados *Unicore User Data Base* (UUDB). O NJS também controla a encarnação dos trabalhos, fazendo a tradução do AJO em seqüência de comandos apropriados que são obtidas na *Incarnation Data Base* (IDB). O TSI recebe informações sobre os trabalhos encarnados do NJS e passa às para o sistema de execução do sistema local.

As funcionalidades do Unicore podem ser sumarizadas por:

- uma interface gráfica auxilia o usuário na criação de trabalhos complexos e trabalhos independentes podem ser executados em qualquer sítio Unicore sem modificações na definição;
- o sistema de gerenciamento de trabalhos fornece controle sobre trabalhos e dados;
- durante a definição do trabalho, o usuário pode especificar qual dado deve ser importado ou exportado para a sua área de trabalho. O Unicore determina quais dados devem ser movimentados e efetua as movimentações em tempo de execução;

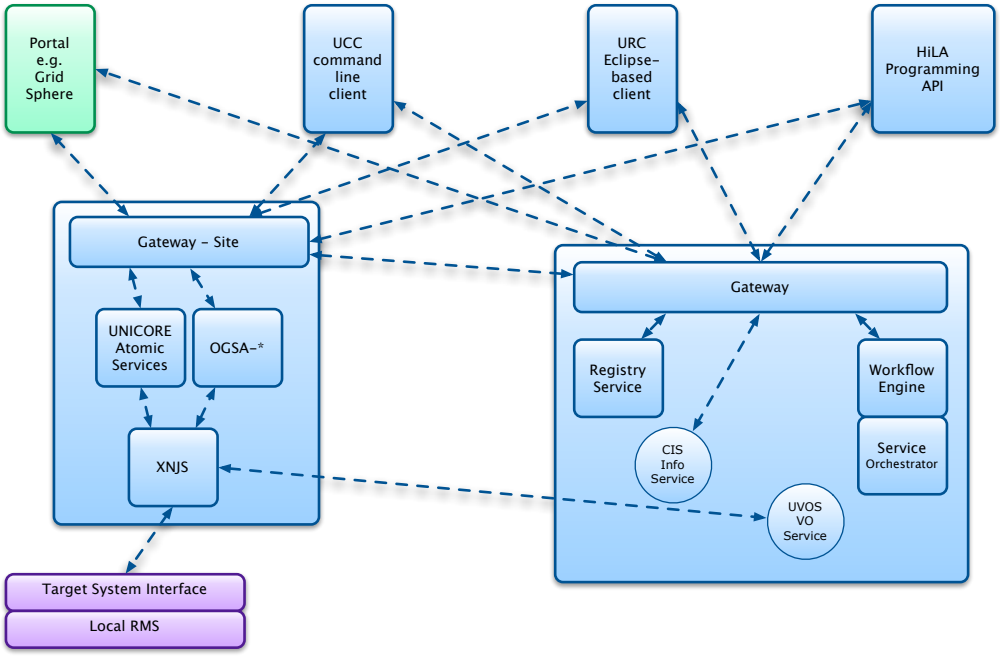


Figura 2.11: Arquitetura do Unicore.

- a interface do usuário é extensível através de *plug-ins*;
- o controle de fluxo dos trabalhos pode ser especificado através de grafos acíclicos;
- a autenticação é efetuada através de certificados X.509v3;
- o Unicore permite que trabalhos legados possam ser executados através de suporte adicional;
- o usuário especifica diretamente os recursos necessários e a interface, em tempo real, informa a disponibilidade dos recursos.

OurGrid

O Ourgrid [63, 77] é um *middleware* baseado em um paradigma ponto-a-ponto, onde uma rede de recursos computacionais é compartilhada entre uma comunidade de usuários durante os momentos em que os computadores não estão sendo utilizados. Esse compartilhamento é feito com base nas regras de doação de recursos que são especificadas por cada usuário de cada ponto. Cada ponto é um consumidor e provedor de recursos.

A principal característica do Ourgrid é o seu sistema de *rede de favores*. Quando um recurso é alocado a um consumidor, essa alocação é considerada um favor. Se

tratando de um favor, é esperado do consumidor o pagamento do favor com o proprietário do recurso alocado. Esse modelo é baseado na esperança que essa dívida seja paga em algum momento futuro e que os consumidores que não se comportam como esperado, têm a sua prioridade diminuída no sistema, fazendo com que ele passe a receber menos recursos.

Outro ponto importante sobre o Ourgrid é que ele é implementado completamente em Java e os trabalhos podem ser executados em máquinas virtuais que podem ser transferidas entre clientes e servidores.

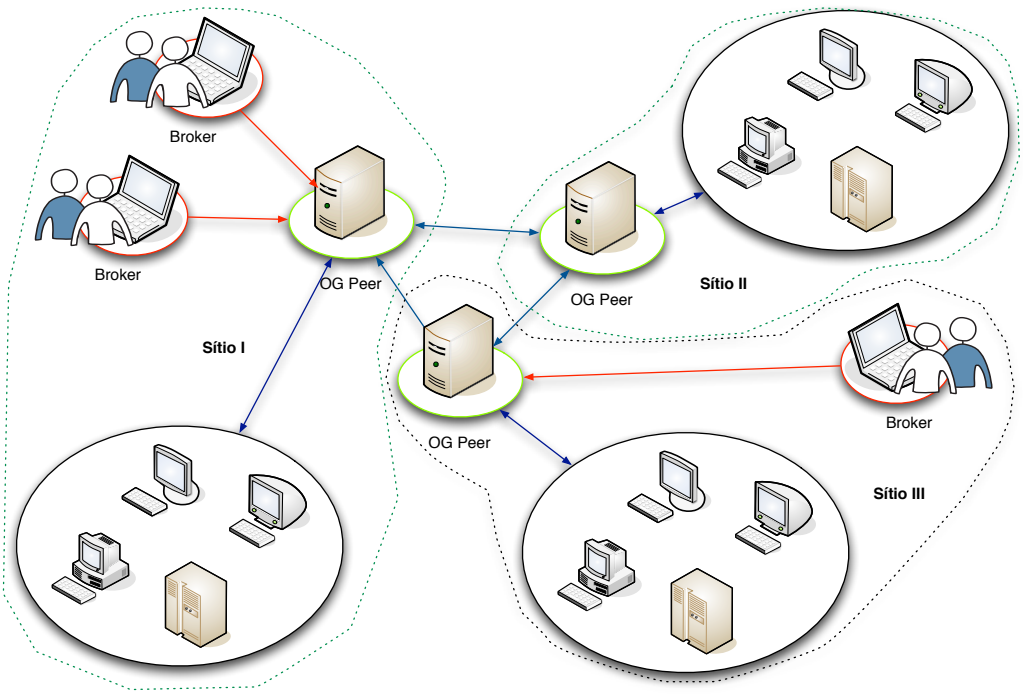


Figura 2.12: Componentes do Ourgrid (adaptado de [1]).

2.3 Sequenciamento

Basicamente, o sequenciamento em uma malha computacional é feito baseado em dois conjuntos de informação fundamentais: as tarefas submetidas pelos usuários com seus requisitos e a informação disponível sobre o estado dinâmico da malha [78]. De posse desses dois conjuntos de informação, o sequenciamento é feito com base em alguma função objetivo e alguma predição do desempenho dos recursos disponíveis. Diferente do sequenciamento feito nos sistemas paralelos e distribuídos tradicionais [79], o sequenciamento na malha normalmente não pode controlar diretamente os recursos envolvidos pois eles pertencem a entidades administrativas diferentes [80, 81].

Além disso, o sistema de sequenciamento pode estar organizado de maneiras dis-

tintas na própria infraestrutura (centralizado, hierárquico ou descentralizado [82]), dependendo do foco em escalabilidade ou desempenho. Na realidade, o sistema de sequenciamento, às vezes chamado de *Metascheduler* [83], não é um componente indispensável da infraestrutura pois não se encontra presente nas distribuições do Globus ou de outros *middlewares*.

O gLite apresenta uma extensão no serviço WMS que permite o sequenciamento de tarefas parametrizadas ou relacionadas por uma seqüência de dependências especificadas em um grafo acíclico. Contudo, esse sistema simplesmente atua como um despacho de tarefas por causa da impossibilidade de se mudar as funções objetivo, ou seja, a cada despacho, somente uma simples verificação de recursos é feita com base nos requisitos fornecidos pelos usuários.

A acurácia da informação disponível sobre o estado dinâmico da infraestrutura, ou seja, quais recursos estão disponíveis em um dado instante, é um ponto crucial para a qualidade das decisões do sistema de sequenciamento, especialmente na malha computacional por causa da diversidade e heterogeneidade dos recursos existentes e assimetrias nos canais de comunicação.

Além disso, existe um outro intermediário entre o sistema de sequenciamento e o recurso computacional: o LRMS. Ele é responsável pelo sequenciamento no domínio local (composto pelas tarefas oriundas da malha e dos usuários locais) e pelo provimento de informações aos sistemas responsáveis (ex. GIS no Globus) pela manutenção da informação sobre o estado dinâmico do sítio na malha.

2.4 Eficiência das *e-Infrastructure*

Recursos que são propriedade de várias instituições diferentes são federados em uma *e-Infrastructure* com base em políticas locais. Essas políticas especificam o que é compartilhado, quem é autorizado a acessar e o quê pode ser feito [84]. Sendo assim, um dois maiores problemas que existe intrinsecamente no paradigma de computação em malha é a coordenação do compartilhamento dos recursos e a execução de tarefas em um ambiente multi-institucional, dinâmico, composto por organizações virtuais formadas por recursos heterogêneos [66].

Nesse contexto, analisamos as evidências que esse ambiente, apesar de proporcionar uma quantidade não trivial de recursos computacionais, também apresenta um problema inerente a sua constituição. Groep et al. [85] apresentam resultados que demonstram que a taxa de falha global de execução de tarefas no EGEE varia entre 25% e 33%. Ou seja, 1 em cada 3 ou 4 tarefas falha antes de completar a sua execução. Esses dados foram obtidos em períodos distintos de operação da infraestrutura e compreende análise de cargas de até 60.000 tarefas.

No EELA, fizemos o levantamentos em 2007 e 2009 sobre a eficiência da infra-

estrutura e encontramos resultados melhores, mas compatíveis. Em dezembro de 2007, a percentagem de falhas da infraestrutura estava em 28,5% e em dezembro de 2009 ela era de 19%. O bom resultado em 2009 deve-se principalmente a que os testes foram feitos com um número menor de RCs, maior engajamento dos administradores, com melhores condições nas redes e sincronização entre os relógios via *Global Positioning System* (GPS).

Além disso, Groep et al. [85] também mostram que somente de 5% a 8% dos jobs falham depois de efetivamente serem transferidos ao CE (depois do passo 2 na Figura 2.3). Ou seja, existem de 20% a 25% de jobs que são perdidos nos passos 1 e 2 da Figura 2.3.

No caso do EGEE, a grande parte do custo do projeto[8] (aproximadamente 60%) é dedicado aos times de operação e ao desenvolvimento das ferramentas de monitoramento. Esse gasto é feito com o intuito de se aumentar a eficiência pois se acredita que com o monitoramento constante podemos aumentar a eficiência identificando o mais cedo possível um RC com algum problema. Para tal, um pacote de ferramentas como o SAM (*Service Availability Monitoring*)[86] foi desenvolvido. O SAM é responsável pela submissão de tarefas-teste na infraestrutura para a verificação do funcionamento correto de diversos módulos do *middleware*. Esses testes são armazenados em uma base de dados e são verificados sistematicamente por operadores que se encarregam de contactar o RC que apresentar problemas. Além do SAM, também é usado uma adaptação do *software* de monitoramento de redes chamado de NAGIOS[9]. A versão de *grid* do NAGIOS submete o mesmo conjunto de tarefas-teste do SAM, contudo ele é considerado mais estável e consome menos recursos para efetuar um mesmo monitoramento.

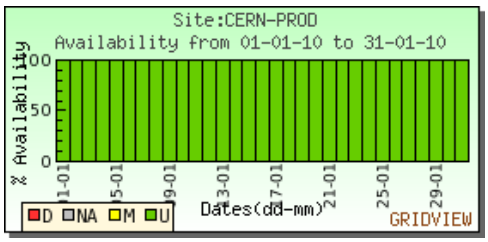
Mesmo com todo esse investimento, a chance de uma tarefa executar na infraestrutura continua em padrões semelhantes. Como exemplo vamos olhar três dos mais importantes centros de recursos do EGEE: *CERN - Tier 0 Production Site* (CERN-PROD), *UK Tier 1 Centre* (RAL-LCG2) e *Centre de Calcul de l'IN2P3 - French Tier 1* (IN2P3-CC)[10].

Nas Figuras 2.13, 2.14 e 2.15 podemos observar os resultados do monitoramento do mês de janeiro de 2010 dos três sites. O gráfico *(a)* em cada figura mostra o resultado da disponibilidade que indica a percentagem do sucesso das tarefas-teste num RC específico, sendo verde para quanto o teste obtém sucesso, amarelo para quando o RC declara que está fechado para manutenção e vermelho para quando o teste falha. No caso do CERN-PROD, a eficiência das tarefas-teste foi de 100%. Já a Figura 2.13(b) mostra o resultado para um único componente do *middleware*,

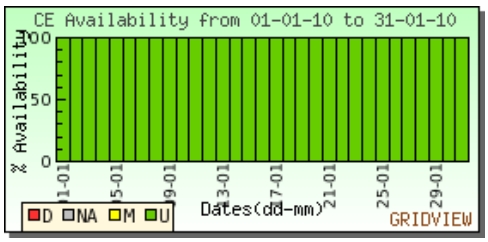
[8] Custo do EGEE é de 32 Milhões de Euros

[9] http://www.nagios.org

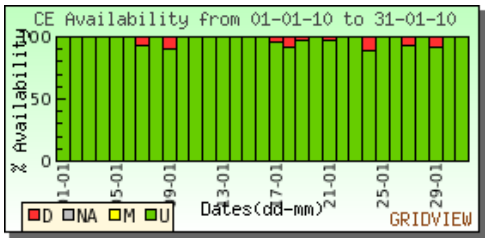
[10] Dos três, somente o RAL-LCG2 não faz parte da infraestrutura do EELA.



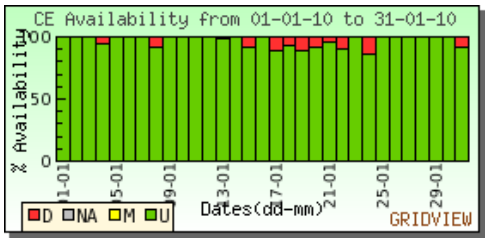
(a) Disponibilidade total



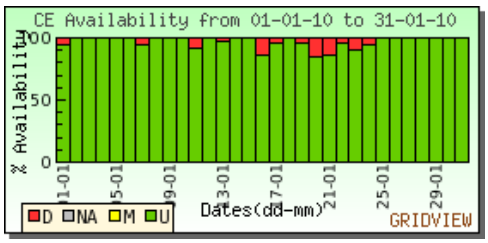
(b) Disponibilidade do serviço CE



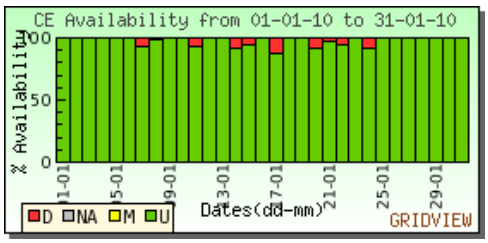
(c) Disponibilidade do ce103.cern.ch



(d) Disponibilidade do ce104.cern.ch

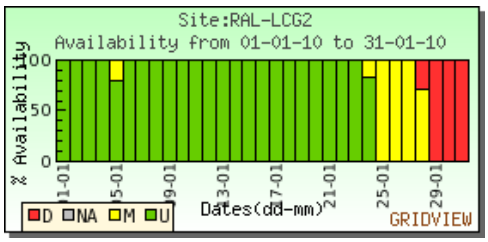


(e) Disponibilidade do ce105.cern.ch

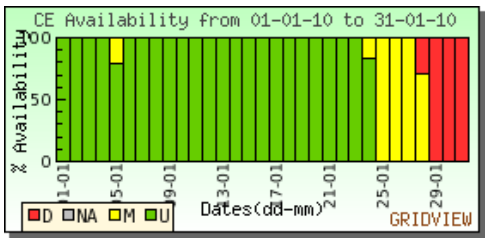


(f) Disponibilidade do ce106.cern.ch

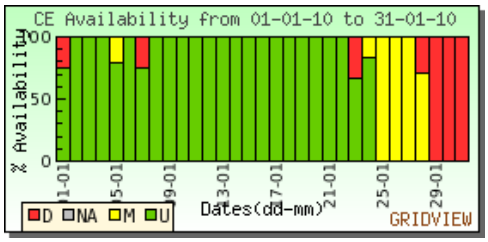
Figura 2.13: CERN-PROD - Tier 0.



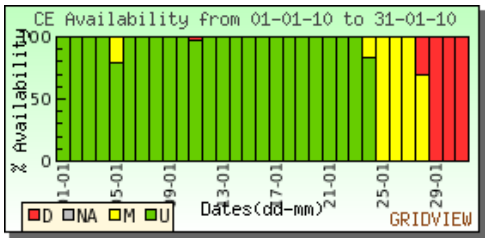
(a) Disponibilidade total



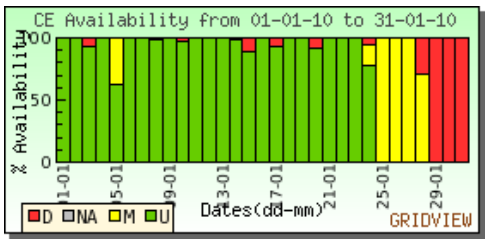
(b) Disponibilidade do serviço CE



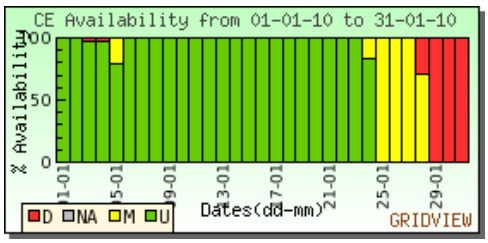
(c) Disponibilidade do ce.ngs.rl.ac.uk



(d) Disp. do lcgce02.gridpp.rl.ac.uk



(e) Disp. do lcgce06.gridpp.rl.ac.uk

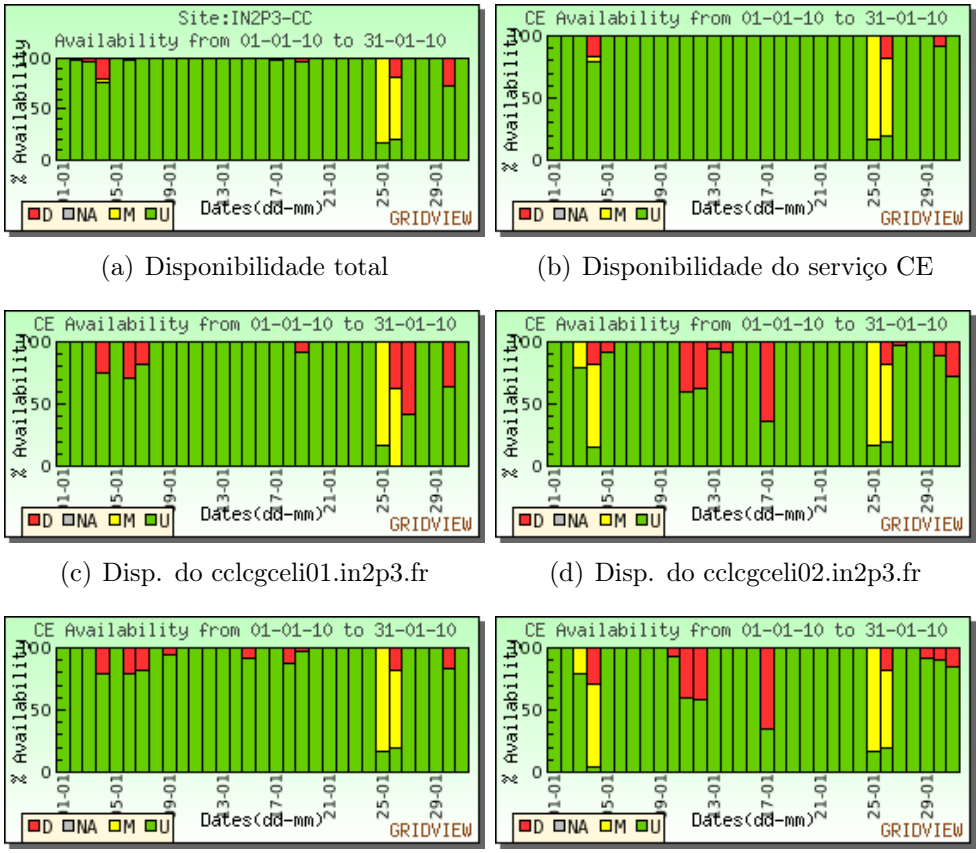


(f) Disp. do lcgce07.gridpp.rl.ac.uk

Figura 2.14: RAL-LCG2 - Tier 1.

nesse caso o CE. No caso do CERN-PROD, a eficiência do serviço CE foi de 100%. Contudo, nos gráficos seguintes, podemos observar que os 100% de disponibilidade foram obtidos as custas do uso de redundância dos serviços, pois nas Figuras 2.13(c), 2.13(d), 2.13(e) e 2.13(f), podemos observar que os CEs apresentaram individualmente falhas durante o período observado. Devemos salientar que se o WMS tivesse escolhido um CE prestes a falhar, a execução da tarefa iria falhar, contribuindo para corroborar os números encontrados por Groep et.al. [85] e por nós na operação da infraestrutura.

Nas Figuras 2.14 e 2.15 podemos observar que mesmo com o uso de redundâncias, a disponibilidade total dos RCs em questão ficou bem abaixo do resultado encontrado no CERN-PROD. Na realidade, tirando o CERN-PROD, todos os RCs do EGEE têm disponibilidades totais menores que 100%.



(a) Disponibilidade total

(b) Disponibilidade do serviço CE

(c) Disp. do cclcgceli01.in2p3.fr

(d) Disp. do cclcgceli02.in2p3.fr

(e) Disp. do cclcgceli03.in2p3.fr

(f) Disp. do cclcgceli04.in2p3.fr

Figura 2.15: IN2P3-CC - Tier 1.

Capítulo 3

Fluxo de processo de aplicações de *e-Science*

A história moderna de fluxo do processo pode ser remetida a Frederick Taylor [87] e Herry Gantt [88]. Juntos, eles começaram um estudo racional da organização do trabalho dentro de um contexto industrial de manufatura. No início do século passado, Frederick Taylor e seus contemporâneos estavam envolvidos primeiramente com fluxos de materiais e de energia. Depois, focaram nas linhas de montagem que continuam como um dos exemplos mais importantes da sua época de fluxo de trabalho ou processos. Geralmente, a noção de fluxo era voltada para uma simples divisão das tarefas ou do processamento e, com o passar dos anos, modelos conceituais de pesquisa operacional, incluindo fluxos de chão de fábrica e teoria das filas, começaram a ser compreendidos [89].

Os exemplos seguintes ilustram a variedade de fluxos de trabalho que são observados em contextos distintos:

(i) no planejamento militar, o conceito de operações é um fluxo de processo que define o tipo particular de missões de combate ou suporte;

(ii) no chão de fábrica, particularmente os regidos por sistemas em *Job Shop*, a viagem de uma peça constituinte de um produto por diversas estações de trabalho constitui um fluxo;

(iii) o processamento dos pagamentos de seguros com sinistro é um exemplo de um sistema intensivo em informação e com um fluxo de processo baseado e controlado por documentos;

(iv) já os sistemas *Wiki*[1] são exemplos de sistemas baseados em fluxos estocásticos.

[1] o maior exemplo de sistema *Wiki* é a Wikipedia - http://wikipedia.org

Apesar dos exemplos apresentarem um uso bem diverso dos fluxos de processos, foi no ambiente empresarial que eles passaram a ser uma ferramenta de compartilhamento do conhecimento[90]. É nesse contexto que eles passam a ser usados para se documentar o trabalho, inclusive como base da memória dos processos existentes na empresa [91–93].

Já no ambiente científico, principalmente nos grandes experimentos *Alice*[35], *Atlas*[36], *CMS*[37], *LHCb*[38] e *Laser Interferometer Gravitational Wave Observatory* (LIGO)[94], por exemplo, a necessidade do uso de *e-Infrastructure*s obrigaram o uso dos fluxos com o mesmo intuito dos ambientes empresariais: a documentação do conhecimento. Nesse caso, todo o fluxo de processamento de dados é documentado, inclusive suas interdependências e características. Em outros casos, a origem do uso não se encontra no volume dos dados ou grandiosidade do ferramental (recursos computacionais), o uso é imposto pela complexidade do processamento e no número de tarefas envolvidas[95].

Além dos motivos apresentados acima, no domínio científico, é crescente a necessidade do uso de sistemas de gerenciamento de fluxo para possibilitar a construção de aplicações e ferramentas que consigam orquestrar o uso dos recursos distribuídos da *e-Infrastructure*. É importante lembrar que nessa infraestrutura os recursos estão espalhados por regiões administrativas diferentes, o que viabiliza a integração de equipes diferentes no uso dos fluxos, uso de recursos específicos e até a redução dos custos de processamento[2].

Como exemplo dessa integração entre equipes multi-disciplinares e multi-institucionais podemos citar uma aplicação que permite o compartilhamento de *espaços de conhecimento*: Tupelo [96], que é um sistema que foi construído com uma combinação de *Web-semântica*, sistemas de gerenciamento de conteúdo e tecnologias de controle de fluxo de processos, tudo isso centrado em um *middleware* que permite a localização, uso, relacionamento e discussão de dados e *meta-dados* em ambientes distribuídos.

Por outro lado, pode ser citado que motivos mais simples levam à adoção de fluxos de processos e de seus sistemas de controle. Marru [97] apresenta que a necessidade do uso de sistemas de controle de fluxo pode surgir simplesmente pois compartilhar aplicações de comando de linha não é trivial por causa de diversas dificuldades praticas e técnicas. Além disso, as aplicações e ferramentas não são fáceis de se instalar e necessitam de versões especializadas de linguagens e sistemas operacionais.

Hoje, a visão de futuro é que os sistemas de fluxo de processo, aliados aos desenvolvimentos da computação semântica [98] possam permitir ao cientista trabalhar com a manipulação dos dados e da *e-Infrastructure* de maneira que essas tecnologias sejam indistintas do processo científico.

O resto do capítulo se organiza da seguinte maneira. Na próxima seção, apresentaremos as definições dos sistemas de gerenciamento de fluxos de processos, seguido pelas características que são desejadas nesses sistemas (*desideratum)*. Na Seção 3.3 apresentaremos a taxonomia que é mais comumente aceita na Academia, seguido por formas de representação do fluxo na Seção 3.4. Finalizamos o Capítulo com as formas de mapeamento dos fluxos nos recursos da *e-Infrastructure* e por uma breve Conclusão.

3.1 Sistemas de gerenciamento de fluxos de processos

Chen [4] define o sistema de gerenciamento de fluxo de processos (Figura 3.1) como um componente de alto nível do *middleware* que suporta a (i) modelagem, (ii) instanciação e (iii) execução de aplicações científicas e comerciais sofisticadas e de larga-escala. Essa definição é fundamentada em diversos trabalhos anteriores [10, 26, 99–102]. Para desempenhar as três componentes, esse sistema deve ser capaz de representar e diferenciar internamente os fluxos especificados e suas instâncias de execução [100, 103, 104].

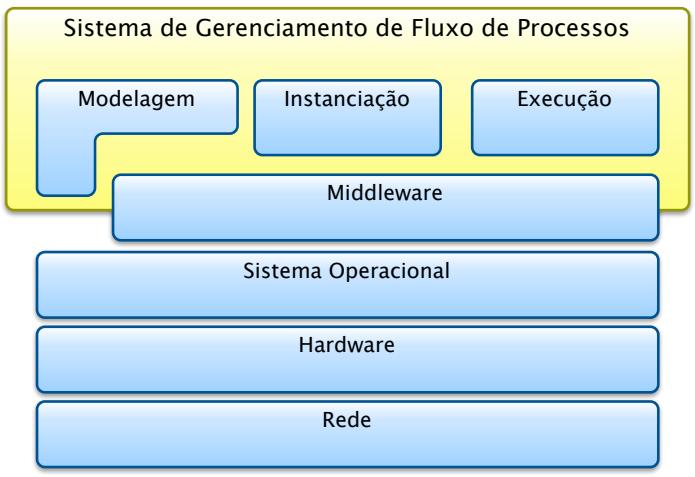


Figura 3.1: Sistema de gerenciamento de fluxo de processos.

De acordo com [105–108], podemos dizer que o controle efetuado em uma instância de execução por um SGFP deve também tratar com computações data-intensiva, assim como as interdependências existentes entre elas. Essa orquestração é normal-

mente feita por serviços de computação em malha [105, 108, 109] e determina a ordem de execução das tarefas que compõem o fluxo [105, 109, 110]. Durante o estágio de execução, as tarefas que se encontram prontas para a execução são despachadas pelo SGFP através da coordenação da execução dos serviços de computação em malha, o que obriga ao SGFP ser construído usando os serviços disponíveis pelo *middleware* utilizado [105, 110]. Esse fato caracteriza o SGFP como um componente de alto nível do sistema de computação em malha, mas não necessariamente parte integrante do *middleware* como apresentado anteriormente.

Além das principais componentes apresentadas acima, outros elementos aparecem com menor ênfase em diversos SGFP. Dentre os mais importantes, podemos citar: a validação das representações internas dos fluxos de processo [4], o controle de autorização do acesso aos dados e aos recursos [111], gerenciamento das falhas de execução [112, 113] e a verificação da corretude temporal de sua execução [114, 115].

A Tabela 3.1 apresenta a lista dos mais importantes SGFPs existentes na atualidade.

Tabela 3.1: Lista dos principais SGFP.

Nome	Referência
DAGMan	[116]
Pegasus	[103]
Triana	[117]
ICENI	[118]
Taverna	[119]
GrADS	[120]
GridFlow	[121]
UNICORE	[122]
Gridbus workflow	[123]
Askalon	[103]
Karajan	[124]
Kepler	[23]
GRAND	[125]
GridWay	[126, 127]

3.2 Características dos fluxos de processos

De uma maneira geral, a tecnologia envolvida com os fluxos de processos em aplicações de *e-Science* está sujeita aos desenvolvimentos relacionados com suas respectivas áreas de aplicação científica[128]. Dessa maneira, no lugar de se listar todas as características de maneira exaustiva, listaremos primeiro alguns padrões que normalmente ocorrem nos fluxos e depois apresentaremos uma revisão das características funda-

mentais que devemos encontrar nos sistemas de gerenciamento de fluxo de processo (SGFP).

3.2.1 Padrões de fluxo de processos

Na Figura 3.2 podemos encontrar os padrões básicos que ocorrem nos fluxos de processos. O primeiro padrão é o de *(i) seqüência*, onde diversas tarefas são executadas de maneira sucessiva e existe dependência entre elas que pode ser de dados ou temporal. O segundo padrão apresentado é o de *(ii) divisão paralela*. Nesse padrão, ocorre no fluxo uma divisão entre o controle de uma tarefa em múltiplas. As tarefas resultantes não necessariamente devem executar em paralelo, mas podem se desejado. Além disso, não existe nenhuma ordem garantida de execução entre as tarefas resultantes.

O próximo padrão básico é o de *(iii) sincronização*. Nele, diversas atividades paralelas, possívelmente criadas em uma divisão paralela, ocorrem e convergem. O controle resultante, depois de todas as tarefas convergentes terminarem, é transferido para uma única tarefa. No caso da *(iv) escolha esclusiva*, somente o ramo com resultado verdadeiro é executado e os outros não são mais avaliados. Já o quinto padrão é o de *(v) fusão* e ocorre onde existe um ponto de decisão para continuação da execução. Nesse caso, o fluxo continua quando a primeira tarefa com resultado verdadeiro ocorre. Observe que como o resultado é analisado, é possível que tarefas que resultem em resultados falsos sejam executadas antes do término da primeira tarefa com resultado verdadeiro.

Apesar de descrever fluxos de processos em ambiente de negócios, van Der Aalst[128] já fazia referência aos cinco padrões apresentados. Além disso, em seu trabalho, ele apresenta outros padrões mais elaborados como os apresentados na Figura 3.3. Neste trabalho, focaremos, em princípio, somente nos padrões básicos, pois eles contemplam a grande maioria das ferramentas e aplicações de *e-Science* existentes quando os comparamos com os trabalhos existentes [2, 3, 23, 103, 122, 129–138].

3.2.2 Características esperadas nos sistemas de gerenciamento de fluxo de processo

Mc Phillips [138] apresenta um *desideratum* dos SGFP. Dentro das características fundamentais podemos citar:

Clareza - um cientista compondo um novo fluxo de processo tem uma boa idéia sobre o que o fluxo deverá fazer durante a execução. Idealmente, o sistema deve confirmar ou contradizer a expectativa do cientista e apresentar uma resposta imediata. Nos sistemas correntes, as expectativas são normalmente

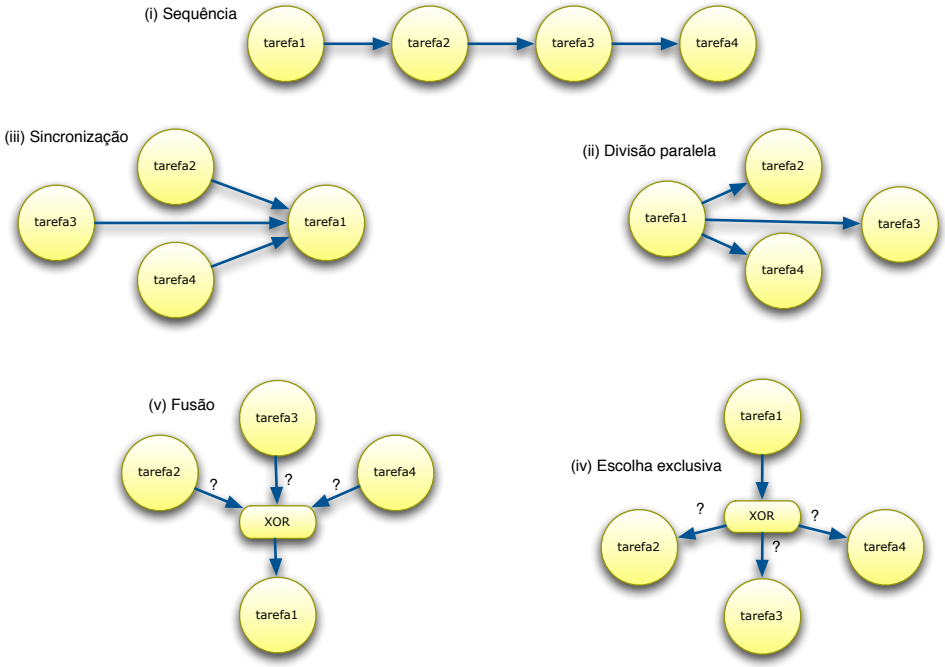


Figura 3.2: Padrões básicos de fluxo de processos.

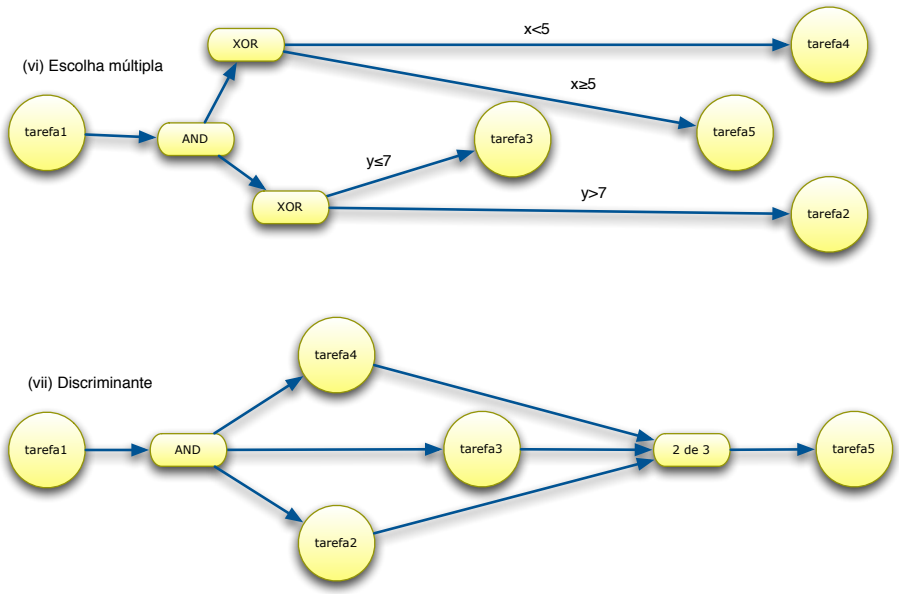


Figura 3.3: Padrões complexos de fluxo de processos.

confirmadas em tempo de execução, com o auxílio dos dados intermediários do processamento. Devido ao fato que pode ser impossível rodar o fluxo para testes, pois o tempo de execução pode ser extenso ou não se encontrar disponível os recursos computacionais, a clareza sobre o comportamento de um modelo do fluxo é muito importante;

Facilidade na criação de fluxos corretos e bem formados - os fluxos são muito mais que simples receitas ou protocolos de pesquisa. Eles são como programas de computador com o objetivo de produzir resultados bem definidos, com entradas bem definidas;

Facilidade na verificação dos resultados - o entendimento do funcionamento de um fluxo a *posteriori* é de fundamental importância para o cientista poder verificar os dados obtidos. Infelizmente, por causa da execução paralela, a gravação dos resultados intermediários pode não ser suficiente para se analisar o que foi executado por um fluxo;

Reportabilidade - deve ser fácil comprovar os resultados científicos produzidos por um fluxo. Cientistas não precisam somente entender o processamento de dados, mas precisam entender como os produtos do fluxo foram produzidos;

Reusabilidade - deve ser fácil reutilizar pedaços dos fluxos já utilizados, pois se aumenta a confiança do cientista quando ele é capaz de usar partes de fluxos com comportamentos já comprovados;

Modelagem de dados - sistemas de gerenciamento de fluxos que não são capazes de gerenciamento de conjunto de dados impossibilitam o compartilhamento de informação dentro da comunidade;

Otimização automática - os sistemas não devem exigir que os cientistas se especializem em otimizar a execução do fluxo, pois eles devem se dedicar a compreensão do problema a ser resolvido.

Não devemos esquecer que mesmo apresentando as características acima, os sistemas de gerenciamento de fluxo são fundamentalmente responsáveis: (a) pela habilidade de se construir aplicações que orquestram o uso de recursos distribuídos; (b) pelo uso das características dos recursos para aumentar o desempenho, diminuindo os custos de execução; (c) por efetuar a execução em domínios administrativos diferentes e (d) a integração de diversos times envolvidos no gerenciamento das diferentes partes do fluxo de processos, o que promove a organização de colaborações entre os times[2].

Por outro lado, a própria execução e coordenação do fluxo apresenta um custo elevado que está relacionado com os seguintes fatores: custo do *middleware*, custo

da transferência de dados, perda de paralelismo e custo relacionado com as atividades do sistema de controle do fluxo [139, 140]. Todos esses custos se somam ao caminho crítico de execução de um fluxo, impactando diretamente no tempo total de execução. Devemos também adicionar que qualquer falha no funcionamento do *middleware* ou uma falha de *hardware* em um WN pode afetar diretamente nos custos de *middleware* e nos custos com as atividades do sistema. Como mostrado no capítulo anterior, entre 25% e 33% das tarefas despachadas no mínimo terão que ser re-despachadas. Para agravar a situação, Iosup et al. [141] mostram que a *Mean Time Between Failures* (MTBF) do *grid* é de somente 12 minutos e de 5 horas para um RC.

3.3 taxonomia

Yu e Buyya [2, 3] apresentam uma das taxonomia dos fluxos de trabalho mais citados. Nela, representada na Figura 3.4, o fluxo pode ser classificado segundo as seguintes dimensões: *desenho*, *sequenciamento*, *tolerância a falhas* e *movimentação de dados*.

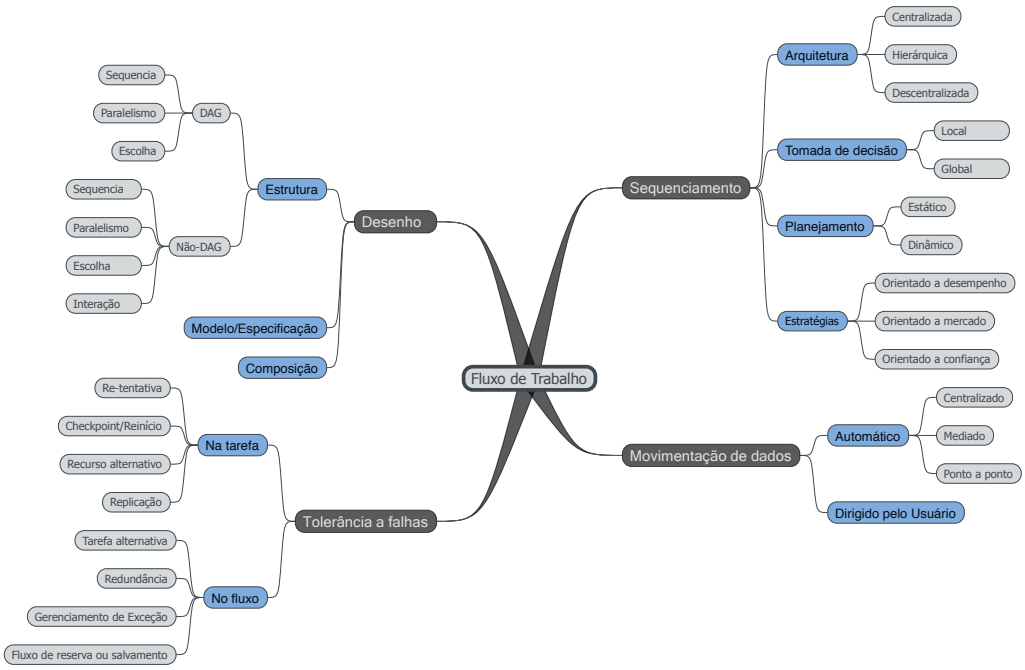


Figura 3.4: taxonomia de fluxos de processo proposta por Yu e Buyya [2, 3].

A dimensão do desenho classifica os fluxos quanto a sua estrutura, ou seja a ordenação temporal de suas tarefas. Essa ordenação temporal pode ser representada por um grafo simples ou por um grafo acíclico, em inglês *Directed Acyclic Graph* (DAG). Além da estrutura, o desenho de um fluxo pode ser classificado pelo modelo empregado, que pode ser abstrato ou concreto. O modelo abstrato ocorre quando

não existe nenhuma ligação entre a estrutura do fluxo e os recursos computacionais nos quais as tarefas serão executadas. Já no modelo concreto existe uma função de mapeamento total ou parcial das tarefas com os recursos computacionais. A última subdivisão do desenho é relacionada com o modo em que o usuário executa a composição do fluxo. A composição pode ser dirigida pelo usuário através de linguagens de especificação ou de ferramentas gráficas. Já na composição automática, o usuário apresenta especificações relacionadas com produtos finais e intermediários.

A dimensão do sequenciamento classifica como os sistemas de gerenciamento de fluxo controlam a execução das tarefas componentes. A primeira classificação é segundo a arquitetura que determina se o controle é centralizado, hierárquico ou descentralizado. No ambiente centralizado, um escalonador central realiza todas as decisões sobre todas as tarefas que compõem um determinado fluxo de processo. No hierárquico, um escalonador central e múltiplos escalonadores escravos trabalham em conjunto sobre as decisões. O central fica responsável pelo controle da execução do fluxo e delega para os escravos partes do fluxo para controle *local*. Por outro lado, o ambiente descentralizado tem diversos escalonadores que tomam as decisões sobre o funcionamento de todo o fluxo de processo em conjunto ou não. Acredita-se que a arquitetura centralizada produz resultados melhores pois toda a informação necessária se encontra no escalonador central. Contudo, essa categoria não consegue escalar com o volume de tarefas, com o número de recursos e classes dos mesmos.

A vantagem do uso da arquitetura hierárquica é que diferentes políticas podem ser usadas no escalonador central e nos escravos [142]. Contudo, tanto na arquitetura central quanto na hierárquica, o escalonador central constitui um ponto central de falhas. O escalonador distribuído consegue trabalhar com sistemas de escalas diferentes, mas encontra maior dificuldade em prover soluções mais próximas do ótimo.

Outra dimensão é relacionada à tomada de decisão que pode ser local ou global. Decisões locais de sequenciamento só afetam a tarefa que está sendo despachada para execução. Já as decisões globais analisam todo o fluxo a cada despacho. Outra classificação é segundo o esquema de planejamento que pode ser estático ou dinâmico. No esquema estático, modelos concretos são gerados antes da execução, de acordo com a informação existente sobre o ambiente de execução. Nesse caso, as mudanças de estado dos recursos não são contabilizadas. Por outro lado, o esquema dinâmico utiliza toda a informação disponível sobre os recursos para a planejamento durante a execução. Esquemas estáticos podem ser classificados em direcionados e simulados. Nos direcionados, os usuários podem emular o processo de escalonamento e efetuar as decisões de mapeamento de recursos de acordo com os critérios de conhecimento prévio, preferência e/ou desempenho. Nos simulados, um "melhor" escalonamento é obtido pela simulação da execução das tarefas dado um conjunto

definido de recursos *a priori*. A simulação pode ser processada com base nas informações estáticas disponíveis ou resultados de uma estimação do desempenho do dado conjunto de recursos.

Esquemas dinâmicos incluem sistemas de previsão e escalonamento *just-in-time*. Esquemas com sistema de previsão utilizam a informação dinâmica em conjunto com os resultados baseados na predição. É similar ao sistema estático baseado em simulação, onde o escalonador necessita prever o desempenho da execução de uma tarefa em um grupo de recursos e gerar o escalonamento próximo do ótimo antes dela iniciar a sua execução. Dessa maneira, o escalonamento vai evoluindo durante a execução do fluxo.

No lugar de se fazer um planejamento durante o escalonamento, o sistema *just-in-time* só efetua a decisão no momento imediato antes da execução da tarefa. Planejamentos *a priori* em ambientes de computação em grade costumam produzir fluxos não ótimos por causa do ambiente dinâmico, onde a utilização e disponibilidade dos recursos varia durante o tempo. Além do mais, não existe uma maneira precisa para prever o tempo de execução de uma tarefa por causa da heterogeneidade do sistema em grade e por que melhores recursos podem sempre ser adicionados em qualquer momento. Contudo, trabalhos recentes mostram que sistemas de reserva de recursos [143] podem melhorar os resultados, aumentando a importância dos esquemas estáticos [144].

A última subdimensão do sequenciamento indica quais estratégias são utilizadas para se comparar as alternativas existentes em cada decisão de despacho. As estratégias orientadas ao desempenho global e de mercado são as mais comuns. Essas estratégias determinam como os recursos da malha serão utilizados para a execução das tarefas. No caso da orientação ao desempenho global, os recursos são utilizados na tentativa de se diminuir o tempo total de execução de em fluxo. Já na estratégia orientada a mercado, modelos de alocação mercadológicos são utilizados na tentativa de se respeitar o paradigma de consumidores e provedores de serviço. Recentemente, estratégias de confiança surgiram [145], onde recursos são selecionados de acordo com sua base histórica de confiabilidade, segurança, vulnerabilidade, etc.

Em ambientes de *grid*, a falha na execução dos fluxos de trabalho pode ocorrer por razões diversas, como falha na rede, condições de sobrecarga em recursos, a indisponibilidade de algum recurso. Como conseqüência, os sistemas de controle de execução do fluxo de processo devem identificar a falha e contornar a situação para permitir a finalização confiável da execução do fluxo. Segundo a tolerância a falhas, os fluxos podem ser divididos em: tolerância na falha da tarefa ou tolerância na falha do fluxo. No primeiro caso, os fluxos são controlados por sistemas que implementam esquemas de resolução de falhas do tipo *retry*, *checkpoint/restart*, recursos alternativos e réplicas. Já no caso de controle de falhas no fluxo inteiro, os siste-

mas implementam os seguintes esquemas de resolução de falhas: tarefa alternativa, redundância, exceção definida pelo usuário e fluxo reserva.

A última dimensão de sequenciamento é relacionada com a maneira em que os dados são movimentados pelo sistema de controle do fluxo. Essa movimentação pode ser automática ou dirigida pelo usuário. No caso de automática, podem se encontrar sistemas centralizados, onde o gerenciador do fluxo toma as decisões sobre a movimentação, mediados, onde os dados intermediários são moviementados por um sistema distribuído, ou ponto-a-ponto.

Devido ao aumento da complexidade dos fluxos, uma outra taxonomia surgiu especificamente para classificar a maneira como a verificação e validação da corretude do fluxo ocorrem. Nessa taxonomia, Chen e Yang [4] mostram seis elementos que compõem a classificação de verificação (estrutura, desempenho, recursos, autorização, custo e temporal) e três que compõem a classificação de validação (representação de processos de negócios ou científicos, representação do poder de expressão do sistema de gerenciamento de fluxo e métodos de validação).

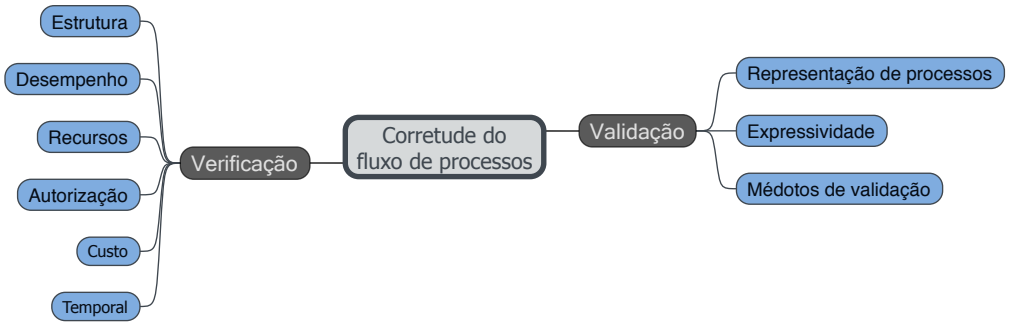


Figura 3.5: taxonomia de fluxos de processo proposta por Chen e Yang [4].

Na Tabela 3.2 apresentamos os mais importantes sistemas de controle de fluxo de processos e suas respectivas classificações.

3.4 Representação

Existem diversas linguagens para a representação de fluxos de processos. A maioria delas é derivada de sistemas completos de controle do fluxo de processos que incluem a possibilidade de descrever fluxos, programar tarefas e atividades e o sistema de controle propriamente dito. Devemos lembrar que a origem dessas linguagens remontam ao uso exclusivo nos ambientes comerciais, onde os últimos desenvolvimentos incluem as linguagens BPEL4WS [146] e WSFL [147]. Essas linguagens foram desenhadas para permitir a coordenação de atividades através de serviços *Web* especificados em XML [148, 149]. Embora as linguagens oriundas dos ambi-

Tabela 3.2: Mapeamento dos sistemas de controle de fluxo na taxonomia

	Desenho de fluxo			Sequenciamento					
Projeto	Estr.	Modelo	Comp.	Arquit.	Decisão	Planejamento	Estratégia	Toler. a Falhas	Mov. de dados
DAGMan	DAG	abs.	usuário	centr.	local	dinâmico	desempenho	tarefa e fluxo	usuário
Pegasus	DAG	abs.	usr e auto	centr.	loca e global	estático e dinâmico	desempenho	DAGMan	mediado
Triana	grafo	abs.	usuário	decent.	local	dinâmico	desempenho	GAT Mgr	p-a-p.
ICENI	grafo	abs.	usuário	centr.	global	dinâmico	desem. e mercado	ICENI midd	mediado
Taverna	DAG	abs. e conc.	usuário	centr.	local	dinâmico	desempenho	tarefa	centr.
GrADS	DAG	abs.	usuário	centr.	local e global	dinâmico	desempenho	tarefa	p-a-p.
Gridflow	DAG	abs.	usuário	hierár.	local	estático	desempenho	tarefa	p-a-p.
UNICORE	grafo	conc.	usuário	centr.	usuário	estático	usuário	UNICORE	mediado
Askalon	grafo	abs.	usuário	decent.	global	dinâmico	desem. e mercado	centr. e usuário	
Karajan	grafo	abs.	usuário	centr.	usuário	usuário	usuário	tarefa e fluxo	usuário
GRAND	DAG	abs.	usuário	hierár.	local	dinâmico	desempenho	tarefa	centr.
Kepler	grafo	abs. e conc.	usuário	centr.	usuário	usuário	usuário	tarefa e fluxo	centr. medi. e p-a-p.
Gridbus	DAG	abs. e conc.	usuário	hierár.	local	estático e dinâmico	mercado	tarefa	centr. e p-a-p.
Gridway	DAG	abs	usuário	centr.	local	dinâmico	desempenho	tarefa e fluxo	usuário

Legenda: abs.: abstrato; centr.: centralizado; conc.: concreto; decent.: decentralizado; p-a-p: ponto-a-ponto

entes comerciais sejam excelentes na descrição e atuação dos fluxos, em particular, elas não são eficazes na representação de fluxos abstratos.

Especificamente para os fluxos de processos científicos, os usuários costumam delegar ao sistema de controle do fluxo as tarefas relacionadas à orquestração e iteração com os recursos da malha computacional, dessa maneira sendo imperativo a possibilidade de especificação abstrata [150]. Além disso, pela própria dinâmica da *e-Infrastructure*, a transformação de abstrato em concreto é normalmente relegada à última instância.

Dentre os mais famosos sistemas de controle de fluxo de processos científicos que apresentam sistemas de representação significativos, podemos citar:

Triana [151] se utiliza de uma linguagem gráfica para a representação de fluxos que inclui elementos de controle de dados e de fluxo. O sistema Triana, com outros serviços *Web* em XML, tem uma base de esquemas de representação de tipos de dados (XSD, [152]), que torna o sistema poderoso mas complexo;

Taverna [119], parte do projeto *myGrid*[119], é um sistema de controle de fluxo de processos científicos focado na área de ciências da vida e saúde. As tarefas são implementadas como classes Java. Além disso, Taverna se utiliza de uma linguagem tipo XML para a especificação do fluxo com um forte sistema de tipagem de dados, que são restritos aos tipos MIME[153] relacionados com ontologias de bioinformática;

VisTrails [154] é um sistema que registra *provenance* para os dados e para os fluxos. Isto permite que o fluxo seja considerado um caderno de anotações do experimento que tem sua evolução junto com o experimento científico. Esse sistema se utiliza de uma especificação gráfica e visual do fluxo e tem o seu foco na execução interativa de cada tarefa;

Kepler [23] se utiliza do ambiente de modelagem do *Ptolemy*[23] e adiciona a habilidade de se testar o fluxo, simular a execução distribuída através de serviços *Web* ou Globus. Apresenta também um sistema para extensão através de atores;

xWFL [123] é uma linguagem com uma estrutura relativamente simples que suporta parametrização através de nome de arquivos, faixas de números e constantes;

GSFL [155] é uma linguagem desenvolvida para o Globus se baseia em diversas soluções encontradas na WSFL [147];

AGWL [156] foi desenhada especificamente para fluxos científicos, balanceando a representação abstrata com grande parte da informação necessária para a otimização durante a transformação do fluxo em concreto;

WOOL [129] compartilha diversas características da AGWL, contudo ela apresenta construtores simplificados para a expressão de paralelismo e sequenciamento. Além disso, apresenta tipos abstratos de dados que facilitam a especificação de dados intermediários.

3.5 Mapeamento de fluxos na *e-Infrastructure*

A necessidade do uso de fluxos abstratos está relacionada com a portabilidade dos mesmos [135]. Além disso, permite compartilhar o fluxo com os diversos membros de uma VO, permitindo a transferência do conhecimento dentro da comunidade. Contudo, para sua execução existe a necessidade de se mapear os fluxos na *e-Infrastructure*.

O mapeamento de fluxos na *e-Infrastructure* é a tradução de um fluxo abstrato em um concreto. Essa tradução pode ser efetuada de diversas maneiras e como apresentado na taxonomia, ela pode ser estática ou dinâmica. Numa tradução estática, os modelos concretos são gerados antes da execução do fluxo e com base na informação existente sobre a *e-Infrastructure*. Por outro lado, a tradução dinâmica utiliza tanto a informação estática quanto a informação corrente sobre o estado dos recursos da infraestrutura, para efetuar decisões durante o tempo de execução de um fluxo.

Esquemas estáticos, como o apresentado na Figura 3.6, também são conhecidos como planejamento *a priori* e podem ser baseados em orientações do usuário ou simulação [3]. No primeiro caso, os usuários emulam o processo de sequenciamento e tomam as decisões de mapeamento dos recursos de acordo com o conhecimento, preferência ou critério de desempenho. Quando a simulação é utilizada, o melhor sequenciamento é obtido através da simulação da execução em um conjunto de recursos antes da execução do fluxo. A simulação pode se utilizar da informação estática ou do resultado de uma estimativa do desempenho.

Esquemas dinâmicos são baseados em sistemas de predição ou sequenciamento *just-in-time*. Sistemas preditivos utilizam a informação corrente do estado da malha computacional e alguns resultados baseados em predição. Os sistemas *just-in-time* só efetuam a decisão de sequenciamento no momento de despacho da tarefa corrente.

Além dos recursos computacionais, os sistemas de mapeamento levam em consideração a disponibilidade dos dados de entrada e intermediários durante a concretização de um fluxo. A Figura 3.7 apresenta um fluxo abstrato e o concreto correspondente com todas as transferências de dados necessárias para a execução que pode ser obtida com heurísticas [136] ou técnicas de inteligência artificial [103] para produzir o fluxo concreto.

Mapeamentos estáticos apresentam um baixo desempenho por causa da natu-

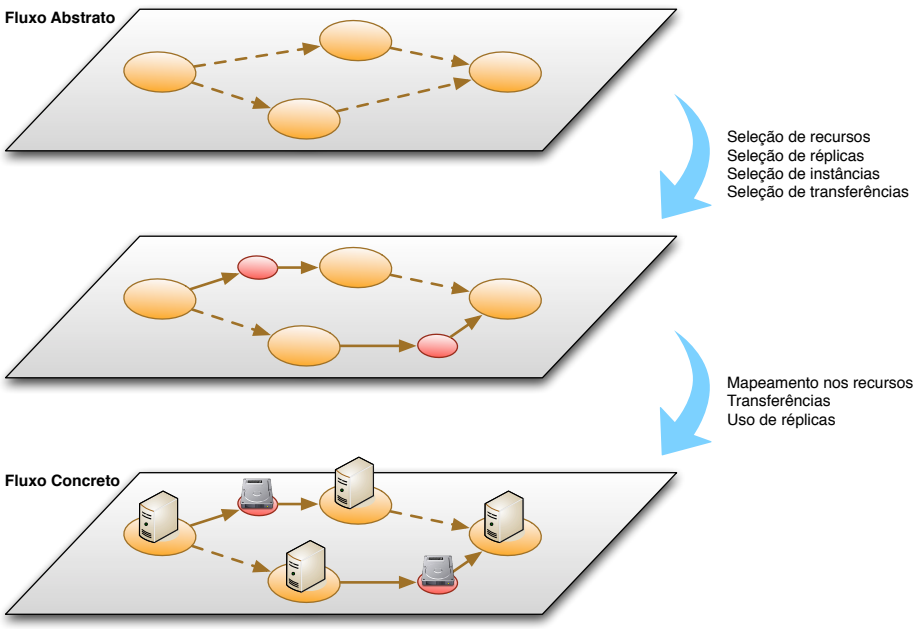


Figura 3.6: Transformação de um fluxo abstrato em um concreto.

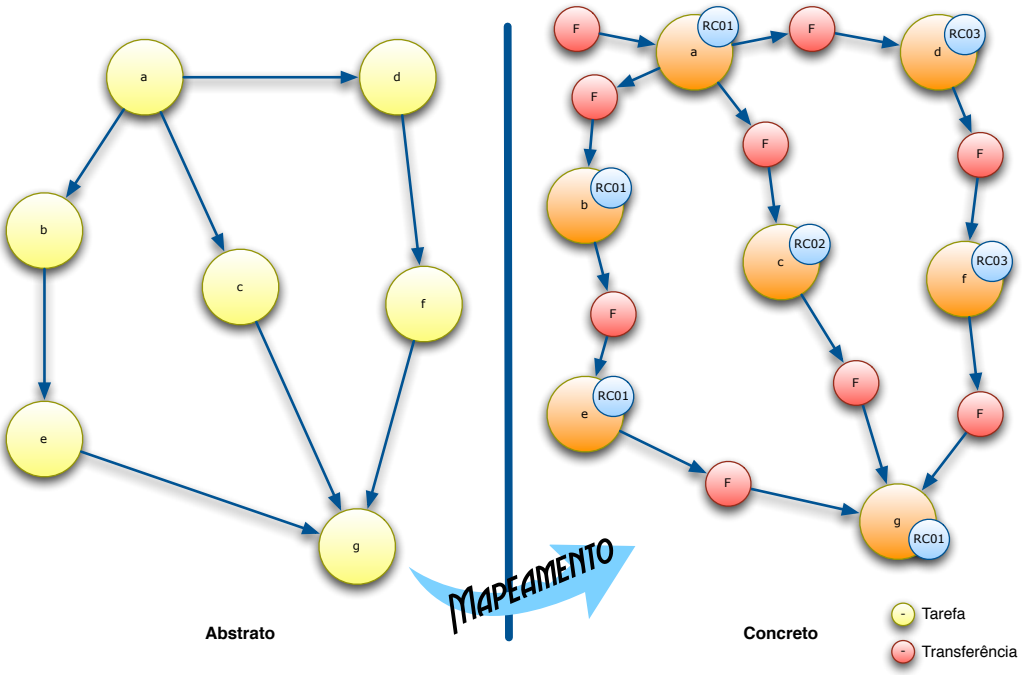


Figura 3.7: Identificação dos produtos intermediários em um fluxo: transferências.

reza dinâmica do ambiente e do estado dos recursos [3], pois recursos podem ficar indisponíveis ou recursos melhores para uma determinada tarefa podem se tornar disponíveis. Além do mais, a reserva antecipada promete se apresentar como uma tecnologia eficiente [157], apesar de necessitar de inúmeras modificações no *middleware* corrente.

3.6 Controle

O controle do despacho das tarefas pode acontecer em três momentos distintos: na hora de delegação do fluxo; na hora do sequenciamento da tarefa ou no momento de liberação do recurso [158]. Além disso, a arquitetura do controle do sequenciamento na infraestrutura é importante para a escalabilidade, autonomia, qualidade e desempenho do sistema [82]. A Figura 3.8 apresenta os três tipos de arquitetura de sequenciamento que existem: centralizado, hierárquico e decentralizado.

Num ambiente centralizado, um escalonador central efetua as decisões de sequenciamento para todas as tarefas do fluxo de processos. O escalonador tem informação completa sobre o fluxo e coleta a informação disponível sobre os recursos. Esse fato leva a crer que o escalonador no ambiente centralizado produz sequenciamentos eficientes pois tem toda a informação necessária [82]. Contudo, esse ambiente não é escalável nas dimensões: número de tarefas, tipos e número de recursos da malha computacional. Além disso, outra desvantagem é que no caso de falha do escalonador central, o sistema inteiro será paralisado.

Diferente do ambiente centralizado, os ambientes hierárquico e decentralizado permitem que tarefas sejam escalonadas por múltiplos escalonadores. Dessa maneira, cada escalonador é responsável por uma partição de um fluxo. Contudo, com a fragmentação da informação, comparado com os sistemas centralizados, o ambiente hierárquico e o descentralizado tendem a obter um desempenho sub-ótimo.

No caso do escalonador hierárquico existe um escalonador central e sub-escalonadores. O central é responsável pelo controle da execução e o ordenamento da distribuição dos sub-fluxos entre os sub-escalonadores. A maior vantagem no uso do sistema hierárquico é que políticas diferentes de escalonamento podem ser usadas em cada componente do sistema.

Além disso, problemas relacionados aos conflitos se tornam mais evidentes [83]. Um exemplo de conflito é quando tarefas de partições diferentes do fluxo são escalonadas por seus escalonadores no mesmo recurso. Um outro fato importante é que em uma *e-Infrastructure* do tipo do EGEE ou do OSG, é inviável o uso de sistemas centralizados e normalmente diversos fluxos estão sendo executados simultaneamente na malha. Ou seja, mesmo que esquemas centrais sejam implementados, a coexistência da execução de diversos fluxos na mesma infraestrutura gera competição entre

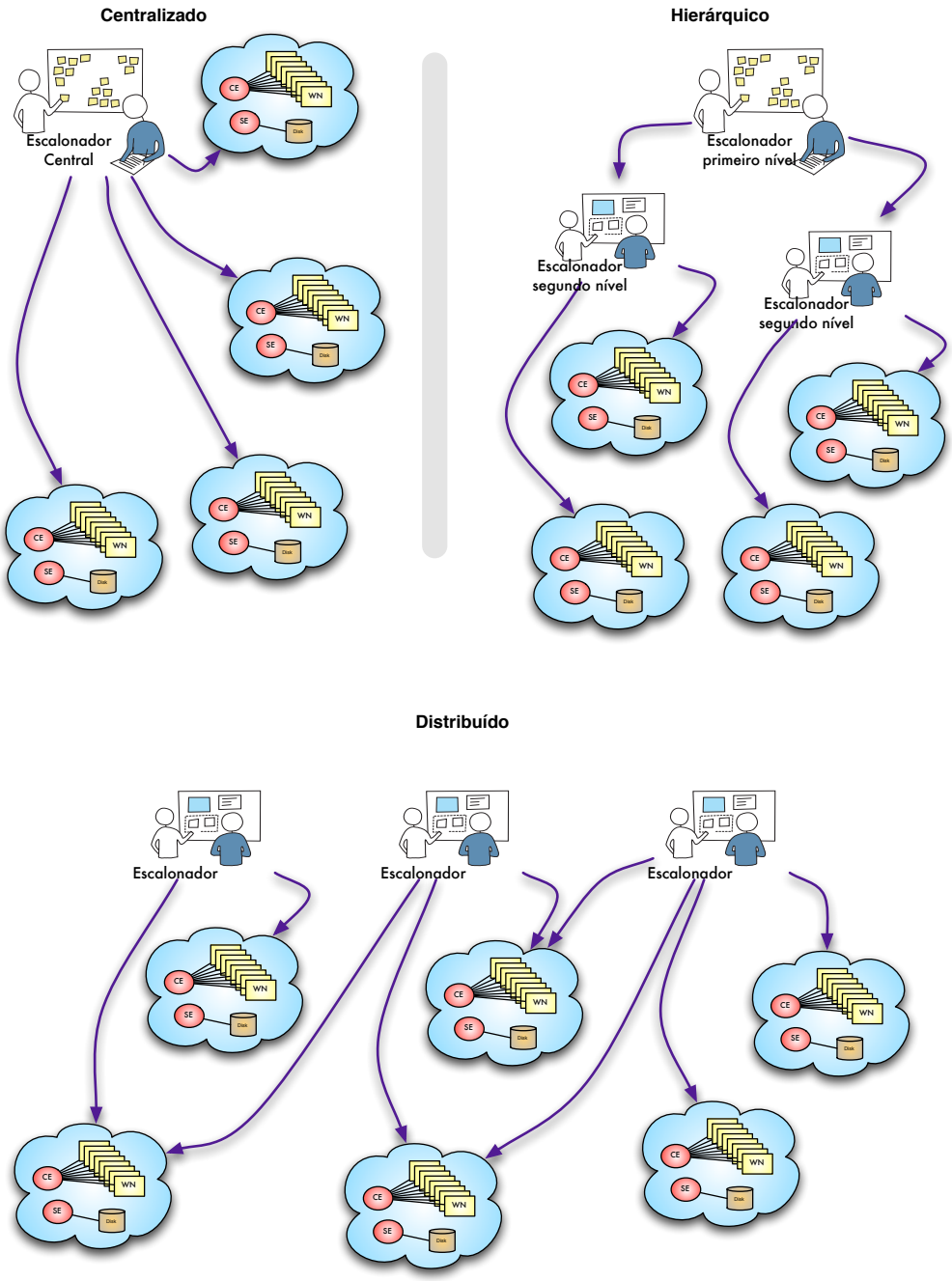


Figura 3.8: Arquiteturas de sequenciamento.

os escalonadores pelos recursos, não importando qual ambiente de controle esteja sendo usado.

Capítulo 4

Controle distribuído do fluxo de processos

4.1 Premissas

Diversos sistemas surgiram na literatura para efetuar o gerenciamento e controle do fluxo de processos em malhas computacionais [23, 78, 95, 103, 108, 112, 116–125, 127, 130, 135, 155, 159–167] contudo, com a evolução e o aumento da adoção das malhas computacionais, novas dificuldades foram acrescentadas, criando a necessidade de novas soluções para o problema do gerenciamento de fluxo de processos.

4.1.1 Imperfeição na informação utilizada

Como um primeiro ponto a ser abordado, gostaríamos de analisar a maneira como os sistemas atuais efetuam as decisões de escalonamento. Na Seção 3.3 apresentamos a taxonomia proposta por Yu e Buyya [2, 3], onde podemos observar que todos os sistemas efetuam o planejamento dinâmico, tentando retardar o máximo possível a decisão do despacho pois planejamentos *a priori* em ambientes de computação em grade costumam produzir fluxos sub-ótimos por causa do ambiente dinâmico, onde a utilização e disponibilidade dos recursos varia durante o tempo. Além disso, não importando se o processo de decisão é *global* ou *local*, os sistemas se valem de previsões ou simulações que levam em consideração que o recurso alocável indivisível seria o processador, *job slot* ou computador. Na realidade, todas as *e-Infrastructures* de produção, ou malhas de serviço, apresentam uma arquitetura que por construção não pode oferecer essa granularidade. Em malhas computacionais como EGEE, EELA, OSG, Naregi, etc, os centros de recursos se apresentam como um único ponto de interfaceamento (*gateway, gatekeeper, peer, computing element*) com os sistemas de escalonamento.

Diversos motivos levam a se ter um único ponto de interfaceamento para cada

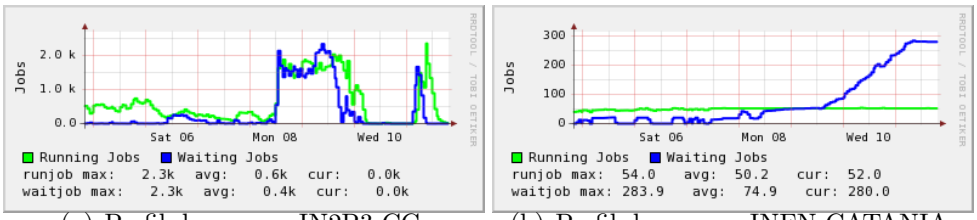
centro de recursos, mas podemos citar como razão principal o fato de facilitar os esquemas de segurança já que a infraestrutura é multi-institucional. Como é observado na especificação do Globus apresentada na Seção 2.2.1, temos que:

> O GRAM [67] é responsável pela execução remota de trabalhos e relatório de estados. Com o uso do GRAM, o usuário demanda a submissão de um trabalho ao processo *gatekeeper* em um computador remoto. O *gatekeeper* verifica se o cliente é autorizado (i.e., o certificado do cliente é mapeado no arquivo de mapeamento local em uma conta existente no computador remoto). Assim que o cliente ganha o acesso, o *gatekeeper* inicia um gerente de tarefas que inicia o trabalho e começa a sua monitorização. Os gerentes de tarefa são criados dependendo do sistema local de gerenciamento de filas (LRMS - *Local Resource Management System*) que pode ser o PBS (*Portable Batch System*), o LSF (*Load Sharing Facility*), etc.

Na realidade, temos que o *gatekeeper* efetua uma interface entre a malha computacional e o sistema local de gerenciamento de filas ou recursos. Com essa estrutura, o recurso alocável indivisível é o LRMS e não um processador, *job slot* ou computador. Esse fato implica que as simulações ou previsões utilizadas no sistema de controle de fluxo de processos deveriam levar em consideração que, além do efeito relacionado com a execução da tarefa em um recurso computacional, os tempos de espera internos ao centro de recursos são significativos. Na Figura 4.1 podemos observar a evidência desse fato. Nela podemos observar em azul a variação das tarefas em espera nas filas dos RCs IN2P3CC (França) e INFN-CATANIA (Itália).

Podemos observar na Figura 4.1(a) que existe sempre uma defasagem entre a demanda existente no RC e a resposta do sistema. Essa defasagem não pode ser inferida com base simplesmente da descrição dos recursos, mas depende também do valor da carga instantânea do RC. Já na Figura 4.1(b) fica evidente que com a sobrecarga em um RC, o tempo de atravessamento do *job* pode, de uma hora para a outra, mudar completamente com a pressão de demanda existente. Christodoulopoulos et al. [168] analisam o atraso imposto pelo escalonamento atual da malha do EGEE e afirmam que toda a carga existente nessa malha poderia ser executada em um "super cluster" equivalente a 34% do tamanho médio do EGEE. Esse número seria uma primeira aproximação do efeito ocasionado pelos atrasos apresentados na malha.

Alguns sistemas [3, 79, 169] tentam armazenar o histórico das execuções anteriores de um centro de recursos e utilizar algoritmos de vitimação com *exponential backoff* [170, 171] para compensar esses fatores. Essa solução é eficiente para banir RCs que estão congestionados mas apresentam o problema de demorar para per-



(a) Perfil de carga - IN2P3-CC. (b) Perfil de carga - INFN-CATANIA.

Figura 4.1: Perfil de carga. A linha verde contabiliza as tarefas em execução e a azul as tarefas em espera.

ceber quando o RC ficou descongestionado. Uma outra solução seria a tentativa de se expor o recurso diretamente ao eslcalonador como feito pelo Askalon [172]. Contudo, essa solução implica na instalação de um novo serviço a ser suportado por cada RC.

Fora das malhas de serviço e produção, existem nos sistemas oportunistas como OurGrid[63], BOINC[162], etc, uma diferença fundamental, pois o recurso computacional se apresenta ao escalonador através dos pares.

Um outro problema relacionado com a qualidade da informação utilizada é relacionado com a taxa de atualização da informação sobre o estado da malha. Por exemplo, como apresentado na Seção 2.2.1 o Globus e o gLite utilizam servidores de informação central baseados em LDAP, extremamente rápidos para a consulta, mas de atualização mais demorada[173]. Devido ao enorme número de entradas (mais de 100K entradas no caso do EGEE) e ao esquema de implementação de atualização, esses servidores efetuam ciclos de atualização em intervalos de 5 ou mais minutos. Dessa maneira, a imagem do estado da infraestrutura disponível para um escalonador de fluxo de processos está freqüentemente desatualizada. Esse fato se potencializa quando diversas instâncias de escalonadores estão orquestrando a execução de fluxo de processos de aplicações diferentes. Nesse caso, é plausível que instâncias distintas compitam pelos mesmos recursos sem saber.

4.1.2 Falhas na *e-Infrastructure*

Na Seção 2.4 foi apresentado o trabalho de Groep [85] que aponta que as falhas na execução global de tarefas no EGEE fica em torno de 25% e 33%, que é corroborado pelas informações obtidas no sistema de monitoramento oficial do EGEE (Figura 4.2). Na infraestrutura EELA, encontramos 28,5% [174] como percentagem de falhas, o que é coerente pois as duas infraestruturas compartilham um número considerável de RCs[1]. Em outras infraestruturas, podemos citar o estudo de Tierney et al. [175]. Nele é mostrado que a taxa de erros no OSG caiu de 30% para 15% num período de 4 anos. Esses valores se apresentam como um ótimo resultado se

[1] a maioria dos sítios europeus pertencem às duas infraestruturas.

levarmos em consideração o trabalho de Thain et al. [176], no qual é observado que aproximadamente 10% dos computadores estão inacessíveis a um dado momento do tempo em um estudo de 5 anos do funcionamento do Condor no globo.

Figura 4.2: Taxa de sucesso total do EGEE e EELA-2.

Na Seção 2.1.1 mostramos que a *e-Infrastructure* EGEE tem o seu monitoramento ativo e distribuído em ROCs numa tentativa pro-ativa de se reduzir as falhas através do aumento da disponibilidade dos RCs, mas como visto na Figura 4.2, os valores continuam em torno de 20%.

Mesmo com esse esforço, a origem das falhas continua incerta, o trabalho de Tierney et al. [175] culpa diretamente falhas em discos e Chawla et al. [177] observaram que os fatores responsáveis pelas falhas eram uma combinação que incluía: erro do usuário, credenciais expiradas, erros de *middleware* e falhas de discos, tudo isso em uma infraestrutura que apresentou uma taxa de falhas global de 30%.

Nos 28,5% encontrados na infraestrutura do EELA, 90% das falhas estavam relacionadas de alguma maneira com o processo de estabelecimento dos contextos de segurança SSL do GSI (Seção 2.2.1). Esse fato vem a corroborar os números encontrados por Groep [85] que somente entre 5% e 8% das tarefas falham depois de serem entregues ao LRMS.

Um outro trabalho muito interessante foi feito por Iosup et al. [178], nele se tenta levantar o comportamento da malha independente da natureza da falha, mas olhando para o sistema como um todo. Nele foi mostrado que a malha apresenta uma MTBF de 12 minutos e de 5 horas para um RC.

De uma maneira geral, trabalhando com taxas globais de falha de aproximadamente 20%, os números indicam que uma tarefa despachada por um escalonador tem em torno de 80% de chance de rodar, o que torna imperativo um sistema de gerenciamento de falhas que seja capaz de lidar com essa taxa de erros na malha.

4.1.3 Outras considerações

Existem outros fatores secundários que afetam o funcionamento do sistema de controle de fluxo de processo. Dentre eles podemos citar:

- *overbook* de recuros - normalmente os RCs suportam mais de uma Organização Virtual fazendo *overbook* dos recursos. Isso aumenta a probabilidade de diversas instâncias de escalonadores (em diferentes VOs) congestionarem os RCs;
- o interfaceamento entre grades de serviço e oportunistas - com a recente ligação entre os dois tipos de malhas computacionais [179], a tendência é que as taxas de falhas aumentem por causa da natureza dos recursos nas malhas oportunistas;

4.2 Solução apresentada

Na tentativa de se endereçar os fatos apresentados na Seção 4.1, desenvolvemos um sistema de gerenciamento de fluxo de processos chamado de *Workflow Agile Scheduler* (WAS) e ele se classifica de acordo com a taxonomia apresentada por Yu e Buyya [2, 3] da seguinte maneira:

1. **arquitetura:** apresenta uma arquitetura distribuída e permite que vários fluxos coexistam na mesma infraestrutura;
2. **tomada de decisão:** executa a tomada de decisão local, contudo a informação sobre o estado da infraestrutura é compartilhada com cada escalonador de fluxo;
3. **planejamento:** efetua o planejamento de forma dinâmica, levando em consideração o estado atual da infraestrutura;
4. **estratégia:** implementa uma estratégia focada no desempenho;
5. **movimentação de dados:** pratica a movimentação de dados semi-automática com base em partições do fluxo fornecidas pelo usuário;
6. **estrutura:** utiliza um fluxo baseado em um DAG;
7. **tolerância à falhas:** implementa um esquema de tolerância a falhas para a tarefa através de re-tentativa.

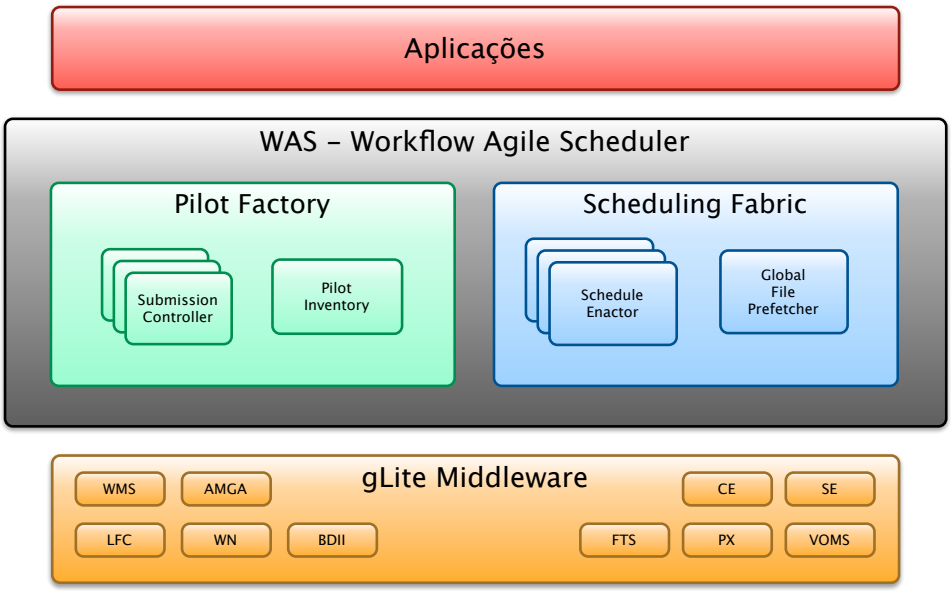


Figura 4.3: WAS - *Workflow Agile Scheduler*.

Na Figura 4.3 apresentamos os componentes centrais do WAS. O primeiro componente é a *Pilot Factory* (PF), que é mostrado na Figura 4.4. Ele é responsável pela principal característica do WAS, que usa o *Pilot Job* (PJ) para implementar uma infraestrutura virtual, onde é possível ter uma menor distorção na informação sobre o estado da infraestrutura e um maior controle direto sobre os recursos.

O PJ vem sendo utilizado em diversas VOs para prover mecanismos que aumentam o controle sobre a gestão de prioridades dentro das VOs de maneira transparente para os RCs [54, 167]. Ele é submetido através dos mecanismos tradicionais e quando ele efetivamente inicia a execução, ele entra em contato com um "gerente" para baixar a aplicação real que será executada. Dessa maneira, os administradores de uma VO podem aumentar a prioridade de um usuário em detrimento de outro. Como a tarefa é submetida em nome do usuário que executa o "gerente" (administrador da VO), um mecanismo específico (gLexec - [180]) é usado para que a tarefa efetivamente rode em nome do usuário que seria o dono da tarefa.

No caso do WAS, os PJs são submetidos por diversas instâncias do *Submission Controller* (SC) que têm por objetivo principal garantir que um determinado número mínimo de PJs estejam efetivamente rodando na infraestrutura. Dessa maneira o usuário pode especificar a largura mínima da máquina, que é a quantidade mínima de PJs ativos simultaneamente na *e-Infrastructure* para a execução do seu fluxo. O sistema se encarrega de manter a submissão através do WMS para manter a largura dessa máquina. No caso do gLite, são submetidas tarefas paramétricas, onde com um único pedido ao WMS podemos gerar um conjunto de PJs.

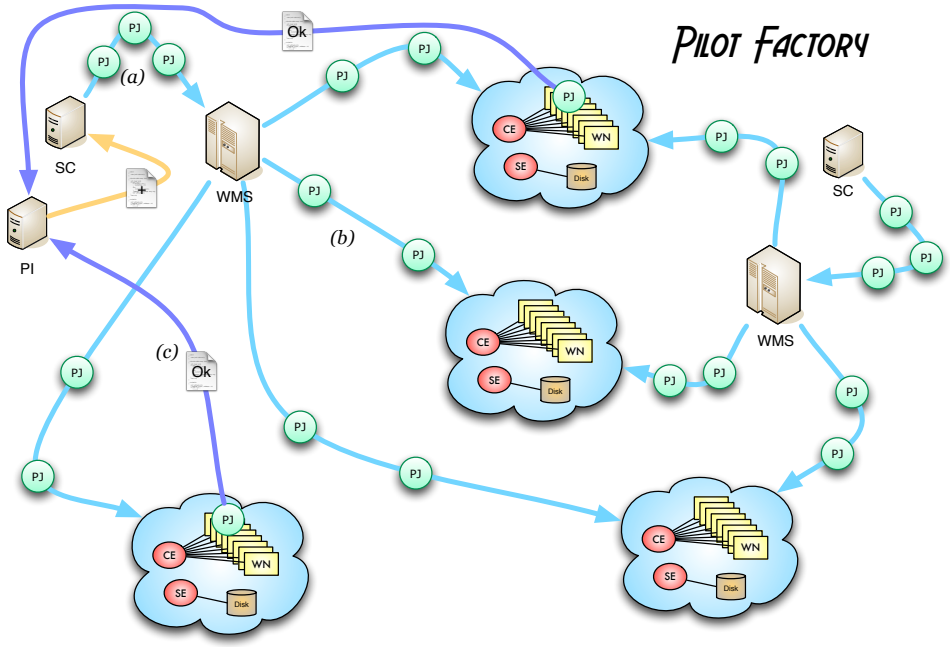


Figura 4.4: Sistema de controle de submissão *Pilot Factory*.

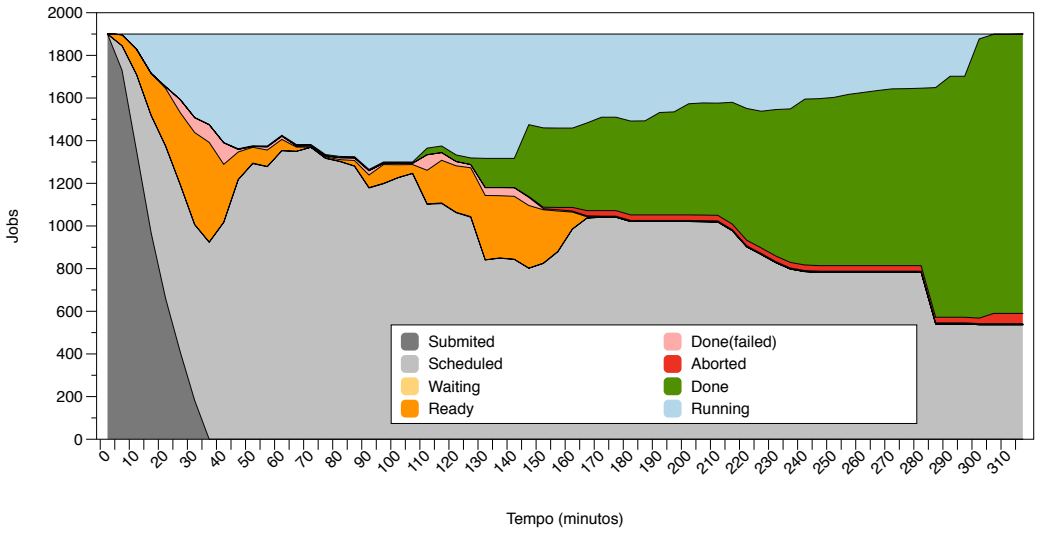


Figura 4.5: Execução de uma instância do SC da *Pilot Factory*.

Na Figura 4.5 podemos observar uma execução de uma instância da PF. No caso da execução dessa instância de exemplo, uma batelada de 1900 tarefas foram submetidas no início do ciclo (área cinza escura na figura). Essas tarefas são despachadas pelo WMS para diversos CEs na infraestrutura e entram nas filas do LRMS em cada RC (área cinza claro). Os PJs passam por diversos estados (Tabela 4.1) e quando começam efetivamente a rodar, entram em contato com o segundo módulo da PF que é o Pilot Inventory (PI). O PI é responsável por contabilizar quais PJ estão rodando na infraestrutura e determinar quando o SC deve submeter uma nova batelada de PJ no caso em que o número de PJs rodando na infraestrutura desceu abaixo do limite predeterminado.

Tabela 4.1: Estados de um *job* no gLite.

Estado	Significado
SUBMITTED	A tarefa foi submetida, mas ainda não foi processada internamente pelo WMS.
WAITING	A tarefa já foi processada internamente no WMS, mais ainda não se escolheu o CE para o despacho.
READY	A tarefa já tem um CE mapeado, contanto ainda não foi transferida para ele.
SCHEDULED	A tarefa se encontra na fila do LRMS do CE
RUNNING	A tarefa se encontra executando em um WN.
DONE	A tarefa terminou
ABORTED	A tarefa foi abortada por um erro de *middleware*
CANCELLED	A tarefa foi cancelada pelo usuário
CLEARED	Os resultados da tarefa já foram removidos do WMS.

Além de manter o SC informado sobre o estado da infraestrutura virtual, o PI é responsável também por informar a cada *Schedule Enactor* (SECT) quais PJ estão rodando. O SECT, um dos componentes do *Scheduling Fabric* (Figura 4.6) é responsável pelo escalonamento do fluxo propriamente dito. Ele escolhe entre os PJs, que estão rodando, os que efetivamente vão receber uma nova tarefa do fluxo de processos para ser executada. Quando um PJ começa efetivamente a executar a tarefa do fluxo, dizemos que ele se encontra no estado *ativo*. Dessa maneira, o SECT vai evoluindo a execução do fluxo de tarefas, efetuando um algoritmo de reversão

de arestas [181] para avaliar quais são as atividades que devem ser despachadas. Entre as diversas instâncias do SECT também é utilizado um algoritmo de reversão [181–184], contudo com o objetivo de se implementar a exclusão mútua para impedir que dois ou mais SECTs tentem ativar PJs no mesmo RC.

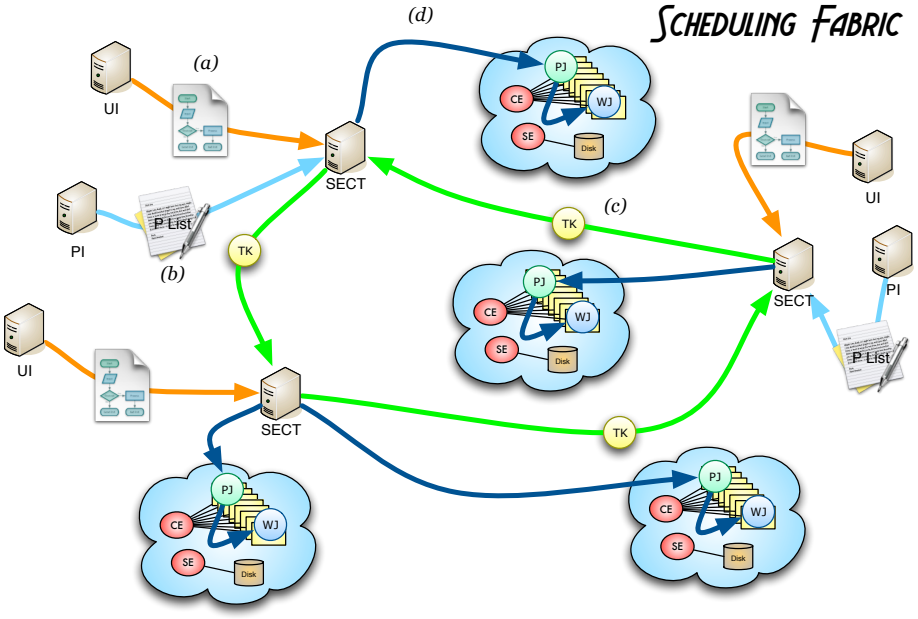


Figura 4.6: Sistema de controle de escalonamento de fluxo de processo *Scheduling Fabric*.

O último módulo do WAS é o *Global File Prefetcher* (GFP). Ele se baseia em uma idéia comum na micro-arquitetura de processadores [185, 186] para efetuar a movimentação semi-automática dos arquivos na infraestrutura virtual. Utilizando informações passadas pelo SECT, o GFP utiliza o serviço gLite FTS para criar réplicas de arquivos na infraestrutura, aumentando o número de RCs elegíveis para se despachar uma tarefa.

O WAS foi desenvolvido nas linguagens *C++*, *Python* e *Bash* e se utilizou as seguintes ferramentas: *Distributed Information Management* (DIM) [187], *Ganga* [188] e *AMGA* [189].

O DIM é um sistema desenhado para prover uma camada de comunicação para todos os processos envolvidos no sistema de aquisição de dados do detetor DELPHI [190] do acelerador LEP, que é predecessor do LHC. Ele provê um sistema de comunicação assíncrono e permite conexões de *um para vários*. No WAS, ele é usado na comunicação entre todos os módulos, exceto em parte da comunicação entre os PJs e SECT. Nesse caso especifico, o SECT e os PJs se comunicam também através do AMGA para o registro do estado do fluxo de processos. O *Ganga* é usado para

o gerenciamento e re-submissão dos módulos do WAS que podem também rodar dentro da própria malha.

Cada tarefa do fluxo é representada por um arquivo no AMGA e o meta-dado de cada um representa o estado corrente de execução da respectiva tarefa. Dessa maneira, o SECT pode levantar quais são as próximas tarefas a serem despachadas e efetivamente conferir quando uma tarefa terminou com estado de sucesso.

4.2.1 Representação de fluxo de processos

O WAS espera como entrada um fluxo de processos abstrato na forma de um DAG que é utilizado para representar as dependências temporais como apresentado na Seção 3.3. Além dessas dependências, o usuário pode fornecer informação sobre as dependências de dados através de Conjuntos de Atividades Relacionadas (CAR). Cada CAR representa as atividades (nós no DAG) que dependem do mesmo arquivo de entrada. Essa informação é utilizada pelo SECT para se particionar o DAG no momento da criação do fluxo concreto.

4.2.2 Particionamento do Fluxo

O WAS utiliza o particionamento do fluxo para aproveitar a localidade dos dados, diminuindo a necessidade de transferências de dados entre RCs. Para tal, o SECT utiliza o conceito de afinidades entre partições do DAG e RCs de acordo com a disposição dos dados na malha.

Na literatura podemos encontrar diversas maneiras de se efetuar uma partição do DAG [191, 192] e no caso específico do WAS, o ideal seria o esquema proposto por França et al.[193] por causa da dinâmica de reversão de arestas empregado. Contudo, o problema de particionamento é NP-Completo [194], o que nos levou ao uso de uma heurística. Além disso, uma característica específica do WAS é que a granularidade do particionamento é o RC e não o nó que executa a tarefa (WN) como nos demais SGFPs. Esse fato se deve pois a tarefa é criada em um WN em um contexto temporário, que perderá validade após a execução da tarefa. Dessa maneira, todos os arquivos que não foram transferidos para um lugar seguro irão desaparecer. No caso da implementação do WAS, os arquivos são movimentados para o SE de um RC e depois copiados para o ambiente da tarefa.

A heurística usada para o particionamento é apresentada na Figura 4.7. Cada CAR recebe uma cor, que é utilizada para marcar cada nó do DAG que utiliza o arquivo correspondente ao CAR. Quando mais de uma cor é atribuída ao mesmo nó, é utilizada a seguinte regra de quebra de simetria:

- o nó fica com a cor do CAR que representa o arquivo cujo o tamanho apresente a maior ordem de grandeza;

- se o empate persistir, é escolhida a cor do CAR que apresenta a maior cardinalidade;
- persistindo o empate, a cor é escolhida de maneira aleatória.

No momento do escalonamento, o SECT vai tentar ativar as tarefas do fluxo que pertençam a uma mesma partição nos RCs que apresentam os arquivos correspondentes aos CARs. Além disso, a cada ciclo de escalonamento, o SECT compara a predição do tempo de despacho as atividades que deverão ser despachadas nós próximos ciclos com o tempo para se transferir[2] os arquivos correspondentes para RCs que não os possuem em seus SEs. No caso do tempo de transferência ser menor que a predição do despacho, o SECT envia o pedido para o *Global File Prefetcher* (GFP) para promover a criação da réplica, na tentativa de se aumentar o número de RCs disponíveis para os futuros despachos.

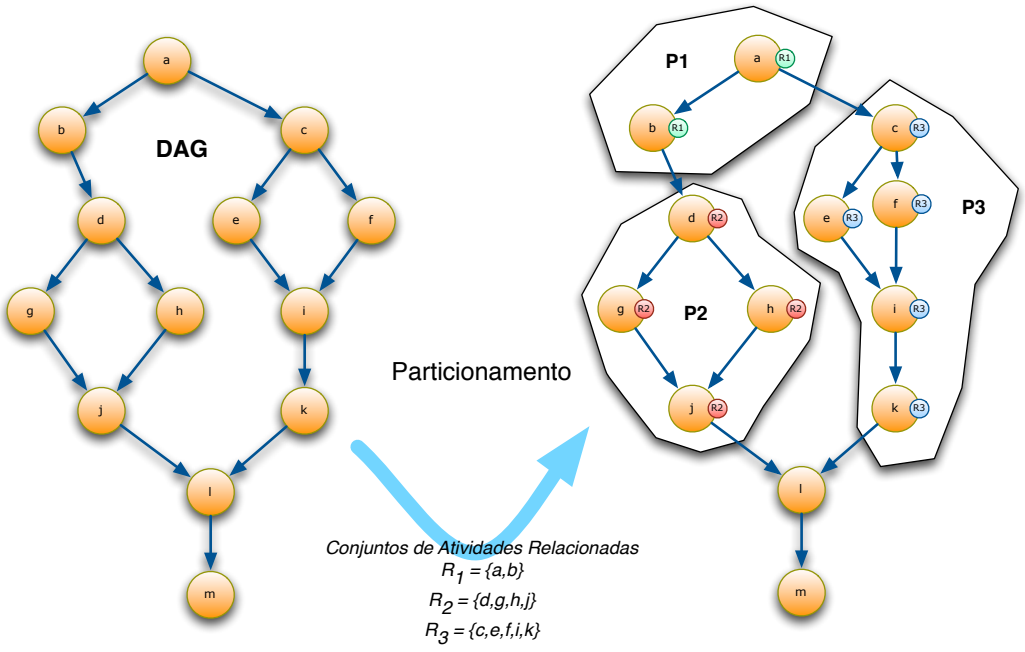


Figura 4.7: Particionamento segundo a heurística apresentada.

4.2.3 Gerenciamento de Falhas

O SC monitora a execução dos PJs através dos inventários fornecidos pelo PI. No protocolo usado, quando um PJ ativo está prestes a encerrar suas atividades, ele deve sinalizar na representação do fluxo de processos que se encontra no AMGA que já terminou todas as atividades e transferiu todos os arquivos de saída. Depois

[2] a capacidade de cada enlace da infraestrutura não se encontra disponível no sistema de informação da malha e foi utilizado um levantamento interno do EELA-2.

dessa atualização, o PJ simplesmente termina sua execução, o que via DIM, vai gerar um sinal no PI. No caso de uma modificação no inventário de PJ ativos sem uma correspondente mudança no estado do fluxo mapeado no AMGA, o SC determina que um PJ abortou sem completar sua tarefa. Nesse caso, o SC despacha a mesma tarefa para um outro PJ como se a execução nunca tivesse ocorrido.

No caso de uma falha em um dos SCs, o PI percebe, via DIM, que um dos SCs abortou. Nesse momento, ele inicia uma nova instância de SC. O mesmo acontece no caso em que um SECT aborte. Nesse caso, o PI dispara um novo SECT configurado para terminar o fluxo que ficou paralisado. No caso de falha do GFP, os SECTs atrelados são responsáveis pelo disparo de uma nova instância.

Se mais de uma instância é iniciada para a mesma função, o sistema de comunicação DIM garante que somente uma das instâncias vai conseguir registrar o seu nome de serviço, dessa maneira, garantindo a corretude.

O único módulo que é iniciado por uma intervenção externa é o PI, que tem o seu funcionamento verificado pelo sistema de monitoramento, que no caso de falha, gera uma nova instância do PI. Pelo maneira que funciona o DIM, todos os PJs ativos vão se re-declarar ao novo PI e a execução dos fluxos é retomada no ponto em que se ocorreu o problema.

4.3 Experimentos

Para a avaliação do funcionamento do sistema de gerenciamento de fluxo de processos WAS, foram feitos diversos experimentos, inclusive uma comparação com o SGFP *Gridway* [126, 127]. O *Gridway* foi escolhido porque: (i) ele é um dos componentes do Globus, que é considerado um *middleware* de referência e (ii) ele é o SGFP de maior aceitação na comunidade formada pelos projetos EELA e EELA-2 [195] que utilizam gLite como *middleware*.

Nos experimentos foram usadas duas configurações da infraestrutura. A primeira, chamada de *configuração completa*, é constituída de todos os 25 RCs do projeto EELA-2, mostrados na Figura 2.6. Essa configuração é formada de 8016 *job slots* e 35,6 TB de área de armazenamento. Os experimentos com essa configuração só foram considerados válidos se a carga (excetuando a gerada pelo próprio experimento) em toda a duração fosse menor que 30% (menos que 2404 *job slots* ocupados). Essa precaução foi tomada para se evitar que outra carga na infraestrutura contaminasse de maneira significativa os resultados do experimento.

A segunda configuração, chamada de *Pilot Testbed* (Figura 4.8) foi usada nos casos em que necessitávamos de maior controle sobre o experimento. Nessa configuração era possível ter o acesso a todas as máquinas da infraestrutura, inclusive dos diversos sistemas de monitoramento disponíveis no EELA-2 Operations Centre

(EOC). Além disso, um outro sistema de monitoramento multi-camada [196] foi instalado para o registro da *provenance* no Pilot Testbed. Essa configuração é composta de 574 *job slots* e 5 TBytes de disco e todos os RCs se encontravam dedicados no momento dos experimentos.

A aplicação usada nos experimentos foi a *Industry@Grid*. Uma breve descrição dela foi publicada no International Science Grid This Week (iSGTW) e se encontra no Apêndice B.

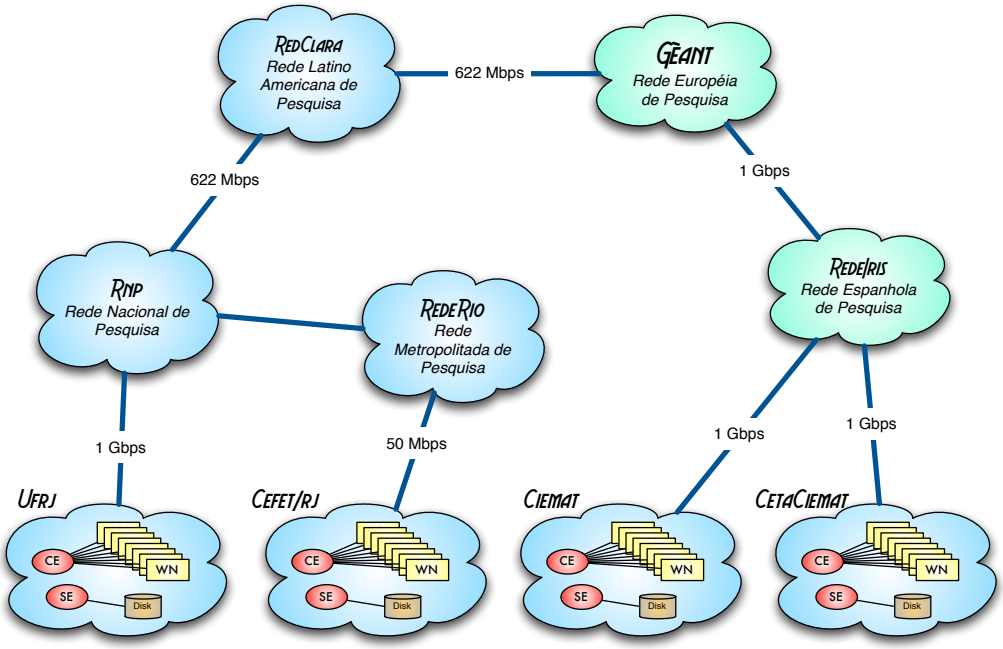


Figura 4.8: *Pilot Testbed*: Infraestrutura de testes.

4.3.1 Resultados

Experimento A

O Experimento A foi executado na configuração completa da infraestrutura e consistiu em execuções de 3 fluxos simultâneos da *Industry@Grid*. Foram feitas 30 medidas para cada um dos SGFP testados, o WAS e o Gridway.

Para o Gridway foi utilizada a *versão 5.4.0* com a configuração padrão e foi adicionado o módulo de submissão de tarefas para o gLite. No WAS, o SC foi configurado para criar uma infraestrutura virtual de 300 *job slots* de largura. Como exemplo, apresentamos a Figura 4.9, que mostra uma das execuções controladas pelo WAS na infraestrutura. Nesse exemplo específico, toda a execução dos 3 fluxos teve a duração de 615 minutos (menor tempo obtido). Na Figura 4.9 podemos identificar em azul escuro os três momentos em que foi necessário a submissão de

tarefas pelo SC, quando o número total de PJs rodando era menor que 300 (faixa em azul claro, no topo do gráfico). Em laranja, podem ser vistas as tarefas em trânsito do WMS aos RCs. Depois de chegar ao RC, as tarefas (em rosa) vão para as filas do LRMS. Em vermelho e salmão se encontram os PJ que apresentaram problemas e abortaram. Os PJs que terminaram a sua execução com sucesso são representados pela área verde.

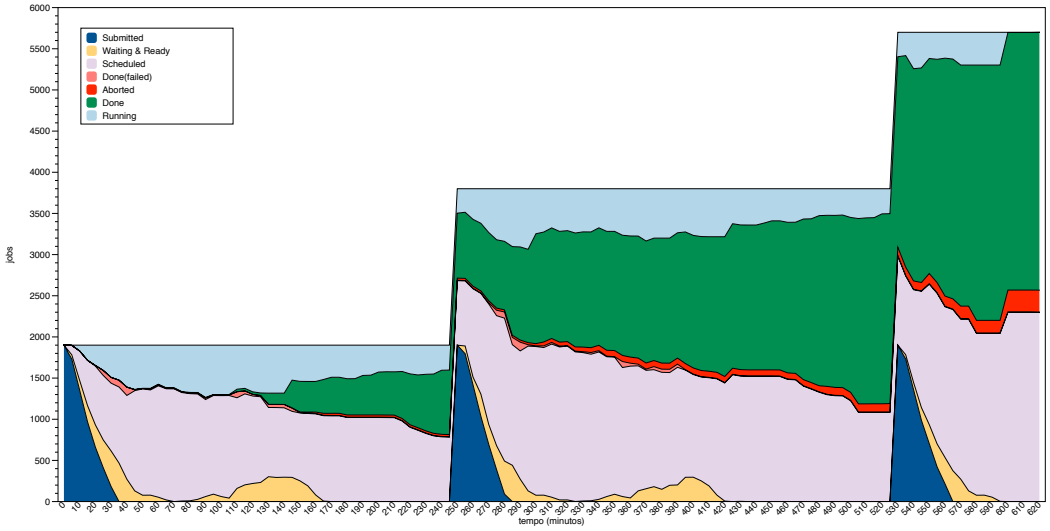


Figura 4.9: Execução da Pilot Factory com três fuxos simultâneos.

Na Figura 4.10, apresentamos a mesma execução da Figura 4.9, contudo no ponto de vista do *Schedule Enactor*. Podemos observar em amarelo os três fluxos sendo executados e em azul os PJ que ainda não tiveram a oportunidade de receber alguma tarefa do fluxo.

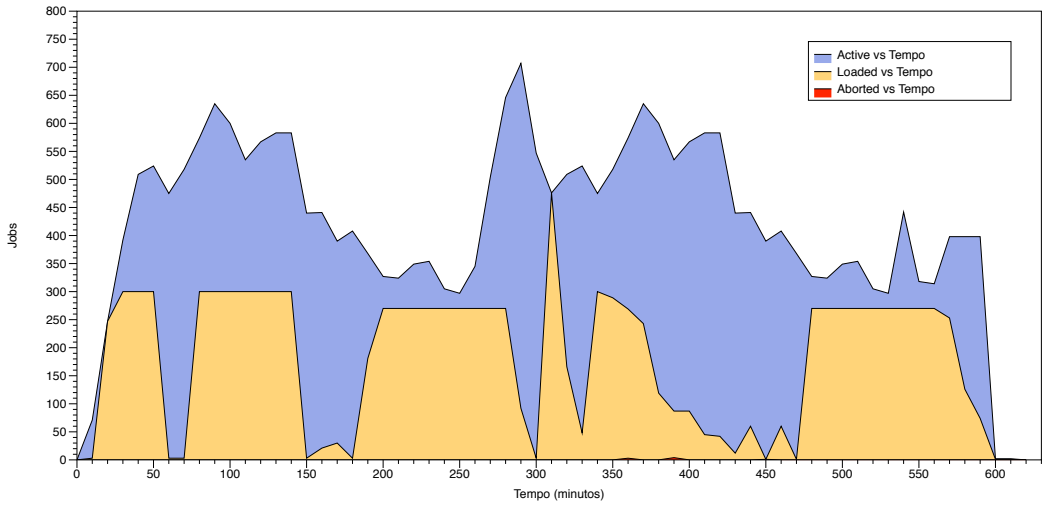


Figura 4.10: Execução usando o *Scheduling Fabric* do WAS.

O tamanho máximo da infraestrutura virtual foi de 736 *job slots* (295 minutos passados do início da execução) e ela apresentou um tamanho médio de 425 *job slots*.

Um fato curioso é que apesar da taxa de PJ abortados na figura (em vermelho) ser em torno de 5%, a eficiência da infraestrutura ficou na faixa dos 68%. Essa discrepância na eficiência ocorre pois a medida instantânea mostrada na figura não contabiliza a resubmissão efetuada pelo WMS durante a execução, o que obriga o cálculo da eficiência *post mortem*. Na Tabela 4.2, apresentamos o resultado da média das 30 medições efetuadas com o WAS e o Gridway, que indica um *speed up* de aproximadamente 1,6.

Tabela 4.2: Resultados do experimento A (em minutos) com $\alpha = 5\%$.

WAS	Gridway
$(815,8 \pm 49,6)$	$(1296,0 \pm 56,6)$

Experimento B

Para avaliar o efeito do GFP, no Experimento B foram executadas 30 rodadas da aplicação *Industry@Grid* sem GFP e mais 30 com o uso do GFP. Esse experimento foi efetuado na configuração *Pilot Testbed* e todos os arquivos foram instalados somente no RC da UFRJ-IF (Figura 4.8) que apresenta o maior enlace e o maior número de *job slots* disponível. Foi constatado que o GFP gerou um *speed up* de 1,3.

Tabela 4.3: Resultados do experimento B (em minutos) com $\alpha = 5\%$.

WAS	WAS sem GFP
$(401,3 \pm 30,6)$	$(526,1 \pm 43,9)$

4.4 Avaliação dos resultados dos experimentos

O primeiro fator significativo para explicar o *speed up* de 1,6 foi a imperfeição da informação disponível no momento do escalonamento. Na Figura 4.11 é mostrado um exemplo do estado dos recursos disponíveis na visão do GridWay. Nela, podemos observar em cada coluna a tradução do estado coletado no sistema de informação da infraestrutura para o estado interno do GridWay, onde as colunas mais relevantes têm o seguinte significado:

- MHZ — velocidade de processamento do recurso;
- MEM — memória disponível no recurso;
- DISK — espaço em disco disponível no recurso (não disponível no exemplo);

- N(U/F/T) — estado dos *job slots* no RC, onde U são os utilizados, F são os disponíveis e T o total de *job slots*.

No estado apresentado na Figura 4.11, os RCs mais rápidos seriam os de HID 14 e 15 (ce-eela.ceta-ciemat.es e ce-eela.ciemat.es, respectivamente), pois são os RCs que apresentam os processadores de maior "velocidade" (3200 MHz). Na realidade, essa informação, oriunda do sistema de informação do gLite, indica uma estimativa do poder computacional médio de todo o RC. O valor é médio e configurado pelo administrador do RC pois o *cluster* pode ser heterogêneo, ou seja, composto por diversas versões e tipos de *hardware*. Em visita aos diversos RCs do projeto EELA e EELA-2, verificamos que a heterogeneidade é um fato corrente por causa das constantes atualizações nos recursos. Essas atualizações ocorrem com uma freqüência anual, enquanto a vida média dos equipamentos é de normalmente 4 anos. Por causa desse motivo, não existe nenhuma garantia de qual será a "velocidade" do recurso disponível para efetivamente executar a tarefa submetida.

No caso do WAS, o valor utilizado na avaliação da "velocidade" de processamento de um recurso é oriundo do recurso propriamente dito, obtido no Linux diretamente no diretório `/proc`[3], pois é coletada no momento inicial da execução do PJ. Além disso, existe uma estimativa inicial melhor sobre a quantidade de memória disponível no momento da execução de uma tarefa do fluxo. O fato de se ter o recurso efetivamente sob controle realmente melhora a qualidade da informação para a tomada de decisão do escalonador.

O segundo fator significativo é a ineficiência do *pooling* executado pelo GridWay para verificar o estado dos *jobs*. O *pooling* é feito através de sucessivas consultas ao CE[4] para se verificar a troca de estados de um *job* e o tempo de resposta está diretamente subordinado à carga corrente no CE. Uma análise *post mortem* evidenciou que diversas tarefas do GridWay foram re-submetidas pois os estados dos *jobs* não eram atualizados e mecanismos de gerenciamento de falhas (através de *timeouts*) eram iniciados. O impacto desse problema ainda é mais grave quando ocorre em uma tarefa que pertence ao caminho crítico do fluxo de processos.

No caso do WAS, o sistema de comunicação implementado entre a tarefa e o SECT não utiliza o CE do RC e, por isso, não fica sujeito as influências da carga existente no sítio. Dessa maneira, o escalonador fica sabendo imediatamente no momento em que uma tarefa terminou, podendo efetivamente disparar as tarefas subseqüentes o mais cedo possível.

[3]diretório no sistema operacional Linux, onde se encontra informações sobre o tipo do processador, velocidade, memória livre e toda as informações disponíveis sobre o estado de todos os processos sendo executados em um computador

[4]na configuração utilizada, o GridWay utilizava a interface Globus (`globus-job-run`) existente no gLite para obter o estado do *job*.

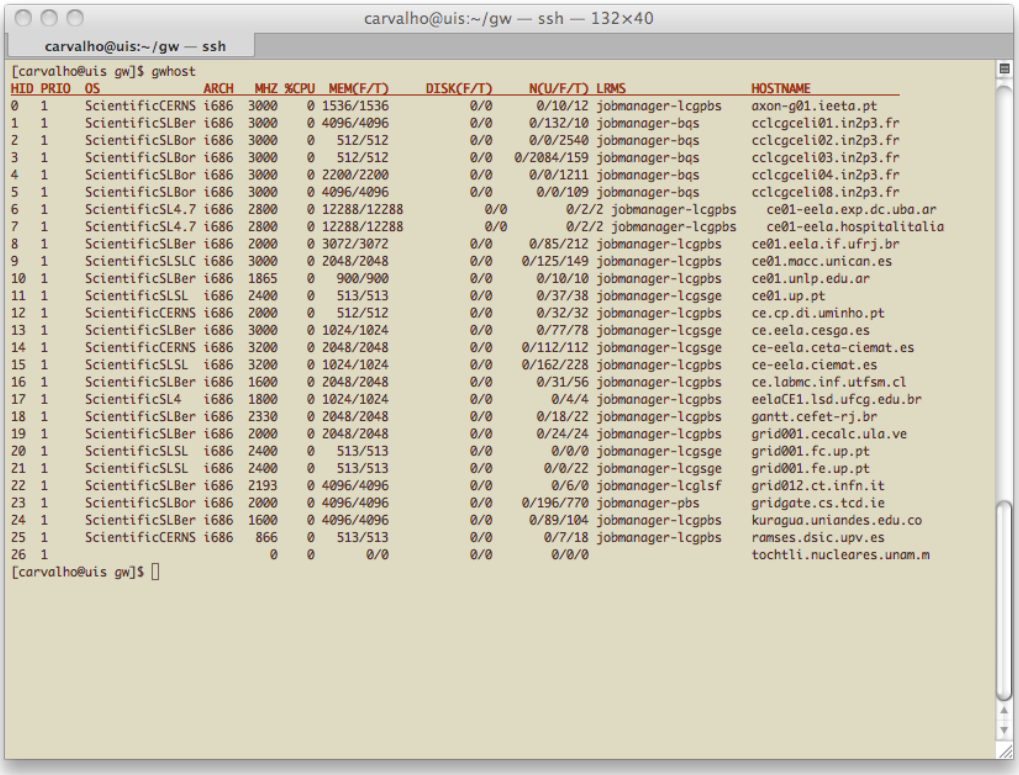


Figura 4.11: Exemplo da informação vista pelo GridWay no momento do despacho.

O terceiro fator significativo é relacionado diretamente com a imperfeição existente na informação sobre o estado da malha e do *overhead* do LRMS. Na análise *post mortem*, foi verificado que no limiar em que o sistema de informação indicava que um RC está com a carga próxima de sua capacidade total, o GridWay continuava com a submissão. Em alguns casos, o RC já tinha recebido outras tarefas para executar[5], obrigando o posicionamento das tarefas submetidas pelo GridWay na fila de espera do LRMS. Nesses casos em que as tarefas do caminho crítico foram enfileiradas, ocorreu um aumento no tempo total de execução do fluxo.

Novamente observamos que a implementação utilizada pelo WAS tem vantagens nesse tipo de situação, pois somente as tarefas que já passaram pelo caminho do LRMS e efetivamente começaram a rodar são apresentadas ao escalonador (SETC) para serem utilizadas na execução do fluxo.

Pelos três fatores apresentados acima, podemos observar que o esquema implementado pelo WAS efetivamente contorna os problemas provocados pela imperfeição na informação existente no sistema de informação da malha e pela exposição dos RCs e não dos recursos existentes na malha (processadores ou *job slots*).

No caso do experimento B, o *speed up* de 1,3 está diretamente relacionado com o número de possíveis candidatos existentes para se rodar uma tarefa no momento do

[5]oriundas de outros fluxos ou outros usuários

despacho feito pelo escalonador. Em uma análise *post mortem* se evidenciou que na versão sem o GFP, o algoritmo de afinidade do SECT concentrou a maior parte da execução dos três fluxos no RC da UFRJ-IF. Mesmo com essa concentração, entre 26% e 39% das tarefas foram executadas em outros RCs. Já na versão com GFP, entre 53% e 62% das tarefas foram executadas em RCs diferentes do UFRJ-IF. Esse número se aproxima da proporção de *job slots* que são fornecidos pelos outros RCs da infraestrutura do *Pilot Testbed*. Com esses resultados, mostramos a importância de se ter um sistema de *prefetch* no sistema de escalonamento quando associado à uma infraestrutura virtual.

Capítulo 5

Conclusão

Esse trabalho apresentou uma revisão das *e-Infrastructures* de serviço em produção e do *middleware* que é empregado na construção das mesmas. Depois, foi revisado o estado da arte do uso de fluxos de processos na *e-Science*, seguido pelos problemas relacionados às falhas que ocorrem nessas infraestruturas, junto com as imperfeições existentes nos sistemas de informação que mantêm o estado da malha. Com esses pontos abordados foi introduzido o WAS, um Sistema de Gerenciamento de Fluxo de Processos que tem por objetivo principal endereçar os problemas de falhas e imperfeição de informação.

Além disso, o WAS apresentou um bom desempenho em uma infraestrutura real, mostrando que é possível se obter um *speed up* de 1,6 quando comparado com uma solução de plena aceitação na comunidade de *grid*. Um outro ponto importante é que se mostrou as qualidades de se utilizar o conceito de infraestrutura virtual, baseadas em *pilot jobs*, pois o seu uso impacta diretamente nos problemas de imperfeição da informação existentes no momento do sequenciamento. É importante lembrar que, no caso do WAS, o sistema de informação da malha não é mais consultado para tomadas de decisões de escalonamento do fluxo de processos, somente a lista dos PJs disponíveis. Apesar de não atacar o problema de falhas diretamente, foi mostrado o potencial de se contornar o problema, tirando proveito de maneira mais eficiente da malha, não importando qual o seu estado corrente.

Por outro lado, podemos também dizer que uma infraestrutura virtual mascara os atrasos gerados pela influência dos LRMSs existentes nos RCs, pois somente os PJs realmente instanciados se apresentam para o escalonamento. Na realidade, uma das mais importantes contribuições é que o momento de tomada de decisão do escalonamento foi levado do momento da submissão da tarefa para o momento em que o recurso está realmente disponível para se usado pelo SGFP. Além disso, mesmo não tendo controle direto sobre o recurso, o SGFP ainda pode escolher entre os recursos realmente presentes em um determinado momento da execução.

Em resumo, como lista de contribuições explícitas do trabalho, gostaríamos de

citar:

- *(i)* segundo a taxonomia de Buyya [2, 3], um SGFP com arquitetura distribuída, tomada de decisão local, planejamento dinâmico, estratégia focada no desempenho, movimentação de dados automática, estrutura de fluxo em DAG e com tolerância à falhas com re-tentativa de submissão.
- *(ii)* um SGFP distribuído desenhado para a execução de diversos fluxos de usuários diferentes simultaneamente;
- *(iii)* um SGFP que executa os fluxos em uma infraestrutura virtual criada com PJs, onde a decisão do escalonamento é feita no momento do despacho da tarefa junto com o momento em que o recurso se torna disponível;
- *(iv)* um gerenciamento de falhas em dois níveis, compartilhado com o *middleware*, diminuindo a complexidade de implementação do SGFP;

Como subprodutos, esse trabalho produziu uma aplicação para a solução de problema de *mix de produtos estocástico* e diversos sistemas de monitoramento e instrumentação da malha e aplicações para se poder observar a execução dos experimentos realizados.

5.1 Trabalho futuro

O WAS abre diversas oportunidades para trabalhos futuros. Como primeiro passo, estudaremos as políticas de geração de PJs pela *Pilot Factory* na tentativa de se encontrar uma maneira mais precisa de controle da infraestrutura virtual. O objetivo principal desse controle seria o de se diminuir a pressão que a *Pilot Factory* efetua durante uma execução na infraestrutura. Depois seriam estudadas novas formas de se particionar os DAGs e novas heurísticas reguladoras do GFP na tentativa de se melhorar o impacto do *prefetch*. Nessa mesma linha, seria estudado o uso de novas técnicas oriundas da microarquitetura como execução e transferência especulativas. Um outro ponto importante seria o estudo da criação de *expositores* (*Pilot Jobs* que rodassem em outros *middlewares*) que poderiam apresentar recursos para o WAS oriundos de outras infraestruturas.

Referências Bibliográficas

[1] ASSIS, L., GAUDENCIO, M., SANTOS-NETO, E., et al. "Uma Arquitetura de Redes Virtuais para um Grid Aberto e Seguro". In: *Anais do VIII Workshop de Teste e Tolerância a Falhas*, 2007.

[2] YU, J., BUYYA, R. "A taxonomy of scientific workflow systems for grid computing", *ACM Sigmod Record*, v. 34, n. 3, pp. 49, 2005.

[3] YU, J., BUYYA, R. "A taxonomy of workflow management systems for grid computing", *Journal of Grid Computing*, v. 3, n. 3, pp. 171–200, 2005.

[4] CHEN, J., YANG, Y. "A taxonomy of grid workflow verification and validation", *Concurr Comp-Pract E*, v. 20, n. 4, pp. 347–360, 2008.

[5] Dutra, I., Corrêa, R., Fiallos, M., et al. (Eds.). *Models for Parallel and Distributed Computation - Theory, Algorithmic Techniques and Applications*, v. 67, *Applied Optimization*. Boston (USA), Kluwer Academic Publishers, 2002.

[6] NADIMINTI, K., DE ASSUNÇÃO, M. D., BUYYA, R. "Distributed Systems and Recent Innovations: Challenges and Benefits", *InfoNet Magazine*, v. 16, n. 3, Sep 2006.

[7] STEVENS, W. R. *TCP/IP Illustrated*. One Jacob Way; Reading, Massachusetts 01867, Addison-Wesley, 1994. ISBN 0-201-63346-9.

[8] BUYYA, R. *High Performance Cluster Computing*. NJ, USA, Prentice Hall, 1999.

[9] SUBRAMANIAN, R., GOODMAN, B. *Peer-to-Peer Computing: Evolution of a Disruptive Technology*. PA, USA, Idea Group Inc., 2005.

[10] FOSTER, I., KESSELMAN, C. *The Grid: Blueprint for a Future Computing Infrastructure*. USA, Morgan Kaufmann Publishers, 1999.

[11] BRAAM, P. J. "The Coda Distributed File System", *Linux Journal*, v. 50, pp. 12–25, jun. 1998. ISSN: 1075-3583.

[12] HEY, T., TANSLEY, S., TOLLE, K. *The Fourth Paradigm: Data-Intensive Scientific The Fourth Paradigm: Data-Intensive Scientific Discovery.* Redmond, Microsoft, 2009.

[13] WEISSMAN, J. B., GRIMSHAW, A. S. "A Federated Model for Scheduling in Wide-Area Systems". In: *HPDC '96: Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing*, p. 542, Washington, DC, USA, 1996. IEEE Computer Society. ISBN: 0-8186-7582-9.

[14] FOSTER, I. "What is the Grid? A Three Point Checklist", *Grid Today*, v. 1, n. 6, 2002.

[15] SHIERS, J., MCCANCE, G., LORENZO, P. M. "Robust and Resilient Services: How to design, build and operate them". In: *Proceedings of the Third EELA Conference*, 2007.

[16] "LCG Technical Design Report". . CERN-LHCC-2005-024, `http://lcg.web.cern.ch/LCG/tdr/`.

[17] GIL, Y., DEELMAN, E., ELLISMAN, M., et al. "Examining the Challenges of Scientific Workflows", *Computer*, v. 40, n. 12, pp. 24–32, 2007. ISSN: 0018-9162.

[18] HOLLINSWORTH, D. "The Workflow Reference Model, Workflow Management Coalition". , 1994. TC00-1003.

[19] BRAUN, T. D., SIEGEL, H. J., BECK, N., et al. "A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems". pp. 330–335, 1998.

[20] CASAVANT, T. L., KUHL, J. G. "A Taxonomy of Scheduling in General-purpose Distributed Computing Systems", *IEEE Transactions on Software Engineering*, v. 14, n. 2, pp. 141–154, 1988.

[21] KRAUTER, K. *A taxonomy and survey of grid resource management systems for distributed computing*, v. 32. New York, NY, USA, John Wiley & Sons, Inc., 2002. doi: http://dx.doi.org/10.1002/spe.432.

[22] ROTITHOR, H. G. "Taxonomy of Dynamic Task Scheduling Schemes in Distributed Computing Systems". 1994.

[23] LUDASCHER, B., ALTINTAS, I., BERKLEY, C., et al. "Scientific workflow management and the Kepler system", *Concurr Comp-Pract E*, v. 18, n. 10, pp. 1039–1065, 2006.

[24] HEY, T., TREFETHEN, A. "Cyberinfrastructure for e-science", *Science*, v. 308, n. 5723, pp. 817–821, 2005.

[25] "LHC Design Report". , . http://ab-div.web.cern.ch/ab-div/Publications/LHC-DesignReport.html.

[26] FOSTER, I., KESSELMAN, C. *Computational Grids", In: The Grid: Blueprint for a New Computing Infrastructure*. 500, Sansome Street, Suite 400, San Francisco, CA 94111, Morgan Kaufmann, 1998.

[27] "Enabling Grid for E-Science". . http://www.eu-egee.org/, em 2007-12-28.

[28] FOSTER, I., KESSELMAN, C. *THE GRID 2, Blueprint for a New Computing Infrastructure*. 500, Sansome Street, Suite 400, San Francisco, CA 94111, Morgan Kaufmann, 2003.

[29] PACINI, F. "Job Description Language (JDL) Attributes Specification (Submission through the WMProxy Service)". . https://edms.cern.ch/document/590869/, May 2005.

[30] COSTA, G. D., DIKAIAKOS, M. D., ORLANDO, S. "Nine months in the life of EGEE: a look from the South". In: *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS'07, Istanbul, 24/10/2007-26/10/2007*, pp. 281–287, http://www.ieee.org/, octobre 2007. IEEE. Disponível em: <ftp://ftp.irit.fr/IRIT/ASTRE/Mascots07.pdf>.

[31] CCITT. "X509, The directory - Authentication framework". , November 1987.

[32] "UK Computing for Particle Physics Grid". . http://www.gridpp.ac.uk/, em 2007-12-28, .

[33] "The NorduGrid Collaboration". . http://www.nordugrid.org/, em 2007-12-28.

[34] "INFN Grid project". . http://grid.infn.it/, em 2007-12-28.

[35] "A Large Ion Collider Experiment - Technical Proposal". , 1995. Disponível em: <http://consult.cern.ch/alice/Documents/1995/01/abstract>. CERN/LHCC-95-71 LHCC/P3.

[36] "Atlas technical proposal". . http://atlas.web.cern.ch/Atlas/TP/NEW/HTML/tp9new/tp9.html, em 2007-12-28. CERN/LHCC/94-43 LHCC/P2.

[37] "The Compact Muon Solenoid Technical Proposal". . http://cmsinfo.cern.ch/TP/TP.html, em 2007-12-28. CERN/LHCC 94-38 LHCC/P1.

[38] "LHCB Technical Proposal". . http://lhcb-tp.web.cern.ch/lhcb-tp/, em 2007-12-28, . CERN LHCC 98-4 LHCC/P4.

[39] "Ba Bar Collaboration". . http://www-public.slac.stanford.edu/babar/, em 2007-12-28.

[40] "The Collider Detector at Fermilab Technical Design Report". . http://www-cdf.fnal.gov/upgrades/tdr/tdr.html, em 2007-12-28. FERMILAB-Pub-96/390-E.

[41] "The D Zero Experiment". . http://www-d0.fnal.gov/atwork/index.html, em 2007-12-28.

[42] "The Zeus Experiment". . http://www-zeus.desy.de/, em 2007-12-28.

[43] "GATE - Geant4 Application for Emission Tomography". . http://www.opengatecollaboration.org/, em 2007-12-28.

[44] "Grid Network Protein Sequence Analysis". . http://gpsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page=/NPSA/npsa_server.html, em 2007-12-28.

[45] "CDSS: Clinical Decision Support System". . http://egee-na4.ct.infn.it/biomed/CDSS.html, em 2007-12-28.

[46] "Grid-enabled docking platform for tropial disease". . http://egee-na4.ct.infn.it/biomed/tropicaldisease.html, em 2007-12-28.

[47] GERMAIN, C., OSORIO, A., TEXIER, R. "A case study in medical imaging and the Grid". pp. 110–118, 2003.

[48] "X-Window-based Microscopy Image Processing Package". .

[49] "Molecular Docking web". . http://egee-na4.ct.infn.it/biomed/GridGRAMM.html, em 2007-12-28, .

[50] GARCIA ARISTEGUI, D. J., MENDEZ LORENZO, P., VALVERDE, J. R. "GROCK: High-Throughput Docking Using LCG Grid Tools". In: *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pp. 85–90, Washington, DC, USA, 2005. IEEE Computer Society. ISBN: 0-7803-9492-5. doi: http://dx.doi.org/10.1109/GRID.2005.1542728.

[51] "E-infrastructure shared between Europe and Latin America". .

[52] CARVALHO, D., MARECHAL, B., BELLO, P. H. R. "Building a Grid in Latin America: The EELA Project e-Infrastructure". In: *LA Grid07 - Seventh IEEE International Symposium on Cluster Computing and the Grid — CCGrid 2007*, 2007.

[53] CARVALHO, D., DUARTE, A., BELLO, P. H. R. "Operating a Transatlantic Grid Infrastructure". In: *Proceedings of the Third EELA Conference*, 2007.

[54] PORDES, R., PETRAVICK, D., KRAMER, B., et al. "The open science grid", *Journal of Physics: Conference Series*, v. 78, pp. 012057, 2007.

[55] SHIERS, J. "The Worldwide LHC Computing Grid (worldwide LCG)", *Computer physics communications*, v. 177, n. 1-2, pp. 219–223, 2007.

[56] "The TeraGrid Project". . `http://www.teragrid.org/`, em 2007-12-19.

[57] MIURA, K. "Overview of Japanese science Grid project NAREGI". In: *Progress in Informatics*, v. 3, pp. 67–75. National Institute of Informatics, 2006.

[58] "India's National grid Computing Initiative - Garuda". . `http://www.garudaindia.in/`.

[59] "South Eastern European Grid". .

[60] "Empowering e-Science across the Mediterranean". .

[61] "The Baltic Grid Project". . `http://www.balticgrid.org/`, em 2007-12-19.

[62] "The Eu China Grid project". . `http://www.euchinagrid.org/`, em 2007-12-19.

[63] CIRNE, W., BRASILEIRO, F., ANDRADE, N., et al. "Labs of the World, Unite!!!" *J Grid Computing*, v. 4, n. 3, pp. 225–246, 2006.

[64] FOSTER, I., KASSELMAN, C. "Globus: a metacomputing infrastructure toolkit", *International Journal of Supercomputing Applications*, v. 11, n. 2, pp. 115–128, 1997.

[65] FOSTER, I., KASSELMAN, C., TUECKE, G. "A security architecture for computational grids". 1998.

[66] FOSTER, I., KASSELMAN, C., TUECKE, S. "The anatomy of the grid: Enabling scalable virtual organizations", *International Journal of High Performance computing Applications*, v. 15, pp. 200–222, 2001.

[67] CZAJKOWSKI, K., FOSTER, I., KASSELMAN, C., et al. "A resource management architecture for metacomputing systems". In: *Proceedings of IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, Orlando, FL, USA, 1998.

[68] BESTER, J., FOSTER, I., KASSELMAN, C., et al. "Data movement and access service for wide area computing systems". In: *Sixth Workshop on I/O in Parallel and Distributed Systems*, Atlanta, GA, USA, 1999.

[69] RESCORLA, E., SCHIFFMAN, A. "The Secure HyperText Transfer Protocol". In: *RFC 2660*, 1999.

[70] FITZGERALD, S., FOSTER, I., KASSELMAN, C., et al. "A directory service for configuring high-performance distributed computations". In: *Proceedings of* 6^{th} *IEEE Symposium on high-performance distributed computing*, Portland, OR, US, 1997.

[71] ALLCOCK, B., BESTER, J., BRESNAHAM, J., et al. "Secure, Efficient Data Transport and Replica Management for High-performance data intensive computing". In: *IEEE mass storage conference*, San Diego, CA, US, 2001.

[72] LAURE, E., HEMMER, F., OTHERS. "Middleware for the Next Generation Grid Infrastructure". In: *Computing in High Energy and Nuclear Physics (CHEP) 2004*, Interlaken, Switzerland, September 2004.

[73] FOSTER, I., KESSELMAN, C., NICK, J., et al. "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration". , 2002. Disponível em: <`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.8105`>.

[74] ET AL., R. B. "GENIUS: a simple and easy way to access computational and data grids", *Future Generation Computer Systems*, 2003.

[75] GRIMSHAW, A., WULF, W. "The Legion vision of worldwide virtual computer", *Communications of the ACM*, v. 40, n. 1, 1997.

[76] ALMOND, J., SNELLING, D. "UNICORE: Uniform Access to Supercomputing as an Element of Eletronic Commerce", *Future Generation Computer Systems*, v. 613, pp. 1–10, 1999.

[77] ANDRADE, N., CIRNE, W., BRASILEIRO, F., et al. "Ourgrid: An approach to easily assemble grids with equitable resource sharing", *JSSPP 2003, LNCS 2862*, pp. 61–86, 2003.

[78] VENUGOPAL, S., BUYYA, R., WINTON, L. "A Grid service broker for scheduling e-Science applications on global data Grids", *Concurr Comp-Pract E*, v. 18, n. 6, pp. 685–699, 2006.

[79] CASAVANT, T., KUHL, J. "A TAXONOMY OF SCHEDULING IN GENERAL-PURPOSE DISTRIBUTED COMPUTING SYSTEMS", *Ieee T Software Eng*, v. 14, n. 2, pp. 141–154, 1988.

[80] BERMAN, F., WOLSKI, R., CASANOVA, H., et al. "Adaptive computing on the grid using AppLeS", *IEEE Trans. Parallel and Distrib. Systems*, v. 14, n. 4, pp. 369–382, 2003. Disponível em: <`http://csdl.computer.org/dl/trans/td/2003/04/l0369.pdf`>.

[81] ALLEN, G., BENGER, W., GOODALE, T., et al. "The Cactus Code: A Problem Solving Environment for the Grid". In: *Proceedings of the 9th International Symposium on High Performance Distributed Computing*, p. 253, 2000. Disponível em: <`http://computer.org/proceedings/hpdc/0783/07830253abs.htm`>.

[82] HAMSCHER, V., SCHWIEGELSHOHN, U., STREIT, A., et al. "Evaluation of Job-Scheduling Strategies for Grid Computing". In: Buyya, R., Baker, M. (Eds.), *Grid Computing - GRID 2000, First IEEE/ACM International Workshop, Bangalore, India, December 17, 2000, Proceedings*, v. 1971, *Lecture Notes in Computer Science*, pp. 191–202. Springer, 2000. ISBN: 3-540-41403-7. Disponível em: <`http://link.springer.de/link/service/series/0558/bibs/1971/19710191.htm`>.

[83] MATEESCU, G. "Qaulity of Service on the Grid via Metascheduling with Resource Co-scheduling and Co-reservation", *The International Journal of High Performance Computing Applications*, v. 17, n. 3, pp. 209–218, Fall 2003. ISSN: 1094-3420.

[84] FOSTER, IAMNITCHI. "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing". In: *International Workshop on Peer-to-Peer Systems (IPTPS), LNCS*, v. 2, 2003.

[85] LI, H., GROEP, D., WOLTERS, L., et al. "Job failure analysis and its implications in a large-scale production grid", *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, p. 27, 2006.

[86] DUARTE, A. N., NYCZYK, P., RETICO, A., et al. "Global grid monitoring: the EGEE/WLCG case". In: *GMW '07: Proceedings of the 2007 workshop*

on Grid monitoring, pp. 9–16, New York, NY, USA, 2007. ACM. ISBN: 978-1-59593-716-2. doi: http://doi.acm.org/10.1145/1272680.1272683.

[87] TAYLOR, F. W. *The Principles of Scientific Management*. Digireads ed. http://www.forgottenbooks.org/info/9781606801123, Republished 2008 by Forgotten Books, 1923.

[88] EMERSON, H. P., NAEHRING, D. C. E. *Origins of industrial engineering: the early years of a profession*. Norcross, GA, Industrial Engineering & Management Press - Institute of Industrial Engineers, Jan 1988.

[89] BARNES, R. M. *Motion and time study : design and measurement of work / Ralph M. Barnes*. 6th ed. ed. New York, USA, Wiley, 1968.

[90] MADHUSUDAN, T., ZHAO, J., MARSHALL, B. "A case-based reasoning framework for workflow model management", *Data & Knowledge Engineering*, 2004.

[91] WARGITSCH, C., WEWERS, T., THEISINGER, F. "An Organizational-Memory-Based Approach for an Evolutionary Workflow Management System - Concepts and Implementation". In: *HICSS (1)*, pp. 174–183, 1998.

[92] WARGITSCH, C., WEWERS, T., THEISINGER, F. "WorkBrain: Merging Organizational Memory and Workflow Management Systems". . Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.8.2148>.

[93] MUTH, P., WODTKE, D., WEISSENFELS, J., et al. "From centralized workflow specification to distributed workflow execution", *Journal of Intelligent ...*, 1998.

[94] BARISH, B. C. "The Laser Interferometer Gravitational-Wave Observatory LIGO", *Advances in Space Research*, v. 25, n. 6, pp. 1165 – 1169, 2000. ISSN: 0273-1177. doi: DOI:10.1016/S0273-1177(99)00980-1. Disponível em: <http://www.sciencedirect.com/science/article/B6V3S-3YKKGP7-1R/2/af74c9314596c64291a35f0bff3a2ef9>. Fundamental Physics in Space.

[95] PANDEY, S., VOORSLUYS, W., RAHMAN, M., et al. "A grid workflow environment for brain imaging analysis on distributed systems", *Concurrency and Computation: Practice and Experience*, v. 21, n. 16, pp. 2118–2139, 2009.

[96] FUTRELLE, J., GAYNOR, J., PLUTCHAK, J., et al. "Semantic middleware for e-science knowledge spaces", *MGC '09: Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, 2009.

[97] MARRU, S., PIERCE, M., HERATH, C. "Open Grid Computing Environment's Workflow Suite for E-Science Projects", *IEEE Fourth International Conference on eScience*, 2008.

[98] PARASTATIDIS, S., VIEGAS, E., HEY, T. "Viewpoint A "smart"cyberinfrastructure for research", *Communications of the ACM*, v. 52, n. 12, 2009.

[99] ABRAMSON, D., KOMMINENI, J., MCGREGOR, J. L., et al. "An Atmospheric Sciences Workflow and its implementation with Web Services", *Future Generation Comp. Syst*, v. 21, n. 1, pp. 69–78, 2005. Disponível em: <http://dx.doi.org/10.1016/j.future.2004.09.025>.

[100] BUYYA, R. "The Gridbus Toolkit for Grid and Utility Computing". In: *CLUSTER*. IEEE Computer Society, 2003. ISBN: 0-7695-2066-9. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/CLUSTER.2003.10011>.

[101] BUYYA, R., VENUGOPAL, S. "The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report", *CoRR*, v. cs.DC/0404027, 2004. Disponível em: <http://arxiv.org/abs/cs.DC/0404027>. informal publication.

[102] BUYYA, R., ABRAMSON, D., VENUGOPAL, S. "The Grid Economy", *Proceedings of the IEEE*, v. 93, n. 3, pp. 698–714, February 2005. doi: 10.1109/JPROC.2004.842784. Disponível em: <http://dx.doi.org/10.1109/JPROC.2004.842784>.

[103] DEELMAN, E., BLYTHE, J., GIL, Y., et al. "Mapping abstract complex workflows onto grid environments", *Journal of Grid Computing*, v. 1, n. 1, pp. 25–39, 2003.

[104] MARINESCU, D. C. *Internet-Based Workflow Management: Towards a Semantic Web*, v. 40, *Wiley Series on Parallel and Distributed Computing*. New York, USA, Wiley-Interscience, 2002.

[105] CYBOK, D. "A Grid workflow infrastructure", *Concurrency and Computation*, v. 18, n. 10, pp. 1243, 2006.

[106] HUANG, Y. "JISGA: A Jini-Based Service-Oriented Grid Architecture", *IJHPCA*, v. 17, n. 3, pp. 317–327, 2003. Disponível em: <http://dx.doi.org/10.1177/1094342003173001>.

[107] AMIN, K., VON LASZEWSKI, G., HATEGAN, M., et al. "GridAnt: A Client-Controllable Grid Workflow System". In: *in 37th Hawai International Conference on System Science, Island of Hawaii, Big Island*, pp. 5–8, 2004. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.2407>.

[108] GUAN, Z., HERNANDEZ, F., BANGALORE, P., et al. "Grid-Flow: a Grid-enabled scientific workflow system with a Petri-net-based interface", *CONCURRENCY AND COMPUTATION*, v. 18, n. 10, pp. 1115, 2006.

[109] CHEN, J., YANG, Y. "Multiple states based temporal consistency for dynamic verification of fixed-time constraints in Grid workflow systems", *Concurrency and Computation: Practice and Experience*, v. 19, n. 7, pp. 965–982, 2007. Disponível em: <http://dx.doi.org/10.1002/cpe.1088>.

[110] KRISHNAN, S., WAGSTROM, P., VON LASZEWSKI, G. "GSFL: A Workflow Framework for Grid Services - In Preprint ANL/MCS-P980-0802, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, 1L 60439, U.S.A., 2002." , 2002. Disponível em: <citeseer.ist.psu.edu/article/krishnan02gsfl.html>.

[111] WU, S., SHETH, A. P., MILLER, J. A., et al. "Authorization and Access Control of Application Data in Workflow Systems", *J. Intell. Inf. Syst*, v. 18, n. 1, pp. 71–94, 2002.

[112] DUAN, R., PRODAN, R., FAHRINGER, T. "DEE: A distributed fault tolerant workflow enactment engine for grid computing", *Lecture notes in computer science*, v. 3726, pp. 704, 2005.

[113] HWANG, S., KESSELMAN, C. "A flexible framework for fault tolerance in the grid", *Journal of Grid Computing*, v. 1, n. 3, pp. 251–272, 2003.

[114] CHEN, J., YANG, Y. "Temporal Dependency for Dynamic Verification of Temporal Constraints in Workflow Systems". In: Jin, H., Pan, Y., Xiao, N., et al. (Eds.), *Grid and Cooperative Computing - GCC 2004: Third International Conference, Wuhan, China, October 21-24, 2004. Proceedings*, v. 3251, *Lecture Notes in Computer Science*, pp. 1005–1008. Springer, 2004. ISBN: 3-540-23564-7. Disponí-

vel em: <http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3251&spage=1005>.

[115] CHEN, J., YANG, Y. "Temporal Dependency for Dynamic Verification of Fixed-Date Constraints in Grid Workflow Systems". In: Zhang, Y., Tanaka, K., Yu, J. X., et al. (Eds.), *Web Technologies Research and Development - APWeb 2005, 7th Asia-Pacific Web Conference, Shanghai, China, March 29 - April 1, 2005, Proceedings*, v. 3399, *Lecture Notes in Computer Science*, pp. 820–831. Springer, 2005. ISBN: 3-540-25207-X. Disponível em: <http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=3399&spage=820>.

[116] TANNENBAUM, T., WRIGHT, D., MILLER, K., et al. *Condor: a distributed job scheduler*. Cambridge, MA, USA, MIT Press, 2002. ISBN: 0-262-69274-0.

[117] PAHWA, J., WHITE, R., JONES, A., et al. "Accessing Biodiversity Resources in computational environments from workflow applications", *Workshop on Workflows in Support of Large-Scale Science (in conjunction with the 15th IEEE International Symposium on High Performance Distributed Computing), Paris, France*, 2006.

[118] MCGOUGH, A. S., LEE, W., DARLINGTON, J. "Workflow Deployment in ICENI II". In: Alexandrov, V. N., van Albada, G. D., Sloot, P. M. A., et al. (Eds.), *Computational Science - ICCS 2006, 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part III*, v. 3993, *Lecture Notes in Computer Science*, pp. 964–971. Springer, 2006. ISBN: 3-540-34383-0. Disponível em: <http://dx.doi.org/10.1007/11758532_130>.

[119] OINN, T., ADDIS, M. J., FERRIS, J., et al. "Taverna: a tool for the composition and enactment of bioinformatics workflows", pp. 3045–3054, jun. 2004. Disponível em: <http://eprints.ecs.soton.ac.uk/10912/1/bth361?ijkey=72HGjsFzej5oQ;http://eprints.ecs.soton.ac.uk/10912/>.

[120] BERMAN, F., CHIEN, A., COOPER, K., et al. "The GrADS Project: Software Support for High-Level Grid Application Development", *The International Journal of High Performance Computing Applications*, v. 15, n. 4, pp. 327–344, nov. 2001. ISSN: 1094-3420. Disponível em: <http://www.netlib.org/utk/people/JackDongarra/PAPERS/gradspaper.pdf>.

[121] CAO, J., JARVIS, S. A., SAINI, S., et al. "GridFlow: Workflow Management for Grid Computing". In: *CCGRID*, pp. 198–205. IEEE Computer Society, 2003. ISBN: 0-7695-1919-9. Disponível em: <http://csdl.computer.org/comp/proceedings/ccgrid/2003/1919/00/19190198abs.htm>.

[122] ERWIN, D., SNELLING, D. "UNICORE: A Grid computing environment", *Concurrency and Computation: Practice and Experience*, v. 14, n. 13-15, pp. 1395–1410, 2002.

[123] YU, J., BUYYA, R. "A Novel Architecture for Realizing Grid Workflow using Tuple Spaces". In: *5th International Workshop on Grid Computing (GRID 2004), 8 November 2004, Pittsburgh, PA, USA, Proceedings*, pp. 119–128. IEEE Computer Society, 2004. ISBN: 0-7695-2256-4. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/GRID.2004.3>.

[124] VON LASZEWSKI, G., HATEGAN, M. "Workflow Concepts of the Java CoG Kit", *J. Grid Comput*, v. 3, n. 3-4, pp. 239–258, 2005. Disponível em: <http://dx.doi.org/10.1007/s10723-005-9013-5>.

[125] VARGAS, P. K., DE CASTRO DUTRA, I., DO NASCIMENTO, V. D., et al. "GRAND: toward scalability in a Grid environment", *Concurrency and Computation: Practice and Experience*, v. 19, n. 14, pp. 1991–2009, 2007. Disponível em: <http://dx.doi.org/10.1002/cpe.1138>.

[126] HUEDO, E., MONTERO, R. S., LLORENTE, I. M. "Coordinated Use of Globus Pre-WS and WS Resource Management Services with GridWay", *Lecture Notes in Computer Science (LNCS)*, v. 3762, n. 234-243, 2005.

[127] LEAL, K., HUEDO, E., LLORENTE, I. M. "A decentralized model for scheduling independent tasks in Federated Grids", *Future Generation Computer Systems*, v. 25, n. 8, pp. 840–852, 2009.

[128] DER AALST, W. V., HOFSTEDE, A. T., KIEPUSZEWSKI, B., et al. "Workflow patterns", *Distrib Parallel Dat*, v. 14, n. 1, pp. 5–51, 2003.

[129] HULETTE, G., SOTTILE, M. "WOOL: A Workflow Programming Language", *ESCIENCE '08: Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pp. 71–78, 2008.

[130] SIRVENT, R., PÉREZ, J., BADIA, R., et al. "Automatic Grid workflow based on imperative programming languages", *CONCURRENCY AND COMPUTATION*, v. 18, n. 10, pp. 1169, 2006.

[131] GIL, Y., DEELMAN, E., ELLISMAN, M., et al. "Examining the challenges of scientific workflows", *Computer*, pp. 24–32, 2007.

[132] DER AALST, W. V., BASTEN, T. "Inheritance of workflows: an approach to tackling problems related to change", *Theoretical Computer Science*, 2002.

[133] DAVULCU, H., KIFER, M., RAMAKRISHNAN, C. "Logic based modeling and analysis of workflows", *Proceedings of the ...*, 1998.

[134] OSTERMANN, S., IOSUP, A., PRODAN, R., et al. "On the characteristics of grid workflows", *Integrated Research in Grid Computing*, pp. 431–442, 2008.

[135] DEELMAN, E., BLYTHE, J., GIL, Y., et al. "Pegasus: Mapping scientific workflows onto the grid", *Lecture Notes in Computer Science*, v. 3165, pp. 11–20, 2004.

[136] MEYER, L., ANNIS, J., WILDE, M., et al. "Planning spatial workflows to optimize grid performance", *Proceedings of the 2006 ACM symposium on Applied computing*, p. 790, 2006.

[137] CHURCHES, D., GOMBAS, G., HARRISON, A., et al. "Programming scientific and distributed workflow with Triana services", *CONCURRENCY AND COMPUTATION*, v. 18, n. 10, pp. 1021, 2006.

[138] MCPHILLIPS, T., BOWERS, S., ZINN, D., et al. "Scientific workflow design for mere mortals", *Future Gener Comp Sy*, v. 25, n. 5, pp. 541–551, 2009.

[139] PRODAN, R., FAHRINGER, T. "Overhead Analysis of Scientific Workflows in Grid Environments", *IEEE Trans. Parallel Distrib. Syst*, v. 19, n. 3, pp. 378–393, 2008. Disponível em: <`http://doi.ieeecomputersociety.org/10.1109/TPDS.2007.70734`>.

[140] LIU, X., CHEN, J., LIU, K., et al. "Forecasting Duration Intervals of Scientific Workflow Activities Based on Time-Series Patterns", *Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pp. 23–30, 2008.

[141] IOSUP, A., JAN, M., SONMEZ, O., et al. "On the dynamic resource availability in grids", *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pp. 26–33, 2007.

[142] HAMSCHER, V., ET AL. "Evaluation of Job-Scheduling Strategies for Grid Computing". In: *1st IEEE/ACM International Workshop on Grid Computing*, 2000.

[143] STEFANO, A. D., FARGETTA, M., PAPPALARDO, G., et al. "Supporting resource reservation and allocation for unaware applications in Grid systems", *Concurrency and Computation: Practice & Experience*, v. 18, n. 8, pp. 851–863, 2006.

[144] DEELMAN, E., BLYTHE, J., GIL, Y., et al. "Workflow management in GriPhyN", *Grid resource management: state of the art and future trends*, pp. 99–116, 2004.

[145] SONG, S. S., KWOK, Y. K., HWANG, K. "Trusted Job Scheduling in Open computational Grids". In: *19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[146] ANDREWS, T., CURBERA, F., DHOLAKIA, H., et al. "Business Process Execution Language for Web Services, Version 1.1". . Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.

[147] LEYMANN, F. "Web Services Flow Language, Version 1.0". . International Business Machines Corporation, 2001.

[148] WORLD WIDE WEB CONSORTIUM. "Extensible Markup Language (XML)". . `http://www.w3.org/XML/`.

[149] XML 2006. *XML 2006*, Boston, Massachusetts, dez. 2006.

[150] HOHEISEL, A. "User tools and languages for graph-based Grid workflows", *CONCURRENCY AND COMPUTATION*, v. 18, n. 10, pp. 1101, 2006.

[151] TAYLOR, I. J., SHIELDS, M. S., WANG, I., et al. "Triana Applications within Grid Computing and Peer to Peer Environments", *J. Grid Comput*, v. 1, n. 2, pp. 199–217, 2003. Disponível em: <`http://dx.doi.org/10.1023/B:GRID.0000024074.63139.ce`>.

[152] BIRON, P. V., MALHOTRA, A. "XML Schema Part 2: Datatypes Second Edition". . World Wide Web Consortium, Recommendation REC-xmlschema-2-20041028, out. 2004.

[153] BORENSTEIN, N., LINIMON, M. "RFC 1437: The Extension of MIME Content-Types to a New Medium". , abr. 1993. Disponível em: <`ftp://ftp.internic.net/rfc/rfc1437.txt,ftp://ftp.math.utah.edu/pub/rfc/rfc1437.txt`>. Status: INFORMATIONAL.

[154] CALLAHAN, S. P., FREIRE, J., SANTOS, E., et al. "Managing the Evolution of Dataflows with VisTrails". In: Barga, R. S., Zhou, X. (Eds.), *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, p. 71. IEEE Computer Society, 2006. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/ICDEW.2006.75>.

[155] KRISHNAN, S., WAGSTROM, P., LASZEWSKI, G. V. "GSFL: A workflow framework for grid services", *Preprint ANL/MCS-P980-0802, Argonne National Laboratory*, v. 9700, 2002.

[156] FAHRINGER, T., QIN, J., HAINZER, S. "Specification of grid workflow applications with AGWL: an Abstract Grid Workflow Language". In: *CCGRID*, pp. 676–685. IEEE Computer Society, 2005. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/CCGRID.2005.1558629>.

[157] SULISTIO, A., BUYYA, R. "A grid simulation infrastructure supporting advance reservation", *16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, 2004.

[158] DEELMAN, E., KOSAR, T., KESSELMAN, C., et al. "What makes workflows work in an opportunistic environment?" *Concurrency and Computation*, v. 18, n. 10, pp. 1187, 2006.

[159] RANJAN, R., HARWOOD, A., BUYYA, R. "A case for cooperative and incentive-based federation of distributed clusters", *Future Gener Comp Sy*, v. 24, n. 4, pp. 280–295, 2008.

[160] LIU, C., BASKIYAR, S. "A general distributed scalable grid scheduler for independent tasks", *J. Parallel Distrib. Comput.*, v. 69, n. 3, pp. 307–314, 2009.

[161] MCPHILLIPS, T., BOWERS, S. "An approach for pipelining nested collections in scientific workflows", *ACM Sigmod Record*, v. 34, n. 3, pp. 17, 2005.

[162] ANDERSON, D. "BOINC: A system for public-resource computing and storage", *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID'04)*, pp. 4 – 10, 2004.

[163] SUBRAMANI, V., KETTIMUTHU, R., SRINIVASAN, S., et al. "Distributed job scheduling on computational grids using multiple simultaneous re-

quests", *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, p. 359, 2002.

[164] BLYTHE, J., JAIN, S., DEELMAN, E., et al. "Task scheduling strategies for workflow-based applications in grids", *Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)-Volume*, v. 2, pp. 759–767.

[165] YU, J., BUYYA, R., RAMAMOHANARAO, K. "Workflow Scheduling Algorithms for Grid Computing", *Metaheuristics for Scheduling in Distributed Computing Environments*, pp. 109–153, 2008.

[166] MONTAGNAT, J., GLATARD, T., PLASENCIA, I. C., et al. "Workflow-Based Data Parallel Applications on the EGEE Production Grid Infrastructure", *Journal of Grid Computing*, v. 6, n. 4, pp. 369–383, 2008.

[167] SFILIGOI, I. "glideinWMS—a generic pilot-based workload management system", *Journal of Physics: Conference Series*, v. 119, pp. 062044, 2008.

[168] CHRISTODOULOPOULOS, K., GKAMAS, V., VARVARIGOS, E. A. "Statistical Analysis and Modeling of Jobs in a Grid Environment", *J Grid Comput*, v. 6, n. 1, pp. 77–101, 2008.

[169] YEO, C. S., BUYYA, R. "Pricing for utility-driven resource management and allocation in clusters", *Int J High Perform C*, v. 21, n. 4, pp. 405–418, 2007.

[170] METCALFE, R., BOGGS, D. "Ethernet: distributed packet switching for local computer networks", *Communications of the ACM*, v. 19, n. 7, 1976.

[171] KWAK, BYUNG-JAE, SONG, et al. "Performance analysis of exponential backoff", *IEEE/ACM Trans. Netw.*, v. 13, n. 2, pp. 343–355, 2005.

[172] DUAN, R., PRODAN, R., FAHRINGER, T. "Run-time optimisation of grid workflow applications", *GRID '06: Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, pp. 33–40, 2006.

[173] WANG, X., SCHULZRINNE, H., KANDLUR, D. "Measurement and analysis of LDAP performance", *IEEE/ACM Transactions on Networking*, v. 16, n. 1, pp. 232–243, 2008.

[174] CARVALHO, D., BELLO, P. H. R., DUARTE, A., et al. "Mining the EELA-2 e-infrastructure". In: *Proceedings of the First EELA-2 Conference*, 2009.

[175] TIERNEY, B., GUNTER, D., SCHOPF, J. "The CEDPS troubleshooting architecture and deployment on the open science grid", *Journal of Physics: Conference Series*, v. 78, pp. 012075, 2007.

[176] THAIN, D., TANNENBAUM, T., LIVNY, M. "How to measure a large open-source distributed system", *Concurrency and Computation: Practice and Experience*, 2006.

[177] CHAWLA, N. V., THAIN, D., LICHTENWALTER, R., et al. "Data Mining on the Grid for the Grid", *IEEE International Parallel & Distributed Processing Symposium '08*, 2008.

[178] IOSUP, A., JAN, M., SONMEZ, O., et al. "On the dynamic resource availability in grids", *GRID '07: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, 2007.

[179] CARVALHO, D., BRASILEIRO, F., DUARTE, A., et al. "An Approach for the Co-existence of Service and Opportunistic Grids: The EELA-2 Case". In: *Latin American Grid - 20th International Symposium on Computer Architecture and High Performance Computing*, 2008.

[180] SFILIGOI, I., KOEROO, O., VENEKAMP, G., et al. "Addressing the pilot security problem with gLExec", *Journal of Physics: Conference Series*, v. 119, 2008.

[181] BARBOSA, V. C. *Concurrency in Systems with Neighborhood Constraints*. Ph.D. thesis, UCLA Computer Science Department, Los Angeles, 1986.

[182] BARBOSA, V., GAFNI, E. "Concurrency in heavily loaded neighborhood-constrained systems", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, v. 11, n. 4, pp. 562–584, 1989.

[183] BARBOSA, V., BENEVIDES, M., FRANÇA, F. "Sharing resources at nonuniform access rates", *Theory of Computing Systems*, v. 34, n. 1, pp. 13–26, 2000.

[184] CARVALHO, D., PROTTI, F., GREGORIO, M. D., et al. "A Novel Distributed Scheduling Algorithm for Resource Sharing Under Near-Heavy Load". In: *OPODIS*, pp. 431–442, 2004.

[185] BAER, J., CHEN, T. "An effective on-chip preloading scheme to reduce data access penalty", *Supercomputing '91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pp. 176–186, 1991.

[186] VANDERWIEL, S., LILJA, D. "Data prefetch mechanisms", *ACM Computing Surveys (CSUR)*, 2000.

[187] GASPAR, C., DÖNSZELMANN, M., CHARPENTIER, P. "DIM, a portable, light weight package for information publishing, data transfer and inter-process communication", *Computer physics communications*, v. 140, pp. 102–109, 2001.

[188] BROCHU, F., EGEDE, U., ELMSHEUSER, J., et al. "Ganga: a tool for computational-task management and easy access to Grid resources", *CoRR*, v. abs/0902.2685, 2009.

[189] SANTOS, N., KOBLITZ, B. "Metadata services on the grid", *Nuclear Inst. and Methods in Physics Research*, 2006.

[190] BARTL, W., ET AL. "DELPHI : TECHNICAL PROPOSAL." . DELPHI-83-66-1 - DELPHI Collaboration., May 1983.

[191] OLOF FJALLSTROM, P. "Algorithms for Graph Partitioning: A Survey", *Linköping Electronic Articles in Computer and Information Science*, v. 3, n. 010, 1998.

[192] HENDRICKSON, B., KOLDA, T. "Graph partitioning models for parallel computing", *Parallel Computing*, v. 26, n. 12, pp. 1519–1534, 2000.

[193] FRANÇA, F. M. G., FARIA, L. "Optimal mapping of neighbourhood-constrained systems". In: *Proceedings, IRREGULAR '95*, v. 980, *Lecture Notes in Computer Science*, pp. 165–170, Lyon, France, 1995. Springer-Verlag.

[194] GAREY, M. R., JOHNSON, D. S. *Computers and intractability: a guide to the theory of NP-Completness*. New York, NY, USA, Freeman, 1979.

[195] CARVALHO, D., MARECHAL, B., BELLO, P., et al. "Applications ported to the EELA e-Infrastructure", *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pp. 852 – 857, 2007.

[196] CARVALHO, D., GAVILLET, P., DELGADO, V., et al. "ON THE RELEVANCY OF EFFICIENT, INTEGRATED COMPUTER AND NETWORK MONITORING IN HEP DISTRIBUTED ONLINE ENVIRONMENT". In: *In Proceedings of the International Conference on Computing in High Energy Physics' 95*. World Scientific Publishing Co. Pte. Ltd., 1995.

[197] PUJARI, N., HALE, T. S., HUQ, F. "A framework for an integrated distribution system optimization model", *International International Journal of Logistics Systems and Management*, v. 4, n. 5, pp. 506–522, 2008.

[198] TAN, K. H., PLATTS, K. "Operationalising strategy: Mapping manufacturing variables", *International Journal of Production Economics*, v. 89, n. 3, pp. 379–393, 2004.

Apêndice A

Pseudo algoritmos

Pseudo Código ScheduleEnactor

```
001 class SchedulerEnactor
002 {
003 public:
004     NewSchedule( File f );
005     PilotJobInterruption( PilotJob PJID );
006     Loop();
007 private:
008     Schedule    itsSchedule;
009     SiteList    itsSiteUniverse;
010 }
011
012 ScheduleEnactor::NewSchedule( File f )
013 {
014     itsSchedule = AMGA.CreateNewSchedule(f);
015     AMGA.CheckReplicas(itsSchedule);
016     PilotInventory.RequestVirtualMachine(itsSchedule);
017 }
018
019 ScheduleEnactor::PilotJobInterruption( PilotJob PJID )
020 {
021     task = AMGA.TaskFromPilotJobID(PJID);
022
023     if ( task.Status() != DONE ) {
024         AMGA.CleanTaskRunningStatus(task);
025     }
026     AMGA.SetTaskReallyDone(task);
027 }
```

```
028
029 ScheduleEnactor::Loop()
030 {
031     SiteList sl;
032
033     TaskList ScheduleTasks =
AMGA.GetReadyTasks(itsSchedule,OrderBy.ExectuionTime);
034
035     for ( Task t : ScheduleTasks) {
036         for ( File f : t.FileList() ) {
037             if ( f.empty() ) {
038                 sl += MAX_CPU;
039             }
040             else {
041                 if ( f.ReplicaList().empty() ) {
042                     sl += MAX_CPU;
043                 }
044                 else {
045                     for ( Replica r : f.ReplicaList() ) {
046                         sl += r.Site();
047                     }
048                 }
049             }
050         }
051     }
052
053     PilotJobSet = PilotInventory.SendRequest(sl);
054
055     for ( Task t : ScheduleTasks) {
056         for ( File f : t.FileList() ) {
057             if ( f.empty() ) {
058
059                 PilotJob = PilotJobSet.Select(MAX_CPU);
060
061                 if( PilotJob.empty() ) {
062                     continue;
063                 }
064
065                 PilotJob.DispatchTask(t);
066
067                 AMGA.UpdateTask(itsSchedule,t,PilotJob);
```

```
068
069             }
070             else {
071
072                 if ( f.ReplicaList().empty() ) {
073                     PilotJob = PilotJobSet.Select(MAX_CPU);
074                     if ( PilotJob.empty() ) {
075                         continue;
076                     }
077                     PilotJob.DispatchTask(t);
078                     AMGA.UpdateTask(itsSchedule,t,PilotJob);
079                 }
080                 else {
081                     for ( Replica r : f.ReplicaList() ) {
082                         PilotJob = PilotJobSet.Select(r.Site());
083                         if ( PilotJob.empty() ) {
084                             continue;
085                         }
086                         PilotJob.DispatchTask(t);
087                         AMGA.UpdateTask(itsSchedule,t,PilotJob);
088                         break;
089                     }
090                 }
091             }
092         }
093     }
094
095     ScheduleTasks =
AMGA.GetReadyTasks(itsSchedule,OrderBy.ExectuionTime);
096
097     RunningTasks = AMGA.RunningTasks();
098     DecisionWindow = RunningTasks.MinETA();
099
100     sl.clear();
101
102     for ( Task t : ScheduleTasks) {
103         for ( File f : t.FileList() ) {
104             if ( f.empty() ) {
105                 continue;
106             }
107             else {
```

```
108                 if ( f.ReplicaList().empty() ) {
109                     continue;
110                 }
111                 else {
112                     for ( Replica r : f.ReplicaList() ) {
113                         sl <- r.Site();
114                     }
115 
116                     EmptyReplicaSiteList = itsSiteUniverse - sl;
117 
118                     for ( Site s : EmptyReplicaSiteList.SiteName() ) {
119                         if ( s.TransferTime(f.Size()) < DecisionWindow
) {
120                             GlobalFilePrefetcher.ScheduleTransfer(f,s);
121                         }
122                     }
123                 }
124             }
125         }
126     }
127 }
128 
129 
130 
131 
```

Pseudo Código PilotInventory

```
001 class PilotInventory
002 {
003 public:
004     RequestVirtualMachine( Schedule sch );
005     SendRequest( SiteList sl );
006     Register( PilotJob PJID );
007     Deregister( Pilot PJID );
008     Loop();
009 private:
010     int itsMachineSize;
011     PilotList itsPilotList;
012 }
013
014 PilotInventory::RequestVirtualMachine( Schedule sch )
015 {
016     itsMachineSize += sch.VirtualMachineSize();
017 }
018
019 PilotInventory::SendRequest( SiteList sl )
020 {
021     PilotList ans;
022
023     for ( Site s : sl ) {
024         if ( s == MAX_CPU ) {
025             PilotJob p = itsPilotList.getFastestCPU();
026             if ( p.empty() ) {
027                 break;
028             }
029             ans += p;
030             itsFreePilotList.remove(p);
031         }
032         else {
033             PilotJob p = itsPilotList.getJobFromSite(s);
034             if ( p.empty() ) {
035                 continue;
036             }
037             ans += p;
038             itsPilotList.remove(p);
039         }
```

```
040     }
041
042     return ans;
043 }
044
045 PilotInventory::Register( Pilot PJID )
046 {
047     itsPilotList.add(PJID);
048 }
049
050 PilotInventory::Deregister( Pilot PJID )
051 {
052     itsPilotList.remove(PJID);
053 }
054
055 PilotInventory::Loop()
056 {
057     // FOREVER
058     for (;;) {
059         if ( itsPilotList.size() < itsMachineSize ) {
060             SubmissionControler.Submit(BATCHSIZE);
061         }
062
063         sleep(SLEEPSIZE);
064     }
065 }}
```

Pseudo Código SubmissionControler

```
001 class SubmissionControler
002 {
003 public:
004     Submit( int size );
005 }
006
007 SubmissionControle::Submit( int site )
008 {
009     gLite.submitParamjob(size);
010 }}
```

Pseudo Código GlobalFilePrefetcher

```
001 class GlobalFilePrefetcher
002 {
003 public:
004   ScheduleTransfer( File f, Site s );
005 }
006
007 GlobalFilePrefetcher::ScheduleTransfer( File f, Site s )
008 {
009     LFN l = f.ConverLFN();
010
011     FTS.ScheduleReplica(l,s);
012 }}
```

Apêndice B

Industry@Grid Application

Release técnico enviado ao iSGTW (`www.isgtw.org`) sobre a aplicação Industry@Grid.

Industry@Grid is grid application developed by the Production Engineering Department of CEFET/RJ and it is composed by a set of grid modules aiming at solving industrial operations problems, such as shop scheduling, product mix (Figure B.1) and supply chain optimisation. Its gridification was achieved during the first year of the EELA-2 project, when several validation tests were performed. During the second year, 14,285 jobs were executed, totalling 727 CPU.days.

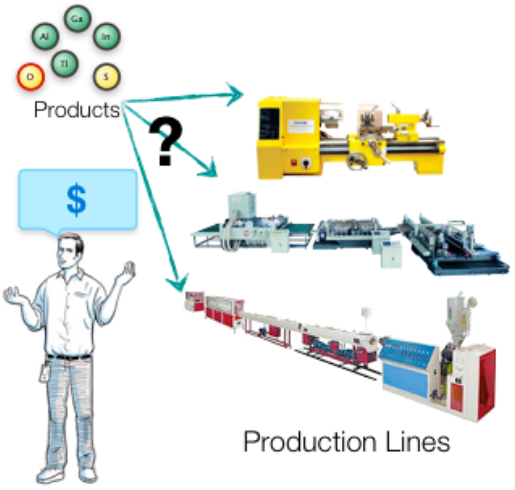


Figura B.1: Product Mix Problem - The problem is to find out which products to include in the production plan and in what quantities these should be produced in order to maximise profit, market share or some other goal.

This application was employed in a real industrial environment and the company studied is composed by three a biaxially oriented polypropylene (BOPP) film factories settled one in Argentina and two in Brazil. More than 180 different products compose its portfolio and its production is sold to the packaging industry of both

countries. BOPP film is used to make a wide variety of wrapping materials and clear bags.

A BOPP production line is able to produce a few hundreds of distinct products, varying weight, painting and material cutting. In the context of Vitopel, which has six production lines (two per facility) and three consumer markets (Argentina, Brazil and Exports), the optimisation of the product mix yields to a mix integer-programming (MIP)[197] problem that has about 4,500 variables and more than 3,000 constraints. Besides the large number of variables and constraints, each production line generates stochastic scrap during its production process through extrusion and cutting, which increases the complexity, since stochastic variables are introduced.

Biaxially oriented polypropylene film (BOPP) is produced in a continuous processing industry. BOPP films are used for different applications, such as: standard and self-adhesive labeling, packaging, for use in conversion, overwrapping and tobacco, and industrial and graphic for graphic lamination and adhesive tapes.

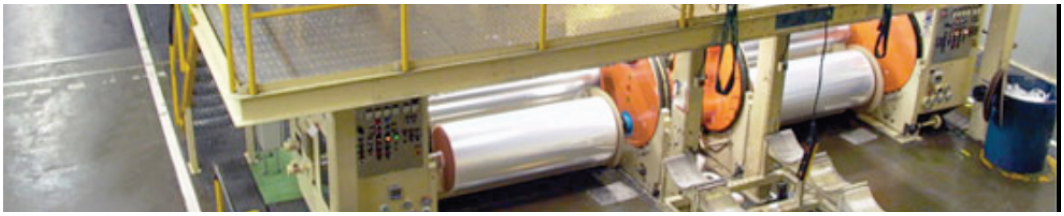


Figura B.2: BOPP Production Line

Figure B.2 depicts that a typical BOPP film production line. The production process consists of many phases grouped into two workstations, one equiped with only one OPP biaxial expander and the other has several OPP splitting machines. The raw materials are consecutively fed into the OPP biaxial expander and then the OPP splitting machine to produce the final products.

The fundamental step in the production of high level property polymer films (such as BOPP) consists in a biaxial stretching of the material. Principally, this can be realized by two different technologies. In the blown-film process, the stretching takes place by blowing-up of a polymer tube in the molten state realizing a deformation in longitudinal and circumferential direction. Alternatively, in the flat-film process, the case studied here, an extruded polymer film is biaxially drawn in a tenter frame after solidification at temperatures below the melting point. The drawing of flat films in two perpendicular directions can proceed in one single step (simultaneous drawing) or in two successive steps (sequential drawing). From a commercial point of view, the sequential drawing of flat films is most important. Compared to the simultaneous process it offers higher production rates. The main advantages over the blown-film technology are resulting from the product quality (transparency

and thickness tolerances of the film) and from the flexibility of the manufacturing process.

The final properties of the film depend on structural changes, taking place in successive steps of the manufacturing process. After cast film production, the second step consists in an uniaxial drawing of the spherulitic film in the machine direction by rollers, operating at different peripheral velocities. It produces the so-called MDO film, that is denominated as "jumbo".

The changeovers occur when the product types switch, the operators usually group the orders with the same thickness together for the sake of convenience. The time, raw materials, equipments to be necessarily prepared for the next job depend on the preceding job. Hence, the setups and the cut formats are dependent on the processing sequence, and should increase the scrap generation due to poor cut patterns allocation. Since the total cost will vary from a different production sequence, and hence the changeovers, the decision maker should strive to use one or more than one sequences with a minimal scrap generation.

The product mix optimisation module generates several instances of a mix integer-programming (MIP) problem in order to maximise contribution margin of each item to the overall profit[198], simulating the scrap distribution for every instance and solving them using the grid. In the context of the company environment, every product mix sample needs an execution of a 50.000 MIP instances, amounting to over 16 CPU.days for a single study cycle.

Thanks to the EELA-2 grid infrastructure, the product mix model with stochastic optimisation problem can be solved in affordable time and company's management reduced the product mix problem from 354 to 28 products (326 with predetermined quantities and production lines). Moreover, company's production lines are now dedicated to a small number of products, decreasing the need of inter-run set-ups and reducing the material handling on shop floor and material logistics.