Universidade do Minho
Escola de Engenharia

Nuno Miguel Domingues Guerreiro

Previsão de disponibilidade em ambientes *Grid*

Novembro de 2008

Previsão de disponibilidade em ambientes *Grid*
Nuno Miguel Domingues Guerreiro
UMinho | 2008

Universidade do Minho
Escola de Engenharia

Nuno Miguel Domingues Guerreiro

Previsão de disponibilidade em ambientes *Grid*

Mestrado em Informática

Trabalho efectuado sob a orientação do
Professor Doutor Orlando Manuel de Oliveira Belo

Novembro de 2008

Para ti, por nunca me deixares desistir e por toda a compreensão...

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais, bem como a toda a minha família, pelo apoio, carinho e por terem tornado possível a minha frequência neste Mestrado. Uma palavra especial para a minha namorada Paula, pela imensa força que me deu durante este último ano, sendo sempre compreensiva, mesmo estando eu constantemente concentrado neste trabalho, nos meus tempos livres. Nunca me deixou pensar senão no cumprimento dos objectivos a que me propus, até nos momentos mais incertos.

Agradeço também aos meus amigos, especialmente aos que viveram mais de perto esta experiência, Vítor e Sandrine, e com os quais passei grande parte dos dias de trabalho, sempre com bom humor e espírito de entreajuda constante.

Este trabalho não seria possível sem a orientação do Professor Orlando Belo. Desde o início, na escolha do tema, até a esta fase final, sempre me ajudou a seguir um rumo claro e calmo. Ajudou-me principalmente a manter os "pés na terra" e a atenuar um dos meus defeitos: a tendência para complicar.

Não poderia deixar de referir a minha entidade empregadora, a *ALERT Life Sciences Computing S.A*, por ter permitido a frequência neste Mestrado, libertando-me durante o tempo necessário. Seria também injusto, se não agradecesse àqueles que trabalham directamente comigo, por compreenderem a minha ausência e por terem ajudado a garantir o cumprimento das entregas estipuladas.

Finalmente, gostaria de reconhecer a disponibilidade e o interesse da equipa responsável pelo desenvolvimento da ferramenta *Gridway*, e em especial a Jose Herrera, por me ter cedido material importante para a criação do meu caso de estudo.

Resumo

Previsão de disponibilidade em ambientes *Grid*

Em qualquer cenário de um mundo ávido de informação são recolhidas vastas quantidades de dados, a cada segundo que passa. Os requisitos de processamento, para extrair informação a partir desses dados, crescem também a uma velocidade incessante. Os ambientes *Grid* apresentam-se às organizações como uma alternativa apelativa aos dispendiosos recursos dedicados, e muitas vezes monolíticos, usados para processamento massivo de dados. Estes sistemas permitem o aproveitamento de recursos heterogéneos distribuídos geograficamente, potencialmente em parceria com outras organizações, sem que os mesmos necessitem de estar dedicados, permitindo reduzir os custos de implementação e aumentando a eficiência global do parque tecnológico de uma organização. Consegue-se, assim, também reduzir o lapso de tempo entre a recolha e a extracção de informação, permitindo decisões baseadas em informação cada vez mais recente. De forma a aproveitar melhor os recursos computacionais, torna-se obrigatório possuir mecanismos eficientes de escalonamento de tarefas em toda a *Grid*. Em particular, um escalonador *grid* deve ter em consideração não só o estado dos recursos e a sua disponibilidade, mas também informações de histórico para que possa prever quais os momentos mais propícios para o lançamento de tarefas, podendo assim optimizar o tempo de execução.

A aplicação de técnicas que permitam efectuar previsão de disponibilidade de uma *Grid* pode ajudar a aumentar a eficácia do processo de escalonamento. Neste contexto, o presente trabalho descreve a instalação, configuração e utilização de um ambiente *Grid,* recorrendo a várias ferramentas *Grid*. Para além disso, apresenta a implementação de um sistema de recolha e armazenamento de estado da *Grid* e das tarefas respectivas. A recolha de dados alimenta a

construção e aplicação de modelos de mineração de dados, com o propósito de fazer previsão de disponibilidade em ambientes *Grid*.

Palavras-chave: Mineração de dados, descoberta de conhecimento, algoritmos de previsão, computação distribuída, Grid, escalonamento de tarefas.

Abstract

Predicting availability on *Grid* environments

In a world thirsty for information, there are vast amounts of data being collected at each second. In the past, so as to extract information from those data sets, powerful computers would have to be acquired, whose performance would not be able to keep up with the increasing growth of collected data. Therefore, *Grid* environments have become an appealing alternative for organizations that seek knowledge within their data. These environments are replacing monolithic and prohibitly expensive super-computers on data processing and analysis. *Grids* enable the usage of geographically dispersed, heterogeneous, non-dedicated and relatively low cost equipments, potentially shared with partner organizations. By cutting costs and allowing any computer to be used for processing, *Grid* environments allow an organization to increase the efficiency of its computing equipments, while not requiring them to be fully dedicated. With the increased efficiency, the elapsed period of time between data gathering and knowledge extraction is also decreased, allowing decisions to be made based on up-to-date information.

In order to obtain higher return on investment, computational resources need to be fully exploited, by using state-of-the-art task scheduling throughout the *Grid*. Thus, *Grid* schedulers should also take historical status data into account, besides current resource status and availability data. As a result, these schedulers would be able to proactively respond to periods of low availability, rather than reactively, effectively optimizing execution times and resource usage.

Using data mining techniques to predict *Grid* availability can help increasing scheduling efficiency. In this context, the current work describes the installation, configuration and usage of a *Grid* environment, by adopting several *Grid* tools. Furthermore, it portrays the implementation of a

Grid status gathering daemon. Finally, the collected data is used to build and test data mining models, with the purpose of predicting availability on *Grid* environments.

Keywords: Data mining, knowledge discovery, prediction models, distributed computing, *Grid*, task scheduling.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

1.1 Computação *Grid*

Os avanços tecnológicos nas comunicações à escala global e na capacidade de armazenamento, entre outros, permitem aos sistemas actuais obter dados originados nas mais variadas fontes. Quer sejam adquiridos através de satélites meteorológicos, sistemas de venda, da banca, de infra-estruturas de telecomunicação ou mesmo de simulações científicas, esses dados são frequentemente armazenados em repositórios centrais de informação, ou *data warehouses*, para suportarem a tomada de decisões. A extracção, processamento e carregamento de dados para esses repositórios é muitas vezes um processo demorado e intensivo, com consumo de recursos poderosos.

Mesmo após a centralização de dados, ainda não é humanamente possível fazer uma análise de dados, que permita a recolha de informação, dado o volume de dados envolvidos. As técnicas de mineração apresentam-se como uma solução para a extracção sistemática de conhecimento não óbvio e implícito, a partir de grandes conjuntos de dados, disponibilizando informação preparada para análise humana.

O conhecimento extraído pode ser de importância vital para as organizações que o procuram, podendo auxiliar, por exemplo, no combate à fraude telefónica e bancária, no *scoring* de clientes na atribuição de empréstimos, nas previsões de venda, ou mesmo determinar quais os clientes com melhor resposta a uma campanha publicitária.

Dada a flexibilidade das técnicas de mineração de dados, tem-se verificado um crescente investimento, quer a nível empresarial quer académico. Contudo, os requisitos de processamento

têm também vindo a tornar-se mais exigentes, à medida que o volume de dados a processar se intensifica.

Como resposta às necessidades de capacidade de processamento, as organizações têm vindo adoptar várias alternativas, desde a aquisição de poderosos *datacenters* à utilização de equipamentos *mainstream*, agrupados em grande quantidade. Esta última é adoptada por empresas com o *Google*, por exemplo, que optam por adquirir equipamentos de relativo baixo custo, em grande número, e distribuindo-os geograficamente. Outra vantagem desta estratégia é a possibilidade de utilização de recursos não dedicados, mas que muitas vezes estão disponíveis em períodos não laborais (*e.g.* noites e fins de semana). Estes sistemas distribuídos seguem o paradigma *Grid*, que assenta, essencialmente, na utilização de recursos heterogéneos, não obrigatoriamente dedicados e distribuídos geograficamente (Figura 1).

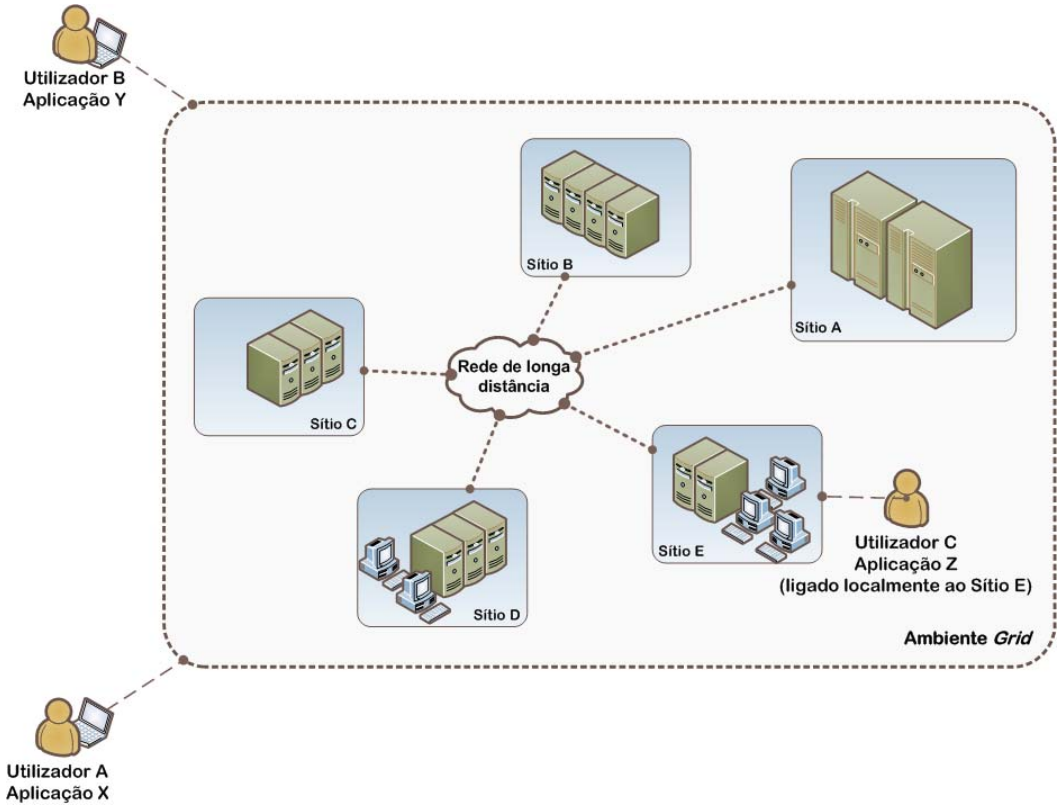

Figura 1 - Exemplo de um ambiente *Grid*, com múltiplos sítios e utilizadores dispersos geograficamente.

A informação do estado e do controlo de execução são também descentralizados, aumentando a escalabilidade e eliminando a dependência de um controlador central. Desta forma, a *Grid* opera independentemente das máquinas que estejam disponíveis, ao contrário de sistemas

centralizados que cessam a operação assim que o controlador central deixe de estar disponível. Para além disso, não há também uma entidade responsável por agregar todos os pedidos, bem como distribuir, monitorizar e assegurar a realização das tarefas.

Devido à inexistência de coordenação central, cada sítio é autónomo na sua gestão e recebe um conjunto de tarefas a executar, provenientes de outros sítios. Um sítio não é mais do que um conjunto de máquinas, sob a mesma autoridade e habitualmente geograficamente próximas. Por sítio, pode existir um ou mais coordenadores responsáveis, por exemplo, pelo escalonamento de tarefas e pela troca de informação com outros sítios. A um nível superior, existe um ou mais coordenadores que determinam qual ou quais os sítios a utilizar para a execução de tarefas. Esta organização hierárquica, com diferentes responsabilidades, faz com que a interoperabilidade entre coordenadores seja fulcral.

1.1.1 Desafios de implementação

A implementação de ambientes *Grid* traz novos desafios às organizações. A utilização de recursos heterogéneos introduz a necessidade do desenvolvimento de código portável, ou da adopção de estratégias de virtualização.

As redes utilizadas apresentam também características distintas (latência, largura de banda e disponibilidade), variando desde redes locais e dedicadas de alta velocidade até redes globais não dedicadas como a Internet. Os recursos de processamento utilizados ficam disponíveis de forma dinâmica, podendo nem pertencer à organização em causa e sendo disponibilizados por terceiros durante certos períodos de tempo, numa estratégia de aprovisionamento a pedido. Surge assim também a necessidade de estabelecer comunicações seguras e de garantir a identificação e autenticação dos intervenientes.

A adopção de ambientes *Grid* constitui também, por si, um desafio. A escolha dos diversos *softwares* de mediação, escalonamento e outras ferramentas não é trivial, para que seja garantida interoperabilidade e também o conjunto de funcionalidades requerido. Além disso, a instalação de toda a infra-estrutura é complexa, sendo por si só um projecto, até porque se trata de uma tecnologia relativamente recente, com grande margem de progressão no que diz respeito à maturidade que apresenta.

A utilização de recursos potencialmente não dedicados faz com que não seja trivial obter o mesmo tipo de garantias apresentadas por sistemas dedicados, no que diz respeito ao tempo de execução. Por exemplo, um retalhista que dependa da execução dos processos de *Extraction-Transformation-Loading (ETL)* num período de apenas 8 horas, consegue que esse período seja

respeitado ajustando os processos e disponibilizando os recursos necessários. Num ambiente *Grid*, a margem de erro é muito superior, uma vez que a disponibilidade de recursos é dinâmica e não estática.

Dado que os recursos num ambiente *Grid* não estão usualmente dedicados a uma única tarefa, a sua disponibilidade é muito variável, dependendo da utilização que os outros processos (não relacionados com a *Grid*) provoquem nas máquinas que se encontram à disposição. Para além disso, o conjunto de máquinas disponíveis para a execução de tarefas varia com o tempo.

Esta conjugação de factores faz com que a determinação da disponibilidade global de uma *Grid* seja complexa e que cada momento seja único, quer na quantidade de recursos disponíveis, quer na qualidade dos mesmos e percentagem de dedicação.

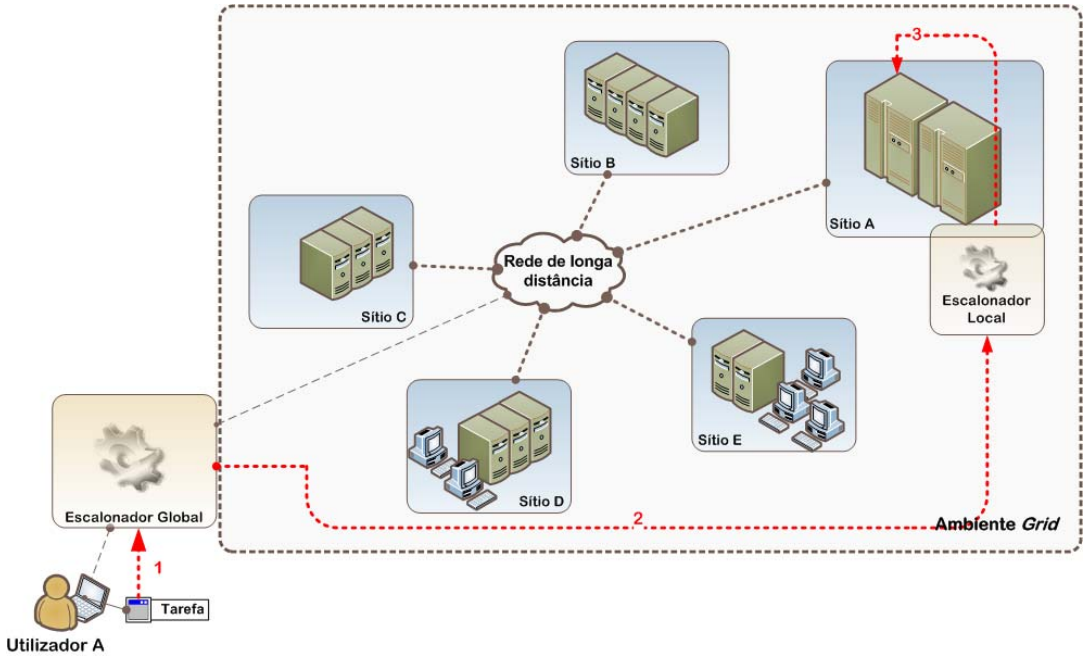

Figura 2 - Submissão de tarefas num ambiente *Grid.*

1.1.2 Escalonamento de tarefas

Devido à existência de diferentes níveis de responsabilidade (por um único sítio, por toda uma organização e multi-organização), existem também vários níveis abstracção. Um escalonador responsável pelas máquinas de um sítio tem acesso a todas as suas características e tem acesso directo a todas elas. Por outro lado, um escalonador que tenha acesso a recursos de várias

organizações tem uma visão mais global, considerando cada sítio ou conjunto de máquinas como que de uma única só máquina se tratasse.

Quando um escalonador multi-organização (ou global) determina que recursos utilizar, não tem possibilidade de escolher qual a máquina que a vai executar, mas apenas o sítio. A escolha da máquina cabe ao escalonador responsável pelo sítio. Desta forma, há uma delegação de responsabilidades: um escalonador global escolhe um sítio de execução e entrega a responsabilidade do escalonamento e seguimento de tarefas ao escalonador local (Figura 2).

A escolha dos sítios e máquinas para a execução de uma dada tarefa pode ter em consideração diversos factores, quer ao nível do hardware quer ao nível do estado de cada máquina. São então considerados parâmetros como número e arquitectura dos processadores, quantidade de memória disponível, carga da máquina, sistema operativo, número de tarefas em execução, número máximo de tarefas, entre outros. Alguns escalonadores permitem restringir o conjunto de máquinas a utilizar, tendo em conta, por exemplo, uma quantidade mínima de memória disponível ou a frequência de funcionamento dos processadores.

1.2 Mineração de dados como suporte ao escalonamento de tarefas

Apesar da sofisticação dos escalonadores actuais e dos algoritmos que disponibilizam, as decisões são ainda tomadas tendo em conta o estado actual das várias máquinas, os sítios disponíveis e as tarefas que devem ser executadas ou já estão em curso. Por não possuir a capacidade de efectuar previsões quanto aos períodos de utilização, os escalonadores podem apenas confiar nos dados presentes, fazendo uma gestão optimizada para o instante e não global, não sendo capazes de prever excesso ou escassez de disponibilidade.

Num exemplo prático, se um operador de telecomunicações móveis implementasse um ambiente *Grid* para processamento de mensagens curtas, não teria como se acautelar automaticamente para períodos críticos como o Natal ou Ano Novo. Caso houvesse a capacidade de prever a falta de disponibilidade do sistema para esses períodos, poderia ser feito o aprovisionamento automático, através do aluguer de recursos a terceiros.

Num outro cenário, se uma organização permitisse que todo o seu parque informático pudesse ser utilizado para execução de tarefas, interligando os todos os seus recursos numa *Grid,* incluindo os computadores de secretária dos seus colaboradores, seria de esperar que a disponibilidade da *Grid* fosse superior em períodos não laborais. Nesta situação, poder-se-ia atrasar ligeiramente a execução de algumas das tarefas mais intensivas, para poder lançá-las na

Grid, num momento mais favorável. Uma outra organização que faça da sua *Grid* a sua principal fonte de rendimentos, através do aluguer por determinados períodos de tempo, poderia também taxar os períodos com base na disponibilidade prevista, oferecendo aos seus clientes *premium* períodos de disponibilidade superior.

A recolha de dados relativos à disponibilidade de uma *Grid* permite que estes sejam posteriormente analisados, para que sejam tomadas decisões de escalonamento mais suportadas. Neste sentido, as técnicas de mineração de dados permitem diversas abordagens à análise dos dados supracitados, com vista ao estabelecimento de previsões para um dado período temporal. Os algoritmos actualmente disponíveis permitem análises com objectivos distintos, desde a criação árvores de decisão com base nos valores das variáveis tidas em conta à descoberta de agrupamentos ou *clusters*. Através da utilização destas técnicas e algoritmos, torna-se possível a descoberta de padrões não óbvios através da análise humana. No contexto apresentado, a descoberta de padrões de disponibilidade pode auxiliar um escalonador possibilitando a tomada decisões mais informadas, através da previsão do estado da *Grid* num determinado período de tempo.

1.3 Objectivos

Com a realização deste trabalho, pretende-se ir ao encontro dos seguintes objectivos:

- Saber fazer a montagem de um ambiente *Grid* protótipo, recorrendo às ferramentas e tecnologias actualmente disponíveis para o efeito.
- Utilizar o ambiente protótipo para a execução de tarefas arbitrárias.
- Determinar métricas relevantes para o cálculo de disponibilidade numa *Grid.*
- Formular um modelo para cálculo da disponibilidade duma *Grid*.
- Recolher periodicamente dados relativos à execução de tarefas e disponibilidade geral da *Grid*.
- Armazenar dados recolhidos numa estrutura intermédia (*e.g.* base de dados relacional) para posterior processamento.
- Extrair conhecimento sobre uma *Grid,* utilizando os dados recolhidos, através da utilização de ferramentas, técnicas e algoritmos de mineração de dados.
- Comparar os modelos de cálculo formulados para análise de representatividade dos mesmos.

- Apontar possibilidades de uso do conhecimento extraído, através de um mecanismo de realimentação para escalonadores *Grid* e também através de acção humana, com vista a um melhor aproveitamento dos recursos disponíveis.

1.4 Organização da tese

Além do presente capítulo, esta dissertação encontra-se organizada em quatro outros capítulos. No próximo capítulo é feita a apresentação dos conceitos mais relevantes relacionado com a computação em *Grid*, bem como da análise do estado da arte das ferramentas e tecnologias *Grid* actuais. No capítulo 3 é descrita a escolha efectuada ao nível do *software* e *hardware*, para criação de um ambiente *Grid* protótipo, bem como a escolha de tarefas a submeter e implementação dos mecanismos de recolha de dados. O capítulo 4 apresenta alguns dos algoritmos disponíveis bem como as ferramentas escolhidas. Apresenta também a definição dos modelos de mineração de dados e os resultados obtidos. Finalmente, no capítulo 5, é feita a análise dos resultados do trabalho realizado, apontando-se conclusões e abrindo perspectivas de trabalho futuro.

Capítulo 2

Ambientes *Grid*

2.1 Enquadramento histórico

O termo *Grid*, como foi descrito em [Kesselman & Foster 1998], surgiu em meados dos anos noventa para designar uma então proposta para a criação de uma infra-estrutura de computação distribuída, com o foco nas comunidades de ciência e engenharia. O interesse destas comunidades, em problemas cada vez mais complexos, resultou na utilização de recursos computacionais com elevado desempenho. Com o galopante aumento da capacidade de armazenamento, não acompanhado pela capacidade de processamento [Berman, F. et al. 2003], tornou-se inevitável contornar essa limitação, com vista a atingir o poder de processar os dados armazenados, revestindo-os assim da utilidade pela qual são recolhidos. As exigentes necessidades ao nível da computação e infra-estrutura, aliadas à vontade de cooperar com a comunidade, fizeram com que fossem estabelecidas redes de trabalho entre as várias organizações. Os avanços nas tecnologias de rede, incluindo a implantação de cablagens de fibra óptica, e o aparecimento da Internet, potenciaram a capacidade de troca de dados entre diversos locais afastados geograficamente.

Com a possibilidade de transportar elevadas quantidade de dados, tornou-se viável, e cada vez mais vantajoso, atribuir o processamento de um conjunto de dados a vários recursos remotos de relativo baixo custo, em detrimento de um recurso local altamente especializado e dispendioso. Assim, a criação de redes de processamento com alto desempenho, baseadas em *hardware* de relativo baixo custo, tornou-se mais acessível do que a aquisição de um super-computador.

Também as redes têm vindo a elevar a sua capacidade mais rapidamente do que os processadores [Berman, F. et al. 2003], perspectivando-se um futuro no qual a taxa de transferência e de armazenamento de dados deixem de ser limitações, sendo o processamento o único responsável pelo afunilamento do desempenho. Por outro lado, deixará de ser necessário transportar conjuntos de dados, previamente ao seu processamento, até ao recurso responsável pela computação. A rede será apenas um meio de alta velocidade para aceder ao armazenamento de dados.

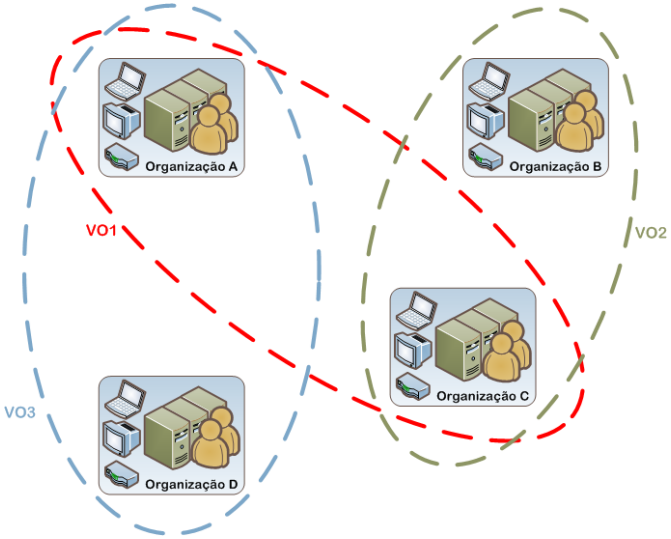

Figura 3 - Envolvimento de organizações na constituição de *VOs*.

2.2 Conceitos *Grid*

Genericamente, o termo *Grid* refere-se à partilha coordenada de recursos, num ambiente dinâmico, heterogéneo e multi-organizacional, com vista à resolução de problemas [Foster, I. et al. 2001]. Esta partilha não se refere simplesmente à troca de dados entre computadores, mas sim ao acesso directo aos mesmos, incluindo o *software*, dados e outros recursos, de forma colaborativa, com vista à resolução de problemas complexos.

Para que possa haver partilha entre os vários intervenientes, de múltiplas organizações, é necessária a definição de qual o papel de cada um, de consumidor ou fornecedor de recursos, de quais os recursos que podem ser utilizados e a quem é permitida a utilização. O conjunto de indivíduos e organizações, que aderem a essas regras de partilha, forma o que se denomina uma organização virtual [Foster, I. et al. 2001] ou simplesmente *VO*, do inglês *Virtual Organization*.

Uma organização pode ser membro de uma ou mais *VOs*, em cada momento (Figura 3) e pode disponibilizar apenas um ou todos os seus recursos.

As *VOs* variam na sua constituição, propósito, duração, organização, entre outras. No entanto, todas elas podem ter em comum um conjunto de requisitos e preocupações, relacionados com as tecnologias adoptadas. Para colmatar as necessidades das *VOs* são necessários mecanismos que sejam flexíveis ao ponto de permitir a partilha de recursos em diferentes formatos, como *cliente-servidor* ou *peer-to-peer*, e diversos tipos de aplicações, muitas vezes críticas e com necessidade de garantia de tolerância a faltas.

Cada *VO* pode agregar uma ou mais arquitecturas distintas de *hardware*, ou até várias gamas de equipamentos, desde vulgares computadores de secretária até a servidores de alto desempenho. Pode também juntar inúmeras instalações de *software* e definir as suas próprias políticas de utilização, segurança e monitorização. Assim, a interligação entre os diversos recursos e indivíduos associados numa *VO* implica a interoperabilidade entre diferentes fornecedores, permitindo assim a partilha entre recursos altamente heterogéneos e também, de certa forma, únicos na sua constituição.

Os ambientes *Grid* estão habitualmente relacionados com aplicações exigentes, complexas e com grande avidez de recursos. Estas podem ser executadas a partir de qualquer ponto geográfico, usando qualquer equipamento embebido na *Grid*. Não existe, assim, nenhum centro nevrálgico nem nenhuma coordenação central nestes ambientes. Pelo contrário, existe uma coordenação descentralizada, em que cada organização membro da *VO* disponibiliza um ou mais responsáveis pelo lançamento e seguimento de tarefas, servindo estes como *gateways* de acesso aos recursos de cada organização. Estes *gateways* são também responsáveis pela criação do ambiente de execução (cópia de bibliotecas, ficheiros e outros recursos necessários) e envio do resultado de cada tarefa para a origem da submissão. Para além disso, garantem também a repetição das submissões, em caso de falha de um dos nós da sua responsabilidade.

Como os ambientes são dinâmicos e não há uma responsabilidade global de uma única organização, a qualquer momento pode ser adicionada ou removida uma máquina, ou até toda uma *VO*, da *Grid*. Assim, não basta uma configuração estática, com uma listagem das máquinas disponíveis. Numa *Grid,* é feita a propagação de informação relativamente aos recursos disponíveis, através das várias organizações interligadas. Desta forma, cada *gateway* envolvido numa *VO* é responsável por anunciar periodicamente quais os recursos pelos quais é responsável, incluindo informação sobre *hardware* e *software*. Esta informação é propagada entre os vários *gateways*, estando assim descentralizada, chegando porém a todos os pontos da *Grid*. Porém, esta

propagação introduz alguns atrasos, fazendo com que os dados publicados nem sempre estejam completamente actualizados. O conceito é semelhante ao utilizado em sistemas como o *Domain Name Service (DNS)*, em que nenhuma máquina centraliza toda a informação, havendo uma organização hierárquica dos servidores e propagação de dados com algum atraso, sendo necessário o refrescamento periódico dos registos armazenados.

2.3 Requisitos estruturais

Para uma organização, as perspectivas de poder optimizar a utilização dos seus recursos e também ter acesso a aprovisionamento a pedido, entre outras, são aliciantes. No entanto, a utilização de ambientes *Grid* traz, por si só, um conjunto de requisitos que devem ser encarados tanto pelos fabricantes de *software* como pelas próprias organizações.

2.3.1 Segurança

A abertura inter-organizações e a utilização de redes não dedicadas como a Internet, aliadas à troca de dados sensíveis e acesso a recursos vitais, fazem com que as preocupações com segurança sejam uma constante. Nenhuma organização está disposta a trocar os seus dados de negócio em canal aberto ou a permitir que terceiros tenham acesso aos seus dados confidenciais. Por outro lado, não quer disponibilizar os seus recursos e o acesso a dados locais a entidades não autorizadas.

Pelo referido, é necessário o estabelecimento de regras e políticas de utilização de recursos, aplicáveis aos indivíduos envolvidos na *VO*. Em particular, é necessário ter em conta aspectos como a certificação, definição de grupos e autorização. Tem então de haver um mapeamento entre as políticas aplicáveis aos participantes na *VO* e as políticas nas organizações às quais os participantes pertencem [Welch, V. et al. 2003]. Tal deve-se ao facto de as próprias organizações já terem habitualmente grandes investimentos em segurança e não quererem abdicar dos mecanismos já implementados.

Mapeamento de políticas de segurança

As políticas de segurança em vigor nas organizações físicas centram-se apenas nos seus utilizadores e grupos de utilizadores locais [Welch, V. et al. 2003]. O acesso dos participantes de uma *VO* tem, desta forma, de ser conseguido através das relações de confiança entre organização física e os seus utilizadores, e também entre estes e as *VOs* em que participam. Logo, para que um qualquer participante tenha acesso a um recurso de uma organização numa *VO*, é necessário que

haja pelo menos um utilizador na organização física que também pertença à *VO*. As políticas de utilização do recurso local são assim herdadas pela *VO*, criando o que é conhecido por domínio de confiança, que funciona como uma camada agregadora de políticas de segurança comuns a todos os participantes. Passa assim a *VO*, através dos seus serviços de segurança, a ser responsável pela atribuição de privilégios aos participantes, sobre os recursos das diversas organizações presentes.

A transposição das políticas de seguranças locais para a *VO* é feita através de adaptadores. Por exemplo, se uma organização utiliza políticas de segurança baseadas em *Kerberos*, então a infra-estrutura *Grid* tem de providenciar um adaptador específico para a interligação com *Kerberos*.

Atribuição de credenciais

Dada a natureza dinâmica dos ambientes *Grid*, para que os seus utilizadores tenham acesso aos recursos, não é possível que os administradores de sistemas atribuam manualmente privilégios de acesso aos recursos da *VO*. Nestes cenários dinâmicos, a autorização de acesso tem de ser feita automaticamente, sem intervenção humana. Para tal, são geradas credenciais após autenticação.

As credenciais atribuídas a um utilizador permitem que este aceda aos recursos na sua e noutras organizações participantes na *VO*, durante a validade da credencial. Torna-se assim possível suportar *VO* de duração limitada, através da limitação da validade das credenciais.

Autenticação única

Pelo facto de serem utilizados sistemas e recursos de terceiros, os utilizadores dos ambientes *Grid* têm de ser identificados de forma transparente em qualquer recurso da *Grid*, sem ser necessário proceder à autenticação em cada recurso a que têm acesso. Doutra forma, a própria utilização da *Grid* torna-se impraticável para os seus utilizadores, funcionando como um sério obstáculo à implementação. Para tal, as credenciais são propagadas por toda a *VO*, acompanhando os pedidos dos seus utilizadores.

Como as infra-estruturas locais não têm informação acerca de todos os utilizadores da *Grid*, dentro da *VO*, é estabelecida uma delegação entre os vários utilizadores, isto é, um utilizador local delega nos utilizadores da *VO* a possibilidade de acederem aos recursos locais, utilizando os mesmos níveis de privilégios pertencentes ao utilizador local. Esta delegação é feita através da criação de *proxies*, que demonstram ao sistema remoto que um utilizador remoto tem acesso aos recursos.

2.3.2 Transferência de dados

Um outro desafio para a implementação e utilização de ambientes *Grid* é a dispersão geográfica dos recursos. Uma vez que o processamento não se encontra concentrado num único local, os dados a processar têm de ser transportados até ao recurso responsável, para execução das tarefas. Desta forma, o mesmo conjunto de dados pode até estar replicado por diversos locais, conforme as necessidades das aplicações desenvolvidas e para evitar latência na transferência de ficheiros. Por outro lado, a produção de dados pode passar a ser distribuída geograficamente, sem a necessidade de centralização, como requisito para o processamento.

Para a execução de aplicações que dependam da transferência de dados, pode ser necessário criar uma réplica local desses mesmos dados, ou até aceder a uma réplica próxima, para garantir um desempenho superior [Allcock, B. et al. 2002]. No entanto, esta complexidade deve estar escondida da aplicação, sendo os ficheiros necessários referenciados através de um endereço único, como por exemplo um *Uniform Resource Locator (URL)*.

Os serviços de transferência de dados devem também incluir uma componente de segurança, garantindo que apenas os utilizadores autenticados e com os privilégios correctos têm acesso aos dados. Por fim, os serviços de transferência de dados na *Grid* têm de garantir transferências com fiabilidade e integridade de dados, caso contrário o processamento em recursos remotos torna-se imprevisível e sujeito a erros.

2.3.3 Coordenação distribuída

Como não existe uma entidade centralizadora de controlo de execução, lançar tarefas para recursos remotos e ter garantias de que serão realmente executadas torna-se um desafio. Tal deve-se ao facto de não haver, em momento algum, conhecimento global acerca da *Grid*, no que diz respeito às tarefas submetidas, canceladas, executadas, entre outras. Os vários coordenadores locais e *gateways* precisam de interagir, trocando informações e responsabilidade acerca das tarefas.

Para o efeito, são necessários protocolos de comunicação para negociar o acesso aos recursos, especificando informações relativas às necessidades para a execução (*e.g.* quantidade de memória) e também para comunicação de operações como criação, destruição e seguimento de tarefas em execução [Foster, I. et al. 2001]. Também a monitorização de tarefas é feita recorrendo ao mesmo tipo de mecanismos, sendo assim possível seguir tarefas (em curso, em execução, etc), podendo uma tarefa ser relançada, caso a sua execução falhe num recurso

remoto. Assim, é possível garantir a execução de tarefas em recursos remotos, nem que para isso seja necessário submeter a mesma tarefa várias vezes até que um recurso a consiga levar a cabo.

Os protocolos de comunicação utilizados têm também de ter suporte para as políticas de segurança definidas pela *VO*. Só desta forma é possível garantir que apenas os membros autorizados da *VO* têm acesso a recursos geridos no âmbito da mesma.

2.3.4 Serviços de informação

A propagação de informação relativa aos recursos disponíveis num ambiente *Grid* constitui um requisito estrutural fundamental. Para que cada coordenador/escalonador global seja capaz de remeter a execução das tarefas da sua responsabilidade para recursos remotos, tem obrigatoriamente de conhecer as características desses recursos. Caso contrário torna-se impossível garantir qualidade de serviço, assim como requisitos mínimos para a execução das tarefas.

Como alternativa, toda a informação teria de ser estática e estar armazenada em todos os nós da *Grid*, o que não se adequa à dinamicidade das *Grids*, onde os recursos são disponibilizados por períodos que podem ir da curta à longa duração.

São vários os tipos de informação que têm de ser propagados a toda a largura de um ambiente *Grid*, fazendo com que sejam necessários serviços de informação capazes de transmitir e agregar dados relativos a todo o estado do ambiente [Czajkowski, K. et al. 2001]. Só desta forma é possível suportar ambientes dinâmicos e heterogêneos.

Tipos de serviço de informação

Em primeiro lugar, são necessários serviços de descoberta e abandono de recursos. Estes serviços injectam informação na rede, alertando para o facto de haver recursos a juntar-se ou a abandonar o ambiente *Grid.* A informação deve incluir características de *hardware*, bem como de *software* instalado. Por exemplo, os serviços de informação podem divulgar que mais um *cluster* de 10 máquinas se juntou à *Grid*, incluindo informação acerca da arquitectura dos processadores, memória, características de rede, bem como sistema operativo e *software* disponível, entre outras. Assim, os escalonadores recebem informação do estado da *Grid*, ao nível dos recursos, que lhes permite escolher o melhor recurso para efectuar uma tarefa.

No que diz respeito à transferência e replicação de dados, são também necessários serviços de informação. Só assim é possível que o sistema de gestão de réplicas consiga escolher qual a

melhor réplica a ser usada como *input* para a tarefa a executar. A informação trocada pode incluir informação acerca da performance de rede, acesso a disco, entre outras.

Por fim, e também segundo [Czajkowski, K. et al. 2001], é através do uso de serviços de informação que se torna possível a adaptação da execução de tarefas ao estado dinâmico da *Grid*. Para que um escalonador possa optar por abortar uma tarefa num recurso e transferi-la para um segundo recurso, necessita de ter informação acerca de ambos, para poder tomar essa decisão. Só desta forma se torna possível aumentar o desempenho na execução de tarefas, ao longo do tempo.

Propagação de informação entre máquinas

Dada a dispersão geográfica presente nos ambientes *Grid*, a informação propagada pelos produtores sofre inevitavelmente de algum atraso, podendo em casos extremos chegar já desactualizada aos consumidores, devido também ao alto dinamismo dos ambientes. Também por razões de escalabilidade e ocupação de largura de banda, o despejo constante de informação detalhada não pode ser considerado como opção.

Com o intuito de não prejudicar a escalabilidade, pode ser utilizada uma alternativa menos intensiva. Numa *VO* podem ser configurados um ou mais agregadores de informação que disponibilizam periodicamente informações resumidas e não detalhadas acerca do estado global da *Grid*. Desta forma, a quantidade de informação disponibilizada não se torna problemática com o crescimento do número de membros da *VO*, até porque os agregadores podem estar organizados de forma hierárquica.

Sempre que seja necessária informação mais detalhada acerca do estado de um recurso ou conjunto de recursos, é contactado o serviço de informação responsável pela publicação de informação da organização em causa. Torna-se assim possível obter informação detalhada e mais actualizada, a pedido, sem inundar de pedidos os serviços agregadores de informação. Como se pode ver na Figura 4, um escalonador global pode contactar um serviço agregador, de forma a obter informação global acerca dos recursos da *Grid*. O serviço agregador pode depois responder com alguma informação menos detalhada, mas indicando onde é possível encontrar informação detalhada. Após recepção dessa informação, o escalonador contacta um novo serviço de informação, obtendo informações detalhadas e que permitem tomar decisões.

Cabe a cada escalonador global decidir quão frequentemente necessita de obter informação acerca do estado da *Grid* e dos vários conjuntos de recursos federados (em cada uma das organizações físicas). Por outro lado, podem ser definidas políticas de acesso à informação

[Czajkowski, K. et al. 2001], por parte das *VOs*, para impedir a proliferação de pedidos. Portanto, estes mecanismos podem ser utilizados para prevenir situações abusivas.

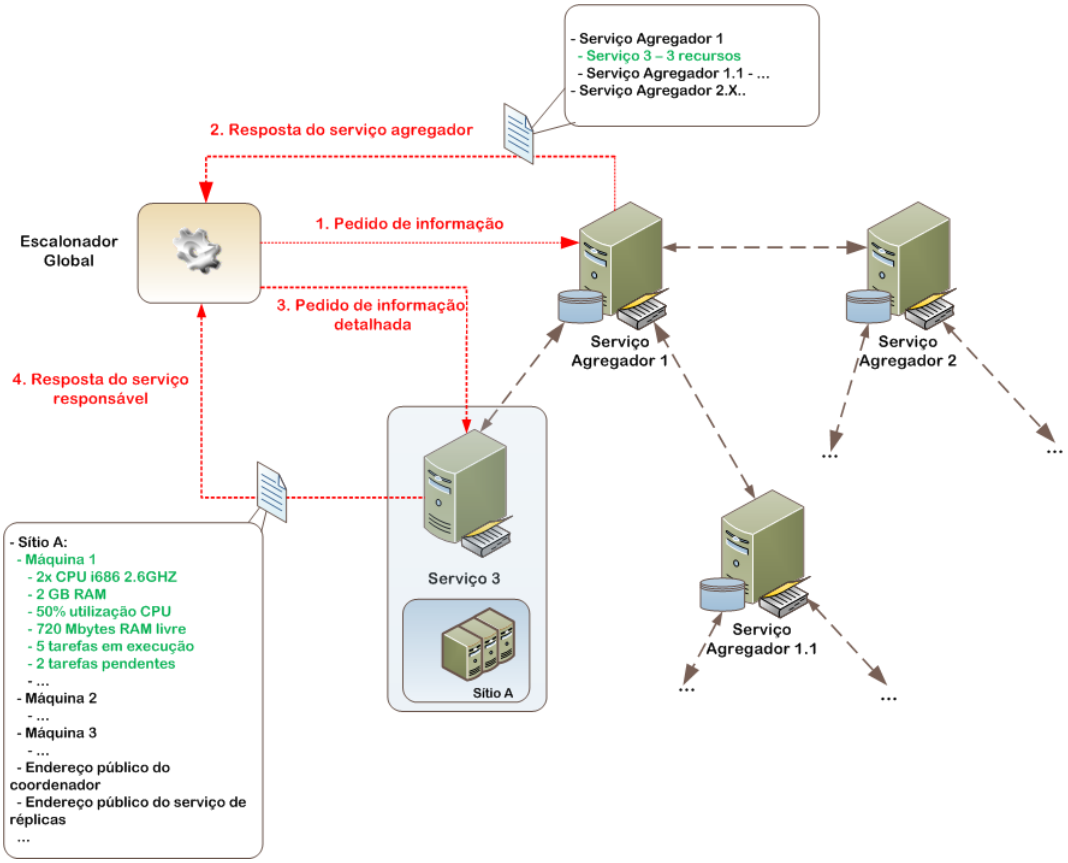

Figura 4 - Propagação de informação num ambiente *Grid*.

2.3.5 Reserva antecipada de recursos

Um outro requisito estrutural desejável num ambiente *Grid* define-se pela possibilidade de reservar antecipadamente recursos. Se é expectável que uma tarefa necessite de um grande conjunto de recursos, ou de recursos especialmente potentes, pode então proceder-se à reserva desses recursos, permitindo a utilização dedicada dos mesmos, dentro do ambiente *Grid*.

Para pôr em prática as funcionalidades de reserva antecipada, é necessário que os vários coordenadores de recursos comuniquem entre si, estabelecendo de reservas. Têm também de ser capazes de armazenar reservas e disponibilizar os recursos necessários para o tempo esperado. Desta forma, quando a tarefa responsável pela reserva começa a executar, os seus recursos estão disponíveis, aumentando assim também a qualidade de serviço.

Do ponto de vista da interoperabilidade, os vários coordenadores têm de ser capazes de reservar e utilizar recursos utilizando outros *softwares*. Consequentemente, como é feito para garantir a interoperabildiade de muitos outros serviços em ambientes *Grid*, são usualmente adoptada estratégias de padronização e implementação de *Web-Services*, por exemplo.

2.3.6 Serviços multi-domínio

Devido às preocupações com segurança de rede, as organizações optam pela instalação de sistemas de controlo de acesso, como por exemplo *firewalls*. A presença destes elementos constitui uma dificuldade adicional, uma vez que a comunicação entre sistemas de várias organizações é imperativa em ambientes *Grid*. Adicionalmente, os recursos estão muitas vezes apenas acessíveis em redes privadas, vedando o acesso ao exterior. Para ultrapassar as dificuldades referidas, têm de existir mecanismos de comunicação que consigam transmitir dados através do previsível emaranhado de redes e mecanismos de controle de acesso.

Aproveitando as vantagens apresentadas pelos *Web-Services* ao nível da interoperabilidade, as ferramentas e *middlewares Grid* podem também beneficiar da permeabilidade que é permitida ao tráfego *web*. Este tipo de tráfego é tendencialmente permitido por omissão nas organizações, não sendo assim necessário estabelecer novas políticas de segurança ou alterar as existentes, para alcançar a comunicação entre serviços e clientes em domínios distintos.

2.4 Estado da arte

Os ambientes *Grid* são compostos por uma miscelânea de pacotes de *software*, responsáveis por efectuar diversas funções, desde gestão de recursos, transferência de ficheiros, escalonamento ou monitorização. Tradicionalmente, os sítios *Grid* são organizados em *clusters* de máquinas, geridos por *software* dedicado. Para permitir a exposição desses sítios em ambientes *Grid*, são utilizados *middlewares*, responsáveis por responder aos requisitos estruturais já referidos. Os próprios *middlewares* são uma composição de vários componentes, potencialmente de fornecedores distintos. Para garantir a interoperabilidade entre estes diferentes *softwares*, são necessários *standards*, que apontam uma visão arquitectural e de comunicação comum, sem muitas vezes tecerem considerações acerca dos detalhes de implementação.

Os escalonadores globais baseiam-se nos serviços fornecidos pelos *middlewares Grid*, levando a cabo a escolha de recursos, atribuição de tarefas e monitorização, entre outras. Estes escalonadores podem ser utilizados através de portais, facilitando assim a utilização por utilizadores não técnicos.

De seguida são apresentadas as iniciativas mais relevantes em cada uma das áreas referidas.

2.4.1 Gestão de *clusters*

Os sistemas de gestão de clusters, também conhecidos por escalonadores *batch*, são os componentes de *software* mais maduros presentes em ambientes *Grid*. Alguns alcançam já as duas décadas desde a altura em que as primeiras versões foram criadas. Os projectos mais importantes nesta área são: *Condor, Portable Batch System, Sun Grid Engine e Load Sharing Facility.*

Condor

Tendo surgido em 1988, na Universidade de *Winconsin-Madison*, o projecto *Condor* [W2] tem como objectivo disponibilizar ferramentas e *frameworks* que suportem a computação de elevado desempenho, utilizando recursos distribuídos. Em particular, este projecto é o responsável pelo desenvolvimento do escalonador e gestor de clusters *Condor.* Este *software* é distribuído de forma gratuita.

Como qualquer escalonador, o *Condor* disponibiliza uma fila de tarefas, para que os utilizadores submetam as suas tarefas. Com base nas tarefas submetidas, nas políticas definidas, e nos recursos configurados, o *Condor* escolhe a máquina que irá executar cada uma das tarefas, sendo responsável pelo seguimento das mesmas e por entregar os resultados aos utilizadores finais. Numa instalação *Condor*, o escalonamento e gestão de recursos são feitos a dois níveis [Litzkow, M. et al. 1988]. Existem assim duas figuras distintas: *coordenador central* e *escalonador local*.

O papel de coordenador central é desempenhado por apenas uma das máquinas, por acumulação com o papel de escalonador local, podendo ser atribuído a uma qualquer outra máquina a qualquer instante. O coordenador é responsável por agregar informação acerca das várias máquinas à sua responsabilidade, incluindo dados relativos ao número de processadores, carga da máquina, número de tarefas em fila, entre outros. Para além disso, cabe ao coordenador atribuir a um escalonador local a possibilidade de utilizar os recursos disponíveis nas várias máquinas. Desta forma, o coordenador não escolhe qual as máquinas a serem utilizadas, apenas reserva uma percentagem da capacidade total para uso por parte de um escalonador local.

O escalonador local é responsável por disponibilizar informação ao coordenador e por levar a cabo a escolha dos recursos e executar as tarefas nesses mesmos recursos. Periodicamente, a pedido do coordenador, o escalonador local devolve os dados relativos à sua máquina, para que os outros escalonadores tenham conhecimento das suas capacidades. Sempre que é submetida uma

tarefa ao escalonador local, este contacta o coordenador para obter recursos a utilizar. Após execução, cabe ao escalonador recolher informação e entregá-la ao utilizador (Figura 5).

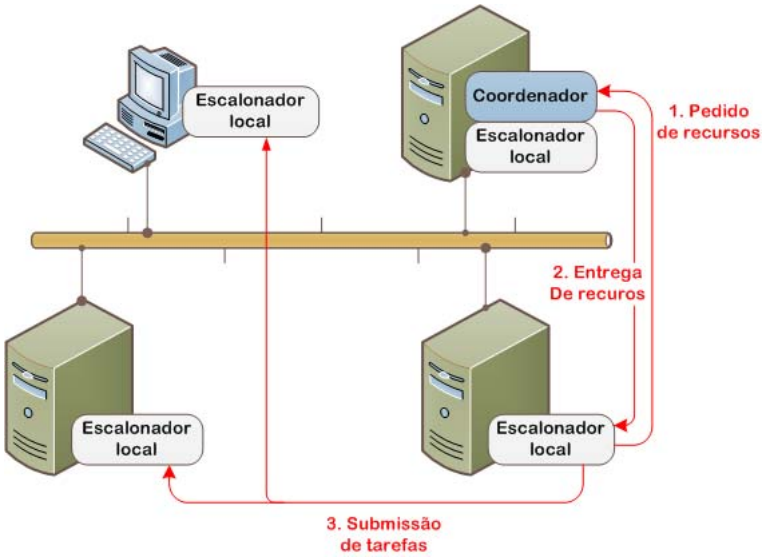

Figura 5 - Submissão de tarefas utilizando *Condor*.

De forma a optimizar a utilização de recursos, o escalonador local do *Condor* monitoriza a actividade da sua máquina. Caso esta esteja a ser utilizada para execução de tarefas submetidas por escalonadores remotos e seja detectada actividade por parte do utilizador (*e.g.* movimentos do rato, teclado, etc.) a tarefa é abortada, obrigando o escalonador remoto a submetê-la novamente. Este mecanismo permite ao *Condor* aproveitar recursos em computadores de secretária ou servidores, sempre que estes não estejam em utilização.

Para além do suporte para dependências entre tarefas, através da construção de grafos acíclicos direccionados, o *Condor* permite também a utilização de recursos noutros domínios administrativos, recorrendo ao *flocking*. Desta forma, sempre que um coordenador esgotar os seus recursos, pode entregar tarefas a coordenadores de outros domínios, para que as mesmas sejam executadas remotamente.

No que diz respeito às plataformas suportadas, o *Condor* pode ser instalado actualmente tanto em ambientes *Windows* como em ambientes *Unix/Linux.*

Portable Batch System

Em 1995, a *NASA* decidiu desenvolver o seu próprio sistema para suportar a execução de tarefas potencialmente paralelas e de processamento massivo em *clusters*. O *Portable Batch*

System (PBS) surgiu como resultado dessa iniciativa, tendo como principal preocupação a possibilidade de definir políticas de escalonamento mais flexíveis [Henderson, R. 1995]**.** Para tal, o *PBS* foi desenvolvido de forma modular, podendo utilizar módulos externos de escalonamento com suporte para diversas políticas de escalonamento. Ao permitir a integração com escalonadores externos, o *PBS* é um sistema capaz de interagir com outros sistemas de escalonamento, numa panorâmica mais alargada, como é a dos ambientes *Grid*.

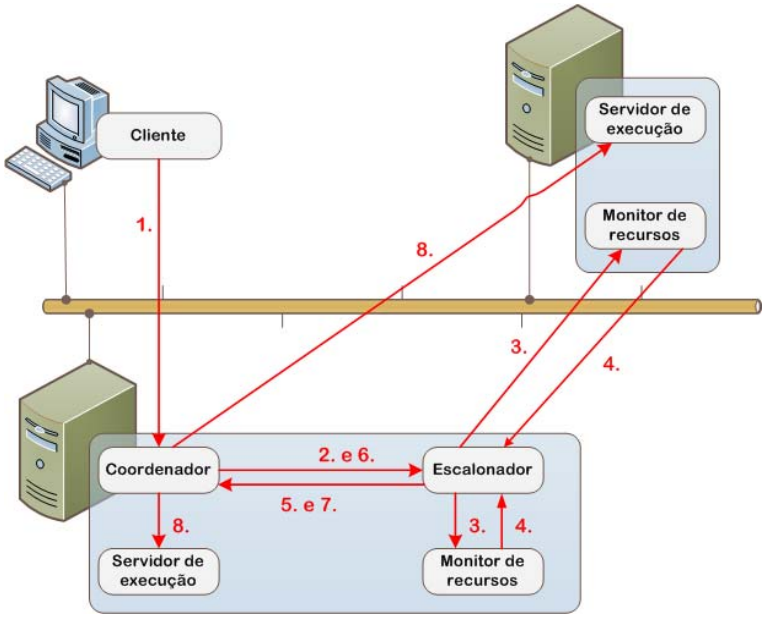

Figura 6 - Submissão de tarefas utilizando *PBS.*

Para além das ferramentas comuns de submissão, modificação e monitorização de tarefas, o *PBS* é composto por quatro módulos distintos: coordenador (um por *cluster*), monitor de recursos (um por máquina), servidor de execução de tarefas[1] (um por máquina) e escalonador (um por *cluster*). A interacção entre os diferentes módulos pode ser descrita pela sequência presente na Figura 6. Numa primeira fase, o cliente submete uma tarefa ao coordenador do *cluster*. Seguidamente, o escalonador é notificado, indicando a existência de uma nova tarefa (passo 2). O escalonador procede então à obtenção de informação relativa aos recursos disponíveis, contactando o monitor de recursos em cada uma das máquinas do *cluster* (passos 3 e 4). Após receber a informação desejada, o escalonador contacta o coordenador para receber informações adicionais acerca da tarefa, nomeadamente quais os requisitos para a sua execução (passos 5 e

[1] O servidor de execução de tarefas é geralmente denominado por *MOM*, por analogia com a figura materna ("*mom"* em inglês), uma vez que é o responsável pela criação e seguimento de tarefas.

6). Com todos os dados necessários para tomar uma decisão, o escalonador escolhe qual ou quais as máquinas que devem ser utilizadas para a execução da tarefa e notifica o coordenador (passo 7). Finalmente, o coordenador contacta os servidores de execução para que estes executem a tarefa nas suas máquinas locais (passo 8).

Embora o fluxo de submissão e execução de tarefas envolva trocas frequentes de informação entre vários intervenientes, o *PBS* consegue garantir elevada disponibilidade, sendo utilizado no super-computador mais potente da actualidade, o *IBM RoadRunner* [Bader, D. 1999] [Gilfeather & Kovatch 2000], com mais de 100 mil *cores* nos seus processadores [W19].

No que diz respeito à disponibilidade de versões do *PBS,* existem actualmente três variantes. A primeira, o *OpenPBS* [W3], é uma versão *open source* descontinuada, resultado da iniciativa original. Como evoluções desta, surgiram o *Torque* [W4] e o *PBS Professional* [W5]. O *Torque* mantém-se uma versão gratuita, com suporte comercial. Quanto às plataformas suportadas, apenas o *PBS Professional* pode ser instalado em ambientes *Windows*, todos os restantes requerem instalação em sistemas operativos *Unix/Linux*.

Como escalonadores externos, estão disponíveis o *Maui* [W4] e o *Moab Cluster Suite* [W4], para além da integração com *middlewares Grid.* O *Maui* pode ser obtido gratuitamente e suporta diversas políticas de escalonamento, atribuição dinâmica de prioridades, reserva antecipada de recursos, entre outras funcionalidades. O *Moab Workload* é baseado no *Maui,* incluindo outras valências como, por exemplo, ferramentas gráficas de administração, portal *Web* para utilizadores finais e suporte para *clusters* virtuais privados. Desta forma, é possível dividir um *cluster* físico em vários *clusters* lógicos, com políticas e gestão distintas.

Sun Grid Engine

Em 2000, a *Sun Microsystems* adquiriu a *Gridware* e criou o *Sun Grid Engine (SGE)* [W16], tendo por base o *CODINE (COmputing in DIstributed Networked Environments)*. O *SGE* é utilizado em *clusters* para submeter, escalonar e gerir a execução remota de tarefas de forma sequencial, paralela e até interactiva.

Para além da gestão de recursos ao nível de *hardware* (*e.g.* processador, memória e disco), o *SGE* permite também a gestão de licenças de *software*, para aplicações cujas licenças sejam controladas através de *FLEXlm* ou *LicenseJuggler*. Adicionalmente, como forma de aumentar a qualidade de serviço, suporta a reserva prévia de recursos para execução de tarefas.

Seguindo um modelo semelhante ao do *PBS*, o *SGE* é constituído por quatro módulos distintos (Figura 7): coordenador (um por *cluster*), escalonador de tarefas (um por *cluster*), servidor de execução de tarefas (um por máquina) e módulo de comunicação (um por máquina).

Opcionalmente, e para permitir elevada disponibilidade, pode ser instalado um módulo coordenador em mais do que uma máquina, ficando esse módulo dormente e sendo apenas activado no caso de o coordenador principal falhar.

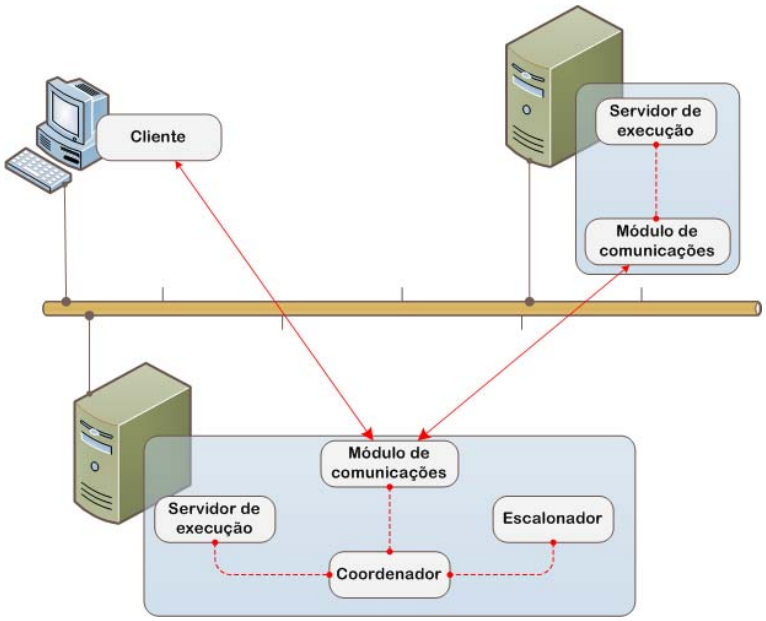

Figura 7 - Apresentação dos vários módulos do *SGE*.

O coordenador é responsável por manter informação acerca das máquinas e respectiva carga, filas de tarefas, e permissões dos utilizadores. Cabe ao coordenador receber as decisões do módulo escalonador e efectuar pedidos aos vários servidores de execução, para que estes executem as tarefas em espera nas filas. Cada uma destas filas pode ter recursos próprios alocados e ter políticas de utilização distintas, de forma a assegurar o cumprimento global dos requisitos de qualidade de serviço.

O servidor de execução tem duas responsabilidades distintas. A primeira é a de receber e executar tarefas na máquina local. Fica também a cargo da monitorização e manutenção de tarefas. A outra responsabilidade deste módulo é a de comunicar periodicamente ao coordenador o estado das tarefas pelas quais é responsável, bem como a carga da máquina onde está instalado.

Após receber informação entregue pelos vários servidores de execução, o coordenador entrega-a ao escalonador, mantendo este uma vista global actualizada de todo o cluster, o que lhe permite tomar as decisões de escalonamento. Estas decisões passam por determinar qual a fila a utilizar para cada tarefa e qual o servidor de execução que irá ser contactado. Todas as comunicações entre máquinas são centralizadas num único módulo de comunicação, não havendo comunicação de rede entre quaisquer outros módulos.

Quanto à implantação em sistemas de produção, o *SGE* tem vindo a ser utilizado em sítios de indústrias com várias centenas de servidores [Gentzsch, W. 2001] e também em projectos de larga escala da academia, como por exemplo o *Ranger* [W20]. Este é um dos mais potentes super-computadores da actualidade, com cerca de 400 servidores e mais de 100 mil *gigabytes* de memória *RAM*.

No que toca à disponibilização, o *SGE* é distribuído gratuitamente, sendo possível instalá-lo em sistemas operativos *Unix/Linux* e *MacOS.* Para além deste produto, a *Sun* distribui também o *N1 Grid Engine*, uma versão comercial do *SGE,* que pode ser instalada em sistemas operativos *Windows* (apenas os serviços de comunicação e execução são suportados) e disponibiliza funcionalidades de geração de relatórios de utilização, contabilidade e monitorização avançada [W16].

Load Sharing Facility

O gestor de recursos *Load Sharing Facility (LSF)*, actualmente comercializado pela *Platform Computing* [W6], teve a sua origem no projecto *UTOPIA* da Universidade de Toronto. Este projecto tinha como linhas gerais a disponibilização de um sistema de uso generalizado, onde pudessem ser executadas aplicações com ligeiras ou nenhumas adaptações, com execução remota transparente, podendo esta ser alterada de forma dinâmica, conforme a carga das máquinas. Foram também tidos em conta princípios como a escalabilidade, robustez e eficiência. Como resultado, o *LSF* é um dos sistemas comerciais com maior implantação, tendo dado origem a uma suite de produtos que inclui sistemas de geração de relatórios e ferramentas analíticas, para optimizar os recursos disponíveis.

A arquitectura original do *UTOPIA* [Zhou, S. et al. 1992] incluía a definição de apenas dois módulos presentes em todas as máquinas do *cluster*: servidor de execução e gestor de informação de carga. O gestor de informação de carga trocava informação de carga com os seus pares, relativa à máquina onde estava instalado. Para além disso, tomavas as decisões acerca da escolha da máquina utilizada para executar cada tarefa. No entanto, as aplicações tinham a possibilidade de forçar a escolha de uma determinada máquina, anulando a decisão de escalonamento. O servidor de execução permitia a execução transparente de aplicações, isto é, criava um processo local e um processo remoto para cada tarefa, sendo que nenhum dos processos tinha noção da deslocalização do outro. O servidor de execução funcionava assim como um intermediário entre processos.

A troca de informação entre gestores podia ser feita a pedido por um deles, contactando assim os seus pares para tomar uma decisão. Para além disso, podia ser utilizada uma estratégica de troca periódica de informação entre gestores. Neste processo, era feita a eleição de um gestor principal. Era da responsabilidade deste gestor agregar a informação de carga obtida por contacto com todos os restantes. A propagação da informação agregada podia ser depois feita periodicamente para todos os gestores. Em alternativa, caberia ao gestor principal tomar todas as decisões de escalonamento.

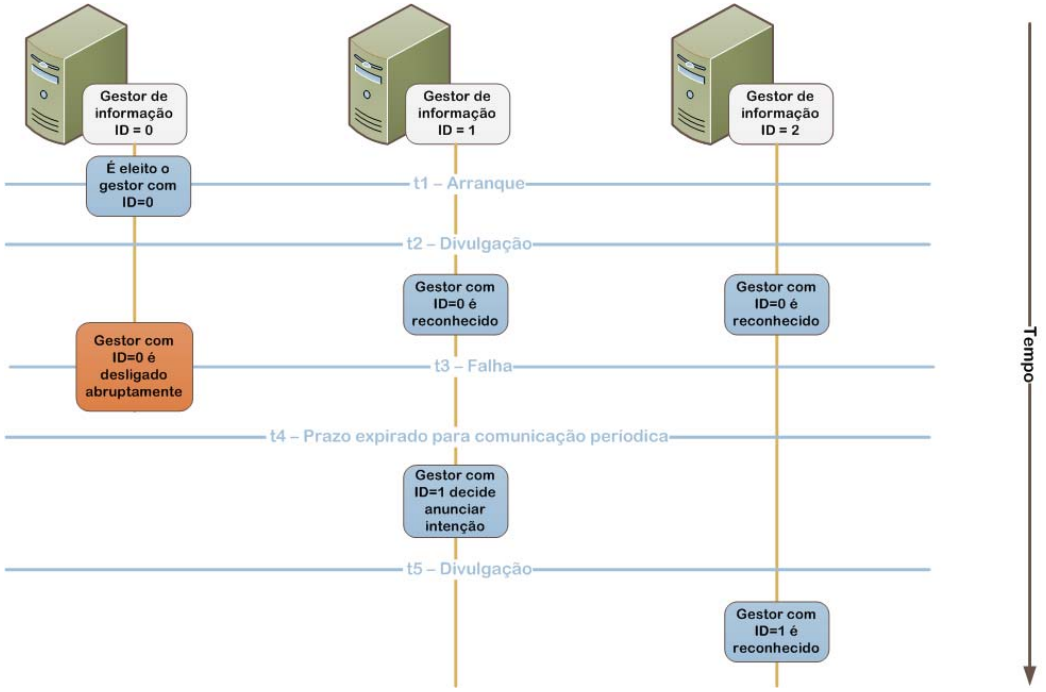

Figura 8 - Eleição de gestores principais de informação utilizando *LSF*.

O processo de eleição era baseado na atribuição de identificadores únicos a cada gestor. No arranque, o gestor com identificador zero seria automaticamente eleito principal. Os restantes gestores ficariam à espera de receber uma notificação. Caso não a recebessem, entrariam num período transitório com duração proporcional ao identificador possuído. Findo esse período, anunciaram a sua intenção de se tornarem no gestor principal e seria eleito aquele que tivesse o identificador mais baixo. O mesmo processo seria repetido caso o gestor principal se encontrasse incontactável durante um período máximo de tempo. Um exemplo deste processo de eleição pode ser observado na Figura 8. Neste caso, após uma eleição inicial, é encontrado um problema com o

gestor eleito e é iniciado um novo processo de eleição, sendo o segundo gestor com identificador mais baixo o novo eleito.

Para contornar o problema da falta de escalabilidade associada a um controlo centralizado, o *UTOPIA* permitia a criação de *clusters* virtuais. Desta forma era possível agrupar máquinas em vários *clusters,* sendo assim possível eleger vários gestores principais em simultâneo. Adicionalmente, o *UTOPIA* permitia a interacção entre *clusters*, virtuais ou não, sendo possível trocar tarefas entre *clusters*. Para além dos módulos providenciados, o *UTOPIA* era também acompanhado por um conjunto de bibliotecas que permitiam embutir suporte para execução remota nas aplicações desenvolvidas. Era desta forma que se tornava possível ignorar as decisões do escalonador, tendo a aplicação poder total.

O *LSF* continua a utilizar os conceitos já referidos, complementando-os através do suporte para a transferência automática de ficheiros entre máquinas, para que possam servir de *input* para as aplicações executadas e para que seja possível obter *output* das mesmas. Adicionalmente, disponibiliza um conjunto de serviços de informação que pode ser acedido através da utilização de uma *API* disponibilizada para o efeito. Estes serviços providenciam informação acerca do estado global do *cluster*, listagens de tarefas em execução, bem como a possibilidade de administração do sistema, entre outras.

O *LSF* apenas está disponível na versão comercial. Esta versão pode ser instalada em sistemas operativos *Windows e Unix/Linux.*

Comparação entre ferramentas

A Tabela 1 apresenta uma breve comparação entre as várias ferramentas de gestão de clusters, enumerando as características consideradas na avaliação de cada uma.

2.4.2 *Standards* e Linguagens *Grid*

Com a evolução dos ambientes *Grid* e com a cada vez maior diversidade de *softwares* envolvidos na criação de uma infra-estrutura *Grid*, torna-se obrigatória a adesão a *standards*, para garantir a interoperabilidade entre elementos de diferentes fornecedores. Com a participação da indústria, foi acelerado o processo de estabilização e melhoria da robustez dos vários componentes, como forma de tornar as *Grids* como verdadeiros ambientes de produção, extravasando o uso meramente experimental [Baker, M. et al. 2005].

	Condor	***PBS***	***SGE***	***LSF***
Distribuição	Gratuita	Gratuita/Comercial	Gratuita/Comercial	Comercial
Plataformas	*Windows* e *Unix/Linux*	*Unix/Linux* (*Windows* na versão comercial)	*Windows*, *Unix/Linux* e *MacOS*	*Windows* e *Unix/Linux*
Reserva antecipada de recursos	Não	Sim	Sim	Sim
Interligação entre vários clusters	Sim	Sim	Sim	Sim
Implantação	Utilização	Utilizado em vários super-computadores, incluindo o *IBM RoadRunner.*	Utilizado em vários super-computadores, incluindo o *Ranger.*	Líder em soluções comerciais
Facilidade de instalação e configuração	Média	Alta	Média	Não aplicável[2]
Possibilidade de integração com *middleware Grid*	Sim	Sim	Sim	Sim
Alterações necessárias para a integração com *middleware Grid*	Sim	Não	Sim	Não

Tabela 1 - Comparação entre ferramentas de gestão de *clusters*.

Actualmente, são várias as organizações dedicadas à criação de normas e linhas de orientação cujo objectivo é uniformizar o desenvolvimento e integração entre os vários componentes de um ambiente *Grid*. Exemplos dessas organizações são o *Open Grid Forum (OGF)* [W8] e a *Organization for the Advancement of Structured Information Standards (OASIS)* [W21].

O *OGF* é uma comunidade de utilizadores, programadores e fornecedores de *software* focada na padronização da computação *Grid e* resultou, em 2006, da fusão entre o *Global Grid Forum* e a *Enterprise Grid Alliance*. É actualmente responsável por diversos *standards* presentes em ambientes *Grid*.

Por sua vez, a *OASIS* é uma organização sem fins lucrativos principalmente focada na criação de *standards* para o desenvolvimento de *Web-Services*, sendo a organização com maior número de *standards* criados, desde a sua criação em 1993. As suas áreas de operação incluem a segurança e *e-business,* no sector público e privado. A *OASIS* conta com mais de 5 mil participantes em cerca de 600 organizações e 100 países.

[2] Devido à não existência de uma versão gratuita, não foi possível avaliar o *LSF*.

Seguidamente, são apresentados alguns dos *standards* mais importantes, adoptados pelos principais fornecedores de *software* para ambientes *Grid.*

Open Grid Services Architecture

Um dos mais importantes *standards* actuais [Baker, M. et al. 2005] é a *Open Grid Services Architecture (OGSA)*, que foi criada em 2002 pelo então *Global Grid Forum*. O seu objectivo é a definição de uma arquitectura aberta e comum para aplicações *Grid*, baseada nos princípios da *Service-Oriented Architecture (SOA).*

Este *standard* foca-se num ambiente *Grid* como sendo um provedor de serviços às aplicações em execução. Para tal, define o serviço *Grid* padrão, tendo em conta as problemáticas da criação, denominação e descoberta de instâncias de serviços com tempo de vida potencialmente limitado [Foster, I. et al. 2002A]. Esta última preocupação está directamente ligada a uma das características das *VOs*: o seu tempo de duração finito.

A *OGSA* define também mecanismos de localização transparente e integração com bibliotecas nativas nas plataformas utilizadas. Para além disso, a *OGSA* utiliza a *Web-Services Description Language (WSDL)*, para definir interfaces, convenções e mecanismos para a criação e composição de sistemas distribuídos. Em particular, define mecanismos de suporte ao elevado dinamismo dos ambientes *Grid*, bem como a notificação entre os diferentes componentes *Grid*. Adicionalmente, tenta responder a requisitos como a invocação fiável de serviços, autenticação, autorização e delegação de responsabilidades em ambientes distribuídos e multi-organizacionais.

Devido à sua abrangência e foco nos aspectos cruciais para a criação de sistemas *Grid*, a *OGSA* é actualmente adoptada pelos principais *middlewares Grid*, produzidos por diversos fornecedores a nível mundial.

Distributed Resource Management Application API

Segundo a sua especificação [Rajic, H. et al. 2007], a *Distributed Resource Management Application API (DRMAA)* providencia um modelo de programação aos programadores de aplicações e gestores de recursos *Grid*, para submissão, monitorização, controlo e obtenção de estado de tarefas executadas em ambientes *Grid*.

Foi a criação da *DRMAA*, e posterior adesão dos principais fornecedores de *software*, que permitiu que fosse possível submeter tarefas entre diferentes escalonadores em sítios distintos.

Actualmente, todos os principais sistemas de gestão de clusters (incluindo os seus escalonadores locais) e escalonadores globais possuem implementações da *DRMAA*. Estas permitem também que sejam desenvolvidas aplicações *Grid* que aproveitem recursos de forma

transparente em toda a *Grid*, sem que tenham de utilizar os mecanismos de submissão de tarefas específicos de cada escalonador. As aplicações podem teoricamente ser instaladas sem a obrigatoriedade da escolha de um escalonador específico. Para além disso, existem implementações de bibliotecas *DRMAA* que podem ser utilizadas em diversas linguagens, incluindo *Java* e *C*.

Job Submission Description Language

A *Job Submission Description Language (JSDL)* foi lançada pelo *Global Grid Forum* em finais de 2005, através da disponibilização da sua especificação [Anjomshoaa, A. et al. 2005]. Trata-se de uma linguagem *XML* para a descrição de tarefas para posterior execução em ambientes de execução distribuída.

Segundo a especificação, a criação da *JSDL* teve em consideração duas preocupações principais. A primeira baseia-se na existência de diferentes sistemas de execução e escalonamento de tarefas, operados por diferentes organizações. Sem uma linguagem comum, a única forma de uma aplicação utilizar os vários sistemas seria suportar todas as linguagens de submissão envolvidas. Com a *JSDL*, torna-se possível criar um único descritivo, podendo sempre ser feita a tradução para uma outra linguagem, caso o sistema de execução não suporte *JSDL*. A segunda preocupação dos criadores da *JSDL* relaciona-se com a delegação de controlo sobre as tarefas entre os vários escalonadores e gestores de recursos. Sem uma linguagem comum, um escalonador global não seria capaz de transmitir ao escalonador local alguns dos requisitos das tarefas a executar. Estes requisitos podem incluir necessidades de *hardware* (*e.g.* número e velocidade dos processadores ou memória *RAM),* mas também outros parâmetros como, por exemplo, ficheiros que devem ser transferidos para que a tarefa possa ser executada, ou quais os limites de tempo para execução da tarefa.

Web-Services Resource Framework

Apesar das vantagens oferecidas pelos *Web-Services,* permitindo interoperabilidade entre sistemas, através da padronização da troca de mensagens, não é possível aos serviços armazenarem estado entre invocações. Tal porque, por definição, um *Web-Service* é *stateless*. Tendo considerado esta característica uma limitação, o *Global Grid Forum* lançou em 2003 a *Open Grid Services Infrastructure (OGSI).* Esta iniciativa aspirava a fornecer uma extensão aos *Web-Services*, sendo assim possível armazenar estado entre invocações e permitindo também que os serviços pudessem ser instanciados e destruídos conforme necessário. Desta forma, tornava-se possível melhorar a gestão de recursos. A possibilidade de armazenar estado entre invocações

permitia, por exemplo, que um serviço de informação de estado pudesse ser implementado utilizado *Web-Services stateful*, através do armazenamento do estado da *Grid*, após inicialização.

Apesar das funcionalidades desejadas, a *OGSI* esteve envolto em controvérsia e foi considerado um fracasso [Baker, M. et al. 2005], pela obrigatoriedade de utilização da orientação a objectos. Para além disso, obrigava à redefinição da *WSDL*, o que obrigaria à modificação das ferramentas utilizadas para o desenvolvimento de *Web-Services*, para a construção de serviços Grid. Por fim, a *OGSI* era considerada demasiado extensa para compor apenas uma única especificação. Como solução para os problemas encontrados, a *Globus Alliance e a IBM* (entre outros) criaram, em 2004, uma nova especificação baseada na *OGSI:* a *Web-Services Resource Framework (WSRF)* [Czajkowski, K. et al. 2004]. Esta especificação foi entretanto aceite pela *OASIS* como um dos seus *standards*.

A *WSRF* define um conjunto de normas que, quando implementadas, permitem ao programador associar um *Web-Service* a um ou mais recursos, sendo o estado armazenado nestes. Desta forma, cada pedido efectuado ao serviço é acompanhado pelo identificador do recurso a ser utilizado, em conjunção com o serviço. Também utilizado este identificador, é possível criar, monitorizar e destruir a instância do sistema, suportando-se assim a noção de ciclo de vida de um serviço.

A especificação *WSRF* define quatro normas distintas: *WS-ResourceProperties, WS-ResourceLifetime, WS-ServiceGroup e WS-BaseFaults*. A primeira, define como consultar e modificar as propriedades dos recursos associados aos serviços. A *WS-ResourceLifetime* define como gerir o ciclo de vida dos serviços *Grid*. A especificação *WS-ServiceGroup* foca-se na representação e gestão de conjuntos de serviços, permitindo a federação de serviços. Por fim, a especificação *WS-BaseFaults* preocupa-se com a comunicação e tratamento de erros durante o processamento de pedidos.

Actualmente, podem ser encontradas implementações *WSRF* em vários *middlewares Grid*, como será apresentado posteriormente neste documento.

Grid Security Infrastructure

A *Grid Security Infrastructure (GSI)* é considerada o *standard de facto* para a implementação de mecanismos de segurança em ambientes *Grid*. A *GSI* recorre a certificação *X.509* para garantir identidades e gerar credenciais que permitem a utilização remota de recursos, dentro de uma *VO*. Para além disso, permite a comunicação segura e autenticada, através de encriptação assimétrica, utilizando criptografia de chave pública.

A *GSI* é a primeira implementação de mecanismos de segurança em conformidade com a *OGSA* [Welch, V. et al. 2003]**.** Utilizando esta infra-estrutura, as aplicações *Grid* não necessitam de implementar mecanismos de segurança próprios. Pelo contrário, a *GSI* fornece as funcionalidades necessárias, para que as políticas de segurança sejam respeitadas e para que apenas os utilizadores autorizados possam ter acesso aos recursos acordados. Para tal, a *GSI* disponibiliza serviços de segurança seguindo o modelo *OGSA* às aplicações, eliminando a necessidade de implementação de funcionalidade de segurança nas aplicações e delegando essa responsabilidade nos servidores aplicacionais incorporados nos *middlewares Grid*.

Além do referido, a *GSI* publica as políticas de segurança, para que um cliente possa conhecer quais as credenciais necessárias para que possa ser estabelecida uma relação de confiança com um serviço. Por fim, a *GSI* especifica quais os *standards* que devem ser utilizados para a comunicação segura entre serviços e clientes, garantindo interoperabilidade [Welch, V. et al. 2003]**.**

A especificação *GSI* é actualmente utilizada em diversos *middlewares Grid*, como base para a criação de mecanismos de segurança, como se poderá ver seguidamente.

2.4.3 *Middlewares Grid*

Os *middlewares* actuais são constituídos por um considerável conjunto de componentes, com ênfase em áreas como a segurança, gestão de recursos, monitorização, entre outras. Alguns desses componentes são partilhados entre *middlewares*, uma vez que são disponibilizados gratuitamente e acompanhados pelo código fonte. Assim, não é de estranhar que alguns *middlewares* surjam como composição entre *middlewares* já existentes e novos componentes de *software*. Seguidamente são apresentados as principais distribuições, sendo estas: *Globus Toolkit, Virtual Data Toolkit (VDT), gLite* e *Uniform Interface to Computing Resources (UNICORE)*.

Globus Toolkit

O *Globus Toolkit* surgiu no final da década de 90, com o objectivo de facilitar o desenvolvimento de aplicações e infra-estruturas distribuídas orientadas a serviços. Actualmente na quarta versão, o *Globus* é encarado como o *standard de facto* para a montagem e implementação de ambientes *Grid* [Cannataro, M. et al. 2004][Ellert, M. et al. 2003][Frey, F. et al. 2002].

Dado o foco no paradigma *SOA*, o *Globus* disponibiliza um conjunto de serviços que potenciam a construção de ambientes e implementação de aplicações *Grid,* seguindo o modelo

proposto pela *OGSA* [Foster, I. 2006]. Os vários serviços *Globus,* que são expostos via *Web-Services*, assentam num conjunto de módulos base internos, como pode ser visto na Figura 9. Para além dos serviços, o *Globus* contém também um servidor aplicacional, onde podem ser disponibilizadas aplicações desenvolvidas pelo programador, bem como um conjunto de bibliotecas utilizadas para o efeito.

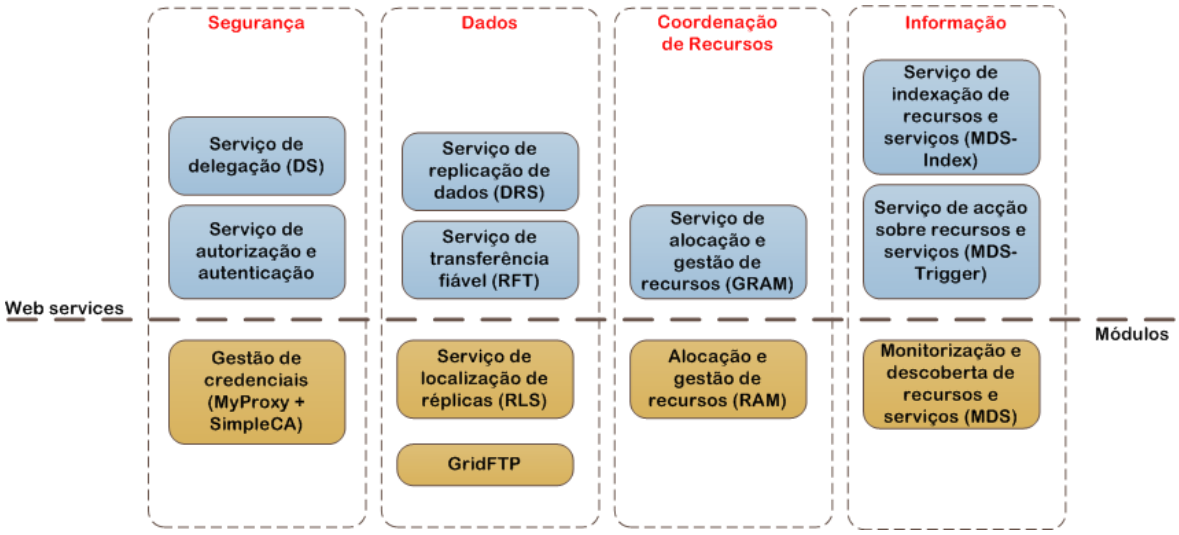

Figura 9 - *Web-Services* e módulos principais do *Globus Toolkit.*

A gestão e transferência de ficheiros em ambientes *Globus* são conseguidas através da utilização dos serviços de transferência fiável e replicação. O serviço responsável pela transferência fiável (*RFT, ou Reliable File Transfer*) é capaz de comunicar utilizando diversos protocolos, incluindo *HTTP*, *HTTPS*, *FTP* ou *GSIFTP*, utilizado o módulo *GridFTP.* O *RFT* publica a funcionalidade do *GridFTP*, aderindo ao *standard WSRF*, permitindo a interoperabilidade com outros sistemas de transferência de ficheiros. Para além disso, não obriga ao estabelecimento de ligações permanentes entre sítios, permitindo assim recuperar facilmente de quebras de comunicação. Quanto à replicação de ficheiros, esta é possível através da utilização do *Replica Location Service (RLS)* que é exposto, conforme a especificação *WSRF,* através do serviço *Data Replication Service (DRS).*

Os mecanismos de segurança do *Globus* estão assentes na *GSI.* Para armazenamento e geração de credenciais, o *Globus* inclui o servidor *MyProxy*. Trata-se de um repositório onde os utilizadores podem obter credenciais, para autenticação em serviços remotos. Uma única credencial abre o acesso a múltiplos serviços, permitindo desta forma mecanismos de autenticação

única. Para a criação e manutenção de certificados *X.509*, o *Globus* inclui também a *SimpleCA*. Este conjunto de ferramentas faz com que seja possível a uma organização gerar os seus próprios certificados, caso não esteja disposta a adquiri-los junto de uma autoridade de certificação. Para permitir a integração com outros sistemas, o *Globus* disponibiliza vários serviços de segurança, incluindo o serviço de delegação (*DS, ou Delegation Service)* e o serviço de autorização e autenticação. O primeiro permite que após a apresentação de credenciais, o utilizador possa aceder aos serviços registados no servidor aplicacional. O segundo é responsável por controlar o acesso aos serviços do servidor aplicacional, mediante autenticação.

A coordenação de tarefas é feita por um conjunto de serviços denominado *Grid Resource Allocation and Management (GRAM).* O *GRAM* permite a localização, submissão, monitorização e cancelamento de tarefas em recursos *Grid*, pertencentes à própria máquina ou em *clusters* associados. Para o efeito, o *GRAM* contém um conjunto de módulos de interligação, que permitem a operação com sistemas gestores de *clusters,* como o *PBS, Condor, LSF e SGE.* Para além disso, e tendo em conta que o não é um escalonador, o *GRAM* disponibiliza a interface para que escalonadores globais possam utilizar recursos locais à instalação de *Globus*.

Quanto à recolha, propagação e disponibilização de informação, é o conjunto de serviços *MDS (Monitoring and Discovery System)* do *Globus* que providencia estas funcionalidades. O módulo *MDS* trata da recolha de estado de recursos e também da descoberta de serviços disponíveis nos vários servidores aplicacionais *Globus*. Essa informação é então publicada pelo serviço de indexação (*MDS-Index)* que permite obter, via *Web-Services*, um estado agregado dos recursos e serviços conhecidos pelo *MDS*. É através deste serviço que os escalonadores globais podem obter informação acerca do estado global da *Grid*, de forma a tomar as decisões de escalonamento. Para além do *MDS-Index*, está também incluído o serviço *MDS-Trigger*, que permite a execução de acções específicas quando certas condições se tornam verdadeiras. O *MDS-Trigger* pode, por exemplo, ser configurado para notificar o administrador de sistemas de uma determinada organização, sempre que uma nova máquina for adicionada ao ambiente *Grid* ou sempre que o espaço em disco numa das máquinas alcançar um valor crítico.

No que diz respeito à disponibilidade e possibilidade de instalação, o *Globus* pode ser obtido gratuitamente e instalado em diversas distribuições de *Unix/Linux* e também em *MacOS*. As componentes *WSRF* do *Globus*, que permitem o desenvolvimento de *Web-Services Stateful*, podem também ser instalada em Windows. Para além da versão gratuita, disponibilizada pela *Globus Alliance*, existe também uma versão comercial do *Globus Toolkit*, fornecida pela *Platform.*

VDT

O *Virtual Data Toolkit (VDT)* [W12] nasceu como produto da *Open Science Grid (OSG)*, um projecto cuja missão é satisfazer os requisitos crescentes da comunidade científica, que pretende recursos de computação de elevado desempenho para utilização em aplicações científicas colaborativas.

Sabendo-se que a instalação de ambientes *Grid* é uma tarefa complexa e demorada, o *VDT* tem como objectivo facilitar a adopção, instalação e manutenção de ferramentas *Grid.* Para tal, o *VDT* inclui um conjunto vasto de *software Grid*, incluindo o *Condor* e o *Globus,* bem como servidores *Web,* aplicacionais e de base de dados. Estes pacotes de *software,* e muitos outros, podem ser instalados de forma fácil, utilizando o instalador *PacMan*.

Para além de incluir o *Globus e Condor,* o *VDT* inclui um terceiro pacote relevante, denominado por *Virtual Organization Membership Service (VOMS).* O *VOMS* é um sistema orientado para a gestão de autorização e privilégios de utilizadores em ambientes multi-institucionais e colaborativos, como é o caso dos ambientes *Grid*. Este sistema permite a manutenção de uma base de dados de perfis e de utilizadores, para além de um conjunto de ferramentas para permitir o acesso a essa base de dados. Para além disso, permite também a geração de credenciais a pedido, permitindo a autenticação única em diversos sistemas.

No que toca à disponibilidade e suporte para sistemas operativos, o *VDT* pode ser obtido gratuitamente, podendo ser instalado nas principais distribuições de *Unix/Linux*, bem como em *MacOs*.

gLite

Nascido a partir de um projecto europeu (ver *EGEE na secção* 2.5), o *gLite* é um dos mais recentes *middlewares Grid*. É composto por uma amálgama de outros *softwares*, contendo componentes próprios e aproveitando componentes de outros *middlewares*, incluindo o *VDT*, e por inerência, o *Globus*. Tal como os últimos, o *gLite* está disponível gratuitamente. No entanto, apenas uma distribuição de *Linux (ScientificLinux)* é oficialmente suportada, como sistema operativo.

Tal como o *Globus*, também o *gLite* é baseado numa arquitectura *SOA*, seguindo os princípios da *OGSA* e da *WSRF,* bem como outros *standards* reconhecidos [Laure, E. et al. 2006]. Desta forma, é possível integrar o *gLite* com outros sistemas semelhantes, incluindo o *Globus*, *VDT* e *UNICORE*. No que diz respeito à integração com sistemas de gestão de clusters, o *gLite* mantém

as potencialidades do *Globus* e também permite a integração através de um outro componente, herdado do *LCG*[3].

Os mecanismos de segurança do *gLite* são baseados na *GSI* e em protocolos como o *Transport Layer Security (TLS)*. Assim, são garantidos princípios como a comunicação segura, não repudiável e autenticada. Para além disso, é também suportada a autenticação única, sendo possível aceder a qualquer recurso na *Grid*, mediante uma única execução do processo de autenticação. O repositório de credenciais do *gLite,* responsável pela atribuição de credenciais para uso em recursos remotos, é também o *MyProxy Server*. No que toca ao mapeamento de políticas entre *VO* e organizações físicas, é usado o *VOMS.* É através deste componente que é possível estabelecer quais os privilégios de segurança de cada um dos utilizadores da *VO*.

A transferência de dados no *gLite* é assegurada por três serviços: armazenamento, catálogo de réplicas de ficheiros e gestor de dados [Laure, E. et al. 2006]. O serviço de armazenamento abstrai um recurso que armazena dados, seja ele um disco rígido, um *array* de discos, ou um directório. O catálogo de réplicas mantém informação acerca da localização de ficheiros, bem alguns metadados, como por exemplo o tamanho ou códigos de verificação de integridade. O serviço gestor de dados guarda informação relativa aos vários serviços de armazenamento e aos catálogos, de forma a escolher as melhores fontes de dados e a garantir a maior eficiência na execução de tarefas dependentes da transferência de ficheiros.

Numa infra-estrutura baseada em *gLite* (Figura 10), os recursos disponíveis são abstraídos através da definição de elementos de computação. Cada um destes elementos pode corresponder a um *cluster* ou a apenas uma máquina, por exemplo. Utilizando o *gLite* é então possível submeter tarefas para que sejam executadas em qualquer um desses elementos. Para além dos habituais serviços de submissão e monitorização de tarefas, o *gLite* inclui também um sistema de gestão de carga que distribui tarefas pelos vários elementos, de forma a optimizar a utilização dos recursos. A escolha dos recursos é feita com base no emparelhamento entre recursos disponíveis e recursos requeridos pela tarefa.

Por cada tarefa que é submetida, executada, ou repetida, por exemplo, é feito o respectivo registo no serviço de informação. Este sistema permite depois que se obtenham informações acerca do estado dos elementos e das tarefas, para posterior análise [Laure, E. et al. 2006].

[3] O LHC Computing Grid *(LCG)* é o projecto responsável por fornecer um *middleware Grid* para utilização no *Large Hadron Collider (LHC)*. Este acelerador de partículas é o maior aparelho de instrumentação do mundo e foi desenvolvido pelo *CERN*.

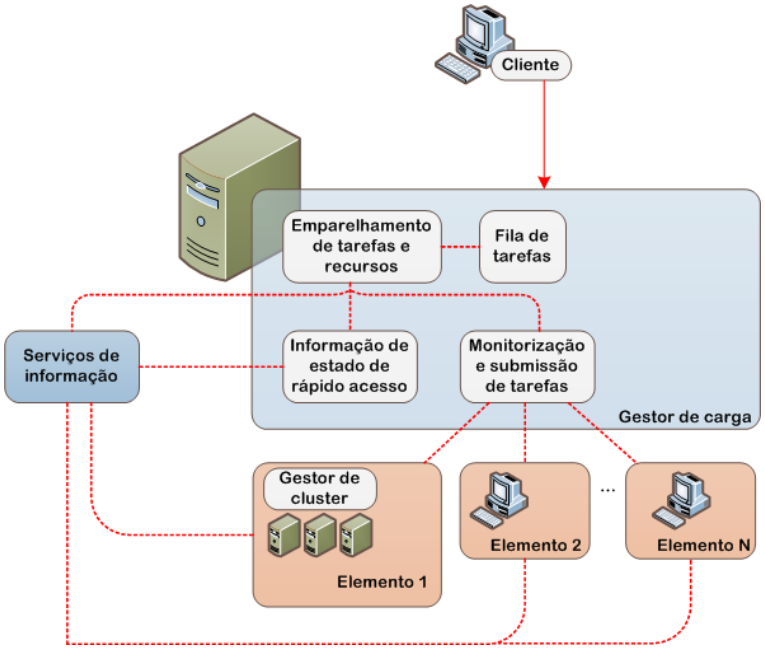

Figura 10 - Arquitectura resumida do *middleware gLite.*

No que diz respeito à integração com sistemas de gestão de *clusters*, é possível submeter tarefas no *gLite* para diversos sistemas, incluindo *PBS, Condor, SGE e LSF.* Adicionalmente, é possível delegar tarefas noutras instalações de *gLite* ou outros *middlewares*, como o *Globus, VDT e UNICORE*, possivelmente em sítios distintos. É também possível submeter tarefas para um sítio que utilize *gLite*, através de vários escalonadores globais.

UNICORE

O *UNICORE* [W22] foi criado na Alemanha em 1997, com o objectivo de facilitar o acesso aos super-computadores alemães, por parte dos seus utilizadores [Erwin, D. 2001]. Para tal, os seus criadores focaram-se em criar um sistema que abstraísse a heterogeneidade existente ao nível do *software* e do *hardware,* incluindo políticas de segurança, sistemas de ficheiros, sistema de gestão de clusters, entre outros. Para além disso, a facilidade de utilização, quer por cientistas ou por engenheiros, foi tida como um dos principais requisitos. Como resultado, o *UNICORE* é um sistema gratuito de simples instalação, tanto no servidor, como no cliente e com uma interface gráfica intuitiva no cliente, em plataformas *Windows e Unix/Linux*.

No que toca à infra-estrutura de segurança, o *UNICORE* baseia-se em certificados digitais *X.509* como forma de identificação dos seus utilizadores. Estes certificados permitem o

mapeamento de utilizadores remotos com utilizadores locais aos recursos onde o servidor *UNICORE* é instalado. Tal permite também o mapeamento de políticas da *VO* com as políticas locais. A autenticação única é suportada, através da utilização de certificados digitais. A comunicação é feita de forma segura, autenticada e não repudiável, através da utilização do protocolo *Secure Socket Layer (SSL)*, tanto entre cliente e servidor, como entre servidores.

No diz respeito à transferência de dados, o *UNICORE* suporta importação e exportação de ficheiros. Assim, é possível importar conjuntos de dados a processar para um recurso remoto, exportando subsequentemente os resultados da computação. No entanto, o *UNICORE* não possui um sistema de gestão de réplicas de ficheiros.

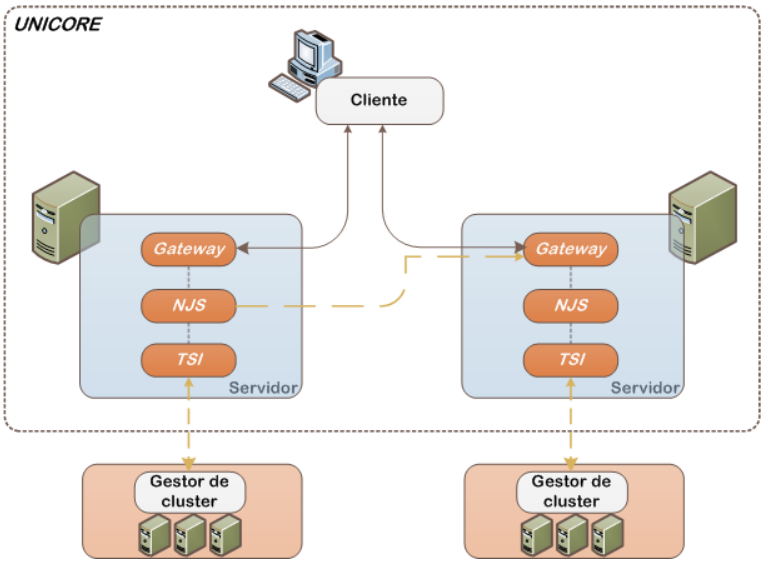

Figura 11 - Arquitectura do *middleware UNICORE*.

O *UNICORE* suporta a submissão, monitorização, cancelamento e repetição de tarefas. A submissão é feita no cliente, de forma gráfica ou linha de comandos. Para tal, o utilizador escolhe o sítio para execução da tarefa. No entanto, a decisão de escalonamento pode ser feita por um escalonador global, de forma a optimizar a utilização de recursos. Após submissão, o utilizador pode seguir a execução das suas tarefas de forma gráfica, no cliente. Do lado do servidor, as tarefas são recebidas por um *gateway*, que entrega os pedidos de execução ao *Network Job Supervisor (NJS)*. Estes pedidos são convertidos em subtarefas que são executadas nos diversos recursos aos quais o *UNICORE* tem acesso, no sítio onde é instalado. O *NJS* pode, no entanto, decidir-se pela utilização de recursos pelos quais um outro *gateway* é responsável. Assim, é possível a troca de tarefas entre vários servidores *UNICORE*.

No que diz respeito à compatibilidade com sistema de gestão de *clusters*, o *UNICORE* suporta a integração com sistemas como o *PBS* (nas suas várias distribuições) e o *LSF*, entre outros. Esta integração é conseguida através do *Target System Interface (TSI)*, que providencia a compatibilidade com os vários sistemas de gestão de *clusters*.

O *UNICORE* permite também a interoperabilidade com outros *middlewares*, como o *Globus* e o *gLite.* Para garantir a correcta comunicação com outros sistemas, o *UNICORE* adere a vários *standards*, nomeadamente *DRMAA, WSRF* e *JSDL.*

A Figura 11 apresenta um resumo da arquitectura do *UNICORE*, evidenciando os seus componentes de gestão de tarefas e ligação entre servidores.

Comparação entre ferramentas

A Tabela 2 apresenta uma breve comparação entre os *middlewares* apresentados, tendo em conta as características avaliadas.

2.4.4 Escalonamento global

Sem a capacidade de tomar partido de recursos distribuídos e de elevado desempenho, de forma transparente e automática, os ambientes *Grid* teriam pouca utilidade para os seus utilizadores finais. Assim, a importância dos escalonadores globais ou *meta-escalonadores* é crucial, para o aproveitamento da infra-estrutura *Grid*. Neste campo, os dois principais escalonadores disponíveis são o *Condor-G* [W2] e o *Gridway* [W7].

Condor-G

Desenvolvido a partir de 2003, pela Universidade de Winsconsin, o *Condor-G* é um sistema de gestão de tarefas em ambientes *Grid*, funcionando como um escalonador global. Este sistema aproveita a componente de gestão do *Condor* e aplica-a a recursos disponibilizados através de um *gateway Globus*. Assim, o *Condor-G* permite aos seus utilizadores tirarem partido do ambiente multi-domínio e distribuído, tal como se tratasse de um único domínio.

O *Condor-G* é responsável pela descoberta e utilização dos recursos apropriados, independentemente da sua localização. Aquando da submissão de uma tarefa, encarrega-se também de inicializar a execução nos recursos escolhidos, transferindo os ficheiros necessários, monitorizando a execução da tarefa e finalmente entregando o retorno da execução ao utilizador final. Em caso de falha, o *Condor-G* possui mecanismos de detecção e re-submissão de tarefas.

	Globus Toolkit	***VDT***	***gLite***	***UNICORE***
Distribuição	Gratuita/ Comercial	Gratuita	Gratuita	Gratuita
Plataformas	*Windows* (apenas *WSRF*), *Unix/Linux e MacOS*	*Unix/Linux* e *MacOs*	*Scientific Linux*	*Windows* e *Unix/Linux*
Implantação	Principal *middleware Grid* a nível mundial e presente em vários outros *middlewares*	Utilizado na *Open Science Grid* e em vários projectos nível mundial	Utilizado no projecto *Enabling Grids for E-sciencE*	Utilizado em alguns projectos incluindo a *Grid* japonesa *NaReGi*.
Facilidade de instalação e configuração	Baixa	Média	Baixa	Alta
Integração com outros *middlewares Grid*	Sim	Sim	Sim	Sim
Autenticação única	Sim	Sim	Sim	Sim
Controlo de acesso a serviços e recursos	Sim	Sim	Sim	Sim
Transferência fiável e replicação de ficheiros	Sim	Sim	Sim	Não
Utilização de recursos em sítios distintos	Sim	Sim	Sim	Sim
Informação de estado	Sim	Sim	Sim	Não
Integração com gestores de *clusters*	Sim	Sim	Sim	Sim
Integração com escalonadores globais	Sim	Sim	Sim	Não
Documentação de configuração e desenvolvimento	Manuais de configuração e desenvolvimento e documentação dos componentes do *middleware*.	Documentação reduzida, focando-se apenas na instalação e configuração	Documentação de configuração, administração e desenvolvimento.	Documentação razoável de configuração e programação.

Tabela 2 - Comparação entre os *middlewares* Grid apresentados.

Para que execução de tarefas em ambientes remotos seja possível, são tidos em conta três principais aspectos: acesso aos recursos através de protocolos *standard*, gestão de recursos e criação ambientes isolados de execução [Frey, F. et al. 2002].

Para o acesso aos recursos, o *Condor-G* utiliza os serviços do *Globus*. A descoberta e monitorização de recursos é feita através do serviço *MDS-Index*. Assim, sempre que um recurso se junta à *Grid*, a abandona, ou altera o seu estado, o *Condor-G* tem acesso a essa informação. Para controlar os recursos descobertos, por exemplo, para entrega de tarefas, é utilizado o serviço *GRAM*.

No que diz respeito à gestão dos recursos, ou seja, a escolha de recursos a utilizar e a atribuição de prioridades a tarefas, são reaproveitados componentes do escalonador *Condor* para *clusters*. Como neste último, é mantida uma fila de tarefas persistente. Para tomar uma decisão (processo também conhecido por *MatchMaking*), o módulo escalonador do *Condor-G* tem em conta o conteúdo da fila, o estado dos recursos disponíveis e também informação anexada à tarefa (ou *ClassAds*). Esta informação é passada pelo utilizador, indicando parâmetros como, por exemplo, memória *RAM* mínima para a execução, número de processadores e suas arquitecturas, entre outros. É assim possível ajudar o escalonador a tomar melhores decisões, se a configuração ideal para a execução de cada tarefa for conhecida de antemão .

Para que o código a executar em cada tarefa não tenha de ter em conta o facto de estar a ser executado num ambiente *Grid*, num recurso remoto, o *Condor-G* recorre à criação de ambientes isolados de execução (ou *sandboxes*). Este mecanismo, também denominado por *GlideIn*, faz com que as chamadas ao sistema operativo, feitas pelo código, sejam reencaminhadas para o sistema de origem [Frey, F. et al. 2002]. Por exemplo, se a tarefa guardar o resultado num ficheiro temporário, esse ficheiro será transportado para o sistema de origem, através da exportação da *sandbox*, no final da execução.

Como mecanismos de submissão de tarefas do *Condor-G*, podem ser utilizadas as ferramentas providenciadas pela instalação (utilizando linguagens de descrição como *JSDL* e *ClassAds*), *Web-Services* e também uma *API DRMAA*. O *Condor-G* permite também a submissão de conjuntos de tarefas em simultâneo e o estabelecimento de dependências entre tarefas, através da definição de grafos acíclicos direccionados. Desta forma é possível reproduzir processos complexos sob a forma de tarefas executadas num ambiente *Grid*, em que o resultado de uma tarefa pode servir como valor de entrada para a tarefa seguinte.

A instalação do *Condor-G* é relativamente simples, após a instalação do *Globus*. Caso se opte pelo *VDT*, a instalação do *Condor-G* é feita automaticamente. Em qualquer uma das opções, o *Condor-G* é distribuído de forma gratuita.

Gridway

Em Setembro de 2002, foi criado o projecto *Gridway*, na Universidade Complutense de Madrid. Em Maio de 2006, este projecto foi integrado na incubadora de projectos da *Globus Alliance*. O seu objectivo é fornecer um escalonador global que se integre com as infra-estruturas *Grid* existentes, de forma a disponibilizar aos seus utilizadores um ambiente integrado em que a utilização de recursos é transparente. Para além disso, pretende ser uma ferramenta para os programadores de aplicações *Grid*, permitindo a utilização da *Grid* através de bibliotecas providenciadas para o efeito.

O principal componente do *Gridway* é um agente modular de submissão e execução de tarefas [Huedo, E. et al. 2005]. Este agente tem como função acompanhar todo o ciclo de vida de uma tarefa, suportando reescalonamento, sempre que seja necessária a migração entre recursos. Tal pode ser necessário caso uma máquina fique sobrecarregada, ou seja detectada uma falha na execução.

A fase de execução de cada tarefa individual é controlada por três módulos distintos: *Prolog, Wrapper* e *Epilog*. O primeiro módulo prepara a execução no recurso remoto, procedendo à cópia dos ficheiros necessários para a execução, incluindo o módulo *Wrapper*. No final desta fase, está criado um ambiente isolado para execução da tarefa, à imagem do que é feito pelo *Condor-G*. Seguidamente, este módulo executa a tarefa solicitada pelo utilizador e envia o retorno da mesma para o *Gridway*. Finalmente, o módulo *Epilog* copia os ficheiros gerados pela tarefa (incluindo o *standard output* e *standard error*) e remove o ambiente criado do recurso remoto, procedendo à limpeza dos ficheiros criados.

Para a escolha dos recursos utilizados para a execução das tarefas são utilizados dois outros módulos: *Resource Selector* e *Performance Evaluator*. O primeiro módulo avalia quais os requisitos e preferências para execução da tarefa, tendo por base a especificação do utilizador (escrita em *JSDL*) e ordena os recursos tendo em conta esses critérios, sempre que é necessário escalonar ou reescalonar uma tarefa. As estratégias utilizadas podem variar, sendo possível utilizar módulos de escalonamento externos ao próprio *Gridway*, para efectuar este processo de *MatchMaking*. Já o módulo de avaliação de performance monitoriza o estado das tarefas e, com base num perfil pré-estabelecido, pode decidir migrar uma tarefa de recurso, caso detecte algum desfasamento entre o perfil e a execução esperada.

Através da utilização do *Gridway*, é possível executar tarefas remotamente em sítios controlados pelo *gateway Globus*. Estas tarefas podem ser submetidas através das ferramentas de linha de comandos providenciadas e também através de uma *API DRMAA*. As tarefas podem ser submetidas individualmente ou em conjunto, sendo também possível estabelecer dependências entre tarefas, através da definição de grafos acíclicos direccionados de tarefas.

	Condor-G	***Gridway***
Distribuição	Gratuita	Gratuita
Plataformas	*Unix/Linux e MacOS*	*Unix/Linux* e *MacOs*
Implantação	Principal *middlware Grid* a nível mundial e presente em vários outros *middlewares*	Utilizado na *Open Science Grid* e em vários projectos nível mundial
Facilidade de instalação e configuração	Alta	Alta
Possibilidade de especificar requisitos preferenciais para a execução de tarefas	Sim	Sim
Suporte para conjuntos de tarefas	Sim	Sim
Suporte para dependências entre tarefas	Sim	Sim
Utilização de recursos em sítios distintos	Sim	Sim
Integração com serviços de informação	Sim *(MDS-Index)*	Sim *(MDS-Index)*
Suporte para módulos de escalonamento externos	Não	Sim
Integração com middlewares *Grid*	Sim (*Globus, VDT e gLite)*	Sim (*Globus, VDT e gLite)*
Documentação de configuração e desenvolvimento	Manuais de instalação	Manuais de instalação, guias de desenvolvimento e especificações das *APIs* disponíveis.

Tabela 3 - Comparação entre ferramentas de escalonamento global apresentadas.

A interacção entre *Gridway* e *Globus* acontece fundamentalmente através da utilização de três serviços: *MDS-Index*, *GRAM* e *RFT*. À imagem do que acontece com o *Condor-G*, é usado o serviço *MDS-Index* para obter informação acerca do estado dos recursos disponíveis. Esta comunicação é feita pelo módulo *Information Manager.* A execução, controlo e monitorização de tarefas em recursos remotos são possíveis através do uso do *GRAM*, sendo o módulo *Execution*

Manager o responsável. A transferência de ficheiros de e para os recursos remotos é feita pelo módulo *Transfer Manager*, em comunicação com o serviço *RFT.*

A instalação do *Gridway* é relativamente simples, podendo ser feita de forma automática, aquando da instalação do *Globus*. Tal como o último, o *Gridway* pode se obtido gratuitamente e instalado em sistemas operativos *Unix/Linux* e *MacOs*.

Comparação entre ferramentas

Na Tabela 3 é apresentado o conjunto de critérios utilizados para a avaliação das ferramentas de escalonamento global supracitadas.

2.5 *Grids* em produção

Embora se trate de um conceito emergente já existem actualmente *Grid* produtivas. Apesar de tudo, o propósito de tais infra-estruturas continua a centrar-se principalmente no campo científico, e não no empresarial. Para além das várias iniciativas regionais e nacionais, têm vindo a ser criados projectos internacionais visando potenciar a investigação científica, através da disponibilização de recursos de alto desempenho. Seguem-se alguns dos mais relevantes na actualidade:

- ***EGEE -*** O projecto *Enabling Grids for E-sciencE (EGEE)* [W9] é uma iniciativa financiada pela Comissão Europeia, com o objectivo de dotar a comunidade científica e o mundo empresarial de uma *Grid* produtiva, com recursos espalhados geograficamente em mais de 50 países e com mais de 250 sítios. Diariamente são submetidos mais de 150 mil tarefas, para cerca de 70 mil processadores. O *EGEE* é multi-faseado, estando actualmente na terceira fase de projecto. A primeira fase centrou-se na montagem da infra-estrutura europeia. Durante a segunda fase foi melhorada a infra-estrutura existente e desenvolvido o *middleware gLite*. Actualmente, visa chegar a mais comunidades, através da expansão da sua rede de recursos, constituindo uma federação de várias *Grids* nacionais.

- ***Teragrid -*** A *TeraGrid* [W13] é uma infra-estrutura científica americana para computação *Grid*, financiada pela *National Science Foundation (NSF)*. Envolve vários parceiros por todo o país, através de redes de alto desempenho, juntando computadores de gama alta. É actualmente a infra-estrutura de maior dimensão a nível mundial e está disponível apenas para investigadores americanos. Após terminada a sua construção em 2004, entrou em

produção em Agosto de 2005. Conta actualmente com mais de 750 *teraflops* de capacidade de processamento e de 30 *petabytes* de capacidade de armazenamento de dados. Quanto ao *software* utilizado, a nível local são utilizadores escalonadores *PBS*, sendo o *Globus Toolkit* o *middleware* de eleição. O escalonamento global fica a cargo do *Condor-G*.

• ***NASA Information Power Grid*** - A *Information Power Grid (IPG)* [W14] é uma *Grid* de alto desempenho construída para uso dos cientistas e engenheiros da *NASA.* Foi criada em 1998, em conjunto com a *NSF,* e tem vindo a basear-se no *Globus Toolkit*. Tem como objectivo fornecer acesso a recursos computacionais heterogéneos, pertencentes a diferentes laboratórios de investigação. A utilização da *IPG* não é exclusiva do pessoal da *NASA,* sendo os recursos também partilhados por outros projectos, como por exemplo, a *TeraGrid*.

Capítulo 3

Instalação, configuração e desenvolvimento do ambiente protótipo

Para que seja possível efectuar previsão de disponibilidade em ambientes *Grid*, é obviamente necessário dispor de uma *Grid* funcional. Além disso, é imprescindível recolher informação de estado da *Grid* e que a mesma tenha disponibilidade variável, caso contrário a análise seria irrelevante e trivial. Embora alguns ambientes estejam publicamente disponíveis, como é o caso de alguns ambientes de demonstração da *EGEE*, a sua utilização obriga a um processo de autorização e adesão a uma *VO* real, mediante a apresentação de recursos de elevada disponibilidade para partilha com os restantes utilizadores da *Grid*. Assim, optou-se pela criação de um ambiente protótipo, para desenvolver a simulação de uma *Grid*, submetendo tarefas periodicamente e monitorizando os recursos do ambiente, criando assim uma *VO* simulada. Não se recorreu à criação de várias *VOs* por ter sido considerado irrelevante para a análise de disponibilidade.

A instalação e configuração de um ambiente *Grid* é um processo moroso e complexo, podendo decorrer vários anos desde a conceptualização até à entrada em produção, como é o caso dos ambientes produtivos apresentados no capítulo anterior. Por vezes, as ferramentas disponíveis não eliminam completamente as necessidades das organizações, sendo necessário desenvolver ferramentas próprias. A evolução das ferramentas *Grid* tem sido precisamente impulsionada pelas necessidades de cada novo projecto.

No contexto do trabalho desenvolvido, tornou-se necessário fazer uma escolha tanto ao nível do *hardware* como ao nível do *software* utilizado. Nas secções seguintes são descritas as

opções efectuadas aos diversos níveis: *hardware*, sistema operativo, gestão de *clusters* e escalonamento local, *middleware* e escalonamento global. Para além do ambiente montado, são também apresentados os sistemas desenvolvidos para geração de carga computacional na *Grid* e recolha de informação de estado da mesma.

3.1 Caracterização do *hardware*, rede e sistema operativo

Devido à inexistência de recursos físicos dedicados, disponíveis durante todo o período necessário para a avaliação de ferramentas, posterior desenvolvimento e recolha de dados, optou-se pela utilização de várias máquinas virtuais, simulando nós físicos. Não sendo a solução ideal, esta configuração não invalida o propósito do ambiente, que é o de servir como fonte de dados para análise de disponibilidade em ambientes *Grid*. Para além disso, o transporte das máquinas virtuais e a possibilidade da sua rápida replicação, permitem a disponibilização de todos os sistemas instalados ou desenvolvidos numa outra infra-estrutura física.

Para criação do ambiente, foram instaladas quatro máquinas virtuais (*gtnode-1, gtnode-2, gtnode-3* e *gtnode-4*), utilizando a ferramenta *VirtualBox* [W24] e simulando dois sítios distintos, numa única *VO*. Não se optou pela instalação de um número superior de máquinas virtuais, para não ultrapassar o número de núcleos de processador disponíveis. Caso esse limite fosse ultrapassado, os resultados ficariam certamente adulterados, devido à partilha efectiva de recursos de processamento entre máquinas virtuais. A cada uma das máquinas virtuais foi atribuído um núcleo de processador com frequência de operação de 2.50GHz. À excepção da máquina *gtnode-1*, foi atribuído 1GB de RAM a cada uma das máquinas virtuais. A máquina *gtnode-1* recebeu 1.5GB de RAM, para alojar serviços adicionais, como servidor gráfico de janelas. No que diz respeito a ligações de rede, foi estabelecida uma ligação simulada a 100Mbps entre máquinas virtuais e entre sítios. Em todas as máquinas foi instalado como sistema operativo Linux Ubuntu Server 7.1, com parâmetros de kernel alterados[4] especificamente para um melhor desempenho em máquinas virtuais.

3.2 Gestão de *clusters*

Como referido, foram criados dois sítios distintos, compostos pelos pares de máquinas *gtnode-1/gtnode-3* e *gtnode-2/gtnode-4*. Optou-se por instalar o sistema *PBS* (variante *TORQUE*,

[4] O período entre verificação de interrupções do processador é ajustado, por omissão, para máquinas físicas. Sem esta alteração, as máquinas virtuais podem consumir até 100% de tempo de processador apesar de se encontrarem inactivas.

na versão 2.3.0) em todas as máquinas, ficando as máquinas *gtnode-3 e gtnode-4* como coordenadores e escalonadores dos respectivos sítios.

A escolha do *PBS* deveu-se essencialmente à grande facilidade de instalação em cada máquina, à integração directa com *middlewares Grid* e à implantação actual. Uma vez que o *PBS* permite gerar um pacote instalador para cada nó adicional do sítio, a instalação em novos nós torna-se relativamente simples, permitindo o rápido crescimento de cada sítio, em termos de nós disponíveis. Já a integração com *middlewares Grid* foi também considerada um requisito obrigatório, e neste campo, o *PBS* é a ferramenta cuja integração é mais simples, não obrigando a qualquer modificação, quer na instalação e configuração do *PBS* quer na instalação do *middleware Grid*. Quanto à compatibilidade com sistemas operativos, não foi considerada uma limitação a instalação apenas possível em sistemas *Linux* na versão gratuita, devido à escolha inicial do sistema operativo e também às restrições da maioria dos *middlewares Grid*.

Para além do *PBS*, foi necessário recorrer a uma ferramenta de monitorização de *clusters*, de forma a publicar dados de estado nos sistemas de informação do *middleware Grid* escolhido (ver secção 3.3). A escolha recaiu no *Ganglia Monitor* [W11], um sistema de monitorização de elevado desempenho para ambientes distribuídos, como *clusters* e *Grids.* Para além das suas capacidades de monitorização e baixo impacto nas máquinas onde é instalado, o *Ganglia Monitor* é suportado pela maioria dos *middlewares Grid*.

3.3 Middleware *Grid*

A escolha e instalação do *middleware* constituem as tarefas mais demoradas e complexas na instalação de um ambiente *Grid*, tendo também consumido grande parte do tempo dedicada ao trabalho presentemente descrito. Tal deve-se ao vasto rol de serviços disponibilizados por um *middleware*, que exigem configuração específica em cada ambiente e cuja documentação ainda terá de sofrer alguma maturação, na maioria dos casos.

Dos vários *middlewares* analisados, foi escolhido o *Globus Toolkit* (versão 4.0.6), pela sua implantação, documentação de instalação/configuração e serviços de informação. O *UNICORE* não foi seleccionado precisamente por não ter serviços de informação, o que obrigaria ao desenvolvimento de um sistema de recolha de informação de estado da *Grid* ou de um adaptador para utilização *Ganglia Monitor*. O *gLite* foi descartado pelo requisito de instalação de *Scientific Linux* e também porque a carga de configuração é superior, uma vez que o próprio *Globus Toolkit* está incluído no *gLite* e também deve ser configurado. Por fim, e apesar da facilidade de instalação, o *VDT* não foi seleccionado uma vez que, por ser um agregador de pacotes de

software, não disponibiliza informação de configuração de cada um deles, obrigando a um trabalho ainda mais demorado de configuração do que do *Globus Toolkit*. A instalação do *Globus Toolkit* foi realizada em todas as máquinas, dando a um escalonador global a possibilidade de efectuar submissão directa de tarefas para cada máquina, ou utilização do *PBS* instalado em cada sítio para a submissão e controlo de tarefas (ver *Coordenação Distribuída)*.

Segurança

Para garantir um funcionamento correcto, o *Globus Toolkit* obriga à utilização de certificados digitais *X.509* em praticamente todas as entidades envolvidas, quer serviços quer utilizadores. Portanto, foi necessário configurar o módulo *SimpleCA,* de forma possibilitar a geração de certificados *X.509*. Este passo não seria necessário, caso fossem adquiridos certificados digitais a uma autoridade de certificação, o que obrigaria, no entanto, a um custo relativamente elevado.

Na posse de uma autoridade de certificação própria, providenciada pelo *SimpleCA*, foi configurada a *GSI*, através do estabelecimento de confiança na autoridade de certificação e geração de certificados para utilizadores, máquina e também para o servidor aplicacional onde são disponibilizados os serviços *Grid*.

O serviço *MyProxy* foi configurado como repositório de credenciais, permitindo assim autenticação única em todas as máquinas e serviços. Tornou-se assim possível utilizar qualquer máquina ou serviço, sem ser necessário introduzir uma palavra-passe, utilizando assim a credencial gerada e armazenada no *MyProxy*. Alternativamente, poderia ter sido feita a partilha do sistema de ficheiros de uma das máquinas, sendo a geração de credenciais feita nessa máquina. No entanto, num cenário real, nem sempre é possível efectuar esta partilha, pelo que são utilizadas soluções como a instalação do *MyProxy.*

Transferência de dados

O *Globus Toolkit* disponibiliza o serviço *RFT*, para garantir a transferência fiável de ficheiros entre as várias máquinas da *Grid*. Todas as transferências efectuadas por uma instância do serviço *RFT* são registadas numa base de dados *PostgreSQL*. Assim, foi necessário, em primeiro lugar, instalar e configurar uma base de dados *PostgreSQL*. Esta base de dados foi também utilizada para armazenar a informação de estado recolhida periodicamente (ver secção 3.6).

Seguidamente, foi efectuada a criação do esquema de base de dados necessário para o funcionamento do *RFT*. O *Globus Toolkit* disponibiliza todos os *scripts* necessários para o efeito.

Após algumas ligeiras configurações, o *RFT* ficou disponível, permitindo a transferência de ficheiros de e para qualquer máquina.

```
* GLUECE:
    o Cluster:
        + SubCluster:
            # Name: main
            # UniqueID: main
            # Host:
                * Name: gtnode-3
                * UniqueID: gtnode-3
                * Processor:
                    o ClockSpeed: 2479
                    o InstructionSet: x86
                * MainMemory:
                    o RAMAvailable: 630
                    o RAMSize: 1011
                    o VirtualAvailable: 1055
                    o VirtualSize: 1445
                * OperatingSystem:
                    o Name: Linux
                    o Release: 2.6.22-14-generic
                * Architecture:
                    o SMPSize: 1
                * ProcessorLoad:
                    o Last15Min: 18
                    o Last1Min: 2
                    o Last5Min: 11
    o ComputingElement:
        + Name: batch
        + UniqueID: batch
        + Info:
            # GRAMVersion: 4.0.6
            # HostName: gtnode-3
            # LRMSType: PBS
            # LRMSVersion: 2.3.0
            # TotalCPUs: 2
        + State:
            # FreeCPUs: 2
            # RunningJobs: 0
            # Status: enabled
            # TotalJobs: 2
            # WaitingJobs: 2
```

Figura 12 – Informação mais relevante recolhida através da utilização do *WebMDS*.

Para os nós de um mesmo sítio, geridos pelo *PBS*, foi necessário gerar chaves públicas *DSA*, de forma a permitir a cópia de ficheiros entre nós. O *PBS* usa preferencialmente o serviço *RSH*, mas também é possível utilizador o serviço *SSH.* Este último permite a autenticação através da apresentação de chaves públicas *DSA,* não obrigando a introdução de palavra-passe.

Coordenação distribuída

No *Globus Toolkit*, a descoberta e coordenação de recursos fica a cargo do serviço *GRAM*, sendo as funcionalidades expostas através do conjunto de *Web-Services WS-GRAM*. O *GRAM* estabelece a ponte entre serviços de gestão de clusters (entre outros) e escalonadores globais.

No ambiente protótipo montado, o *GRAM* foi configurado de duas formas, dependendo da máquina em questão. Nas máquinas onde foi instalado o coordenador do *PBS* (*gtnode-2* e *gtnode-4)* foi feita a configuração integrando o *GRAM* e o *PBS*. Em todas as máquinas foi também feita a configuração que permite o lançamento de tarefas na própria máquina onde está instalado o *Globus Toolkit*, através da criação de processos locais. Para que o serviço *WS-GRAM* pudesse lançar tarefas em nome de utilizadores da *Grid*, foi também necessário atribuir privilégios especiais ao utilizador local criado para o efeito e inserido na *VO* simulada.

Serviços de informação

O serviço *MDS-Index* do *Globus Toolkit* permite a consulta periódica de informação de estado dos recursos. Estes recursos não são exclusivamente computacionais, podendo ser serviços de armazenamento de dados, por exemplo. A informação dos serviços *RFT* é directamente publicada no índice. Quanto à informação relativa ao estado de cada uma das máquinas, esta é publicada de duas formas. Como cada máquina tem uma instalação do *Globus Toolkit*, a informação de estado é periodicamente publicada no seu índice local. Para além desta, também é publicada informação recolhida pelo *Ganglia Monitor* e *PBS*, por cada um dos dois sítios. Para tal, foi necessário configurar o módulo *UsefulRP*, disponibilizado pelo *Globus Toolkit*.

Para evitar que um escalonador tenha de consultar múltiplos serviços de índice, o *MDS-Index* pode ser configurado de forma hierárquica. Assim, um serviço pode agregar informação de múltiplos serviços, disponibilizando informação agregada aos consumidores da mesma. No ambiente montado, o serviço *MDS-Index* da máquina *gtnode-1* foi configurado como sendo o agregador dos serviços das outras máquinas. Desta forma, o escalonador global instalado (secção 3.4*)* consulta este serviço para determinar qual o sítio e/ou máquina responsável pela execução de cada tarefa e quais os serviços que disponibilizam determinados dados para transferência. No entanto, esta informação tem sempre algum atraso, relativamente ao estado efectivo das máquinas. Esta situação é inevitável e característica de ambientes *Grid.*

Para análise humana dos dados publicados no *MDS-Index,* foi instalada a ferramenta *WebMDS*, que transforma os dados publicados num formato *HTML*. Um exemplo de alguns dos

dados apresentados pode ser visto na Figura 12. Esta ferramenta permitiu visualizar e seleccionar os dados relevantes recolhidos para a posterior a análise de disponibilidade.

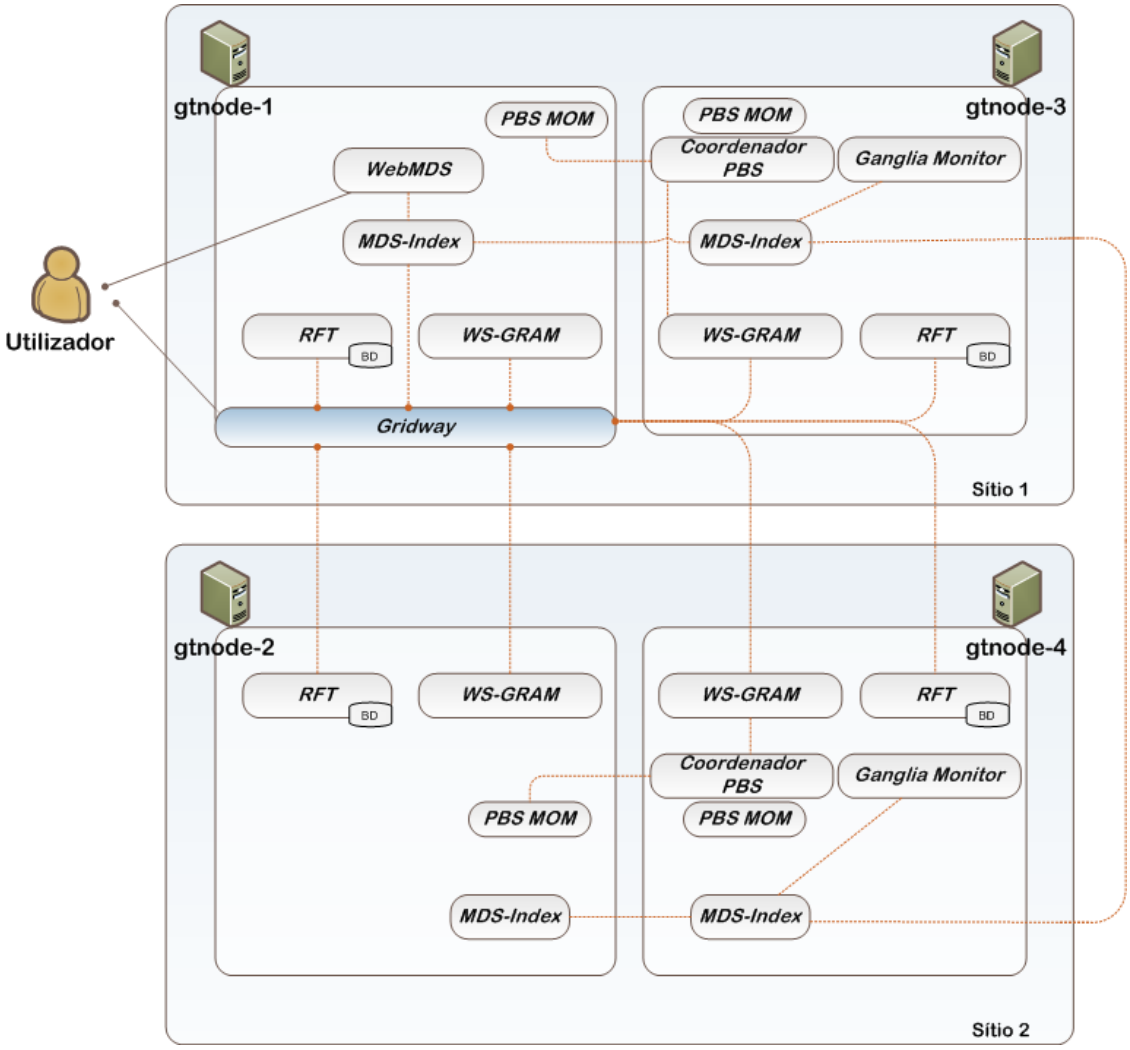

Figura 13 - Diagrama simplificado do ambiente Grid protótipo.

Outros serviços

Os serviços referidos são apenas alguns dos que o *Globus Toolkit* providencia. Os serviços de delegação *(DS)* e localização de réplicas de ficheiros *(RLS)* não obrigaram a nenhuma configuração adicional, ficando operacionais após instalação. Alguns dos serviços não foram utilizados, como é o caso do serviço *MDS-Trigger*.

3.4 Escalonamento global

No que diz respeito ao escalonamento global, a ferramenta seleccionada foi o escalonador *Gridway* (versão 5.2.3). Para além da fácil integração com o *Globus Toolkit*, até por fazer parte da incubadora de projectos da *Globus Alliance*, o *Gridway* apresenta também documentação que não só permite instalação fácil, como também potencia o desenvolvimento de aplicações *Grid*, nomeadamente pela adopção da *DRMAA.* Para além disso, e ao contrário do *Condor-G*, o *Gridway* permite a adopção de um módulo de escalonamento externo. Desta forma, torna-se possível desenvolver um escalonador que beneficie da análise desenvolvida neste trabalho, relativamente à previsão de disponibilidade em ambientes *Grid*.

A instalação do *Gridway* pode até ser feita em conjunto com a instalação do *Globus Toolkit*. No entanto, e para não obrigar à escolha de um dos *middlewares* compatíveis em particular, a instalação foi realizada em separado na máquina *gtnode-1*[5]. A configuração é relativamente simples e resumiu-se à indicação de quais os *URLs* para os serviços *MDS-Index* pretendido, bem como quais as versões dos serviços *RSL e WS-GRAM* disponíveis.

Após configuração, o *Gridway* pode ser disponibilizado como *daemon* a todos os utilizadores, ou pode ser instanciado por cada utilizador. Optou-se pela disponibilização comum a todos os utilizadores, para simular um cenário real, apesar da existência de apenas um utilizador activo. Tendo-se concluído a configuração do *Gridway*, o ambiente *Grid* protótipo ficou operacional para utilização, podendo uma representação simplificada do mesmo ser vista na Figura 13, que apresenta os serviços mais significativos instalados em cada uma das máquinas.

3.5 Bateria de carga

A recolha de informação de uma *Grid* sem qualquer actividade tornar-se-ia num processo desnecessário, à imagem do que aconteceria com a análise da disponibilidade ao longo do tempo. Para simular a utilização real de uma *Grid,* tornou-se essencial a submissão periódica de tarefas. À imagem do trabalho descrito em [Herrera, J. et al. 2004][6], também foi utilizado um conjunto de *benchmarks* denominado *NAS Parallel Benchmarks* [Bailey, D. et al. 1991], desenvolvido pela divisão *NAS* da *NASA*, para avaliar o desempenho de sistemas paralelos. Foram seleccionados três desses benchmarks: *LU, SP e BT*. A escolha recaiu apenas nestes por pretenderem simular

[5] Optou-se por esta máquina por ser aquela com mais memória *RAM* disponível, tentando assim o diminuir o impacto desta ferramenta na disponibilidade do sistema.

[6] Os autores tiveram a gentileza de ceder o código fonte utilizado na abordagem descrita no artigo, que serviu de orientação para o mecanismo de submissão periódica de tarefas.

aplicações científicas exigentes ao nível do processamento, ao contrário dos restantes. No entanto, foram escolhidos os parâmetros menos exigentes, para evitar que a execução de cada tarefa se prolongasse por várias horas. Foi também desenvolvido um *daemon* de submissão de tarefas, responsável por entregar tarefas para execução ao escalonador *Gridway.* Este programa gera números de vírgula flutuante, entre 0.0 e 1.0, de forma aleatória a cada 10 segundos. Este valor foi escolhido de forma a não causar carga excessiva relacionada com a própria criação da bateria de carga, na máquina *gtnode-1*.

Para simular a utilização de aplicações com características diferentes, foram criados três perfis. O primeiro perfil aplicacional corresponde a uma aplicação utilizada durante todo o dia, mas com baixa probabilidade de utilizar a *Grid*, lançando apenas uma tarefa de cada vez. O segundo perfil corresponde a uma aplicação utilizada durante o período laboral, com picos de utilização a meio da manhã e tarde, bem como uma pausa para almoço. Este perfil pode levar à submissão de entre uma ou duas tarefas simultaneamente, dependendo do número aleatório gerado. Por fim, o último perfil corresponde a uma aplicação intensiva utilizada durante o período nocturno, como é o caso de um processo de *ETL*, em que várias tarefas complexas são lançadas a meio da noite e se prolongam até ao início da manhã. Este último perfil obriga à execução de quatro tarefas em simultâneo. Todos os perfis correspondem a aplicações executadas apenas durante os dias úteis da semana, ficando a *Grid* inactiva durante os fins-de-semana.

Para cada perfil aplicacional criado, e sempre que o número obtido periodicamente é igual ou inferior ao valor de probabilidade atribuído ao período temporal presente, é gerado um conjunto de tarefas que posteriormente são submetidas utilizando o *Gridway*.

	0h	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h	11h
24h	.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025
Laboral	0	0	0	0	0	0	0	0	.005	.02	.02	.03
Nocturno	.08	.05	.02	0	0	0	0	0	0	0	0	0
	12h	13h	14h	15h	16h	17h	18h	19h	20h	21h	22h	23h
24h	.0025	.0025	.0025	0.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025	.0025
Laboral	.04	0	.01	0.03	.02	.015	0	0	0	0	0	0
Nocturno	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 4 - Probabilidade de lançamento de tarefas para cada um dos perfis, por hora.

Os valores de probabilidade de lançamento (Tabela 4), bem como os perfis aplicacionais, foram ajustados de forma a permitir que o ambiente *Grid* tivesse capacidade de resposta para

executar as tarefas lançadas, sem aumentar a fila de tarefas indefinidamente. Por outro lado, pretendeu-se sobrecarregar a *Grid* em alguns períodos, alternando com períodos de baixa e média actividade, fazendo com que uma análise de previsão de disponibilidade tenha real utilidade.

Para a submissão de tarefas, optou-se pelas bibliotecas *DRMAA* disponibilizadas pelo *Gridway*. Desta forma, o código desenvolvido para criação da bateria de carga é independente de todos os componentes do ambiente *Grid*. Assim, poderá ser utilizado qualquer outro escalonador ou *middleware*, desde que seja disponibilizada uma implementação da *DRMAA.* Tal não aconteceria caso se tivesse optado pela utilização da *JSDL*, o que obrigaria à utilização de ferramentas específicas do *Gridway,* diferentes ou inexistentes noutros sistemas.

3.6 Recolha e armazenamento de estado

De forma a recolher e armazenar informação de estado de cada uma das máquinas do ambiente *Grid*, foi desenvolvido um segundo *daemon*, que consulta periodicamente serviço *MDS-Index* do *Globus Toolkit*. Entre outros dados menos relevantes, disponibilizados pelo serviço *MDS-*Index, foram escolhidos para recolha os seguintes:

- Nome da máquina.
- Nome e tipo do sistema local para gestão de tarefas (*Fork*[7] ou *PBS).*
- Número total de processadores.
- Número de processadores disponíveis para utilização.
- Frequência de operação dos processadores.
- Número de processos em espera no último minuto.
- Número de processos em espera nos últimos cinco minutos.
- Número de processos em espera nos últimos quinze minutos.
- Quantidade total de memória RAM.
- Quantidade total de memória RAM disponível.
- Número de tarefas em execução.
- Número de tarefas em espera.

A este conjunto de dados foi também acrescentada a data de recolha, uma vez que este é um dado essencial para a criação de modelos de previsão.

[7] O nome deste sistema faz alusão à chamada ao sistema operativo *fork()*, uma vez que para cada tarefa submetida é imediatamente criado um novo processo e iniciada a execução. Não se trata portanto de um sistema evoluído de controlo e escalonamento, mas apenas um mecanismo para lançar processos imediatamente. Este sistema é disponibilizado por omissão pelo *Globus Toolkit*.

```
<ns1:GLUECE xmlns:ns1="http://mds.globus.org/glue/ce/1.1">
  <ns1:Cluster ns1:Name="My Cluster - 2" ns1:UniqueID="My Cluster - 2">
   <ns1:SubCluster ns1:Name="main" ns1:UniqueID="main">
    <ns1:Host ns1:Name="gtnode-2" ns1:UniqueID="gtnode-2">
     <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0" ns1:CacheL1I="0" ns1:CacheL2="0"
          ns1:ClockSpeed="2479" ns1:InstructionSet="x86"/>
     <ns1:MainMemory ns1:RAMAvailable="604" ns1:RAMSize="1011" ns1:VirtualAvailable="1028"
          ns1:VirtualSize="1445"/>
     <ns1:OperatingSystem ns1:Name="Linux" ns1:Release="2.6.22-14-generic"/>
     <ns1:Architecture ns1:SMPSize="1"/>
     <ns1:FileSystem ns1:AvailableSpace="4929" ns1:Name="entire-system" ns1:ReadOnly="false"
          ns1:Root="/" ns1:Size="8231"/>
     <ns1:NetworkAdapter ns1:IPAddress="192.168.5.2" ns1:InboundIP="true" ns1:MTU="1500"
          ns1:Name="gtnode-2" ns1:OutboundIP="true"/>
     <ns1:ProcessorLoad ns1:Last15Min="0" ns1:Last1Min="0" ns1:Last5Min="0"/>
    </ns1:Host>
   </ns1:SubCluster>
  </ns1:Cluster>
  <ns1:ComputingElement ns1:Name="default" ns1:UniqueID="default">
    <ns1:Info ns1:GRAMVersion="4.0.6" ns1:HostName="gtnode-2" ns1:LRMSType="Fork"
         ns1:LRMSVersion="1.0" ns1:TotalCPUs="1"/>
    <ns1:State ns1:EstimatedResponseTime="0" ns1:FreeCPUs="1" ns1:RunningJobs="0"
         ns1:Status="enabled" ns1:TotalJobs="0" ns1:WaitingJobs="0" ns1:WorstResponseTime="0"/>
    <ns1:Policy ns1:MaxCPUTime="-1" ns1:MaxRunningJobs="-1" ns1:MaxTotalJobs="-1"
         ns1:MaxWallClockTime="-1" ns1:Priority="0"/>
  </ns1:ComputingElement>
</ns1:GLUECE>
```

Figura 14 – Exemplo de informação recolhida numa consulta ao *MDS-Index.*

A comunicação com o *MDS-Index* é feita via *Web-Services*, respeitando a norma *WS-ResourceProperties.* Esta norma tem como objectivo normalizar a terminologia, conceitos e operações necessárias para publicar as propriedades de um recurso. Cada recurso é também representado por uma norma de *Web-Services* própria: a norma *WS-Resource.* Para obter informações acerca de recursos de computação, deve ser utilizada a operação *GetResourceProperties*.

De forma a simplificar a utilização dos serviços disponibilizados, o *Globus Toolkit* é acompanhado por um conjunto de bibliotecas que incluem *stubs*[8] para comunicar com os serviços. Assim, não é necessário desenvolver ou gerar os *stubs* a partir da especificação dos serviços em *WSDL.* Os pedidos feitos ao *MDS-Index* são acompanhados por uma expressão, que permitem seleccionar a informação desejada.

No caso presente, foi utilizada uma expressão que permite seleccionar apenas informação relacionada com elementos de computação. Estes são os únicos tidos em conta para o cálculo de disponibilidade, uma vez que são também os únicos considerados pelo escalonador *Gridway*, para efectuar as suas decisões. Na Figura 14 é possível ver um exemplo da informação recolhida junto do *MDS-Index*, para uma das máquinas da *Grid.*

Alguns dos parâmetros recolhidos não são publicados nem pelo escalonador *PBS* nem pelo sistema *Fork*, pelo que tiveram de ser ignorados nesta análise. Exemplos desses casos são os valores de tempo esperado de resposta, pior tempo de resposta e todos os valores relacionados com as políticas de escalonamento.

HOSTUNIQUE_ID	gtnode-3	LRMSVERSION_ID	1.0	RAM_TOTAL	503
LRMSUNIQUE_ID	default	CPU_CACHE_L1	0	VM_FREE	490
LRMSTYPE_ID	Fork	CPU_CACHE_L1D	0	VM_TOTAL	925
FULL_DT	12-09-2008	CPU_CACHE_L1I	0	OS_NAME	Linux
YEAR_DT	2008	CPU_CACHE_L2	0	OS_RELEASE	2.6.22-14-generic
MONTH_DT	9	CPU_CLOCK_SPEED	2404	SPACE_FREE	4102
DAY_DT	12	CPU_COUNT	1	SPACE_TOTAL	8231
WEEKDAY_DT	6	CPU_FREE	1	JOBS_RUNNING	0
HOUR_DT	9	CPU_LOAD_15MIN	0	JOBS_TOTAL	0
HOUR24_DT	21	CPU_LOAD_5MIN	6	JOBS_WAITING	0
MINUTE_DT	50	CPU_LOAD_1MIN	22	ENABLED	true
SECOND_DT	5	RAM_FREE	79		

Tabela 5 – Exemplo de um registo referente a uma recolha junto do serviço *MDS-Index.*

Uma vez que a informação no *MDS-Index* não é actualizada imediatamente (por omissão, é actualizada a cada 5 minutos), optou-se pela recolha periódica de dados a cada 10 minutos, também para minimizar a carga na máquina onde o *daemon* foi instalado *(gtnode-1)*. Cada recolha resulta no processamento e armazenamento de dados numa base de dados relacional. Uma vez que o *RFT* obriga à instalação de uma base de dados *PostgreSQL*, optou-se por utilizar o mesmo motor já instalado, criando uma base de dados própria. No entanto, todo o código desenvolvido é

[8] Um *stub* é uma porção de código que disponibiliza exactamente a mesma funcionalidade que um serviço remoto. Assim, permite ao programador abstrair-se de toda a complexidade de comunicação e serialização de dados, fazendo com que o serviço seja utilizado independentemente da sua localização ou meio de comunicação, de forma transparente.

independente do motor de base de dados. Na Tabela 5 é apresentado o exemplo de um registo armazenado após consulta do serviço *MDS-Index*.

Para cada par constituído por nome da máquina *(HOSTUNIQUE_ID)* e nome do sistema de gestão de tarefas (*LRMSUNIQUE_ID)*, é armazenado um registo, contendo as informações mais relevantes recolhidas junto do índice, à data da recolha. Para além destes dados, o significado de cada um dos restantes dados relevantes é apresentado de seguida:

- Número total de processadores – *CPU_COUNT*.
- Número de processadores disponíveis para utilização – *CPU_FREE*.
- Frequência de operação dos processadores – *CPU_CLOCK_SPEED*.
- Número de processos em espera no último minuto – *CPU_LOAD_1MIN*.
- Número de processos em espera nos últimos cinco minutos – *CPU_LOAD_5MIN*.
- Número de processos em espera nos últimos quinze minutos – *CPU_LOAD_5MIN*.
- Quantidade total de memória RAM – *RAM_TOTAL.*
- Quantidade total de memória RAM disponível – *RAM_FREE*.
- Número de tarefas em execução – *JOBS_RUNNING*.
- Número de tarefas em espera – *JOBS_TOTAL*.
- Data e hora de recolha – *FULL_DT.*
- Ano da recolha – *YEAR_DT.*
- Mês da recolha – *MONTH_DT*.
- Dia do mês da recolha – *DAY_DT*.
- Dia da semana da recolha – *WEEKDAY_DT* (Domingo=1; Sábado=7).
- Hora da recolha – *HOUR_DT* (formato 12h).
- Hora da recolha – *HOUR24_DT* (formato 24h).
- Minuto da recolha – *MINUTE_DT*.
- Segundo da recolha – *SECOND_DT*.

3.7 Cálculo de disponibilidade

Os dados recolhidos e armazenados não expressam directamente uma métrica de disponibilidade de cada uma das máquinas. Os parâmetros de carga, que indicam o número de processos em espera nos últimos minutos, expressam de certa forma a falta de disponibilidade do sistema. No entanto, não são suficientes, uma vez que não reflectem a capacidade real da máquina. Por exemplo, se uma máquina poderosa tiver em média 0.5 processos em espera de entre milhares em execução, poderá certamente receber mais trabalho para executar. O mesmo

não poderá provavelmente acontecer com uma máquina com baixo desempenho e com apenas dezenas de processos em execução.

$$(kCPU \times kCPU_{arquitectura} \times CPU_{frequência} \times CPU_{livres} + kMEM \times RAM_{disponível}) \times coef_{disponibilidade}$$

Figura 15 – Fórmula de cálculo de disponibilidade associada a cada registo recolhido.

Para reflectir um valor mais fidedigno de disponibilidade, foi estabelecida uma fórmula de cálculo (Figura 15), que será aplicada a cada um dos registos recolhidos. Esta fórmula considera essencialmente quatro factores: capacidade de processamento dos processadores; quantidade de processadores disponíveis; quantidade de memória *RAM* disponível; carga da máquina.

A capacidade e número de processadores disponíveis são, talvez, os factores mais importantes para a disponibilidade de um sistema. A componente do cálculo de disponibilidade relacionada com os processadores é determinada com base na multiplicação de um factor configurável *(kCPU)* que atribui um peso relativo ao processador no valor final, por uma constante atribuída à arquitectura do processador *($kCPU_{arquitectura}$)*, pela frequência de operação do processador *($CPU_{frequência}$)* e também pelo número de processadores livres em cada instante (*CPU_{livres})*. Dado que a arquitectura e modelo de todos os processadores (virtuais) é exactamente igual, assumiu-se um valor unitário para *$kCPU_{arquitectura}$*. Caso houvesse distinção entre tipos de processador, esta constante poderia permitir fazer a diferenciação em termos de desempenho.

A memória disponível é também determinante para a disponibilidade de um sistema, podendo até ser requisito obrigatório para execução de tarefas. O cálculo da porção de disponibilidade relacionada com memória é feito com base na multiplicação de um factor configurável *(kMEM)* pela quantidade de memória *RAM* disponível no sistema. Após calculadas as componentes de processamento e memória, é feita uma última multiplicação, por um coeficiente de disponibilidade *($coef_{disponibilidade}$)*. Este coeficiente está relacionado com o número de processos em espera nos últimos minutos[9]. Caso a máquina tenha em média mais do que um processo em espera, o coeficiente toma o valor zero. Tal deve-se ao facto de o *MDS-Index* marcar como indisponíveis todas as máquinas cujo valor de processos em espera supere a unidade. No caso de o número de processos em espera ser inferior à unidade, então o coeficiente toma o valor da diferença em relação à unidade, como pode ser visto na Figura 16.

[9] No caso presente, o número de minutos escolhido foi de cinco, por coincidir com o período entre actualizações do *MDS-Index.*

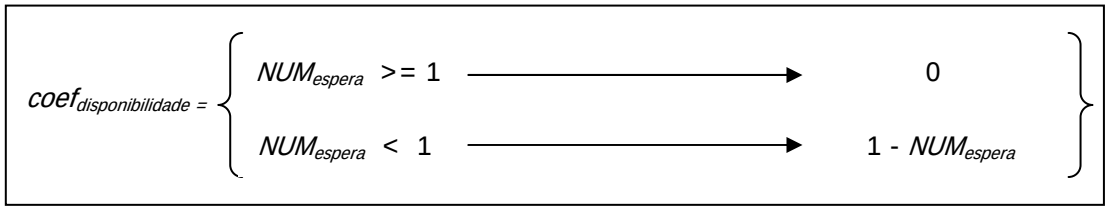

Figura 16 – Cálculo do coeficiente de disponibilidade.

No cálculo de disponibilidade não foram considerados factores como capacidade efectiva de transferência de dados entre nós, nem desempenho no acesso a disco. Estes factores ajudariam certamente nas decisões de escalonamento, mas não são considerados por escalonadores como o *Gridway*. Para além disso, dada a homogeneidade entre máquinas neste aspecto, a sua recolha seria irrelevante.

Por fim, a fórmula de cálculo apresentado permite estabelecer um equilíbrio entre capacidade de processamento e memória, que pode ser adaptado às necessidades de utilização da *Grid*. Assim, caso o ambiente *Grid* seja principalmente utilizado para oferecer capacidade de processamento, sendo a quantidade de memória disponível menos relevante, é possível atribuir um peso mais substancial à capacidade de processamento, por ajuste de *kCPU*. Caso contrário, é também possível ajustar *kMEM*, atribuindo mais relevância aos valores de memória disponíveis.

Capítulo 4

Previsão de disponibilidade

Com o ambiente *Grid* em estado funcional e com carga simulada, a recolha periódica de informação acerca do seu estado constitui uma fonte válida de dados para uma análise recorrendo a técnicas de mineração de dados. Com este estudo, pretende-se extrair conhecimento a partir dos registos recolhidos, criando modelos de mineração de dados que permitam a previsão e classificação da disponibilidade de ambientes *Grid* ao longo do tempo. Para além de uma análise por cada máquina, pretende-se examinar o comportamento global de todas as máquinas em conjunto.

Ao longo deste capítulo serão apresentadas as ferramentas que foram seleccionadas para suportarem a análise referida, bem como os mecanismos de preparação prévia de dados, os algoritmos de mineração de dados utilizados, ou como os modelos de previsão de disponibilidade foram criados. Por fim, são apresentados os resultados obtidos através da aplicação dos modelos de previsão a dados de estado do ambiente *Grid.*

4.1 Ferramentas para construção de modelos de mineração de dados

De seguida serão apresentadas as ferramentas de mineração de dados analisadas para a aplicação de algoritmos e criação de modelos de mineração de dados - foram apenas consideradas ferramentas que estão disponíveis de forma gratuita.

Weka

Desenvolvido na Universidade de Waikato, na Nova Zelândia, o *Weka* [W17] é constituído por um conjunto de ferramentas e um rol extenso de implementações de algoritmos de mineração de dados gratuitamente disponíveis. Este conjunto inclui algoritmos de classificação, regressão, segmentação e criação de regras de associação. Estas implementações podem ser utilizadas de forma programática, através de uma *API Java* disponibilizada para ao efeito, ou através das ferramentas incluídas no *Weka*. Esta suite suporta também os vários estágios da análise de dados, desde o pré-processamento de dados, até à visualização.

Os dados a analisar com o *Weka* podem estar armazenados numa qualquer base de dados acessível por *Java DataBase Connectivity (JDBC)* ou num ficheiro *ARFF*, num formato de texto próprio. Já o modo de interface com utilizador disponibilizado é simples, mas a sua usabilidade deixa um pouco a desejar.

RapidMiner

O *RapidMiner* [W18], anteriormente conhecido por *YALE*, é uma ferramenta de mineração de dados disponibilizada em versões gratuitas e também comerciais. Esta é considerada a ferramenta gratuita e *open-source* com maior taxa de utilização em todo o mundo.

A sua utilização é relativamente fácil e intuitiva, sendo apoiada por documentação de boa qualidade. A interface gráfica permite a rápida construção de modelos, sendo que são disponibilizados mais de 400 operadores para o efeito, incluindo algoritmos, módulos de pré-processamento, filtragem, transformação de dados, representação gráfica e avaliação de desempenho de modelos. Para além dos algoritmos disponibilizados directamente pelo *RapidMiner*, são também importados todos os algoritmos presentes no *Weka*, sendo assim possível retirar partido do melhor que ambas as ferramentas têm para oferecer. Para além da interface gráfica, é também possível utilizar o *RapidMiner* através da linha de comandos, possibilitando desta forma a utilização de forma automatizada, sem intervenção do utilizador.

Para além do que já foi referido, o *RapidMiner* tem também grande flexibilidade no acesso a dados, sendo suportadas diversas fontes de dados, incluindo ficheiros *ARFF, Excel, CSV, BibTex, SPSS, dBASE* e também múltiplas bases de dados, via *JDBC*. Assim como o *Weka*, também o *RapidMiner* é desenvolvido em *Java*, podendo ser instalado em múltiplas plataformas, como *Linux, Windows, MacOS,* entre outras.

Pelos motivos apresentados, e por herdar toda as funcionalidades também disponibilizadas *Weka*, o *RapidMiner* foi seleccionado para o estudo descrito neste documento.

4.2 Preparação de dados

Para realizar a análise descrita seguidamente, foram armazenados dados correspondentes a uma semana completa de operação da *Grid*, num total de cerca de 7 mil e 600 registos. No entanto, e devido à morfologia da *Grid*, optou-se por reduzir esse número para cerca de 3 mil e 800 registos. A justificação fica a dever-se à forma como as máquinas ficaram interligadas. Como estas foram configuradas com os sistemas *Fork* e *PBS*, na realidade são recolhidos dados praticamente em duplicado, a cada instante. Por exemplo, a máquina *gtnode-1* é publicada via *Fork*, sendo também publicada em conjunto com a máquina *gtnode-3*, via *PBS*. O mesmo acontece com as máquinas *gtnode-2* e *gtnode-4.* Assim, foram apenas seleccionados os dados publicados via *Fork*, por permitirem observar o estado de cada máquina individualmente, para além de uma análise global. Caso se optasse pelos dados publicados via *PBS,* a análise da disponibilidade global da *Grid* teria certamente o mesmo significado, não sendo possível diferenciar entre as várias máquinas de cada sítio.

Posteriormente, os dados dos vários conjuntos foram agregados por máquina e por hora. Não se optou por uma granularidade maior, devido ao período substancial de tempo entre recolhas (10 minutos). Com uma granularidade mais fina, obteríamos resultados com base em dados baseados de recolhas muito similares ou repetidas. Uma granularidade menor evitaria a possibilidade de ser feita previsão de disponibilidade para períodos de uma hora, devido ao baixo número resultante de recolhas.

A agregação de dados foi feita através de vistas na base de dados, calculando a média dos valores numéricos obtidos para cada máquina, no período de uma hora. Para além dos valores recolhidos, foi também obtida a média do valor calculado de disponibilidade, utilizando o modelo descrito na secção 3.7. Para a análise de disponibilidade global, foi feita uma agregação adicional, através da soma dos valores obtidos para cada uma das máquinas individualmente. O valor de disponibilidade ficou assim armazenado com o nome de *AVAILABILITY.*

Dado que alguns algoritmos não suportam cálculos baseados apenas em valores numéricos, os valores de disponibilidade tiveram também de ser apresentados em categorias (ou classes). Assim, cada valor de disponibilidade foi dividido pelo valor máximo de disponibilidade recolhido, multiplicado por dez e posteriormente arredondado (Figura 17). Conseguiu-se desta forma criar 6 categorias distintas (*C0 – C5*), correspondendo *C0* à disponibilidade mínima e *C5* à disponibilidade máxima. Caso o número de categorias fosse maior, maior seria também a proximidade entre a classe atribuída e o valor real. No entanto, tornar-se-ia mais fácil errar na previsão, devido à menor diferença entre os valores de cada classe. Se pelo contrário, se optasse por um número de

classes menor, a previsão poderia ser feita de forma com mais precisão, mas no limite, todas as instâncias seriam classificadas em apenas uma classe, perdendo-se assim a utilidade do processo.

$$round\left(\frac{disponibilidade \times 5}{disponibilidade_{máxima}}\right)$$

Figura 17 – Fórmula utilizada na categorização de valores de disponibilidade.

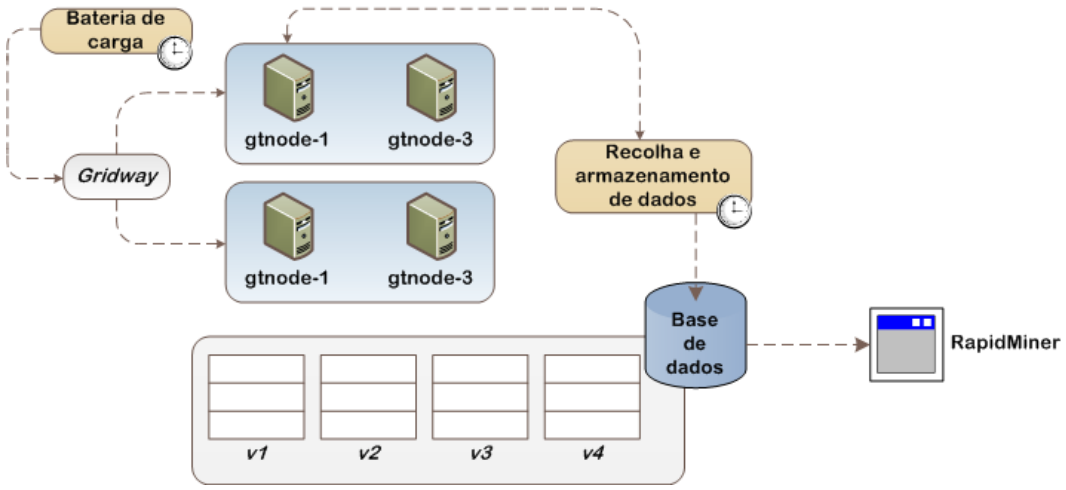

Figura 18 - Ilustração da produção, recolha, armazenamento e processamento de dados.

Como resultado da preparação de dados, ficaram disponíveis quatro vistas que serviram como fonte de dados para criação de modelos de mineração de dados. A primeira das vistas *(v1)* dá acesso à disponibilidade por máquina, bem como aos factores que a originaram (informação relativa ao processador, memória, entre outros). Uma segunda vista *(v2)* dá uma visão global, agregando dados de todas as máquinas. As duas vistas restantes *(v3* e *v4)* apresentam os dados de disponibilidade categorizados, por máquina e de forma global, respectivamente. Para além disto, as vistas permitem recalcular os valores de disponibilidade, através do ajuste dos parâmetros *kCPU* e *kMEM*, também armazenados na base de dados e passíveis de alteração.

A Figura 18 apresenta, de forma esquemática, todo o processo desde a criação de carga na *Grid* através da submissão de tarefas via *Gridway*, posterior recolha de dados e armazenamento numa base de dados, para posterior utilização do *RapidMiner*, de forma a construir modelos de mineração de dados.

4.3 Algoritmos de mineração de dados

Os modelos de mineração de dados têm actualmente aplicações muito diversas, desde o combate à fraude à extracção de conhecimento em aplicações ligadas à medicina. Essa flexibilidade é, em parte, explicada pelo vasto conjunto de algoritmos disponíveis, bem como técnicas de preparação de dados, métodos de visualização e avaliação de resultados.

Os algoritmos de mineração de dados podem ser divididos em dois grupos, tendo em conta significado atribuído às variáveis consideradas. Os algoritmos de aprendizagem supervisionada distinguem as variáveis entre variáveis de previsão e variáveis de resposta. O objectivo destes algoritmos é estabelecer uma relação entre as diversas variáveis de previsão, de forma a prever o valor da variável de resposta. Por sua vez, os algoritmos de aprendizagem não supervisionada são mais orientados para a exploração de dados, não fazendo distinção entre variáveis. Como propósito, estes algoritmos pretendem encontrar relações quando não se pretende determinar um valor de resposta em concreto. Este tipo de algoritmos é habitualmente utilizado para obter um conhecimento inicial dos dados, partindo-se depois para algoritmos supervisionados.

No contexto do estudo apresentado, e de entre os algoritmos disponibilizados pelo *RapidMiner*, foram seleccionados algoritmos de três tipos: *clustering*, regras de associação e árvores de decisão. Seguidamente é feita uma breve descrição de cada tipo.

4.3.1 Algoritmos de *clustering*

Os algoritmos de *clustering* são de aprendizagem não supervisionada e permitem estabelecer agrupamentos naturais de dados, tendo em conta similaridade entre valores [Witten & Frank 2005]. Cada registo é assim colocado num ou mais grupos (ou *cluster*), juntamente com os registos mais similares ou através do cálculo de uma probabilidade de vir a pertencer a esse grupo.

O *K-Means* é um dos algoritmos clássicos mais populares de *clustering.* A sua utilização é relativamente simples e obriga a que cada registo (ou instância) pertença a um único *cluster*. Para além disso, o algoritmo apenas permite a utilização de variáveis numéricas, obrigando a transformação prévia.

O processo de execução deste algoritmo é antecedido pela escolha do seu único parâmetro (*k),* correspondente ao número de *clusters* a criar, de forma a agrupar as instâncias. Este parâmetro tem valor recomendado correspondente à raiz quadrada de metade do número de instâncias, não sendo obrigatoriamente o valor ideal. Seguidamente são aleatoriamente criados *k clusters*, sendo os seus pontos centrais determinados ou obtidos através da geração de pontos. Em seguida, as instâncias são atribuídas ao *cluster* cujo centro lhe esteja mais próximo. O valor de

proximidade é calculado a partir da distância Euclidiana[10]. Terminado este passo, são repetidamente calculados os novos centros e atribuídas as instâncias aos *clusters* respectivos, até que os centros estabilizem e se mantenham constantes após iterações sucessivas.

Este algoritmo, em si, é relativamente simples e eficaz, minimizando o total da distância Euclidiana. No entanto, este mínimo é local a cada distância e não corresponde garantidamente ao mínimo global, que equivale a uma situação óptima. Para além disso, e devido à possível aleatoriedade, é imprevisível, podendo ser necessário efectuar várias execuções até encontrar o melhor modelo. Os resultados obtidos por este algoritmo são tipicamente analisados de forma visual, sob a forma de gráficos. A análise permite determinar quais os *k* perfis instâncias mais prováveis de serem observados. Para além disso, permite identificar quais as instâncias com maior probabilidade de mudarem de *cluster*/perfil, por serem as mais distantes dos centros do respectivo *cluster*.

No contexto de previsão de disponibilidade num ambiente *Grid*, espera-se que a utilização do algoritmo permita identificar *k* períodos típicos de disponibilidade similar. Adicionalmente, espera-se determinar quais as instâncias recolhidas com maior probabilidade de mudança de *cluster*.

O *RapidMiner* disponibiliza duas implementações baseadas no algoritmo *K-Means*: *KMeans* e *W-SimpleKMeans*. Optou-se pela utilização do algoritmo *KMeans*, por permitir uma maior facilidade de análise de resultados, devido a um maior conjunto de visualizações gráficas disponíveis, sendo os dois algoritmos em tudo similares nos restantes aspectos.

4.3.2 Algoritmos para a criação de regras de associação

Os métodos de descoberta de regras de associação são largamente utilizados, principalmente numa fase inicial em que não se conhece os padrões procurados. Esta variante de extracção de conhecimento é particularmente popular em casos de *Market Basket Analysis* [Witten & Frank 2005], para encontrar associações entre produtos adquiridos por clientes. É esta a família de técnicas responsável pela famosa associação entre a compra simultânea de fraldas e de cerveja nos hipermercados. Os algoritmos utilizados são não supervisionados e permitem a previsão de qualquer variável, não sendo assim necessário estabelecer uma variável de resposta.

As regras de associação são compostas por uma premissa e uma conclusão. A premissa é uma condição necessária para a verificação de uma conclusão. Ambas podem ser combinações

[10] A distância Euclidiana determina a distância entre dois pontos num espaço de dimensão *N*. O cálculo é feito através da raiz quadrada do somatório do quadrado da diferença entre cada uma das *N* coordenadas/valores dos pontos.

lógicas de várias expressões, relacionando variáveis entre si e/ou com valores constantes. Na Figura 19 é apresentado um exemplo de uma regra de associação: se o tempo actual é de sol e apenas há vento fraco, então é um bom dia para ir à praia.

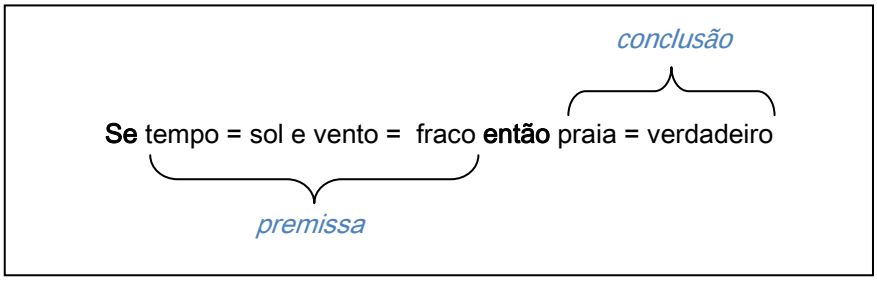

Figura 19 – Exemplo de uma regra de associação.

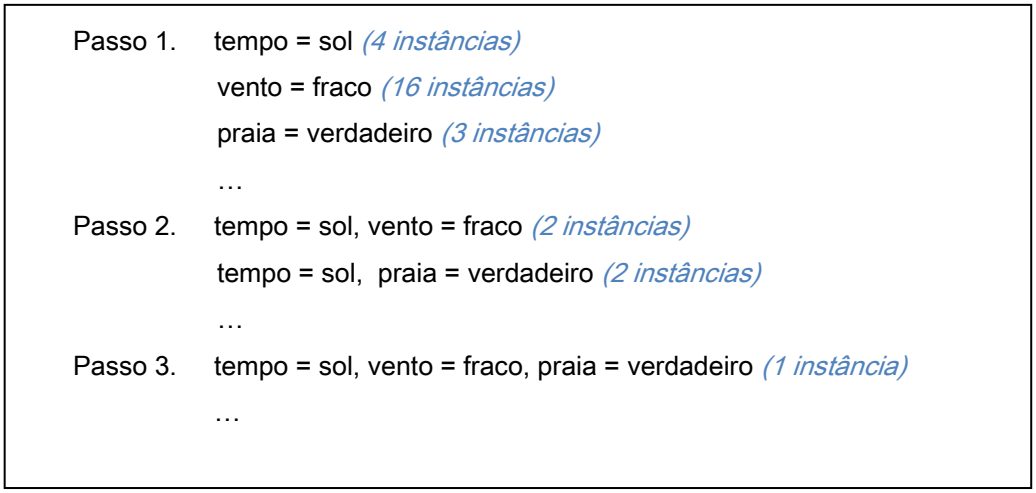

Figura 20 – Divisão de condições em conjuntos de itens.

Dado que um pequeno conjunto de instâncias pode permitir deduzir múltiplas regras de associação, é necessário avaliar as regras quanto à sua qualidade. As principais métricas de qualidade de uma regra são os seus valores de cobertura e de confiança. A cobertura de uma regra indica o número (ou percentagem do total) de instâncias em que se verifica tanto a premissa como a conclusão, de forma independente. A confiança de uma regra indica o número (ou percentagem do total) de instâncias em que se se verifica a premissa, então também se verifica a conclusão. Através do estabelecimento de mínimos para ambos os valores, é possível eliminar regras menos significativas, reduzindo assim o conjunto de regras de associação.

O primeiro passo para a criação de uma regra de associação passa pela formação de conjunto de itens (Figura 20). Inicialmente são criados conjuntos de apenas uma condição, tendo

em conta o limite mínimo de cobertura. A partir dos conjuntos de instâncias de um único elemento são criados conjuntos de dois elementos, em que ambas as condições dos dois conjuntos se verificam. Novamente são considerados apenas os conjuntos com um mínimo de cobertura. O processo repete-se até que não seja possível estabelecer conjuntos de maior dimensão.

A partir dos conjuntos criados, são geradas regras de associação, com determinada cobertura e confiança, combinando as condições dos vários conjuntos em premissas e conclusões. Na Figura 21 é apresentado um exemplo das regras geradas a partir de um conjunto.

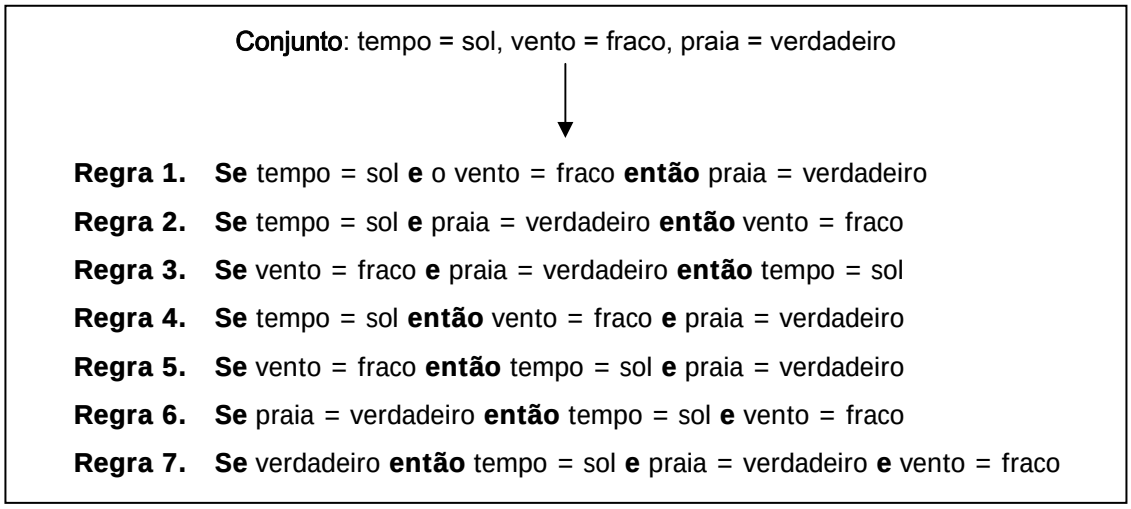

Figura 21 – Geração de regras de associação a partir de um conjunto de itens.

No âmbito do estudo de previsão de disponibilidade num ambiente *Grid*, as regras de associação podem contribuir para determinar o estado da *Grid*, sempre que se verifiquem as premissas das regras calculadas, permitindo potencialmente prever, por exemplo, o impacto que a execução de uma tarefa adicional pode causar na *Grid*.

São duas as possibilidades para criar regras de associação através do *RapidMiner*. Pode ser utilizado o algoritmo *W-Apriori* fornecido pelo *Weka* ou utilizar o algoritmo *FPGrowth* em conjunção com o algoritmo *AssociationRuleGenerator*.

O *W-Apriori* executa todo o processo, desde a criação de conjuntos de itens à geração de regras. No entanto, a análise de dados é relativamente limitada, sendo apenas possível observar as regras em formato de texto, somente acompanhadas pelo valor de confiança.

Caso se opte para utilização do *FPGrowth,* é necessário aplicar também o algoritmo *AssociationRuleGenerator*, uma vez que o *FPGrowth* apenas calcula os conjuntos de itens, sendo as regras posteriormente criadas pelo *AssociationRuleGenerator*. Ao contrário do que acontece com o

W-Apriori, é neste caso possível visualizar as regras de forma tabular, acompanhadas por um conjunto adicional de informações, incluindo cobertura e confiança. Assim, foi esta a opção seleccionada para cálculo de regras de associação.

4.3.3 Algoritmos para criação de árvores de decisão

As árvores de decisão são construídas com algoritmos de aprendizagem supervisionada, sendo assim necessário definir uma variável de resposta. Esta variável pode ter como significado, por exemplo, uma determinada classe de valores. Nesse caso, a árvore é utilizada para a construção de um modelo de classificação. Os algoritmos de construção de árvores decisão são bastante utilizados devido à grande facilidade de leitura dos resultados.

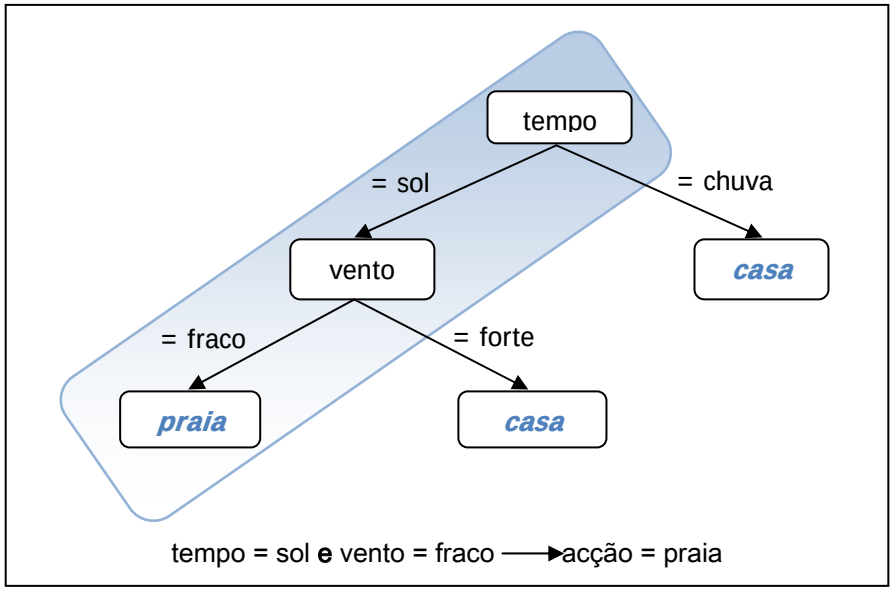

Figura 22 – Exemplo de uma árvore de decisão e de uma condição para previsão.

A construção de árvores de decisão é um processo recursivo, envolvendo criação de ramificações de decisão [Witten & Frank 2005]. Cada um dos ramos da folha indica uma condição relativa ao valor do nó superior da árvore. Como resultado, cada uma das folhas expressa valor ou classe prevista, sendo o caminho ao longo dos ramos, desde a raiz até às folhas, que determina o conjunto de condições necessárias para que a previsão se materialize (Figura 22).

Os aspectos mais importantes envolvidos no processo de criação de árvores de decisão são a escolha das raízes da árvore e de cada sub-árvore e quais os ramos a serem criados após escolhida a raiz. Estas escolhas determinam a profundidade da árvore, bem como o número de nós

e quais as previsões possíveis de obter, pelo que são da maior relevância. São vários os critérios que podem ser utilizados, relativos ao nível de informação de cada um dos ramos.

A cada sub-árvore a criar é atribuído um nível de informação, que é medido em *bits*. Ao contrário do que possa parecer, estes *bits* podem ser representados por valores fraccionados. Quanto maior o número de *bits* de uma sub-árvore, maior a quantidade de informação ainda necessária para chegar a uma decisão. O cálculo é feito com base em funções que respeitem as seguintes três regras:

- Sempre que todos os valores de previsão obtidos através da travessia para um ramo são idênticos então o nível de informação é 0.
- O nível de informação é tanto maior quanto maior for o número de valores de previsão alcançados.
- O nível de informação é cumulativo, ou seja, independentemente das sub-ramificações feitas e do seu número, o nível de informação da raiz é sempre igual.

Uma das funções utilizadas para o cálculo do nível de informação é a função de entropia, representada na Figura 23. Habitualmente são utilizados logaritmos de base 2. Cada valor p_n corresponde à fracção de valores previstos para a variável de resposta, na sua sub-árvore. Por exemplo, se numa sub-árvore estiverem contidos cinco valores previstos com dois tipos distintos, com dois e três elementos de cada tipo, então os valores de p_1 e p_2 são de 2/5 e 3/5 respectivamente, levando a um nível de informação de 0.971 *bits*.

$$entropia\,(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 - \ldots - p_n \log p_n$$

Figura 23 – Cálculo da função de entropia.

Outro valor relevante para além do nível de informação é o ganho de informação de cada ramo. Este valor corresponde à diferença entre o nível de informação de um ramo em relação à soma da média do nível de informação de cada um dos seus sub-ramos. Na prática, este valor é relacionado com quanto um ramo contribui para a previsão, permitindo diferenciar as várias instâncias consideradas. Na Figura 24 é apresentado um exemplo em que é calculada tanto o nível de informação de cada sub-árvore como o ganho da raiz.

Alguns algoritmos efectuam a ramificação com base no valor mais alto de informação de cada potencial raiz, dado que quanto maior o nível de informação, maior a contribuição da ramificação para distinguir as instâncias. No entanto, esta abordagem pode levar a escolha de variáveis como por exemplo identificadores, em que todos os valores são distintos, eliminando a

validade da previsão para novos dados. Para tal, é por vezes utilizada a relação entre ganho de informação e nível de informação de cada nó para efectuar a ramificação.

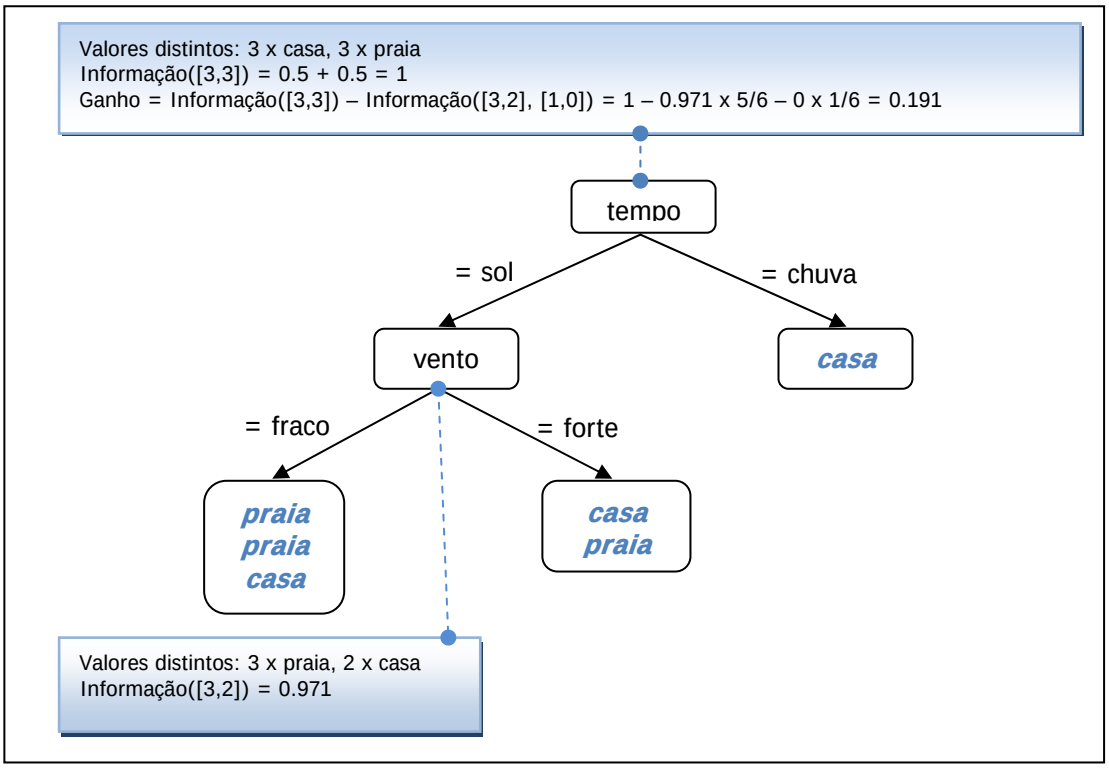

Figura 24 – Informação e ganho dos ramos de uma árvore de decisão, utilizando entropia.

De forma a evitar que a árvore se ramifique indefinidamente, é possível estabelecer um nível de profundidade máximo e também um nível de confiança mínimo. Assim, se o número de ramos que separam a sub-árvore a ser avaliada e a raiz da árvore for imediatamente inferior ao nível de profundidade a ramificação é interrompida na sub-árvore. O mesmo acontece sempre que a ramificação cria sub-árvores, em que alguma caso houvesse mais alguma ramificação, seria ultrapassado o limite mínimo de confiança. Este valor de confiança indica a percentagem de instâncias a que a previsão se aplica, por travessia na árvore, até chegar à sub-árvore a criar.

Para a previsão de disponibilidade em ambientes *Grid*, espera-se que as árvores de decisão contribuam com a classificação dos períodos de disponibilidade da *Grid*, e de cada máquina individualmente, ao longo do tempo. Desta forma, tornar-se-ia possível determinar qual classe de disponibilidade esperada, caso um conjunto de factores de verificasse em simultâneo.

O *RapidMiner* disponibiliza um algoritmo de cálculo de árvores de decisão, denominado *DecisionTree*, que permite fazer a ramificação com base no ganho de informação e no rácio entre nível e ganho de informação. Assim, e por razões óbvias, foi este o algoritmo utilizado.

4.4 Modelos de mineração de dados

Tendo em conta os algoritmos seleccionados, foram criados três grupos de modelos, numa relação de um para um com os algoritmos: *clustering*, regras de associação e árvores de decisão. Em cada grupo foi feita a análise de disponibilidade agregada global e também por máquina da *Grid*. Os valores escolhidos para as constantes no cálculo de disponibilidade *(kCPU=0.005, kMEM=0.0005)* procuram representar maior importância do processador, em relação ao valor de memória. As constantes têm valores reduzidos, uma vez que tanto a frequência de operação do processador como o número de *megabytes* de memória disponíveis são números relativamente grandes.

4.4.1 Modelos de *clustering*

Sendo um algoritmo não supervisionado, o *KMeans* não obriga a nenhuma fase prévia de treino para que seja utilizado, isto porque não pretende prever uma variável de resposta. Assim, para esta análise, foram considerados os dados do conjunto total de instâncias. Para avaliar visualmente os *clusters*, foi utilizado o operador *SVDReduction*, que permite a redução dos valores das variáveis a *N* dimensões, permitindo representar os *clusters* num plano bidimensional ou tridimensional, por exemplo.

Para análise de disponibilidade por máquina, e após agregação de valores por hora, foram utilizados cerca de 670 registos, recorrendo à vista *v1*. Já para a análise de disponibilidade global da *Grid* foram utilizados cerca de 170 registos, utilizando a vista *v2*.

De forma a facilitar a compreensão dos resultados, foi utilizado *k=5*, como número de clusters a criar pelo *KMeans.* Valores inferiores reduziram a utilidade destes modelos, sendo que valores superiores criariam exponenciais possibilidades de análise.

Uma vez que o *KMeans* apenas suportada variáveis numéricas, os nomes das máquinas foram mapeados removendo a *string "gtnode-"* de cada uma delas. Para além disso, os valores de memória disponível foram divididos pela centena, de forma a permitir uma análise visual mais clara. Caso contrário os valores de memória seriam demasiado superiores aos das restantes variáveis, causando dificuldades devido às escalas dos gráficos. O mesmo aconteceu com o valor relativo ao ano, que foi dividido por mil.

Para avaliar a dispersão dentro e entre *clusters*, foi utilizado o operador *ClusterCentroidEvaluator*, com a métrica *Davies-Boulding.* Esta métrica representa um rácio entre a dispersão dentro de um cluster e a separação entre *clusters*. O valor em módulo é tanto menor quanto mais compactos forem os *clusters* e quanto maior for a distância entre eles.

4.4.2 Modelos de criação de regras de associação

Assim como acontece com os algoritmos de *clustering*, os algoritmos de criação de regras de associação são também não supervisionados. Pelas mesmas razões, não foi necessária uma fase de treino, tendo sido utilizados os dados na íntegra, para todas as instâncias.

O conjunto de instâncias utilizado para a análise de disponibilidade por máquina foi composto por cerca de 670 registos, recorrendo à vista *v1*. Quanto o conjunto tido em conta pela análise de disponibilidade global da *Grid* foi formado por cerca de 170 registos, utilizando a vista *v2*. Nesta análise, foram consideradas todas as colunas disponíveis, apesar de tal criar muitas regras óbvias, cuja filtragem foi feita de forma não automática.

Ambos os modelos foram constituídos utilizando em os algoritmos *FPGrowth e AssociationRuleGenerator*, em sequência. O valor mínimo de cobertura utilizado no *FPGrowth* foi de 0.2, sendo utilizado o valor de 0.5, como valor mínimo de confiança no *AssociationRuleGenerator.*

Dado que o algoritmo *FPGrowth* não suporta valores numéricos, todos os valores numéricos foram pré-processados. Para cada valor de cada variável, foi criada uma variável acessória, com significado verdadeiro caso a variável original tivesse o valor correspondente. Por exemplo, para a variável *CPU_FREE* e para o valor 1.0, foi criada a variável *CPU_FREE_1.0*, com valor verdadeiro se *CPU_FREE=1.0.*

A avaliação de cada regra é feita com base nos valores de suporte e confiança, não sendo necessário utilizar qualquer operador adicional.

4.4.3 Modelos baseados em árvores de decisão

Ao contrário dos dois modelos anteriores, os modelos de árvores de decisão obrigam a uma fase inicial, em que o modelo é treinado, de forma maximizar a sua apetência para calcular o valor da variável de resposta. Assim, numa primeira fase, o modelo foi criado e treinado, com base no conjunto de treino. Posteriormente foi feito um teste com um outro conjunto de dados, sendo o modelo finalmente validado com um terceiro conjunto de dados. Para tal, os dados foram separados aleatoriamente em 3 partes, cada uma delas de diferente cardinalidade.

Aproximadamente 66% dos dados foram atribuídos ao conjunto de teste e os restantes divididos em duas metades de 17%, uma para teste e outra para validação.

Quanto ao número de instâncias, e para a análise de disponibilidade por máquina, foram usadas cerca de 460 instâncias para o conjunto de teste e cerca de 100 para os restantes conjuntos, utilizando a vista *v3* como fonte. Já para a análise global, foram utilizadas cerca de 100 instâncias para o conjunto de teste e aproximadamente 30 instâncias para cada um dos conjuntos remanescentes, recorrendo à vista *v4*.

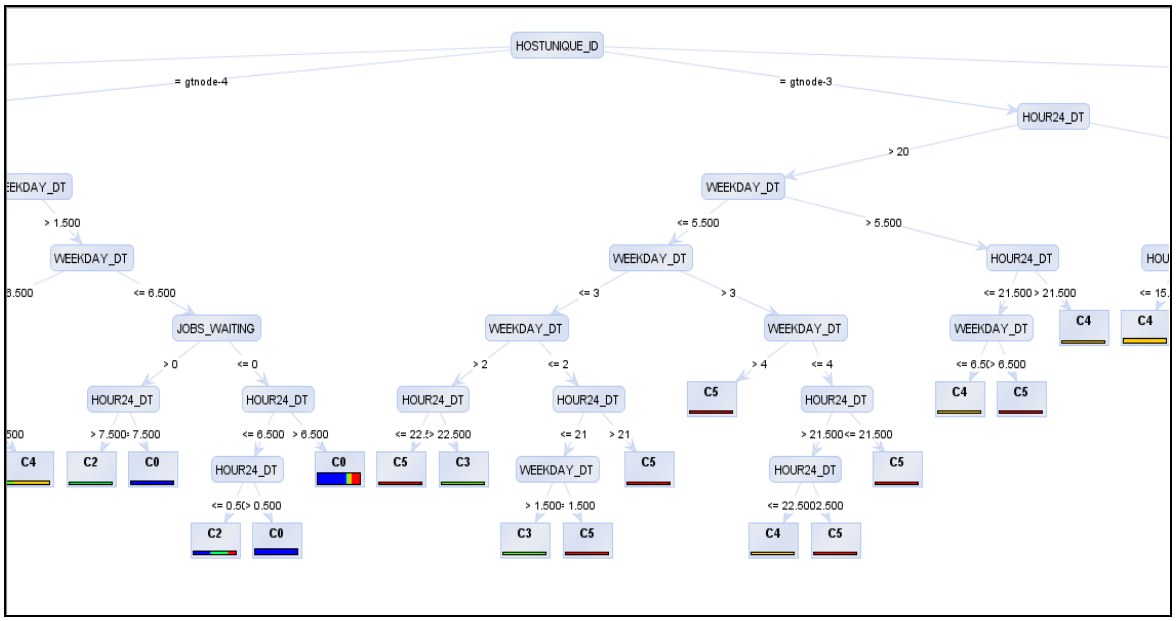

Figura 25 – Exemplo parcial de uma das árvores de decisão criadas.

Para criar os modelos recorreu-se, como já referido, ao algoritmo *DecisionTree*, utilizando o ganho de informação como critério de ramificação, usando valores de ganho de informação mínimo e confiança mínima de 0.005 e 0.25, respectivamente. A altura das árvores foi limitada a 30 ramos. Após a criação do modelo, foi feita a sua aplicação aos conjuntos de teste e validação, sendo avaliada o desempenho de classificação através dos operadores *ModelApplier* e *ClassificationPerformance*. A precisão da classificação foi a métrica utilizada para avaliação de desempenho. Esta métrica indica a percentagem de instâncias de teste e validação que são correctamente classificadas pelos modelos. O conjunto de variáveis considerado foi limitado, de forma a eliminar variáveis relacionadas aritmeticamente. Assim, foram apenas incluídas as seguintes variáveis: *YEAR_DT, MONTH_DT, WEEKDAY_DT, HOUR24_DT, HOSTUNIQUE_ID* (apenas na análise de disponibilidade por máquina), *JOBS_WAITING, JOBS_RUNNING*, e

AVAILABILITY. Esta última resultou do cálculo e categorização do valor de disponibilidade, sendo a variável de resposta do modelo.

Devido ao grande número de previsões, não é possível apresentar senão um exemplo parcial de uma representação gráfica de uma árvore de decisão (Figura 25). Através da travessia dos ramos das árvores, é possível extrair previsões de disponibilidade e as respectivas condições para que a previsão se aplique. A título de exemplo, são apresentadas algumas travessias de cada modelo, na Tabela 6 e na Tabela 7.

Expressão de travessia	Disponibilidade
HOSTUNIQUE_ID = gtnode-1 && HOUR24_DT > 20 && WEEKDAY_DT > 6.500	C2
HOSTUNIQUE_ID = gtnode-3 && JOBS_WAITING > 0 && HOUR24_DT <= 7.500;	C0
HOSTUNIQUE_ID = gtnode-4 && HOUR24_DT <= 19 && WEEKDAY_DT <= 1.500 && HOUR24_DT <= 15.500	C4
HOSTUNIQUE_ID = gtnode-2 && WEEKDAY_DT > 1.500 && HOUR24_DT > 17 && WEEKDAY_DT <= 2.500 && HOUR24_DT > 20.500	C5

Tabela 6 – Exemplos de travessias na árvore de previsão de disponibilidade por máquina.

Expressão de travessia	Disponibilidade
HOUR24_DT > 20 && WEEKDAY_DT > 3 && WEEKDAY_DT > 6	C4
HOUR24_DT > 20 && WEEKDAY_DT <= 3 && HOUR24_DT > 22	C4
HOUR24_DT <= 20 && WEEKDAY_DT > 1.500 && WEEKDAY_DT <= 6.500 && HOUR24_DT <= 8.500 && HOUR24_DT > 0 && WEEKDAY_DT > 4	C0
HOUR24_DT <= 20 && WEEKDAY_DT > 1.500 && WEEKDAY_DT > 6.500 && HOUR24_DT <= 11.500 && HOUR24_DT <= 0	C3

Tabela 7 – Exemplos de travessias na árvore de previsão de disponibilidade da *Grid*.

4.5 Resultados obtidos

Tendo sido recolhidos e tratados os dados necessários, e após a criação dos vários modelos de mineração de dados, foi feita a aplicação de cada modelo a um conjunto de dados definido. Alguns dos modelos permitem um infindável conjunto de representações gráficas e tabulares, pelo que nesta secção apenas são apresentados os resultados obtidos considerados mais relevantes.

Estes têm como alvo de análise a disponibilidade por máquina e também a disponibilidade global da *Grid*.

4.5.1 Modelos de *clustering*

Os resultados apresentados para os modelos de *clustering* são revelados através da enumeração dos pontos centrais de cada um dos *clusters* gerados, bem como contagem de instâncias por *cluster,* métricas de avaliação de cada um dos *clusters* e vários gráficos, com significados distintos. Os gráficos dos pontos centrais apresentam os valores de cada variável relativa a cada ponto. Os gráficos de dispersão apresentam a distribuição das várias instâncias por cada *cluster*. Por fim são apresentados gráficos de relação de várias variáveis com os *clusters* gerados. As variáveis consideradas para análise são as que compõem os pontos centrais.

Disponibilidade por máquina

A Tabela 8 apresenta os dados obtidos relativamente aos cinco *clusters* gerados pelo modelo *clustering* de disponibilidade por máquina. São apresentadas as contagens de instâncias por *cluster*, bem como os valores de cada um dos pontos centrais e o valor do índice *Davis-Bouldin***.**

	Cluster 0	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Todos**
Cardinalidade (nº instâncias)	92	154	154	143	125	668
Pontos centrais						
HOSTUNIQUE_ID	2.446	2.390	2.727	2.545	2.376	
YEAR_DT/100	2.008	2.008	2.008	2.008	2.008	
MONTH_DT	9.000	9.000	9.000	9.000	9.000	
DAY_DT	16.815	16.610	14.565	15.699	16.824	
WEEKDAY_DT	4.043	3.883	3.929	4.168	4.104	
HOUR24_DT	8.935	15.760	7.786	20.483	2.456	
HOUR_DT	8.935	3.760	4.747	8.483	2.456	
CPU_FREE	0.337	0.293	0.995	0.997	0.267	
CPU_COUNT	1.000	1.000	1.000	1.000	1.000	
RAM_FREE/100	4.131	5.766	4.912	5.399	5.426	
RAM_TOTAL/100	11.555	11.638	10.704	11.176	11.176	
JOBS_WAITING	0.594	0.061	0.000	0.000	1.151	
JOBS_RUNNING	0.000	0.000	0.000	0.000	0.000	
AVAILABILITY	1.687	1.206	10.631	11.130	0.936	
Davis-Bouldin						0.651

Tabela 8 - Dados relativos ao modelo de *clustering* de disponibilidade por máquina.

A mesma informação relativa aos dados dos pontos centrais pode ser vista sob a forma de uma representação gráfica, na Figura 26. No eixo horizontal é representado o nome de cada uma das variáveis dos pontos centrais, sendo o seu valor representado no eixo vertical. Os vários *clusters* podem ser distinguidos pela cor das linhas.

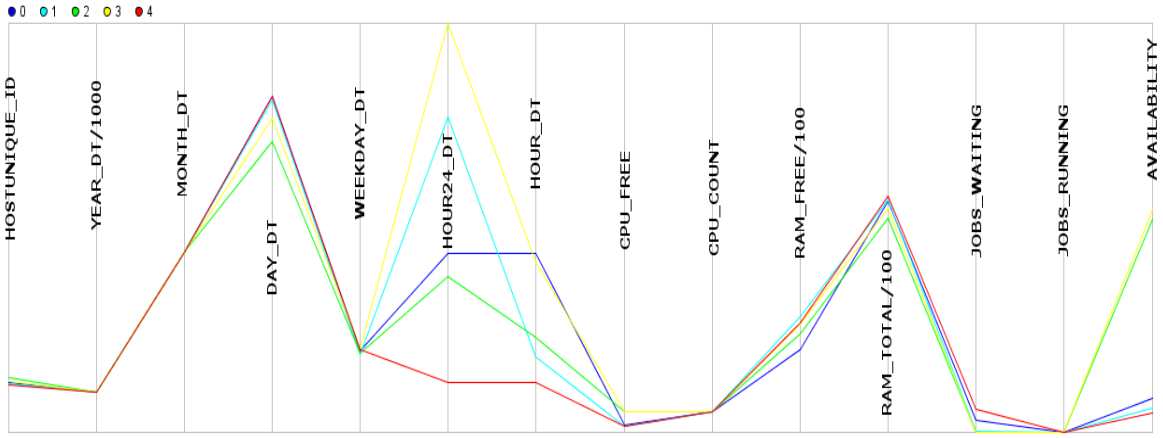

Figura 26 - Pontos centrais - disponibilidade por máquina.

A Figura 27 permite visualizar a dispersão dos *clusters* gerados, através de uma representação *2D* de cada uma das instâncias consideradas. A redução para duas dimensões foi efectuada pelo algoritmo *SVDReduction.* Os vários *clusters* podem ser distinguidos pela cor dos pontos.

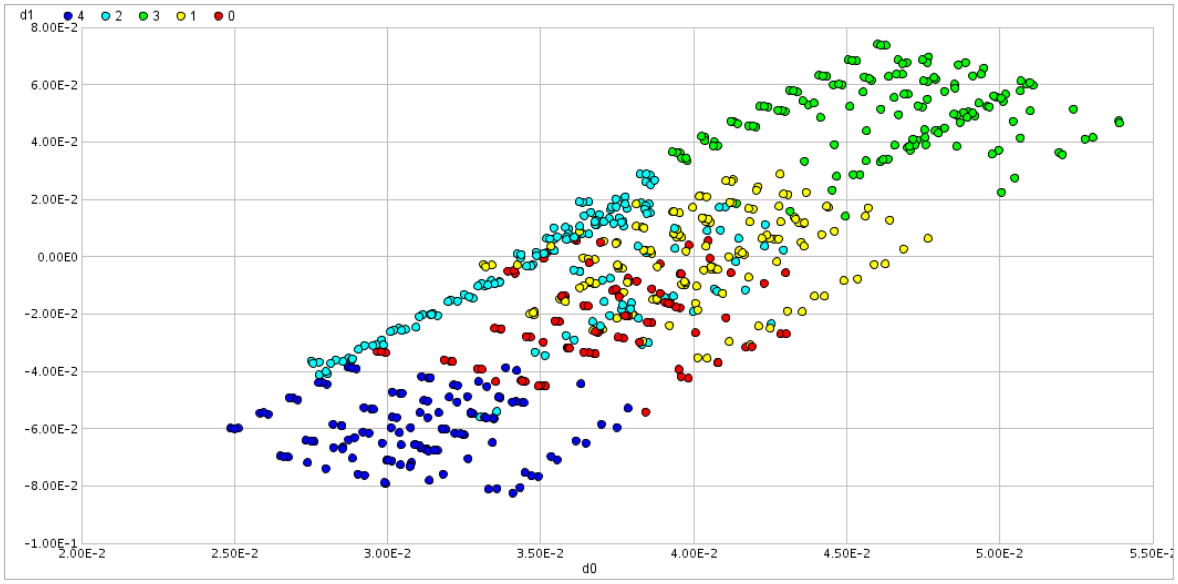

Figura 27 - Dispersão dos *clusters* - disponibilidade por máquina.

O gráfico da Figura 28 apresenta a disponibilidade de cada um dos pontos de cada *cluster*, sob a forma de bolhas. A dimensão da bolha representa a disponibilidade de cada instância, para cada hora do dia (eixo horizontal), máquina (eixo vertical). O *cluster* a que cada instância pertence é representado pela cor da bolha.

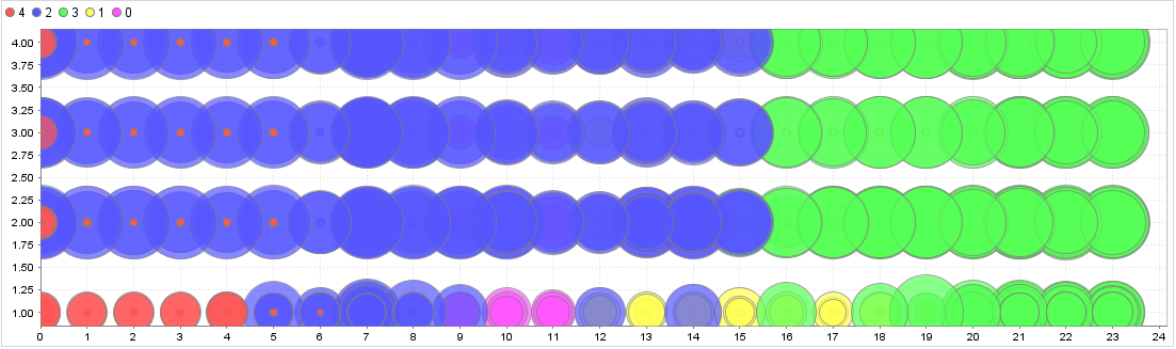

Figura 28 – Bolhas de disponibilidade - disponibilidade por máquina.

Disponibilidade da *Grid*

De seguida, são apresentados os dados obtidos relativamente aos cinco *clusters* gerados pelo modelo *clustering* de disponibilidade da *Grid*.

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Todos
Cardinalidade (nº instâncias)	6	65	36	32	41	169
Pontos centrais						
YEAR_DT/100	2.008	2.008	2.008	2.008	2.008	
MONTH_DT	9.000	9.000	9.000	9.000	9.000	
DAY_DT	15.000	17.056	15.500	17.344	14.244	
WEEKDAY_DT	2.000	4.056	4.056	4.344	3.976	
HOUR24_DT	3.500	8.574	20.389	13.219	7.098	
HOUR_DT	3.500	4.130	8.389	5.719	4.756	
CPU_FREE	0.000	0.867	3.852	1.824	3.939	
CPU_COUNT	4.000	3.944	4.000	3.875	3.976	
RAM_FREE/100	4.049	24.888	22.412	13.067	21.505	
RAM_TOTAL/100	45.520	44.958	45.285	44.256	44.551	
JOBS_WAITING	28.722	0.000	0.000	1.109	0.000	
JOBS_RUNNING	0.000	0.000	0.000	0.000	0.000	
AVAILABILITY	0.000	2.746	41.929	14.698	38.016	
Davis-Bouldin						1.036

Tabela 9 - Dados relativos ao modelo de *clustering* de disponibilidade da *Grid*.

Na Tabela 9 é apresentado o número de instâncias por *cluster*, bem como os valores de cada um dos pontos centrais e o valor do índice *Davis-Bouldin***.**

Desta feita para a disponibilidade da *Grid*, é apresentada a informação relativa aos dados dos pontos centrais, de forma gráfica, na Figura 29. O eixo horizontal corresponde a cada uma das variáveis dos pontos centrais e o eixo vertical é relativo ao valor de cada uma das mesmas. A cor das linhas representa o *cluster* a que cada ponto corresponde.

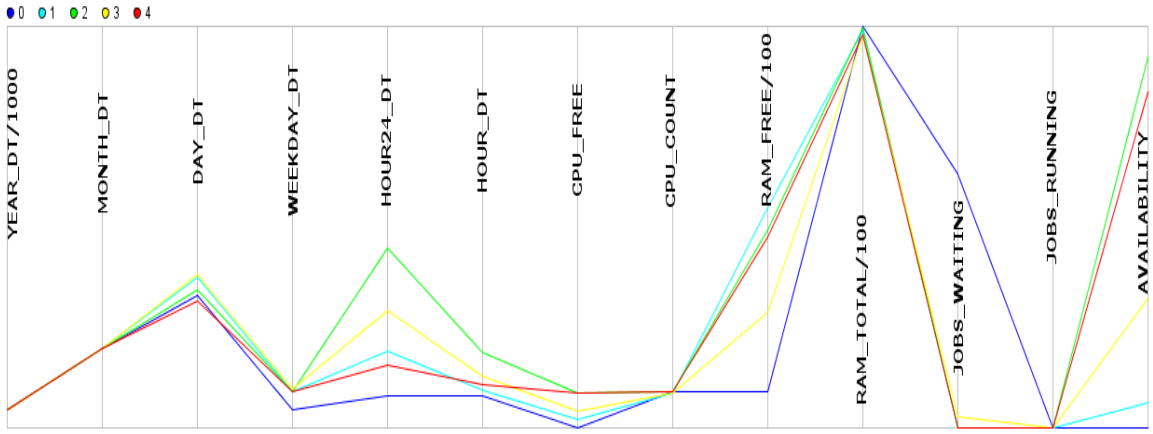

Figura 29 - Pontos centrais - disponibilidade da *Grid*.

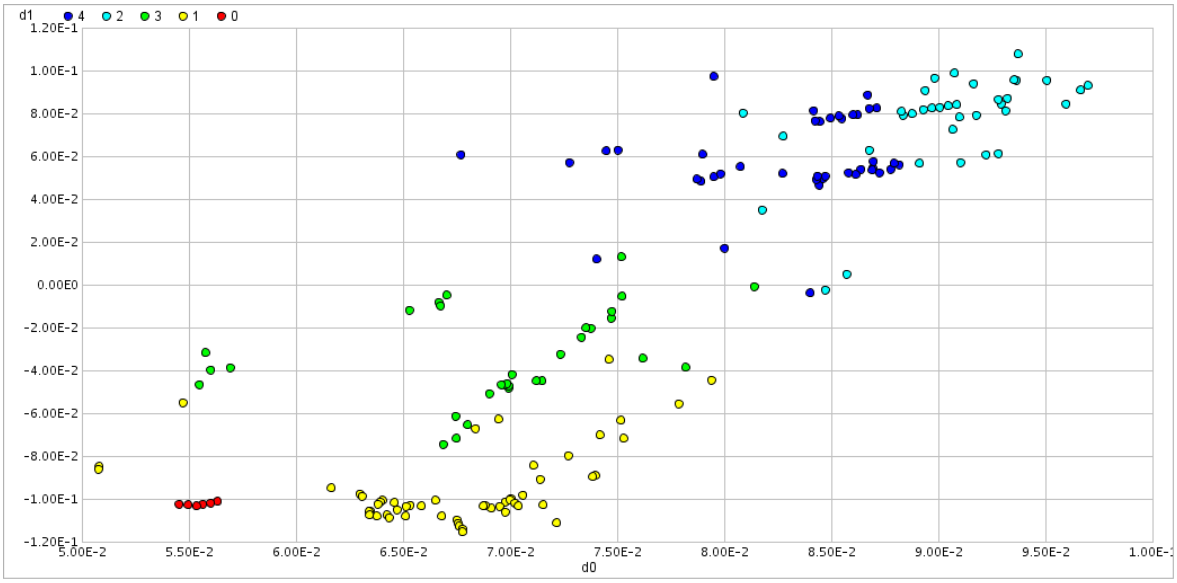

Figura 30 - Dispersão dos *clusters* - disponibilidade da *Grid*.

A dispersão dos *clusters* gerados pode ser visualizada na Figura 30. As instâncias estão distribuídas num espaço bidimensional, graças à transformação resultante da utilização do algoritmo *SVDReduction.* O cluster atribuído a cada instância pode ser observado pela cor de cada ponto.

Os resultados podem também ser vistos na Figura 31. Nestes, cada uma das bolhas representa uma instância. A cor da bolha corresponde ao *cluster* atribuído à instância pelo modelo. Já a dimensão é relativa ao valor correspondente de disponibilidade. No eixo horizontal é apresentada a hora da recolha de cada instância, num formato 24h, sendo o eixo vertical relativo ao dia da semana (Domingo=1; Sábado=7).

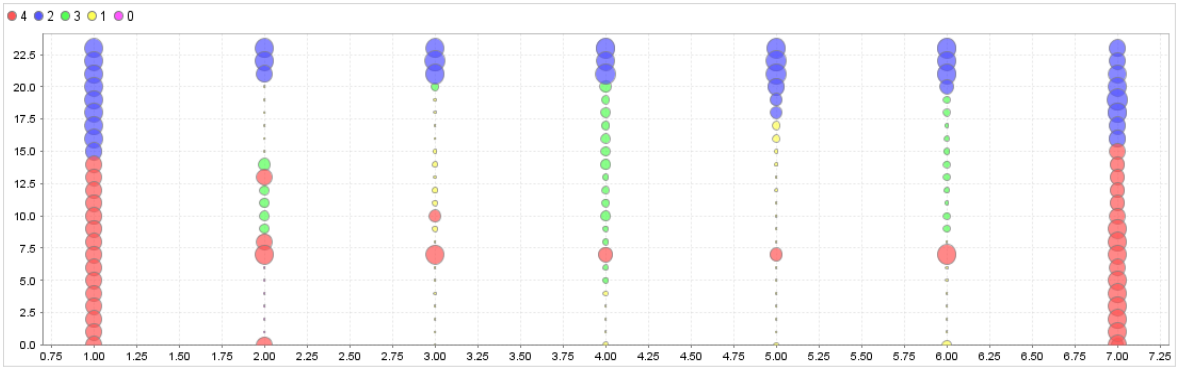

Figura 31 – Bolhas de disponibilidade - disponibilidade por máquina.

4.5.2 Modelos de criação de regras de associação

Os resultados obtidos correspondem às dez regras mais relevantes, e não óbvias, por objectivo de análise, obtidas pela aplicação dos modelos de regras de associação. O critério de relevância utilizado para filtragem tem em conta os valores de cobertura e confiança. Quase todas as regras óbvias eliminadas têm valor de confiança muito próximo 1, relacionando pares valores como, por exemplo, dia da semana e dia do mês, hora em formato 12h e hora em formato 24h, mês e ano, entre outras.

No total, foram obtidas cerca de 400 regras de associação na análise de disponibilidade por máquina (Tabela 10) e cerca de 900, na análise de disponibilidade da *Grid* (Tabela 11).

4.5.3 Modelos baseados em árvores de decisão

Após criação e treino dos modelos baseados em árvores de decisão, foram obtidos os resultados apresentados, através da aplicação dos modelos aos conjuntos de teste e validação. As

tabelas apresentadas relacionam o número de classificações reais e previstas das instâncias de cada conjunto. Para cada classe de previsão é também apresentada a medida de precisão (ou percentagem de acerto). É também apresentada a percentagem de acerto global de cada modelo, obtida através da divisão entre previsões correctas e previsões totais.

Premissas	Conclusões	Cobertura	Confiança
CPU_FREE=1 & WEEKDAY_DT=7	JOBS_WAITING=0	0.148	1
HOSTUNIQUE_ID=gtnode-4	RAM_TOTAL=1011	0.253	1
WEEKDAY_DT=1	JOBS_WAITING=0 & CPU_FREE=1	0.144	1
WEEKDAY_DT=7	CPU_FREE=1	0.148	0.990
WEEKDAY_DT=1	CPU_FREE=1	0.144	1
HOSTUNIQUE_ID=gtnode-3	CPU_FREE=0	0.130	0.515
RAM_TOTAL=1519	HOSTUNIQUE_ID=gtnode-1	0.241	1
WEEKDAY_DT=1, DAY_DT=14	RAM_TOTAL=1011	0.108	0.750
JOBS_WAITING=0, CPU_FREE=0	RAM_TOTAL=1011	0.281	0.847
JOBS_WAITING=0	CPU_FREE=1	0.561	0.597

Tabela 10 - Regras de associação obtidas durante análise de disponibilidade por máquina.

Premissas	Conclusões	Cobertura	Confiança
RAM_TOTAL=4552	CPU_COUNT=4	0.905	1
CPU_COUNT=4, CPU_FREE=4	JOBS_WAITING=0	0.420	0.947
RAM_TOTAL=4552, CPU_FREE=4	CPU_COUNT=4	0.396	1
WEEKDAY_DT=7	CPU_FREE=4	0.142	0.960
WEEKDAY_DT=1	CPU_FREE=4	0.142	1
CPU_FREE=4	JOBS_WAITING=0	0.420	0.947
WEEKDAY_DT=7	JOBS_WAITING=0	0.148	1
WEEKDAY_DT=1	JOBS_WAITING=0	0.148	1
JOBS_WAITING=0	CPU_COUNT=4	0.893	0.950
CPU_FREE=4	CPU_COUNT=4	0.444	1

Tabela 11 - Regras de associação obtidas durante análise de disponibilidade da *Grid*.

Na Tabela 12 são apresentados os resultados obtidos para o conjunto de teste, por aplicação da árvore de decisão de disponibilidade por máquina. O mesmo é feito na Tabela 13, para o conjunto de validação.

	C0 Real	C1 Real	C2 Real	C3 Real	C4 Real	C5 Real	Precisão
C0 Prevista	**40**	3	0	0	0	2	88.89%
C1 Prevista	0	**0**	0	0	0	0	0.00%
C2 Prevista	3	0	**6**	0	1	0	60.00%
C3 Prevista	0	0	1	**0**	2	0	0.00%
C4 Prevista	1	0	0	0	**12**	6	63.16%
C5 Prevista	1	0	1	2	3	**23**	76.67%
Global							75.70%

Tabela 12 - Previsões de disponibilidade por máquina utilizando o conjunto de testes.

	C0 Real	C1 Real	C2 Real	C3 Real	C4 Real	C5 Real	Precisão
C0 Prevista	**37**	0	1	0	1	1	92.50%
C1 Prevista	2	**0**	1	0	1	0	0.00%
C2 Prevista	0	0	**10**	0	1	0	90.91%
C3 Prevista	0	0	1	**0**	3	0	0.00%
C4 Prevista	0	0	0	1	**13**	6	65.00%
C5 Prevista	2	1	2	0	6	**14**	56.00%
Global							71.15%

Tabela 13 - Previsões de disponibilidade por máquina utilizando o conjunto de validação.

Analogamente ao que é apresentado para a análise de disponibilidade por máquina, são também apresentados resultados das previsões efectuadas pelos modelos de árvores de decisão, nos conjuntos de teste e validação. Os resultados são apresentados pela Tabela 14 e pela Tabela 15, respectivamente.

	C0 Real	C1 Real	C2 Real	C3 Real	C4 Real	C5 Real	Precisão
C0 Prevista	**4**	1	0	1	0	1	57.14%
C1 Prevista	0	**4**	0	1	0	0	80.00%
C2 Prevista	0	0	**3**	0	0	0	100.00%
C3 Prevista	0	0	0	**0**	1	0	0.00%
C4 Prevista	0	0	0	1	**7**	3	63.64%
C5 Prevista	0	0	0	0	1	**1**	50.00%
Global							65.52%

Tabela 14 - Previsões de disponibilidade da *Grid,* utilizando o conjunto de teste.

	C0 Real	C1 Real	C2 Real	C3 Real	C4 Real	C5 Real	Precisão
C0 Prevista	**7**	1	0	0	2	0	70.00%
C1 Prevista	1	**2**	0	1	0	0	50.00%
C2 Prevista	0	0	**1**	0	0	0	100.00%
C3 Prevista	0	3	0	**1**	1	0	20.00%
C4 Prevista	0	0	0	0	**5**	0	100.00%
C5 Prevista	0	0	0	0	0	**0**	0.00%
Global							64.00%

Tabela 15 - Previsões de disponibilidade da *Grid*, utilizando o conjunto de validação.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Discussão de resultados

O ambiente *Grid* protótipo criado foi alvo da aplicação de uma bateria de carga, de forma a introduzir carga variável, ao longo do tempo. Esta bateria simulou a execução de três aplicações distintas, com perfis diferentes e com carga diferente durante o dia. Os dados recolhidos periodicamente, junto dos serviços de informação da *Grid*, foram utilizados para a criação e aplicação de modelos de mineração de dados, cujos resultados foram apresentados no capítulo anterior. A discussão destes resultados é feita de seguida, para cada grupo de modelos de mineração de dados. No final, é feita uma comparação entre os vários modelos, no que diz respeito às potencialidades de cada um.

Pretende-se com esta análise perceber em que medida pode a aplicação de técnicas de mineração de dados contribuir para a análise e previsão de disponibilidade em ambientes tão empiricamente imprevisíveis, como é o caso dos ambientes *Grid*.

5.1.1 Modelos de *clustering*

Para ser feita uma exploração inicial dos dados recolhidos, foram criados dois modelos de *clustering*. Estes pretender ajudar na avaliação da disponibilidade de cada máquina e da *Grid* como um todo. Desta forma, pretendeu-se conhecer o comportamento isolado das máquinas, bem como o resultado da aplicação da bateria de carga no ambiente *Grid.*

O modelo de *clustering* obtido para a avaliação da disponibilidade de cada máquina apresenta-se consideravelmente compacto, havendo bastante sobreposição entre *clusters*, principalmente entre os *clusters* 0, 2 e 3 (Figura 27). Tal sobreposição é comprovada pelo índice *Davis-Bouldin* de apenas 0.651 (Tabela 8), apesar da distância entre os pontos centrais ser relativamente baixa, como se pode comprovar na representação *2D* do *cluster*. Esta sobreposição é causada, em parte, pelo baixo número de *clusters*.

Quanto aos pontos centrais, deve-se realçar dois deles, os dos *clusters* 2 e 3, agregando pontos de elevada disponibilidade (Tabela 8 e Figura 26), sendo os restantes *clusters* centrados em pontos de disponibilidade reduzida. Os valores de *JOBS_RUNNING* podem, à primeira vista, indicar que não estavam a ser executadas tarefas nas máquinas analisadas. No entanto, dado que cada máquina possuía dois sistemas de execução de tarefas *(Fork e PBS)* e apenas foi analisado um deles, estes valores são normais e não diferentes do esperado. De relembrar que foi feita esta opção, de analisar apenas um dos sistemas, precisamente por cada máquina ser publicada por ambos os sistemas, o que poderia indicar uma disponibilidade maior do que a real, fruto da duplicação de valores. Assim, optou-se por analisar apenas os dados do sistema *Fork*, por permitirem um estudo individual de cada máquina. Para além dos valores de *JOBS_*RUNNING, outros valores como *WEEKDAY_DT* e *DAY_DT* são compreensivelmente aproximados, devido ao reduzido número de *clusters*.

Quanto à disponibilidade e distribuição de máquinas pelos vários *clusters,* observou-se alguma homogeneidade entre as máquinas *gtnode-2, gtnode-3* e *gtnode-4* (Figura 28). Tal explica-se pela maior carga da máquina *gtnode-1*, devido a esta ser a máquina que foi escolhida para a submissão de tarefas, para a recolha de dados, bem como para outros serviços. Apesar da maior capacidade de memória, o processador acabou por ser o factor mais limitante. Esta máquina é a que mais varia entre *clusters* e a que invariavelmente apresenta menor disponibilidade ao longo do tempo. Também se observa uma clara repartição de períodos de maior disponibilidade ao longo do tempo, pelos *clusters* 2 e 3. O *cluster* 2 acaba por estar mais relacionado com os períodos dos dias não úteis, estando o *cluster* 3 ligado à alta disponibilidade no período correspondente ao final da tarde e início de noite. Para além disso, é observável a baixa disponibilidade associada ao *cluster* 4, correspondente aos períodos de execução da aplicação nocturna intensiva.

No que diz respeito à disponibilidade global, o primeiro dado relevante é o número de instâncias por *cluster* (Tabela 9). O *cluster* 0 tem um reduzido número de instâncias, estando principalmente associado à execução da aplicação nocturna intensiva, dada a hora do dia (*HOUR24_DT*) e o número de tarefas em espera *(JOBS_WAITING).* Mais uma vez, são observados

dois períodos de elevada disponibilidade, correspondentes aos *clusters* 2 e 4 (Figura 29). A disponibilidade do *cluster* 2 é referente ao período do final da tarde, em que não há grande submissão de tarefas, senão da aplicação 24h. Fica também comprovado que esta aplicação não tem grande impacto na *Grid*, por ter baixa probabilidade de submissão e tarefas relativamente ligeiras.

Quanto à dispersão (Figura 30), é nesta análise mais significativa, principalmente devido ao baixo número de instâncias analisadas. No entanto, o *cluster* 0 mostra grande homogeneidade, mais uma vez devido à grande intensidade de tarefas, a que a *Grid* é submetida no período entre as 0h e as 6h da manhã.

Analisando o cruzamento entre disponibilidade, período temporal e *cluster* (Figura 31), é possível concluir que aos fins-de-semana a utilização da *Grid* é francamente baixa. Um outro período interessante é novamente o período compreendido entre as 0h e as 6h, aproximadamente. Embora as tarefas da aplicação nocturna não sejam submetidas durante todo esse período, a sua execução arrasta-se até ao início da manhã, algumas horas antes do início da aplicação diária. Esta carga nocturna é seguida de um período de acalmia, aproximadamente entre as 7h e 8h da manhã. Durante o período estabelecido como laboral, é observada bastante variabilidade em termos de disponibilidade e até de atribuição de instâncias a *clusters.* Tal fica-se a dever à média intensidade de submissão de tarefas, combinada com três tipos diferentes de *benchmarks*. Estes variam em intensidade e tempo de execução, pelo que a sua escolha aleatória acaba por não provocar um comportamento absolutamente previsível durante o período referido. Por fim, e apesar de não serem submetidas tarefas durante o período relativo à hora de almoço, não é observável um aumento de disponibilidade neste período. Esta situação é criada devido a alguma diferença entre a data de submissão da tarefa e a data de execução efectiva, e também devido ao tempo de execução elevado de algumas tarefas, fazendo com que nem sempre as mesmas sejam concluídas na mesma hora em que foram submetidas.

Em resumo, os modelos de *clustering* permitiram encontrar períodos similares, quer ao nível de disponibilidade, quer ao nível do período temporal, principalmente na análise global a toda a *Grid*. A sua análise permite extrapolar, em certa medida, qual o comportamento da *Grid* ao longo da semana. Já a análise por máquina permitiu identificar uma máquina claramente mais sobrecarregada que as restantes (*gtnode-1*), sendo a máquina mais imprevisível e que mais alterna entre *clusters*.

5.1.2 Modelos de criação de regras de associação

Os resultados obtidos pela aplicação de criação de regras de associação são talvez os menos positivos. Apesar do grande número de regras obtidas (cerca de 1300 no total), a grande maioria retratava, por exemplo, a relação aritmética existente entre alguns valores e associações óbvias entre dias da semana e do mês.

Das regras consideradas relevantes, filtradas por análise humana, de destacar os altos níveis de confiança em grande parte dos casos. Já os valores de suporte são raramente altos, até porque algumas das regras estão relacionadas com o dia da semana e daí automaticamente limitadas a um sétimo dos dados, sendo este o valor máximo de suporte para qualquer regra deste género.

Algumas das regras acabam por ajudar a prever, de certa forma, a disponibilidade das máquinas e da *Grid*. Três das regras obtidas para a disponibilidade por máquina (Tabela 10 e Tabela 11) indicam que nos dias não úteis o processador atribuído a cada máquina está grande parte das vezes disponível. É também apresentado, se bem que com um nível mais baixo de confiança, que a máquina *gtnode-3* está muitas vezes carregada e sem qualquer processador disponível. Também na disponibilidade por máquina é estabelecida a relação entre tarefas em espera e número de processadores livres. No entanto, o valor de confiança desta regra é baixo.

No modelo de disponibilidade da *Grid*, é novamente encontrada a relação entre dias não úteis e processadores livres, com alta confiança. É também encontrada a relação entre número de processadores livres e tarefas em espera, bem como entre dias não úteis e tarefas em espera. Para além disso, são encontradas regras, que embora não óbvias, espelham a constância dos valores de memória e número de processadores, por exemplo.

Ambos os modelos de criação de regras de associação acabaram por sair prejudicados pelo facto de os valores de *JOBS_WAITING e JOBS_RUNNING* não corresponderem à soma de valores de *Fork e PBS*, pelo que nem sempre foi possível estabelecer uma relação entre estes valores e os restantes, com altos níveis de confiança e suporte. Adicionalmente, a grande homogeneidade da *Grid*, pode ter contribuído para limitar o conjunto de regras de associação. Apesar destes factores, algumas das regras ajudam efectivamente a prever a disponibilidade, principalmente durante os períodos de menos actividade, correspondentes aos dias não úteis. Não permitem, no entanto, prever impactos pela alteração de uma das condições, como a submissão de uma tarefa adicional.

5.1.3 Modelos baseados em árvores de decisão

Com a criação de modelos baseados em árvores de decisão, era esperado que fosse possível prever a classe de disponibilidade referente a um determinado período no tempo. Os modelos

criados para disponibilidade por máquina e da *Grid* permitem efectivamente responder às expectativas.

O modelo utilizado para prever a disponibilidade por máquina apresenta valores de previsão com sucesso acima dos 70%, tanto para o conjunto de teste (Tabela 12) como para o conjunto de validação (Tabela 13). Os piores valores de classificação são referentes à classe de disponibilidade *C3,* em que não houve nenhum acerto em nenhum dos conjuntos de teste e validação. No conjunto de validação, a classe *C1* é também erradamente prevista, não sendo sequer observada no conjunto de testes.

Para além do referido, é curioso o facto de o primeiro critério para ramificação ser precisamente a identificação de cada máquina, permitindo fazer uma previsão de forma individual, para cada máquina. Este critério é automaticamente escolhido, por permitir um maior ganho de informação, devido às diferenças existentes entre algumas das máquinas, quanto à disponibilidade, nomeadamente entre a máquina *gtnode-1* e as restantes.

No que diz respeito à disponibilidade global da *Grid*, os valores alcançados de precisão de previsão de disponibilidade, são relativamente mais baixos, rondando os 65%, quer para o conjunto de teste (Tabela 14) ou para o conjunto de validação (Tabela 15). Este facto pode ser explicado pelo conjunto reduzido de instâncias utilizado no teste e validação do modelo, uma vez que os dados agregados de todas as máquinas resultam num número de instâncias relativamente curto.

No conjunto de teste, a classe *C3* foi novamente prevista com 0% de precisão, sendo a precisão de apenas 20% no conjunto de validação. Em ambos os conjuntos a classe *C2* foi prevista com 100% de acerto, assim como a classe *C4* no conjunto de validação. Neste último conjunto não foi observada qualquer instância com *C5* real, mais uma vez devido ao número reduzido de instâncias.

Ambos os modelos, referentes a cada máquina e à *Grid*, apresentam taxas de sucesso relativamente altas, sendo os casos de insucesso maioritariamente relacionados com a classificação numa classe vizinha. Caso o número de classes fosse diminuído para apenas três, os resultados de precisão teriam sido consideravelmente superiores. No entanto, perder-se-ia alguma granularidade. Observa-se também que a classe de disponibilidade *C3* é a classe mais problemática. A razão para este facto está relacionada com a grande variabilidade dos períodos de média disponibilidade, como já foi observado nos modelos de *clustering* (5.1.1). Para além disso, e como esperado, torna-se mais fácil prever a disponibilidade de cada máquina do que da *Grid* como um todo. Tal porque prever a disponibilidade da *Grid* está, em certa medida, relacionada com a

previsão da disponibilidade média de cada máquina. Por fim, e talvez devido à maior flutuação de disponibilidade de cada máquina, em relação à *Grid* como um todo, o modelo de previsão de disponibilidade por máquina acaba por ser significativamente mais complexo e extenso do que o modelo de previsão de disponibilidade da *Grid*.

5.2 Comparação de modelos

Os vários modelos apresentam resultados úteis para a previsão de disponibilidade em ambientes *Grid*. Os modelos de *clustering* disponibilizam resultados que permitem fazer a análise dos dados de forma simples. Desta forma é possível detectar máquina sobrecarregadas, saber a flutuação da disponibilidade ao longo do tempo e também extrapolar qual o comportamento da *Grid* de semana para semana. No entanto, a análise automática, com vista à previsão de disponibilidade não é possível, uma vez que a análise destes modelos obriga a ter algum conhecimento acerca dos dados analisados. Isto, porque o processo de obtenção de resultados tem de ser conduzido em direcção a padrões, sobre os quais haja algum tipo de suspeita.

Os modelos de criação de regras de associação possibilitam a utilização automática, por representarem expressões lógicas. No entanto, é dúbio se os resultados obtidos trariam algum benefício a um escalonador, como mecanismos de previsão de disponibilidade e suporte à decisão de escalonamento.

Por fim, os modelos baseados em árvores de decisão permitem tanto uma análise fácil humana como automática. Na sua forma gráfica (Figura 25), uma árvore de decisão é de fácil compreensão, sendo possível determinar rapidamente a classe prevista para um determinado período do tempo. Por serem constituídas por expressões lógicas, as árvores de decisão podem perfeitamente ser adoptadas por sistemas automáticos. É provável, dadas boas percentagens de acerto, que estes modelos auxiliem um escalonador na escolha do melhor período para execução de tarefas. Estas percentagens poderiam eventualmente melhorar com a recolha de dados num período superior ao de uma semana.

5.3 Considerações finais

A instalação e configuração de ambientes *Grid* é uma tarefa morosa e complexa, em parte devido ao grande número de ferramentas cujos papéis se sobrepõem e, por vezes, à baixa cobertura da documentação apresentada por alguns dos projectos. No trabalho apresentado, este processo foi baseado em excessivas ocorrências de tentativa e erro, atingindo-se o sucesso apenas ao fim de algumas instalações. Assim, para que possam tornar-se de uso corrente, as várias

ferramentas *Grid* têm ainda de maturar, de forma a serem acessíveis às organizações, sem ser necessário um estudo aprofundado. Iniciativas com os objectivos do projecto *VDT* irão certamente continuar a surgir, facilitando a utilização de tecnologias *Grid*.

Com um ambiente protótipo disponível, criou-se um mecanismo de recolha de informação baseado no serviço de informação *MDS-Index*, recolhendo assim dados utilizados para o cálculo e previsão de disponibilidade. Dados os resultados obtidos, o modelo de cálculo utilizado permite efectivamente representar o conceito de disponibilidade, podendo eventualmente ser afinado com outros parâmetros.

Estando o mecanismo de recolha de informação disponível, foi feita a submissão de tarefas, mediante três perfis aplicacionais, definidos pela carga e probabilidade de geração de tarefas por hora. Utilizando os dados recolhidos como fonte, e após processamento prévio, foi feita a análise de disponibilidade, extraindo conhecimento acerca do comportamento das máquinas e da *Grid*, por recurso a técnicas, algoritmos e modelos de mineração de dados. Nenhum dos princípios dos mecanismos apresentados é aplicável em exclusivo ao *middleware* e escalonadores seleccionados. Caso o *middleware* disponibilize um serviço de informação, bastará adaptar o sistema de recolha de informação, para que seja possível fazer análise e previsão de disponibilidade num qualquer ambiente *Grid*.

Não conhecendo as aplicações executadas na *Grid*, como é o caso nos ambientes reais, a análise humana, sem o recurso a técnicas de mineração de dados, é impraticável para os padrões não óbvios. Apresentados e discutidos que estão os resultados obtidos, conclui-se que a mineração de dados pode realmente permitir a análise e previsão de disponibilidade de ambientes *Grid*. Esta conclusão baseia-se no facto de ter havido uma boa correspondência entre os resultados obtidos e os padrões de carga introduzidos na *Grid*, pela aplicação da bateria de carga. Assim, os conceitos de mineração de dados utilizados possibilitaram a descoberta de períodos de franca ocupação da *Grid* e de cada uma das máquinas, bem como de alguns períodos de disponibilidade quase total. Permitiram também identificar máquinas sobrecarregadas e com comportamento distinto das demais. *No* entanto, foi observada alguma flutuabilidade de resultados, característica de sistemas não dedicados como é o caso dos ambientes *Grid*. No caso presente, foram utilizadas máquinas virtuais para simular nós físicos, o que pode ter também originado alguma dessa variação.

A análise feita recorrendo a modelos como os apresentados pode coadjuvar tanto os vários escalonadores, como os administradores humanos das *Grids*. A adaptação dos *softwares* de escalonamento é possível, dado o carácter lógico de alguns dos resultados obtidos, podendo as expressões obtidas ser certamente aplicadas no processo de decisão de escalonamento. Já os

intervenientes humanos podem tirar partido de modelos para identificar períodos de estrangulamento da *Grid*, máquinas sub ou sobreutilizadas, entre outras informações.

Apesar dos resultados apresentados permitirem expor algumas das vantagens da utilização de técnicas de mineração de dados para a previsão de disponibilidade, não foi possível considerar todos os factores que a influenciam, num ambiente *Grid*. Devido à infra-estrutura de *hardware* utilizada, factores como a elevada latência, baixa largura de banda e adulteração de dados, presentes habitualmente em ambientes interligados em redes de longa distância, não foram considerados. Também não foram considerados factores como contenção no acesso a disco. Adicionalmente, as máquinas virtuais utilizadas partilhavam a mesma arquitectura, ao nível do *hardware*, tornando a *Grid* um pouco mais homogénea do que o desejado. Para além disso, o número de máquinas utilizado foi também reduzido. Por fim, a utilização da *Grid* foi simulada, variando certamente de ambiente para ambiente, e de organização para organização.

5.4 Trabalho futuro

Tendo em conta o trabalho descrito e algumas das questões levantadas, são várias as perspectivas de trabalho futuro que neste momento consideramos. Em primeiro lugar, poderá ser feita a aplicação dos mesmos conceitos a uma *Grid* de dimensões reais, idealmente heterogénea e com múltiplos utilizadores e inúmeras tarefas diferentes, recolhendo dados durante largos períodos de tempo. Desta forma seria possível confirmar se os modelos podem ser facilmente adaptados e transpostos para cenários, permitindo também determinar se as ferramentas e algoritmos utilizados são escaláveis ou se é necessário seguir outra abordagem como, por exemplo, mineração incremental de dados.

Uma outra abordagem poderia passar pela utilização da própria *Grid* para execução dos modelos de mineração de dados. Essa perspectiva torna-se possível pela utilização do *RapidMiner* ou do *Weka*, através da linha de comandos, podendo até a aplicação ser enviada juntamente com a tarefa, para que esta seja executada. Para além disso, a utilização de ambientes reais poderá levar à obtenção de dados que permitam melhorar o desempenho de modelos como o de criação de regras de associação.

Uma vez que foi observado um decréscimo de disponibilidade de uma das máquinas da *Grid*, pelo facto de alojar o sistema de submissão de tarefas e recolha de dados, será útil fazer uma comparação com uma outra orgânica, em que haja máquinas dedicadas para este tipo de tarefas, libertando os nós computacionais para que sirvam apenas a execução de tarefas.

Apesar de o modelo de cálculo de disponibilidade apresentar resultados consistentes, será útil avaliar em que medida contribui para a melhor ou pior previsão de disponibilidade em ambientes *Grid*. Assim, um estudo comparativo entre modelos, utilizando constante *kCPU e kMEM* diferentes poderá permitir afinar os modelos para resultados mais positivos. Através dos dados recolhidos, fica a ideia de que é mais fácil prever a utilização do processador do que da memória. Isto porque mesmo os processos inactivos continuam a consumir memória, potencialmente durante longos períodos de tempo, tanto em alturas de elevada como baixa disponibilidade.

Para além da previsão de disponibilidade, poderão ser feitas análises similares para outras métricas, como por exemplo taxas de insucesso, tempos de execução e ritmo de submissão de tarefas, por exemplo. Este tipo de análise poderão ajudar a aumentar a eficiência em ambientes *Grid*, a um nível mais fino, como é o caso da tarefa.

Por fim, e talvez a perspectiva mais aliciante, passa pela criação de mecanismos de realimentação nos vários escalonadores utilizados. Desta forma, seria possível influenciar as decisões dos escalonadores, com base na análise feita aos dados históricos recolhidos. Estes escalonadores modificados poderiam determinar prioridades e máquinas óptimas para execução, com base em informações do presente e previsões para períodos futuros, determinadas a partir da utilização de modelos de mineração de dados, por análise de dados históricos de estado. Assim, seria determinado o impacto que uma análise como a descrita neste trabalho pode ter no processo de escalonamento e até que ponto é possível melhorar os algoritmos existentes e em vigor, em alguns casos, há dezenas de anos.

Siglas e acrónimos

CODINE – COmputing in DIstributed Networked Environments

DNS – Domain Name Service

DRMAA – Distributed Resource Management Application API

DRS – Data Replication Service

DS – Delegation Service

DSA – Digital Signature Algorithm

EGEE – Enabling Grids for E-sciencE

ETL – Extraction-Transformation-Loading

FTP – File Transfer Protocol

GRAM – Grid Resource Allocation and Management

GSI – Grid Security Infrastructure

GSIFTP – GSI File Transfer Protocol

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

IPG – Information Power Grid

LHC – Large Hadron Collider

LCG – LHC Computing Grid

JDBC – Java Database Connectivity

JSDL – Job Submission Description Language

LSF – Load Sharing Facility

MDS – Monitoring and Discovery System

NJS – Network Job Supervisor

NSF – National Science Foundation

OASIS - Organization for the Advancement of Structured Information Standards

OGF – Open Grid Forum

OGSA – Open Grid Services Architecture

OGSI – Open Grid Services Infrastructure

OSG – Open Science Grid

PBS – Portable Batch System

RLS – Replica Location System

RFT – Reliable File Transfer

RSH – Remote Shell

SGBD – Sistema de Gestão de Base de Dados

SGE – Sun Grid Engine

SOA – Service-Oriented Architecture

SSH – Secure shell

SSL – Secure Socket Layer

TLS – Transport Layer Security

TSI – Target System Interface

UNICORE – Uniform Interface to Computing Resources

URL – Uniform Resource Locator

VDT – Virtual Data Toolkit

VO – Virtual Organization

VOMS – Virtual Organization Membership Service

WSDL – Web-Services Description Language

WSRF – Web-Services Resource Framework

XML – eXtensible Markup Language

Bibliografia

[Berman, F. et al. 2003] Berman, F. et al. (2003). *Grid computing: Making the Global Infrastructure a Reality.* Chichester: John Wiley and Sons, pp. 4-27.

[Czajkowski, K. et al. 2001] Czajkowski, K. et al. (2001). 'Grid Information Services for Distributed Resource Sharing', *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 181-184.

[Foster, I. 2006] Foster, I. (2006). 'Globus Toolkit Version 4: Software for Service-Oriented Systems', *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pp. 2-13.

[Foster, I. et al. 2001] Foster, I. et al. (2001). 'The Anatomy of the Grid: Enabling Scalable Virtual Organizations', *Lecture Notes in Computer Science*, vol. 2150.

[Kesselman & Foster 1998] Kesselman, C. & Foster, I. (1998). *The Grid: Blueprint for a New Computing Infrastructure.* San Mateo: Morgan Kaufmann Publishers.

[Welch, V. et al. 2003] Welch, V. et al. (2003). 'Security for Grid Services', *12th IEEE Internation Symposium on High Performance Distributed Computing.* Los Alamitos: IEEE Computer Society Press, pp. 48-57.

[Foster, I. et al. 2002A] Foster, I. et al. (2002). 'Grid Services for Distributed System Integration', *IEEE Computer,* vol. 35, no. 6, pp. 37-46.

[Allcock, B. et al. 2002] Allcock, B. et al. (2002). 'Data management and transfer in high-performance computational grid environments', *Parallel Computing*, vol. 28, pp. 749-771.

[Foster, I. et al. 2002B] Foster I. et al. (2002). 'The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration'.

[Litzkow, M. et al. 1988] Litzkow, M. et al. (1988). 'Condor – a hunter of idle workstations', *8th International Conference on Distributed Computing Systems*, pp. 104-111.

[Henderson, R. 1995] Henderson, R. (1995). 'Job Scheduling Under The Portable Batch System', *Lecture Notes in Computer Science*, Londres: Springer-Verlag, vol. 949, pp. 279-294.

[Bader, D. 1999] Bader, D. et al. (1999). 'Design and Analysis of the Alliance / University of New Mexico Roadrunner Linux SMP SuperCluster', *Proceedings of the IEEE International Workshop on Cluster Computing*, pp. 9-18.

[Gilfeather & Kovatch 2000] Gilfeather, F. & Kovatch, P. (2000). 'Supercluster: TeraClass computing', *Proceedings of 4th International Conference on High Performance Computing in the Asia-Pacific Region*, vol. 2-2000, pp. 874-883.

[Gentzsch, W. 2001] Gentzsch, W. (2001). 'Sun Grid Engine: towards creating a compute power grid'. *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid,* pp. 35-36.

[Zhou, S. et al. 1992] Zhou, S. e tal. (1992). 'UTOPIA: A Load Sharing Facility for Large, Heterogenenous Distributed Computer Systems', *Computer Systems Research Institute, University of Toronto, Technical Report CSRI-257*.

[Baker, M. et al. 2005] Baker, M. et al. (2005). 'Emerging Grid Standards', *IEEE Computer,* vol. 38, no. 4, pp. 43-50.

[Rajic, H. et al. 2007] Rajic, H. et al. (2007). *Distributed Resource Management Application API Specification 1.0*, Global Grid Forum.

[Anjomshoaa, A. et al. 2005] Anjomshoaa, H. et al. (2005). *Job Submission Description Language (JSDL) Specification, Version 1.0*, Global Grid Forum.

[Czajkowski, K. et al. 2004] Czajkowski, K. et al. (2004). *The WS-Resource Framework Version 1.0*, Global Grid Forum.

[Laure, E. et al. 2006] Laure, E. et al. (2006). 'Programming the Grid with gLite', *Computational Methods in Science and Technology*, vol. 12, no. 1, pp. 33-45.

[Erwin, D. 2001] Erwin, D. (2001). 'UNICORE: A Grid Computing Environment', *Lecture Notes in Computer Science*, Londres: Springer-Verlag, vol. 2150, pp. 825-834.

[Cannataro, M. et al. 2004] Cannataro, M. et al. (2004). 'Distributed Data Mining On Grids: Services, Tools and Applications', *IEEE Transactions on Systems Man and Cybernetics*, vol. 34, no. 6, pp. 2452-2465.

[Ellert, M. et al. 2003] Ellert, M. et al. (2003). 'The NorduGrid project: using Globus toolkit for building GRID infrastructure', *Proceedings of the VIII International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, pp. 407-420.

[Huedo, E. et al. 2005] Huedo, E. et al. (2005). 'The Gridway Framework for Adaptive Scheduling and Execution on Grids', *Scalable Computing: Practice and Experience*, vol. 6, no. 3, pp. 1-8.

[Frey, F. et al. 2002] Frey, F. et al. (2002). 'Condor-G: A Computation Management Agent for Multi-Institutional Grids', *Cluster Computing*, vol. 5, no. 3, pp. 237-246.

[Bailey, D. et al. 1991] Bailey, D. et al. (1991). 'The NAS Parallel Benchmarks', *The International Journal of Supercomputer Applications*, vol. 5, no. 3, pp. 63-73.

[Herrera, J. et al. 2004] Herrera, J. et al. (2004). 'DRMAA Implementation within the GridWay Framework', *Workshop on Grid Application Programming Interfaces*.

[Witten & Frank 2005] Witten, I & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Amesterdão: Morgan Kaufmann, 2ª edição, pp. 61-82, 96-105, 112-119, 136-139, 144-157.

Referências WWW

[W1] http://www.globus.org/ [2008/08/05]

A *Globus Alliance* é uma comunidade de organizações e indivíduos envolvidos no desenvolvimento de tecnologias *Grid*, com vista à partilha de recursos de processamento, bases de dados e instrumentos. É a responsável pelo desenvolvimento do *Globus Toolkit*.

[W2] http://www.cs.wisc.edu/condor/ [2008/08/05]

O objectivo do projecto *Condor* é o desenvolvimento, implementação e avaliação de mecanismos e políticas que suportam computação de alto desempenho, recorrendo a um vasto número de recursos computacionais distribuídos geograficamente. Está envolvido em projectos como o escalonadores *Condor* e *Condor-G*, por exemplo.

[W3] http://www.openpbs.org/ [2008/08/05]

O *OpenPBS* foi desenvolvido inicialmente pela *NASA*, como escalonador de tarefas *batch*. Actualmente é mantido comercialmente pela *PBS Gridworks*, uma divisão da *Altair Grid Solutions*.

[W4] http://www.clusterresources.com/ [2008/08/05]

O *TORQUE* é uma versão gratuita e *open-source* do *OpenPBS*. Está disponível neste sítio *Web* para *download*. Podem também ser encontradas ferramentas complementares que incluem, por exemplo, algoritmos alternativos de escalonamento.

[W5] http://www.pbsgridworks.com/ [2008/08/05]

Sítio da *PBS Gridworks*, uma divisão da *Altair Grid Solutions* e responsável pelo desenvolvimento do *PBS Professional.*

[W6] http://www.platform.com/ [2008/08/05]

A *Platform* desenvolve o escalonador e gestor de recursos *LSF.* Este produto permite a distribuição e virtualização de tarefas em ambientes de alto desempenho.

[W7] http://www.gridway.org/ [2008/08/05]

O *Gridway* é um escalonador global desenvolvido pela *Globus Alliance*. Trata-se de uma ferramenta capaz de interagir com diversos escalonadores locais como, por exemplo, *Condor*, *PBS*, *SGE* ou *Condor.*

[W8] https://www.ogf.org/ [2008/08/05]

O *Open Grid Forum* agrega utilizadores, programadores e fornecedores de *software*. Trata-se de uma organização que lidera o esforço global para a padronização da computação *Grid*. Resultou da fusão da *Global Grid Forum* e da *Enterprise Grid Alliance*. É responsável por *standards* como o *OGSA*, *OGSI* e *JSDL*.

[W9] http://www.eu-egee.org/ [2008/08/05]

O projecto *Enabling Grids for E-sciencE* almeja dotar a comunidade científica e empresarial de uma infra-estrutura global que potencie a investigação. Deste iniciativa nasceu o *middleware gLite*.

[W10] http://glite.web.cern.ch/glite/ [2008/08/05]

O *gLite* é um *middleware* para construção de aplicações *Grid*, nascido do empenho colaborativo de vários centros de investigação académicos e industriais pertencentes ao projecto *EGEE*.

[W11] http://www.ganglia.info/ [2008/08/05]

O *Ganglia* é um sistema de monitorização escalável para ambientes de alto desempenho, como *clusters* e *Grids*.

[W12] http://vdt.cs.wisc.edu/ [2008/08/05]

O objectivo do *Virtual Data Toolkit* é facilitar a instalação de *software* com vista à criação de *Grids*. Trata-se de um pacote de instalação e configuração que inclui ferramentas de monitorização, *middlewares* e escalonadores.

[W13] http://www.teragrid.org/ [2008/08/05]

A *TeraGrid* é uma infra-estrutura *Grid* aberta à descoberta científica, envolvendo recursos de alto desempenho em diversos locais geográficos.

[W14] http://www.nas.nasa.gov/ [2008/08/05]

A divisão *NASA Advanced Supercomputing* tem como função disponibilizar ambientes de computação de alto desempenho, para acelerar missões da *NASA* e potenciar o avanço da ciência. É a organização responsável pela manutenção da *Information Power Grid*.

[W15] http://www.unicode.eu/ [2008/08/05]

O *UNICORE* é um dos mais reconhecidos *middlewares* para construção de sistemas *Grid*, actualmente utilizado em sistemas de produção, dedicados à investigação científica.

[W16] http://gridengine.sunsource.net/ [2008/08/05]

Sendo suportado pela *Sun Microsystems*, o *Sun Grid Engine* é um escalonador local para uso em *clusters*. É também a base para o produto comercial denominado *N1 Grid Engine.*

[W17] http://www.cs.waikato.ac.nz/ml/weka/ [2008/08/05]

O *Weka* é disponibilizado como um conjunto de algoritmos para extracção de conhecimento, utilizando técnicas de mineração de dados. Para além de apresentar ferramentas para estas tarefas, é acompanhado por uma *API Java* que permite utilizar o *Weka* embebido noutras aplicações.

[W18] http://www.rapidminer.org/ [2008/08/05]

O *RapidMiner* é uma aplicação *open-source* para mineração de dados. Permite efectuar extracção de conhecimento e análise de resultados de forma visual, através do uso de diversos *plugins* para extrair, filtrar, transformar e analisar dados em diversos suportes, como ficheiros de texto ou bases de dados.

[W19] http://www.top500.org/system/9485/ [2008/08/08]

Sítio com informações relativas ao super-computador *IBM Road Runner*, o actual líder da lista de super-computadores, a nível mundial.

[W20] http://www.top500.org/system/9257 [2008/08/08]

Sítio com informações relativas ao super-computador *Ranger*, o actual quarto classificado da lista de super-computadores, a nível mundial.

[W21] http://www.oasis-open.org/ [2008/08/09]

A *OASIS* é uma organização sem fins lucrativos com vista ao desenvolvimento, convergência e adopção de *standards* em ambientes cuja troca de informação é vital.

[W22] http://www.unicore.eu/ [2008/08/16]

Sítio do *UNICORE*, um dos mais importantes *middlewares* Grid.

[W23] http://www.postgresql.org/ [2008/09/15]

Sítio do sistema de gestão de base de dados *PostgreSQL*. Este é considerado o *SGBD* open-source mais evoluído do mundo.

[W24] http://www.virtualbox.org/ [2008/09/15]

A *VirtualBox* é uma ferramenta de virtualização que permite a instalação de máquinas virtuais utilizando diversos operativos (*Windows, Linux, OpenBSD, DOS, etc)* em sistemas anfitriões físicos. Também nos anfitriões são suportadas múltiplas plataformas. A *VirtualBox* é distribuída gratuitamente sob licença *GPL* havendo, no entanto, uma versão comercial.