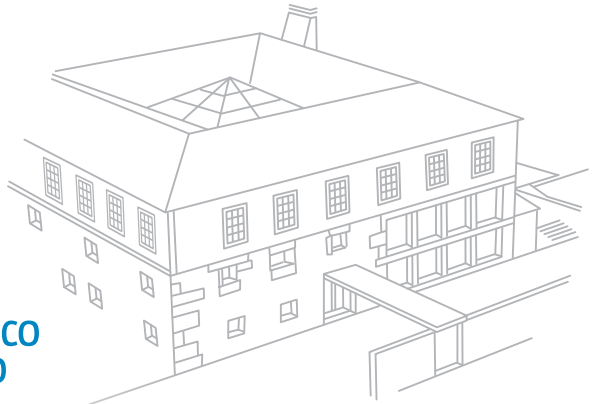


ESTGF | POLITÉCNICO DO PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Escalonamento de Processos ETL em Ambientes Grid

DESIGNAÇÃO DO MESTRADO
Mestrado em Engenharia Informática

AUTOR
Rui Manuel Sousa da Silva

ORIENTADOR(ES) Orlando Manuel de Oliveira Belo
Vasco Nuno Caio dos Santos

ANO
2012

A todos aqueles que nunca desistem de realizar os seus sonhos…

Agradecimentos

Agradeço aos meus orientadores, Professor Orlando Belo e Professor Vasco Santos, pela sua orientação, disponibilidade e compreensão no decorrer da realização desta dissertação. A sua contribuição para esta dissertação foi de grande importância, pois todas as opiniões dadas foram bastante valiosas.

Um agradecimento especial ao meu colega Bruno Oliveira, que acompanhou e trabalhou em conjunto comigo para a realização deste trabalho, tendo debatido com ele questões importantes sobre os conceitos base que suportam este trabalho.

Agradeço também à Escola Superior de Tecnologia e Gestão de Felgueiras por todo o acolhimento prestado durante estes últimos anos de formação, e também a todos os Professores por todo o conhecimento que me transmitiram e que permitiu enriquecer a minha formação.

À minha família, por toda a compreensão, confiança e apoio.

Aos meus amigos.

Resumo

Cada vez mais as organizações necessitam de estar preparadas para enfrentar um mundo em constante evolução, onde é necessário agregar um conjunto de informações provenientes de diversas áreas de negócio, de forma a tomar decisões que influenciam o desempenho da organização no seu meio competitivo. Para tal, as organizações utilizam Sistemas de *Data Warehousing* (SDW) que aglomeram e integram dados recorrendo a um processo de Extracção, Transformação e Carregamento (ETL). Os processos de ETL apresentam uma grande complexidade, pois têm de aceder a um conjunto de sistemas fonte, muitas vezes heterogéneos, de forma a realizar tarefas de transformação e limpeza de dados de acordo com as regras de negócio, exigindo para isso um elevado poder computacional. Com o crescimento de um SDW, o seu processo de ETL possui cada vez mais dados para processar. No entanto, é desejável que o tempo de processamento dos dados não comprometa o sistema, independentemente do volume de dados a tratar. Recorrendo à paralelização de tarefas, é possível reduzir o tempo de processamento dos dados, uma vez que algumas tarefas independentes podem ser executadas por máquinas diferentes ao mesmo tempo. O principal conceito dos ambientes *Grid* assenta na reutilização e aproveitamento de recursos, beneficiando assim do poder de processamento distribuído de forma a reduzir o impacto do crescimento de dados a tratar. Desta forma, é possível utilizar um ambiente *Grid* para realizar o escalonamento de um processo ETL, reduzindo o impacto oriundo do crescimento de dados, uma vez que os ambientes *Grid* permitem tirar partido dos recursos distribuídos disponíveis.

Palavras Chave: Data Warehouse, ETL, Ambientes Grid, Processamento em Paralelo, Escalonamento.

Abstract

Organizations need to prepare themselves to a changing world, and gathering and storing information from the various business areas will enhance decision making processes that affect the organization's performance in its competitive environment. To do this, organizations use Data Warehousing Systems (DWS) as data repository, where they store and integrate data using an Extraction, Transformation and Loading (ETL) process. The ETL process is known for its great complexity, mainly because it has to access a set of source systems, often heterogeneous, in order to extract data, perform cleaning tasks and process the data according to business rules, which requires great computational power. With the growth of a DWS, its ETL component has increasingly more data to process. However, it is desired that the data processing time remains within its window of opportunity regardless of the volume of data to be processed. Using task parallelization, it is possible to reduce the data processing time, since some independent tasks can be performed by different machines at the same time. The main concept of Grid environments is to reuse and harness resources, making it possible to benefit from the distributed processing power to reduce the impact of data growth. Thus, it is possible to use a Grid environment to perform the scheduling of an ETL process, reducing the impact of the data growth, since Grid environments allow the use of available distributed resources.

Keywords: Data Warehouse, ETL, Grid environment, Parallel processing, Scheduling.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Acrónimos

CA	Certificate Authority
DAG	Directed Acyclic Graph
DN	Distinguished Name
DW	Data Warehouse
DWS	Data Warehousing Systems
ETL	Extract, Transform and Load
FTP	File Transfer Protocol
GASS	Globus Access to Secondary Storage
GRAM	Globus Resource Allocation Manager
GSI	Grid Security Infrastructure
LFN	Logical File Name
LSF	Load Sharing Facility
MDS	Monitoring and Discovery Service
NJS	Network Job Supervisor
OSG	Open Science Grid
PBS	Portable Batch System
PKI	Public Key Infrastructure
QoS	Quality of Service
RLS	Replica Location Service

RSL	Resource Specification Language
SDW	Sistema Data Warehousing
SLA	Service-level Agreement
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TSI	Target System Interface
VDT	Virtual Data Toolkit
VO	Virtual Organization
VOMS	Virtual Organization Membership Service
WSRF	WS Resource Framework
XPDL	XML Process Definition Language

1 Introdução

1.1 Contextualização

Um Sistema de *Data Warehousing* (SDW) tem como principal objetivo armazenar informação proveniente de um ou mais sistemas fonte, tornando-se assim num repositório de dados centralizado. O processo de Extração, Transformação e Carregamento (ETL) é o processo mais crítico e moroso na construção de um SDW, pois visa a extração de dados a partir de um conjunto heterogéneo de sistemas fontes, transformando-os, e posteriormente carregando-os num *Data Warehouse* (DW). A Figura 1 apresenta a arquitetura tipo do processo ETL [1].

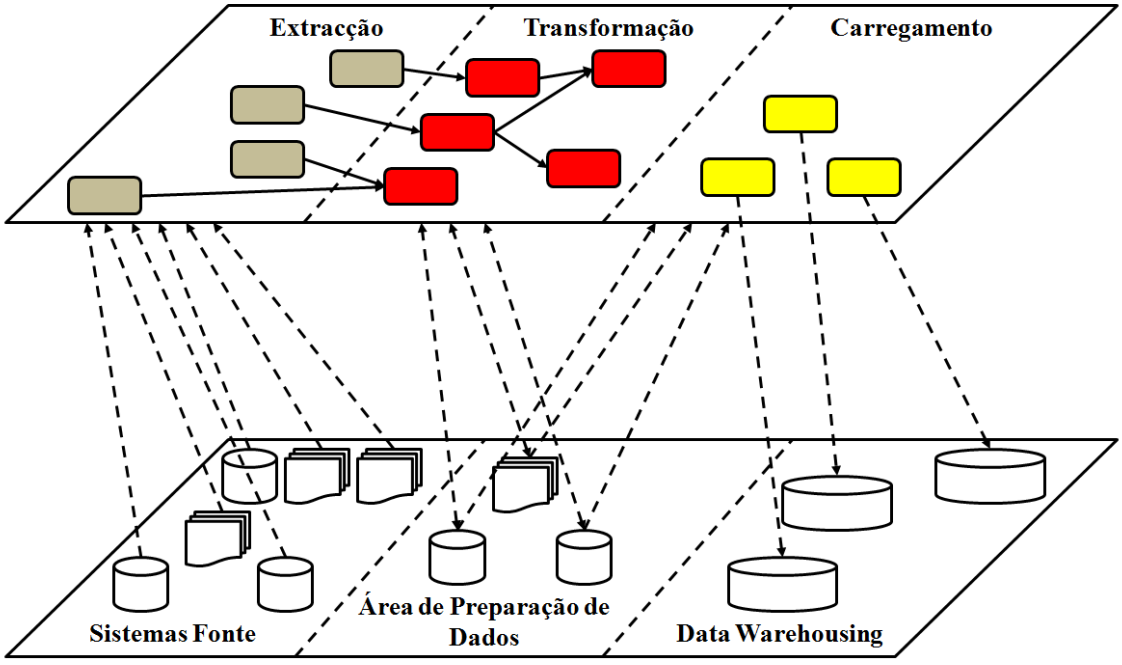


Figura 1 - Ilustração de um Processo de ETL [1]

O processo de ETL, cuja complexidade reside não só no acesso aos dados, mas principalmente na sua transformação e limpeza, requer um elevado poder computacional. Considerando o trabalho de

Kimball [2], um SDW tende a abranger cada vez mais áreas de negócio de uma organização e, consequentemente, o volume de dados a tratar com vista à sua integração no DW tende também a aumentar. Como tal, um SDW é um repositório de dados em contínuo crescimento. Desta forma, a componente de ETL de um SDW tem cada vez mais dados para tratar, sendo desejável que o tempo de processamento não se estenda para além do tempo disponível, independentemente do volume de dados a processar [3]. Esta particularidade torna-se um problema no desenvolvimento de um SDW pois é necessário definir, antecipadamente, estratégias e abordagens que permitam o processamento dos dados com sucesso, no tempo estipulado. Cada vez mais as organizações precisam de estar preparadas para este problema e, como tal, é necessário possuir recursos computacionais que permitam acompanhar o crescente volume de dados gerado, tanto a nível de armazenamento no SDW como no poder de processamento necessário para a concretização das tarefas especificadas no processo de ETL. Os recursos que as organizações dispõem, para este tipo de tarefas, são dispendiosos e muitas vezes não acompanham a crescente quantidade de dados gerada, tornando-se rapidamente obsoletos. Assim, as organizações necessitam de realizar atualizações periódicas a todos esses recursos ou adquirir novos recursos computacionais, tornando este processo extremamente dispendioso.

No entanto, nos dias de hoje as organizações possuem um grande número de computadores, cujo poder computacional é cada vez maior, sem que, no entanto, consigam retirar maiores proveitos das suas capacidades. A maior parte destes recursos encontram-se, grande parte do tempo, desaproveitados, seja por estarem inativos ou por serem utilizados por aplicações pouco exigentes em termos de recursos computacionais, tais como aplicações de navegação na *Web*, processadores de texto ou gestores de folhas de cálculo. Tendo em vista tal característica de utilização, o aproveitamento destes recursos é de vital importância para uma organização. Na última década tem-se assistido a várias experiências e abordagens para tirar partido de recursos distribuídos em prol de tarefas ou serviços que requerem grande poder computacional ou até elevada tolerância a falhas. Neste sentido, surgem conceitos como *Grids* computacionais ou computação na Nuvem. Embora com propósitos diferentes, ambas as abordagens assentam em processamento distribuído, no entanto, as *Grids* computacionais estão mais dirigidas ao processamento intensivo de tarefas ao invés da computação na Nuvem que se baseia na disponibilização de serviços. Como tal, e tendo em conta a particularidade de um sistema ETL, é proposta a utilização de uma *Grid* computacional como suporte à execução do processo de ETL.

Uma *Grid* computacional tem como princípio a execução paralela de processos utilizando todos os recursos heterogéneos disponíveis por uma ou mais organizações. Desta forma, uma organização pode utilizar eficientemente todos os recursos que dispõe para gerir e processar grandes quantidades de dados, tornando-se assim numa solução de baixo custo e com grande poder computacional.

Para distribuir um conjunto de tarefas num ambiente *Grid* é necessário possuir um escalonador, que é responsável por selecionar os recursos que vão suportar a execução de um conjunto de tarefas,

utilizando para o efeito um leque de métricas que permitam avaliar e distinguir os melhores recursos disponíveis. Normalmente, as métricas consideradas no processo de tomada de decisão baseiam-se, por exemplo, na análise da capacidade de processamento, disponibilidade, memória, largura de banda, sistema operativo, entre outras.

De forma a não sobrecarregar os melhores recursos com tarefas em excesso, o escalonador pode optar por utilizar algumas estratégias de balanceamento de carga, dividindo as tarefas por vários recursos de forma a utilizar total partido dos recursos disponíveis. Assim sendo, os algoritmos que definem o escalonador assumem um papel fundamental no processo de escalonamento num ambiente *Grid*, uma vez que determinam o seu desempenho e eficiência. O algoritmo de escalonamento pode ser considerado o núcleo do ambiente *Grid*, estando relacionado com os módulos de monitorização de recursos, gestão de trabalhos, transferência de ficheiros e autenticação. Assim, para que o escalonamento seja bem-sucedido, deverá considerar questões de otimização no processo das tarefas, e também de critérios de qualidade de serviço.

Tipicamente, o processo de ETL contém um conjunto de operações que dependem do resultado de outras operações. Assim é possível representar todo o processo de ETL através de um fluxo de tarefas que possa ser submetido e interpretado pelo escalonador de um ambiente *Grid*. Tendo em consideração que grande parte das operações existentes no processo de ETL são dependentes de resultados de operações antecedentes, é necessário ter em conta o volume de dados que transita de operação para operação e o seu impacto num ambiente distribuído. Para atenuar este problema torna-se necessário embutir algumas estratégias de seleção de conjuntos de nodos com melhor disponibilidade de largura de banda.

1.2 Objetivos

Com a realização deste trabalho pretende-se implementar um escalonador, que opere num ambiente *Grid*, e que efetue a divisão do processo de ETL pelos diversos nodos deste tipo de ambiente distribuído, sendo para isso necessário alcançar os seguintes objetivos:

- Conhecer os diferentes *middlewares* de gestão de ambientes *Grid*.
- Proceder à implementação de um ambiente *Grid* para testes.
- Selecionar e estudar os diferentes serviços de informação disponíveis para ambientes distribuídos.
- Conhecer APIs de interação com ambientes *Grid*.
- Desenvolver um conjunto de aplicações para gestão e controlo de um ambiente *Grid*.
- Extrair periodicamente informação relativa aos nodos pertencentes ao ambiente *Grid*.
- Desenvolver um escalonador de tarefas para um ambiente *Grid*.
- Considerar a largura de banda disponível nos nodos, no processo de tomada de decisão realizado pelo escalonador.

1.3 Organização da dissertação

Além do presente capítulo, esta dissertação encontra-se dividida em mais outros seis capítulos. No segundo capítulo é realizada a apresentação dos conceitos de Ambiente *Grid*, *Middleware Grid* e Serviços de Informação, sendo também efetuada uma breve análise comparativa entre os diferentes *middlewares Grid* e serviços de informação. De seguida, no terceiro capítulo é realizada uma descrição sobre a configuração e execução de processos ETL em ambiente *Grid*, apresentando um conjunto de requisitos operacionais, políticas e estratégias de escalonamento de processos ETL em ambiente *Grid*. O quarto capítulo apresenta o caso de estudo escolhido, acompanhado pelo conjunto de resultados alcançados com a sua implementação e execução no ambiente *Grid* selecionado. Por fim, no capítulo cinco, é realizada uma análise crítica sobre os resultados obtidos, apresentadas algumas conclusões sobre o trabalho realizado e apontadas algumas das suas possíveis linhas de desenvolvimento futuro.

2 Ambientes para Computação em GRID

Segundo Foster [4], o termo *Grid* refere-se à partilha coordenada de recursos, num ambiente dinâmico, heterogéneo e multi-organizacional, com vista à resolução de problemas (Figura 2). Por sua vez, Plaszczak e Wellner [5] definem *Grid* como uma tecnologia que permite a virtualização e reserva de recursos, e a partilha de serviços e recursos entre organizações. Para levar a cabo a partilha entre vários intervenientes, de múltiplas organizações, é necessário definir o papel desempenhado por cada um, ou seja, é importante definir quem vai utilizar e quem vai fornecer os recursos. O conjunto de indivíduos e organizações, que aderem a essas regras de partilha, formam uma *Virtual Organization* (VO) [4].

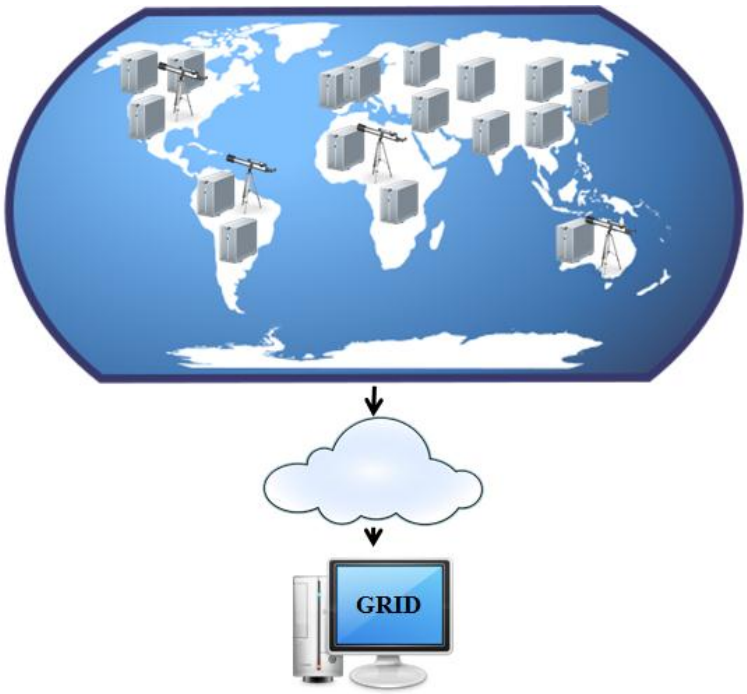


Figura 2 - Ilustração de um Ambiente Grid

Em [4] é ainda apresentado um outro conceito de *Grid*, focando-se principalmente na sua vertente tecnológica, sendo o termo *Grid* definido como um tipo de sistema paralelo e distribuído que permite a partilha, seleção e agregação dinâmica de recursos autónomos, geograficamente distribuídos, em tempo de execução, considerando a sua disponibilidade, capacidade, performance, custo e requisitos de qualidade de serviço exigidos pelo utilizador final.

Num ambiente *Grid*, a execução de trabalhos ou tarefas pode ser efetuada a partir de qualquer ponto geográfico, utilizando para isso qualquer recurso existente na *Grid*. Como este tipo de ambientes são por natureza dinâmicos e não existe uma responsabilidade global de uma só organização, a qualquer momento pode ser adicionado ou removido um recurso ou até mesmo uma ou várias VOs.

Ian Foster em [6] apresenta uma lista com os principais aspetos de uma *Grid*:

- A coordenação dos recursos não é submetida a um controlo central.
- Utilização de protocolos e interfaces *standards*.
- Qualidade de serviço não trivial.

Em [7] são definidas as principais características de um ambiente *Grid*:

- Ambiente Heterogéneo: Um ambiente *Grid* é normalmente composto por uma grande variedade de *software* e *hardware*, dispersos por diferentes domínios administrativos.
- Diversidade de recursos: Os recursos de um ambiente *Grid* incorporam recursos computacionais, armazenamento de dados, *links* de comunicação, *software*, licenças, equipamento especial, supercomputadores e *clusters*[1]. O ambiente *Grid* providencia acesso consistente, independente e transparente aos recursos.
- Centrado no utilizador: O principal objetivo de um ambiente *Grid* é satisfazer as necessidades das aplicações submetidas pelo utilizador final, ou seja, o recurso escolhido para execução da aplicação é escolhido através do ponto de vista do utilizador final, maximizando assim a performance da aplicação, independentemente do efeito sobre o sistema como um todo.

Embora os ambientes *Grid* sejam utilizados há bastante tempo, existem ainda um conjunto de desafios [7] na sua implementação. Um dos principais desafios de um ambiente *Grid* é a gestão e o manuseamento da heterogeneidade dos recursos, que resulta da grande diversidade de tecnologias, tanto a nível de *software* como de *hardware*, abrangidas pelo ambiente *Grid*. A existência de recursos heterogéneos, em ambientes *Grid*, faz com que seja necessário criar código portável para as aplicações que nele executam, de forma a permitir que as aplicações sejam independentes do sistema operativo ou

[1] *Cluster* é um aglomerado de computadores. Normalmente um *cluster* é constituído por um conjunto de computadores interligados entre si através de uma rede, operando como se fosse uma única máquina com grande capacidade de processamento e armazenamento.

da arquitetura da máquina onde são executadas, podendo ser adotadas estratégias de virtualização de forma a unificar o ambiente *Grid*.

Outro desafio envolve a manipulação dos recursos que se encontram difundidos através de fronteiras políticas e geográficas e sobre o controlo administrativo de diferentes organizações. Este desafio faz com que a performance e disponibilidade dos recursos seja de difícil previsão, uma vez que os pedidos de utilização dentro do domínio administrativo podem ganhar prioridade sobre os pedidos provenientes de domínios administrativos externos. Os ambientes *Grid* operam numa vasta variedade de infraestruturas de rede, desde redes locais, redes dedicadas de alta velocidade até redes globais como a *internet*. Desta forma, a velocidade, latência, largura de banda e disponibilidade da infraestrutura de rede é afetada tanto pelo tráfego gerado pelo ambiente *Grid* como pelo tráfego externo ao ambiente *Grid*, assumindo um estado dinâmico ao longo do tempo, que pode afetar abruptamente a conectividade entre os nodos do ambiente *Grid*. Este desafio apresenta-se crítico no momento da elaboração do plano de escalonamento de tarefas, podendo afetar significativamente a performance do escalonador. Como estratégia de resolução podem ser aplicadas técnicas de previsão de largura de banda, que permitem prever aproximadamente a largura de banda entre dois nodos num determinado período de tempo.

Estes ambientes são por definição dinâmicos, isto é, os atributos que caracterizam um nodo podem sofrer mudanças de valores em função do tempo. Todos os recursos pertencentes ao ambiente *Grid* podem ser dedicados, ou seja só executam tarefas provenientes da *Grid*, ou não dedicados, como por exemplo um posto de trabalho que executa tarefas para um ou mais utilizadores. Estes recursos possuem uma disponibilidade de processamento dinâmico, quer seja por executarem tarefas da *Grid* quer seja por o tipo de utilizadores que possuem. Este desafio faz com que o escalonador do ambiente *Grid* tenha de considerar a disponibilidade de um recurso de forma a melhorar a sua performance. Para isso, podem ser utilizados serviços de informação, que permitem consultar o estado de um recurso, ou técnicas de previsão de performance que conjugadas com o serviço de informação podem prever o estado de um recurso num determinado período de tempo.

Outro grande desafio que se coloca centra-se na problemática da segurança num ambiente Grid. A segurança é um fator de grande importância devido à necessidade de o ambiente *Grid* ter de se adaptar às políticas de segurança das organizações virtuais, garantindo segurança na comunicação entre tais organizações, possibilitando a identificação e autenticação de todos os intervenientes no processo. Em [7] é apresentada uma lista com os principais desafios a alcançar pelos ambientes *Grid*:

- Utilização dos recursos de forma ótima e eficiente.
- Partilha eficiente de recursos entre diferentes organizações.
- Autenticação e autorização dos utilizadores.

- Segurança no armazenamento de dados e programas.
- Segurança nas comunicações.
- Controlo centralizado ou semi-centralizado.
- Qualidade de Serviço (QoS) e contratos de nível de serviço (SLA).
- Interoperabilidade entre diferentes *Grid*.
- Suporte para processos transacionais.

2.1 Middlewares Grid

Um *Middleware Grid* é um *software* que permite a partilha de recursos heterogéneos, sendo constituído vulgarmente por um conjunto de componentes especializados em áreas como, por exemplo, a segurança, gestão de recursos, monitorização, serviços de informação, entre outros. O seu principal objetivo passa por fornecer um conjunto de aplicações e serviços que permitem a construção de um ambiente *Grid*, fornecendo também um conjunto de ferramentas e utilitários para manipular e gerir toda a *Grid*. Um dos principais problemas na sua utilização surge quando existe a necessidade de interligar duas VOs com diferentes *middlewares*, uma vez que estes podem utilizar diferentes protocolos de comunicação. Tendo isso em conta, e também pelo facto de atualmente não existir um *standard* para a maioria dos serviços disponibilizados, torna-se difícil interligar dois *middlewares*. Como solução, pode-se recorrer a *brokers* desenvolvidos especificamente para efetuar essa interligação. Além disso, a utilização de diferentes versões do mesmo *middleware* pode ser também problemática, uma vez que podem existir incompatibilidades de suporte ou funcionalidades que não sejam suportadas por todas as versões. Tendo isso em conta, é recomendado que todos os nodos do ambiente *Grid* tenham a mesma versão instalada e atualizada, de forma a evitar perdas de desempenho e possíveis erros de execução.

Normalmente grande parte dos *middlewares Grid* incluem de base, no seu pacote de instalação, um conjunto de *brokers* que permite efetuar a interligação com serviços externos, como por exemplo, um serviço de informação ou um gestor de trabalhos. Desta forma os *brokers* permitem completar a informação e funcionalidades base disponibilizadas. Como tal, a escolha de um *middleware Grid* é influenciada pelo conjunto de *brokers* disponibilizados.

2.1.1 Globus Toolkit

O *Globus Toolkit*[2] é um *software* de código livre, que pode ser utilizado para implementar *Grids* computacionais e aplicações baseadas em ambientes *Grid*. Com a sua utilização é permitida a partilha de poder computacional, base de dados e outras ferramentas. O *Globus Toolkit* apresenta-se como um

[2] http://www.globus.org/toolkit/

sistema seguro que pode operar sobre fronteiras empresariais, institucionais e geográficas, sem sacrificar a autonomia local de uma organização. O núcleo de serviços, interfaces e protocolos presentes no *Globus Toolkit* permite aos utilizadores aceder remotamente aos recursos enquanto que, simultaneamente, é preservado o controlo local sobre utilizadores que acedem aos recursos. Os componentes deste *software* estão divididos em áreas de segurança, gestão de recursos, serviços de informação e gestão de dados (Figura 3), sobre as quais se falará com mais detalhe de seguida.

Figura 3 - Arquitetura funcional do Globus Toolkit

Segurança

O *Grid Security Infrastructure* (GSI) [8] providencia métodos para autenticar os utilizadores da *Grid* e garantir uma comunicação segura, estando baseado no *Secure Sockets Layer* (SSL), *Public Key Infrastructure* (PKI) e na utilização de certificados X.509. Para um utilizador aceder a um recurso é necessária a existência de um certificado, mapeado numa conta de utilizador, na máquina remota. É também necessário que a *Certificate Authority* (CA) que emitiu o certificado seja de confiança. As CAs também fazem parte da definição do conceito de VO [4], pois as VOs são colaborações verticais em que os utilizadores que são certificados por uma CA da organização virtual ganham acesso aos recursos autenticados pela mesma CA. Neste cenário, as VOs podem cooperar entre si através do reconhecimento e confiança das CAs de cada uma das organizações virtuais, sendo assim possível aos utilizadores aceder a recursos de forma colaborativa. Para armazenar as credenciais dos utilizadores o *Globus Toolkit* utiliza o *myProxy Server*, que serve como repositório de credenciais e é responsável pela sua atribuição para uso em recursos remotos.

Gestor de Recursos

O pacote de gestão de recursos permite a alocação de recursos através da submissão de trabalhos, preparação de ficheiros executáveis, monitorização de trabalhos e recolha de resultados. O pacote de gestão de recursos possui os seguintes componentes:

- ***Globus Resource Allocation Manager* (GRAM)** [9] – Este é o responsável por providenciar a execução remota de trabalhos e reportar o estado da execução. Para tal um cliente deve realizar o pedido de submissão de um trabalho ao *daemon*[3] *Gatekeeper* da máquina remota, que por sua vez irá verificar se o cliente está autorizado. Caso a autenticação termine com sucesso, o *Gatekeeper* inicia o gestor de trabalhos, que inicializa a execução e monitorização do trabalho. Tendo em conta que o GRAM possui interfaces para vários escalonadores locais, tais como *Portable Batch System* (PBS), *Load Sharing Facility* (LSF) e *Load Leveler*, o gestor de trabalhos é criado dependendo do escalonador local do sistema. Os detalhes de um trabalho são especificados pelo *Globus Resource Specification Language* (RSL), que é parte constituinte do GRAM, providenciando uma sintaxe constituída por pares *<atributo, valor>* para descrever os recursos requeridos para o trabalho incluindo, por exemplo, o mínimo de memória, o número de CPUs, entre outros.

- ***Globus Access to Secondary Storage* (GASS)** [10] - Trata-se de um mecanismo de acesso a ficheiros que permite às aplicações procurar, abrir e escrever em ficheiros remotos. Este serviço é utilizado para recolha de resultados assim que o trabalho termina. Para tal é utilizado o protocolo HTTPS para transferência de dados, que contém um conjunto de funções, pertencentes à camada GSI, para reforçar as permissões de acesso aos dados e armazenamento.

Serviços de Informação

O pacote de serviços de informação disponibiliza informação dinâmica e estática sobre os nodos que estão interligados na *Grid*. Este serviço é denominado *Monitoring and Discovery Service* (MDS) [11], e é o responsável por providenciar suporte para agregar, publicar, consultar e pesquisar informação sobre os recursos pertencentes à *Grid*.

Gestor de dados

O pacote de gestão de dados disponibiliza utilitários para transferir, armazenar e gerir grandes quantidades de dados. Os componentes deste pacote são os seguintes:

- ***Grid* FTP -** É uma extensão do protocolo padrão *File Transfer Protocol* (FTP) que possibilita a transferência de dados com segurança, eficiência e fiabilidade em ambientes *Grid*. Adicionalmente em comparação ao protocolo FTP, o *Grid* FTP fornece suporte ao GSI para

[3] *Daemon* é um programa/serviço que se encontra em execução como um processo de *background*, para receber instruções/pedidos de outros programas para executar uma determinada ação.

transferência de dados com autenticação, invocação de transferência de dados de terceiros e transferência de dados paralela e parcial.

- **R*eplica Location and Management* -** Este componente suporta múltiplas localizações na *Grid* para o mesmo ficheiro. Usando as funções de gestão de réplicas, um ficheiro pode ser registado no *Replica Location Service* (RLS) e as suas réplicas podem ser criadas ou eliminadas. Com o RLS, um ficheiro é identificado pelo seu *Logical File Name* (LFN), e é registado como uma coleção lógica, ficando assim guardado num ficheiro um apontador para a sua localização física. Esta informação está disponível através de consultas e pesquisas ao RLS.

2.1.2 Virtual Data Toolkit

O *Virtual Data Toolkit*[4] nasceu como produto da *Open Science Grid* (OSG). O grande objetivo da OSG consiste em satisfazer os requisitos crescentes da comunidade científica, que procura recursos de computação de grande desempenho para utilização em aplicações científicas colaborativas. Este *middleware* é um agregador de pacotes de *software*, especializado em computação distribuída, que permite uma instalação e configuração simples e rápida através do *software PacMan*[5]. O seu principal objetivo é facilitar a implementação, utilização e manutenção, por parte dos utilizadores, de *software* de computação distribuída. O VDT contém uma grande variedade de *software* tais como *Condor-G* e *Globus Toolkit*.

2.1.3 gLite

O *gLite*[6] é um *middleware Grid* gratuito que agrega um conjunto de componentes de outros *middlewares Grid*, permitindo a interoperabilidade com outros *middlewares*, tais como o *Globus*, VDT e UNICORE. Os seus componentes estão divididos em áreas tais como segurança, transferência de dados, gestor de recursos, gestor de trabalhos e sistema de informação, sobre as quais se falará com mais detalhe seguidamente.

Segurança

A segurança no gLite é obtida através da utilização do GSI e do protocolo *Transport Layer Security* (TLS), obtendo-se, desta forma, uma comunicação autenticada e segura. A autenticação única também é suportada, permitindo assim aceder a qualquer recurso na *Grid*, mediante uma única execução do processo de autenticação. Para armazenar as credenciais dos utilizadores o gLite utiliza o *myProxy*

[4] *http://vdt.cs.wisc.edu/*
[5] http://atlas.bu.edu/~youssef/pacman/
[6] *http://glite.cern.ch/*

Server. Já o mapeamento de políticas entre organizações virtuais e organizações físicas é realizado através do *Virtual Organization Membership Service* (VOMS), que é um sistema que realiza a gestão de autorização e privilégios de utilizadores em ambientes multi-institucionais e colaborativos. Através deste sistema é permitida a manutenção de uma base de dados de perfis e de utilizadores, e a geração de credenciais a pedido, permitindo a autenticação única em diversos sistemas. Através deste sistema o gLite consegue definir quais os privilégios de segurança de cada utilizador da VO.

Transferência de Dados

A transferência de dados no gLite é assegurada por três serviços: armazenamento, catálogo e réplica de ficheiros, e gestor de dados [12]. O serviço de armazenamento abstrai os recursos de armazenamento, sejam eles um disco rígido, um *array* de discos, ou um diretório. Já o catálogo de réplicas gere a informação relativa à localização física dos ficheiros. Por fim, o serviço gestor de dados guarda informação relativa aos vários serviços de armazenamento e seus respetivos catálogos, de forma a escolher as melhores fontes de dados e garantir uma maior eficiência na execução de trabalhos dependentes de transferência de ficheiros.

Gestor de Recursos

No gLite é criada uma abstração dos recursos computacionais, através da definição de elementos de computação, que podem corresponder a um *cluster* ou a apenas uma máquina.

Gestor de Trabalhos

Permite realizar a gestão de trabalhos em qualquer elemento de computação. Além do serviço de submissão e monitorização de trabalhos, o gLite inclui também um sistema de gestão de carga que distribui o trabalho por vários elementos de computação, de forma a otimizar a utilização dos recursos.

Sistema de Informação

O sistema de informação do gLite armazena informação relativa a cada trabalho que é submetido, executado ou repetido, para que a informação acerca do estado dos elementos de computação e dos trabalhos possa ser consultada para posterior análise [12].

2.1.4 UNICORE

O *UNICORE*[7] foi desenvolvido na Alemanha em 1997 com o principal objetivo de facilitar o acesso aos supercomputadores alemães, por parte dos seus utilizadores [13]. É um *middleware* gratuito que

[7] *http://www.unicore.eu/*

possui uma instalação simples, tanto em modo servidor como em modo cliente, e a sua interface gráfica é bastante intuitiva, estando disponível tanto para sistemas Windows como Unix/Linux. A interoperabilidade com outros *middleware Grid* é permitida, suportando sistemas tais com o *Globus* e o gLite. Os componentes do UNICORE estão organizados por áreas, nomeadamente segurança, transferência de dados, gestor de trabalhos e sistema de gestão de *clusters*. De seguida é apresentada cada uma destas áreas.

Segurança

A segurança no UNICORE é alcançada através da utilização do protocolo SSL. A identificação dos utilizadores no ambiente *Grid* é possível devido à utilização de certificados digitais X.509, a partir dos quais é permitido mapear um utilizador remoto com um utilizador local aos recursos onde o servidor UNICORE se encontra.

Transferência de Dados

Este serviço possibilita a transferência de conjuntos de dados para serem processados por um recurso remoto, sendo ainda possível transferir o resultado da computação. Ao contrário dos *middlewares* anteriores, o UNICORE não possui um sistema de gestão de réplicas de ficheiros.

Gestor de Trabalhos

O gestor de trabalhos no UNICORE suporta a submissão, monitorização, cancelamento e repetição de trabalhos. A submissão de trabalhos é efetuada através do cliente e os trabalhos são recebidos do lado do servidor por um *gateway*, que entrega os pedidos de execução ao *Network Job Supervisor* (NJS). Estes pedidos são transformados em sub-trabalhos que, por sua vez, são executados nos diversos recursos disponíveis na *Grid*. O NSJ pode, no entanto, optar pela utilização de recursos pelos quais outro *gateway* é responsável, tornando-se deste modo possível a troca de trabalhos entre vários servidores UNICORE.

Sistema de Gestão de Clusters

A integração com vários sistemas de gestão de *clusters*, tais como o PBS, LSF, entre outros, é possível recorrendo ao seu sistema de gestão de *clusters*. Esta integração é obtida através do *Target System Interface* (TSI), que permite a compatibilidade com os vários sistemas de gestão de *clusters*.

2.2 Comparação entre middlewares Grid

Para comparar os diferentes *middlewares Grid* foi considerado um conjunto de características pertinentes para análise: distribuição, plataformas, facilidade de instalação/configuração, documentação e integração com outros *middlewares Grid*.

No que diz respeito à distribuição, foi dada particular relevância aos sistemas gratuitos, pois integram componentes desenvolvidos por uma larga comunidade científica, apresentando assim uma maior robustez e menores custos relacionados com a aquisição de licenças. Na avaliação dos *middlewares* foram privilegiados os sistemas que mais plataformas suportam, uma vez que um ambiente *Grid* tende a ser heterogéneo, ou seja, composto por diferentes recursos, inclusive *middlewares Grid*. A facilidade de instalação/configuração destes sistemas apresenta-se como um fator bastante subjetivo. Os sistemas classificados como de *alta facilidade*, possuem usualmente uma interface gráfica para instalação e configuração do sistema, enquanto que os sistemas com *média facilidade* possuem uma interface assente numa linha de comandos, que permite a sua instalação e configuração seguindo um conjunto de passos e questionários. Por fim, os sistemas classificados com *baixa facilidade* exigem a configuração e edição de ficheiros de configuração específicos seguindo um conjunto de regras.

Na análise da documentação existente, o *Globus Toolkit* obteve classificação de *muito bom* uma vez que é um projeto com grande maturidade e que possui um repositório de documentação extenso, que explica passo a passo todas as regras de configuração e instalação. Adicionalmente, o facto de este ser bastante utilizado no meio científico, contribui para o aparecimento de vários fóruns de discussão e tutoriais especializados. O VDT, apesar de conter o *Globus Toolkit* embutido, possui outros pacotes de *software* que não se encontram bem documentados, obtendo assim uma avaliação final de *médio*. O gLite e o UNICORE possuem bons repositórios de documentação, contudo ainda não existe muita documentação externa a página oficial do projeto. A integração com outros *middewares* apresenta-se como um fator importante pois contribui para a diversificação e reutilização de recursos externos que possuam outros *middlewares* instalados. Na Tabela 1 é apresentado uma breve análise comparativa entre os diferentes *middlewares Grid* analisados.

Tabela 1 - Comparativo entre os diferentes middlewares Grid

	Globus Toolkit	VDT	gLite	UNICORE
Distribuição	Gratuita	Gratuita	Gratuita	Gratuita
Plataformas	Windows, Unix/Linux e MacOs	Unix/Linux e MacOs	Scientific Linux	Unix/Linux e MacOs
Facilidade Instalação/Configuração	Baixa	Média	Baixa	Alta
Documentação	Muito Boa	Média	Boa	Boa
Integração com outros *middlewares Grid*	Sim	Sim	Sim	Sim

2.3 Serviços de Informação

A consulta de informação relativa aos recursos disponíveis num ambiente *Grid* é um requisito fundamental, sendo por isso essencial a existência de um serviço responsável pela sua recolha. Para tal existem os serviços de informação, cuja função consiste em recolher um conjunto de informações relativas a cada nodo da *Grid*, que podem ser descrições de *hardware*, *software* ou serviços. Toda essa informação será posteriormente disponibilizada para consulta. Os serviços de informação permitem também alertar para o facto de existirem recursos a integrar ou a abandonar o ambiente *Grid*. Segundo Fitzgerald [14], é através do uso deste tipo de serviços que se torna possível a adaptação da execução de tarefas ao estado dinâmico da *Grid*. Só desta forma é possível suportar ambientes dinâmicos e heterogéneos. Para que um escalonador possa decidir a submissão de uma tarefa a um recurso precisa de conhecer antecipadamente quais os recursos presentes no ambiente *Grid* bem como as suas características. Só assim é possível avaliar o cumprimento dos pré-requisitos das tarefas e comparar recursos com vista à escolha do nodo mais adequado, aumentando, consequentemente, ao longo do tempo, o desempenho na execução de tarefas.

Uma vez que a *Grid*, por definição, é um ambiente geograficamente distribuído, por razões de escalabilidade e ocupação de largura de banda, a consulta constante de informação detalhada não pode ser considerada uma opção. Para não se prejudicar a escalabilidade pode ser escolhida uma alternativa menos intensiva, ou seja numa organização virtual podem ser configurados um ou mais agregadores de informação que publicam periodicamente informação específica, não detalhada, sobre o estado global da *Grid*. Desta forma, a dimensão da informação publicada não se torna problemática com o crescimento do número de nodos na organização virtual, até porque os agregadores podem estar agrupados de forma hierárquica. Sempre que seja necessário consultar informação detalhada acerca do

estado de um recurso ou conjunto de recursos, é contactado o serviço de informação responsável pela publicação de informação sobre o nodo pretendido.

2.3.1 Serviço de Descoberta e Monitorização de Recursos

O MDS4 é um serviço de descoberta e monitorização de recursos, incluído no *Globus Toolkit*, que define um conjunto de protocolos padrão para aceder e disponibilizar informação através de esquemas *standards* para representar informação, permitindo a recolha de informação de monitores de *clusters* (tais como *Ganglia*[8], *Hawkeye*[9], *Clumon*[10] e *Nagios*[11]), serviços (ex.: GRAM, RFT e RLS) e sistemas de fila de espera (ex.: PBS, LSF, Torque). A informação é disponibilizada pelo MDS4 através de um esquema XML baseado no *standard* GLUE. Este serviço baseia-se em protocolos e interfaces de consulta, subscrição e notificação definidos pelo *WS Resource Framework* (WSRF) e *WS-Notification* que são implementados pelo *GT4 Web Services Core*. A especificação 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.

A especificação WSRF define quatro normas:

- **WS-Resource Properties,** que define como consultar e modificar as propriedades dos recursos associados ao serviço;
- **WS-Resource Life Time,** que define como gerir o ciclo de vida dos serviços da *Grid*;
- **WS-Service Group,** que se foca na representação e gestão de conjuntos de serviços, permitindo a associação de serviços;
- **WS-Base Faults,** que se preocupa com a comunicação e tratamento de erros durante o processamento de pedidos.

Um conjunto de fornecedores de informação implementados no MDS4 permite que seja recolhida informação proveniente de outras fontes, permitindo ao MDS4 recolher informação de outras ferramentas e sistemas, tais como o monitor de *clusters* Ganglia e os escalonadores PBS e Condor. Posteriormente, a informação disponibilizada pode ser utilizada, por exemplo, por escalonadores, portais, sistemas de alarme, entre outros. O MDS4 providencia dois serviços de alto nível: um serviço de índices, que recolhe e publica informação agregada acerca das fontes de informação, e um serviço de *triggering*, que recolhe informação acerca dos recursos e realiza uma ação quando uma determinada condição for verificada. Adicionalmente, uma interface baseada na *Web* denominada *Web* MDS providencia uma transformação XSLT que permite obter uma interface visual sobre os dados.

[8] *http://ganglia.sourceforge.net/*
[9] *http://research.cs.wisc.edu/condor/hawkeye/*
[10] *http://clumon.ncsa.illinois.edu/*
[11] *http://www.nagios.org/*

2.3.2 Ganglia

O Ganglia é um sistema de monitorização escalável e distribuído para sistemas computacionais de alta performance tais como *Clusters* e *Grids*, tendo sido desenvolvido por Matthew L. Massie como um projeto de código livre. Este sistema utiliza XML para representar os dados e é composto por vários *daemons*, uma interface *Web* em PHP e algumas ferramentas. A monitorização que efetua, recolhe valores a partir de várias métricas tais como carga do CPU, memória disponível, utilização do disco, carga de utilização da rede, versão do sistema operativo, entre outros.

O *daemon* de monitorização utilizado pelo Ganglia é o Gmond, que é um *daemon multi-thread* que deve ser executado em todos os nodos monitorizados. Este *daemon* monitoriza variáveis que estão definidas num ficheiro de configuração. Já o *daemon* Gmetad permite ao Ganglia organizar hierarquicamente a informação, recolhida de vários *clusters*, em forma de árvore.

A interface Web disponibilizada pelo Ganglia permite visualizar a informação recolhida para que administradores de sistemas e utilizadores comuns possam extrair informação vital sobre os nodos dos *clusters*, sendo ainda possível agrupar essa mesma informação por *clusters* ou por redes. A estrutura do Ganglia permite que sejam usadas apenas partes do *toolkit*.

2.3.3 Nagios

O Nagios é um servidor de monitorização de sistemas, que providencia informação para um cliente. O pacote de *software* Nagios deve ser instalado em todas as máquinas pertencentes ao ambiente *Grid*, sendo necessária a ativação de todos os serviços de forma a monitorizar as características de cada máquina do ambiente *Grid*. É possível criar uma hierarquia da rede, que permite ao sistema parar de monitorizar recursos que não possam ser alcançados. Isto significa que se por exemplo um *router* for desligado, todas as consultas sobre os nodos hierarquicamente inferiores deste serão paradas. Este sistema combina várias funcionalidades de monitorização num enorme pacote, que permite monitorizar *hardware*, *software* e serviços que são especificados pelo administrador de sistemas. É ainda permitido enviar alertas no caso de ocorrer algum evento particular. Além disso, este sistema pode monitorizar protocolos de rede tais como POP3, SMTP, ICMP e NNTP, disponibilizando ainda um interface *Web* que pode ser opcionalmente configurado para consultar o estado da rede, as notificações e um histórico de problemas.

2.4 Comparativo entre Serviços de Informação

Para comparar os diferentes serviços de informação foram considerados um conjunto de características base, a saber: monitorização de serviços *Grid* disponibilizados pelo *middleware Grid*, monitorização de *hardware*, serviço agregador de informação e interface *Web*. A monitorização de serviços *Grid*

disponibilizados pelo *middleware Grid* permite consultar informação sobre serviços específicos dos *middlewares Grid*, tais como serviço de gestão de trabalhos, serviço de gestão de transferências, entre outros. A monitorização de *hardware* permite a recolha de informação relacionada com as características do *hardware*, como por exemplo a velocidade de processamento, a memória disponível ou o espaço em disco. O serviço de agregação de informação permite agregar a informação dos recursos num servidor central, providenciando um repositório de informação atualizado, reduzindo, consequentemente, os custos de comunicação efetuados na consulta da informação. A interface *Web* permite ao utilizador consultar a informação sobre os recursos a partir de um navegador *Web*, simplificando deste modo o trabalho do gestor do ambiente *Grid*. Na Tabela 2 está apresentado uma pequena análise comparativa entre os vários serviços de informação analisados.

Tabela 2 - Comparação entre os vários serviços de informação

	MDS4	Ganglia	Nagios
Monitorização de serviços *Grid* disponibilizados pelo *middleware Grid*	Sim	Não	Não
Monitorização de *hardware*	Não	Sim	Sim
Serviçoagregador de informação	Sim	Sim	Sim
Interface *Web*	Sim	Sim	Sim

3 Configuração e Execução de Processos ETL em *Grids*

3.1 Caracterização e descrição de um ambiente para acolhimento de um sistema ETL

O processo de ETL, tal como foi referido anteriormente, é uma componente crítica no sucesso da implementação de um SDW. Isto deve-se ao facto de o seu desenho e a sua implementação envolver o desenvolvimento de processos complexos que interagem com grande parte dos componentes de um SDW e que necessitam de aceder a dados processados, os quais, maioritariamente, se encontram armazenados em sistemas transacionais cuja disponibilidade de acesso é condicionada e limitada. Com a adição de um maior número de áreas de negócio a um SDW, este deverá ter a capacidade de evoluir, incorporando operações mais complexas e morosas, condicionando assim a escolha da infraestrutura de suporte à execução do seu processo de ETL. Sendo assim, o planeamento da sua infraestrutura deve responder dinamicamente à quantidade de dados a processar, considerando um aumento dos dados consoante o tempo e tendo em conta o poder computacional exigido, para que estas operações terminem dentro de um período de tempo estipulado [3].

3.2 Políticas e estratégias para o escalonamento de ETL em GRID

A execução de um processo de ETL encontra-se diretamente relacionada com a infraestrutura em que é executado. O processamento em paralelo permite melhorar a performance global do processo ETL nas situações em que se verifica a existência de um grande volume de dados a ser tratado. Considerando uma metodologia descentralizada [3], na qual os dados, ou sistemas fontes, se encontram representados em documentos XML, torna-se possível realizar a sua divisão em documentos de menor dimensão, reduzindo assim o tempo de processamento e permitindo o acesso em paralelo a diferentes partes do ficheiro. É também possível recorrer à duplicação dos sistemas

fonte, isto é dos documentos XML, para que seja possível executar vários processos em diferentes fluxos de dados. Na fase de transformação de um processo de ETL é também possível decompor as transformações em aplicações, cuja execução possa ser realizada em paralelo, isto é, que possa executar simultaneamente diversos componentes do mesmo fluxo de dados.

3.2.1 Cálculo de disponibilidade

De modo a ser possível levar a cabo a avaliação de um dado recurso, ou seja, verificar se este é adequado para executar uma dada tarefa, tendo em conta a sua disponibilidade, deve-se recorrer ao cálculo de disponibilidade. Para efetuar este tipo de cálculo é necessário efetuar a recolha das características de todos os recursos pertencentes ao ambiente *Grid* em causa. Para realizar a recolha e armazenamento periódico do estado do ambiente *Grid* foi criado um *daemon* que consulta periodicamente o servidor MDS. A informação é disponibilizada através de um esquema XML baseado no *standard* GLUE. Na Figura 4 está apresentado um exemplo da informação disponibilizada pelo MDS.

```
    </ns44:AggregatorConfig>
    <ns44:AggregatorData>
     <ns1:GLUECE xmlns:ns1="http://mds.globus.org/glue/ce/1.1">
      <ns1:Cluster ns1:Name="debian4" ns1:UniqueID="debian4" xsi:type="ns1:ClusterType" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance">
       <ns1:SubCluster ns1:Name="main" ns1:UniqueID="main">
        <ns1:Host ns1:Name="debian1.ciicesi.intra.estgf.ipp.pt" ns1:UniqueID="debian1.ciicesi.intra.estgf.ipp.pt">
         <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0" ns1:CacheL1I="0" ns1:CacheL2="0" ns1:ClockSpeed="2391"
ns1:InstructionSet="x86"/>
         <ns1:MainMemory ns1:RAMAvailable="666" ns1:RAMSize="1252" ns1:VirtualAvailable="3910" ns1:VirtualSize="4510"/>
         <ns1:OperatingSystem ns1:Name="Linux" ns1:Release="2.6.35-22-generic"/>
         <ns1:Architecture ns1:SMPSize="1"/>
         <ns1:FileSystem ns1:AvailableSpace="65527" ns1:Name="entire-system" ns1:ReadOnly="false" ns1:Root="/"
ns1:Size="77326"/>
         <ns1:NetworkAdapter ns1:IPAddress="172.20.109.91" ns1:InboundIP="true" ns1:MTU="0"
ns1:Name="debian1.ciicesi.intra.estgf.ipp.pt" ns1:OutboundIP="true"/>
         <ns1:ProcessorLoad ns1:Last15Min="8" ns1:Last1Min="2" ns1:Last5Min="6"/>
        </ns1:Host>
        <ns1:Host ns1:Name="debian2.ciicesi.intra.estgf.ipp.pt" ns1:UniqueID="debian2.ciicesi.intra.estgf.ipp.pt">
         <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0" ns1:CacheL1I="0" ns1:CacheL2="0" ns1:ClockSpeed="1495"
ns1:InstructionSet="x86"/>
         <ns1:MainMemory ns1:RAMAvailable="70" ns1:RAMSize="376" ns1:VirtualAvailable="1196" ns1:VirtualSize="1510"/>
         <ns1:OperatingSystem ns1:Name="Linux" ns1:Release="2.6.26-2-686"/>
         <ns1:Architecture ns1:SMPSize="1"/>
         <ns1:FileSystem ns1:AvailableSpace="31320" ns1:Name="entire-system" ns1:ReadOnly="false" ns1:Root="/"
ns1:Size="39172"/>
         <ns1:NetworkAdapter ns1:IPAddress="172.20.109.92" ns1:InboundIP="true" ns1:MTU="0"
ns1:Name="debian2.ciicesi.intra.estgf.ipp.pt" ns1:OutboundIP="true"/>
         <ns1:ProcessorLoad ns1:Last15Min="0" ns1:Last1Min="0" ns1:Last5Min="0"/>
        </ns1:Host>
        <ns1:Host ns1:Name="debian3.ciicesi.intra.estgf.ipp.pt" ns1:UniqueID="debian3.ciicesi.intra.estgf.ipp.pt">
         <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0" ns1:CacheL1I="0" ns1:CacheL2="0" ns1:ClockSpeed="2391"
ns1:InstructionSet="x86"/>
```

Figura 4 - Extrato da informação recolhida através do servidor MDS

Após realizar a consulta da informação providenciada pelo MDS foram selecionados para armazenamento os seguintes dados:

- Nome da máquina.
- Número total de processadores.

- Frequência do processador.
- Tamanho total da memória RAM.
- Quantidade atual de memória RAM disponível.
- Tamanho total de espaço em disco.
- Quantidade atual de espaço em disco disponível.
- Carga de utilização do processador no último minuto.
- Carga de utilização do processador nos últimos cinco minutos.
- Carga de utilização do processador nos últimos quinze minutos.
- Data e Hora da recolha.

Todos os dados recolhidos foram armazenados num ficheiro CSV de forma a garantir uma grande portabilidade e uma fácil manipulação. Depois, é possível realizar a divisão do ficheiro CSV e submeter a análise de cada fragmento às máquinas presentes na *Grid*, reduzindo assim o tempo total necessário para processar os dados de todos os recursos do ambiente *Grid*. Consequentemente, este processo também diminui o tempo de escalonamento, o que é uma vantagem para ambientes *Grid* de grandes dimensões, uma vez que os dados recolhidos sobre os recursos demoram muito tempo para serem processados. Com esta metodologia é possível submeter a análise e processamento dos dados de recolha dos vários recursos presentes no ambiente *Grid*. A Figura 5 apresenta um exemplo do ficheiro de recolha.

HostName	TotalCpus	ClockSpeed	RamSize	RamAvailabel	FileSystemSize	FileSystemAvailableSpace	ProcessorLoadLast1min	ProcessorLoadLast5min	ProcessorLoadLast15min	Year	Month	Day	DayWeek	Hours	Minutes
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	94	77326	65040	95	95	90	111	5	8	3	17	22
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	36	39172	31269	0	0	0	111	5	8	3	17	22
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	23	39848	30383	18	17	16	111	5	8	3	17	22
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	708	38616	29878	0	0	0	111	5	8	3	17	22
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	48	77326	65025	47	72	88	111	5	8	3	18	9
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	32	39172	31269	0	0	0	111	5	8	3	18	9
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	22	39848	30383	27	15	18	111	5	8	3	18	9
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	708	38616	29877	0	0	0	111	5	8	3	18	9
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	48	77326	65025	47	72	88	111	5	8	3	18	10
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	32	39172	31269	0	0	0	111	5	8	3	18	10
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	22	39848	30383	27	15	18	111	5	8	3	18	10
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	708	38616	29877	0	0	0	111	5	8	3	18	10
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	48	77326	65025	47	72	88	111	5	8	3	18	11
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	32	39172	31269	0	0	0	111	5	8	3	18	11
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	22	39848	30383	27	15	18	111	5	8	3	18	11
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	708	38616	29877	0	0	0	111	5	8	3	18	11
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	108	77326	64998	61	57	49	111	5	13	1	15	42
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	5	39172	31264	38	19	6	111	5	13	1	15	42
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	76	39848	30364	65	61	36	111	5	13	1	15	42
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	668	38616	29880	0	0	0	111	5	13	1	15	42
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	108	77326	64998	61	57	49	111	5	13	1	15	42
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	5	39172	31264	38	19	6	111	5	13	1	15	42
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	76	39848	30364	65	61	36	111	5	13	1	15	42
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	668	38616	29880	0	0	0	111	5	13	1	15	42
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	73	77326	64998	39	64	55	111	5	13	1	15	54
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	6	39172	31264	0	3	3	111	5	13	1	15	54
debian3.ciicesi.intra.estgf.ipp.pt	1	2391	494	47	39848	30381	17	34	32	111	5	13	1	15	54
debian4.ciicesi.intra.estgf.ipp.pt	1	2793	1518	688	38616	29880	0	1	0	111	5	13	1	15	54
debian1.ciicesi.intra.estgf.ipp.pt	1	2391	1252	73	77326	64998	39	64	55	111	5	13	1	15	55
debian2.ciicesi.intra.estgf.ipp.pt	1	1495	376	6	39172	31264	0	3	3	111	5	13	1	15	55

Figura 5 - Exemplo do ficheiro de recolha

Tendo em vista a otimização da distribuição do processo de ETL na *Grid*, é necessário avaliar os recursos existentes relativamente à sua performance e disponibilidade num determinado instante. Para isso foi considerado o trabalho desenvolvido em [15], em especial os aspetos relacionados com o cálculo de disponibilidade de uma *Grid*. A disponibilidade computacional mede a máxima capacidade de poder de processamento que uma máquina é capaz de disponibilizar num determinado momento,

considerando a sua carga computacional em utilização. Em [15] a disponibilidade computacional é o factor de decisão que mais influencia o tempo de execução de tarefas e a sua taxa de sucesso. A fórmula (1) permite quantificar a disponibilidade computacional de cada nodo.

$$Perf_{AVAILABILITY} = (kCPU \times kCPU_{ARCH} \times CPU_{FREQ} \times Free_{CPUs} + kMEM \times Avail_{RAM}) \times Avail_{COEF}. \quad (1)$$

O cálculo de disponibilidade é afetado pelos seguintes fatores:

- **kCPU**, que representa o peso atribuído ao processador, este fator pode ser configurável conforme a importância que o processador tenha para uma determinada operação ETL.
- **kCPUarch**, que expõe o peso atribuído à arquitetura do processador.
- **CPUfreq**, que é a frequência de operação do processador.
- **FreeCpus**, que indica o número de núcleos de processamento disponíveis.
- **kMEM**, que representa o peso atribuído à memória RAM, este fator pode ser configurável conforme a importância que a memória RAM tenha para uma determinada operação ETL.
- **AvailRam,** que é a quantidade total de memória RAM disponível.

Esta disponibilidade assenta na avaliação de fatores como o tipo, velocidade e importância relativa do processador, em comparação com a disponibilidade e importância da memória local para a execução do trabalho. Este cálculo de disponibilidade é ainda afetado por um fator que é inversamente proporcional à carga de cada nodo da *Grid*. Neste trabalho propõem-se a utilização da carga computacional média dos últimos cinco minutos, que pode ser determinada da seguinte maneira:

$$Avail_{COEF} = 1 - CPU_{Load\ 5\ Min}. \quad (2)$$

3.2.2 Avaliação da largura de banda

De forma a avaliar a largura de banda disponível entre as diversas máquinas presentes no ambiente *Grid*, foi criado um *daemon* que é responsável por transferir diariamente ficheiros entre as máquinas que constituem o ambiente de execução. Foi elaborado um cenário de testes para avaliar a performance do *daemon* criado, no qual se injectou um ficheiro cujo tamanho foi definido em 1 megabyte, de forma a não causar impacto e congestionamento nos nodos da *Grid* que possuam uma reduzida largura de banda. O *daemon* desenvolvido considera também um histórico de trabalhos executados na *Grid*, fazendo com que assim seja possível verificar os tempos de transferência dos

trabalhos, evitando transferências desnecessárias de ficheiros e poupando largura de banda na *Grid* no momento de testar a sua performance.

A informação sobre a transferência dos ficheiros é armazenada num servidor que agrega a informação proveniente de todas as máquinas. A recolha da informação é efetuada através de um *socket* TCP que o *daemon* disponibiliza, para posterior consulta. A informação pode estar organizada em forma de árvore, o que evita congestionamentos na troca de informação. A Figura 6 apresenta um esquema da arquitetura criada.

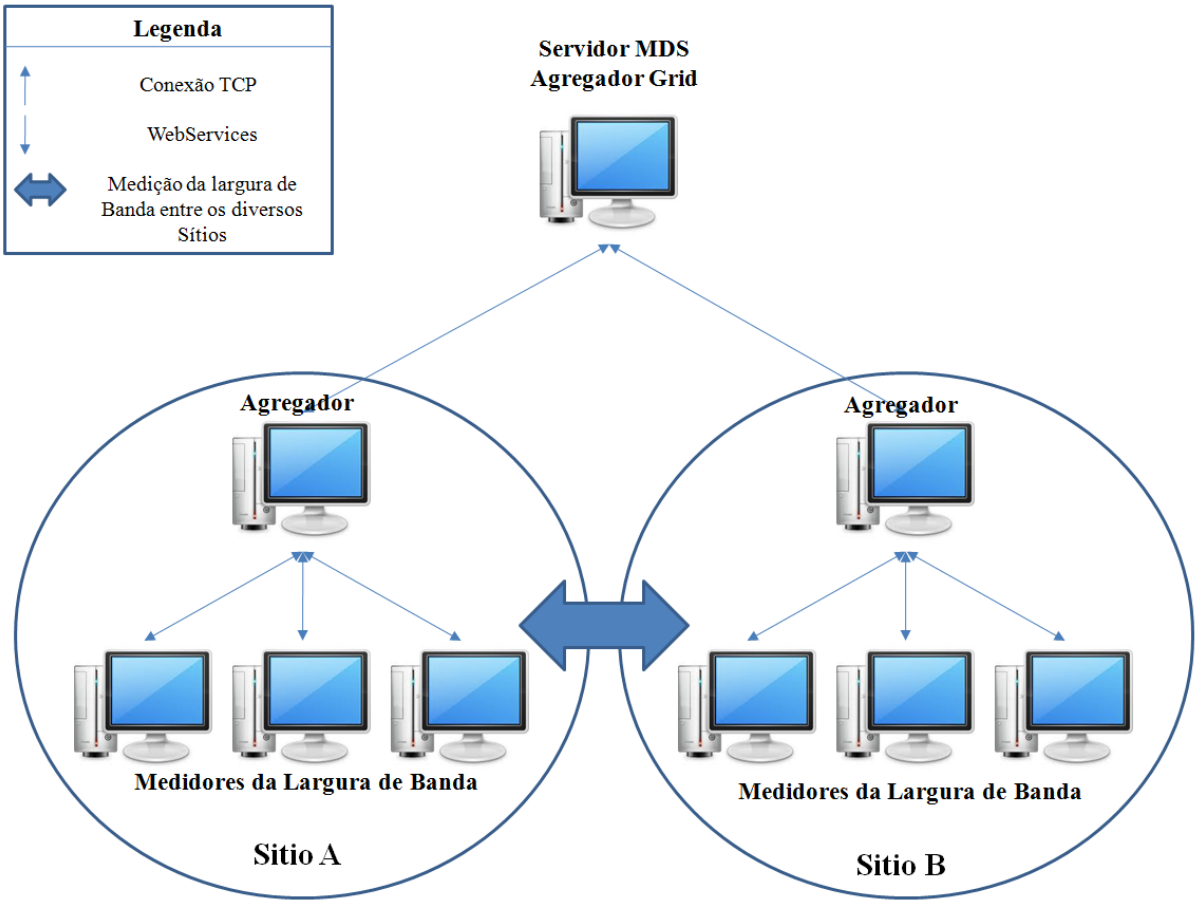


Figura 6 - Arquitetura do sistema de consulta da largura de banda do ambiente Grid

Todas as máquinas possuem um *daemon* que consulta o servidor MDS para obter informações sobre as restantes máquinas que pertencem ao ambiente *Grid*. Após obter a informação, o *daemon* começa a testar a conectividade da sua máquina com as restantes, através do envio de um ficheiro. Após a conclusão do teste de conectividade, o *daemon* reporta todos os dados recolhidos ao servidor agregador de nível superior através de *sockets* TCP. Este processo só é efetuado quando não existe informação suficiente no ficheiro de histórico sobre os trabalhos executados no ambiente *Grid*, permitindo assim avaliar a largura de banda entre os nodos.

De forma a calcular a largura de banda disponível entre duas máquinas é utilizada a fórmula (3). Esta fórmula não apresenta um valor real sobre a largura de banda. No entanto, ela providencia uma forma de comparar a diferença entre larguras de banda, sendo esta diferença um valor com maior importância para a escolha entre recursos com disponibilidade semelhantes.

$$LarguraBanda[i,j] = TamanhoFicheiro[i,j] / Tempo[i,j] \quad (3)$$

O cálculo da largura de banda é afetado pelos seguintes fatores:

- **LarguraBanda[i,j]**, que representa a largura de banda disponível para o envio de ficheiros da máquina **i** para a máquina **j**.
- **TamanhoFicheiro[i,j]**, que é o tamanho do ficheiro que é enviado para testar a largura de banda da máquina **i** para a máquina **j**.
- **Tempo[i,j]**, que possui o tempo total da transferência, do ficheiro de teste, da máquina **i** para a máquina **j**.

3.2.3 Previsão de disponibilidade

Para alcançar melhores resultados na previsão de performance de um recurso, foram utilizados alguns algoritmos de *clustering* apresentados em [15], com o objetivo de descobrir padrões nos dados que foram recolhidos. Com a utilização desses algoritmos foi possível explorar visualmente a performance de disponibilidade nos recursos, detetando-se assim grupos de recursos que partilham a mesma classe de performance. O algoritmo de *clustering* utilizado foi o KMeans[12]. Com a utilização deste algoritmo determinou-se quais os dados que eram mais relevantes para recolha e os que mais afetavam a classe de performance de uma máquina. Após a obtenção desses dados foi necessário proceder à avaliação dos recursos, recorrendo-se, para isso, a algoritmos de árvores de decisão. De acordo com [15], para aplicar esse tipo de algoritmo é necessário subdividir os nodos em seis classes de performance (C0 – C5) calculadas em (4), uma vez que os algoritmos de árvores de decisão não conseguem prever valores numéricos contínuos, sendo assim necessário realizar a conversão para um valor não numérico denominado classe.

$$(Perf_{AVALIABILITY} \times 5) / (max(Perf_{AVALIABILITY})) . \quad (4)$$

[12] http://www.cs.cmu.edu/~dpelleg/kmeans.html

Seguidamente, utilizando o algoritmo de árvores de decisão foi criado o modelo de previsão (Figura 7) utilizando a ferramenta RapidMiner[13], para que fosse possível prever a performance e a largura de banda.

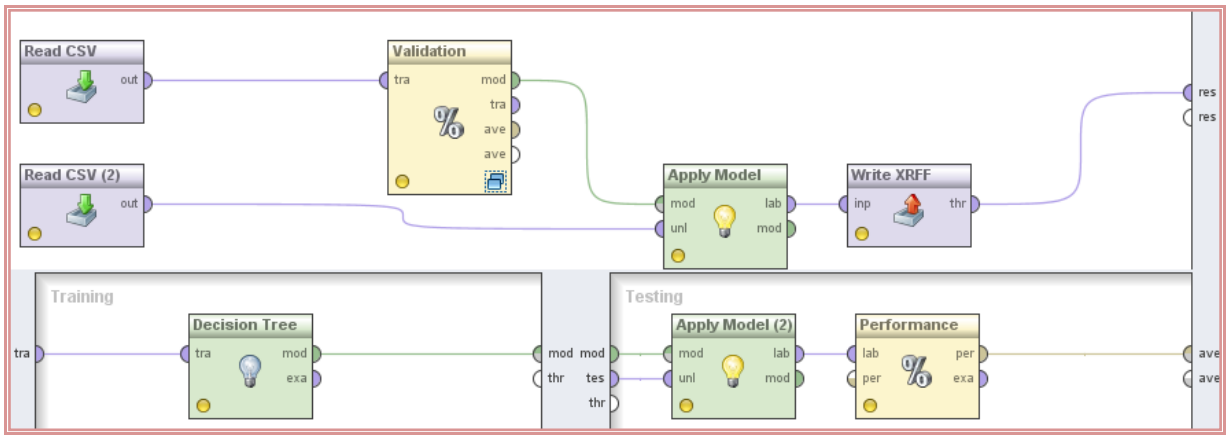


Figura 7 - Modelo de previsão desenvolvido

O algoritmo de árvore de decisão foi configurado utilizando os seguintes parâmetros definidos em [15]:

- **Critério**: ganho de informação.
- **Tamanho mínimo para dividir** (tamanho mínimo de um nodo de forma a permitir uma divisão): 4.
- **Tamanho mínimo** (tamanho mínimo de todas as folhas): 2.
- **Ganho mínimo** (o ganho mínimo a atingir de forma a produzir uma divisão): 0.0050.
- **Profundidade máxima** (profundidade máxima da árvore): 30.
- **Confiança** (o nível de confiança usado para calcular o valor de erro mais pessimista): 0.25.

Na previsão de performance e previsão de largura de banda pode ser utilizado o mesmo modelo, alterando apenas os dados de *input* provenientes do ficheiro CSV e os dados de *output* exportados para o ficheiro XRFF.

3.2.4 Escalonamento

O escalonador do ambiente *Grid*, também conhecido como *resource broker*, tem como principal objetivo mapear tarefas aos recursos, respeitando os seus requisitos e propriedades, encapsulando toda a complexidade existente neste tipo de ambientes. Contudo, o escalonador não possui total controlo sobre os recursos. Embora estes sejam partilhados no ambiente *Grid*, a sua principal função é servir os

[13] http://rapid-i.com

utilizadores da organização que os possuam, e, como tal, a sua utilização assume um papel dinâmico que muitas vezes é complicado de prever e de medir. Adicionalmente, outro fator que afeta a performance de um escalonador é a diversidade das características dos recursos pertencentes ao ambiente *Grid*. O escalonamento em *Grid* pode ser definido como o processo de tomada de decisão aglomerando recursos de vários domínios administrativos.

Taxonomia de escalonamento

Em [16] é proposta uma taxonomia para descrever o escalonamento em sistemas distribuídos. A taxonomia proposta está representada na Figura 8.

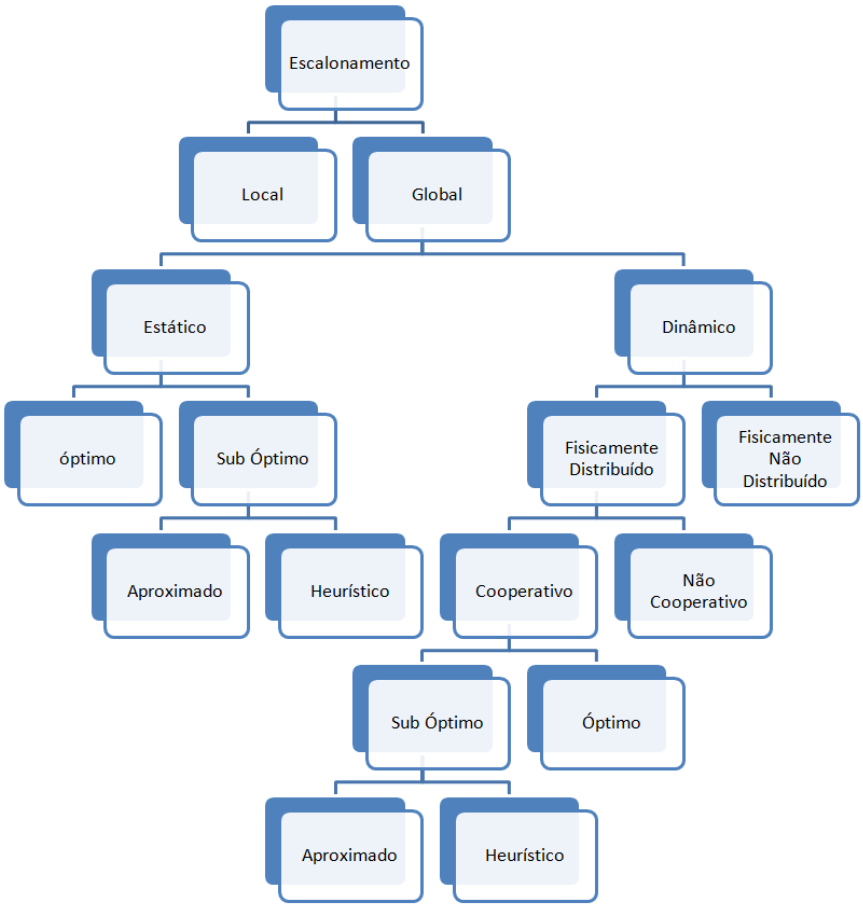


Figura 8 - Taxonomia dos escalonamentos em ambientes distribuídos

Cada tipo de escalonamento possui um conjunto de características, nomeadamente:

- **Escalonamento local,** que realiza a atribuição de intervalos de tempo de um processador, enquanto o **escalonamento global** atribui processadores para executarem as tarefas.
- **Escalonamento estático,** que realiza as decisões de escalonamento *à priori*, enquanto que no **escalonamento dinâmico** as informações dos recursos e das tarefas variam consoante o

tempo, isto é, as decisões de escalonamento são determinadas no momento antes da execução, de forma a que a informação recolhida (estado dos recursos) se encontre o mais atualizada possível, contribuindo para uma estimativa de custo com mais qualidade.

- **Escalonamento ótimo,** que é utilizado quando se possui bastante informação sobre o estado do ambiente Grid. Este tipo de escalonamento é também utilizado quando o custo para realizar a computação dos critérios de seleção não é elevado. No caso em que o custo de computação é inviável ou demasiado elevado, aplica-se o **escalonamento sub-ótimo** para resolver o problema.
- **Escalonamento aproximado,** que é utilizado para alcançar boas soluções. Para isso são realizadas otimizações sucessivas recorrendo a métricas previamente definidas. No **escalonamento heurístico** são procuradas soluções por aproximações sucessivas avaliando-se a qualidade das soluções até que um dado critério de paragem seja cumprido.
- **Escalonamento não distribuído,** em que a operação é executada num único processador, já no **escalonamento distribuído** uma operação pode ser dividida e distribuída fisicamente entre vários processadores.
- **Escalonamento cooperativo,** no qual as operações podem colaborar entre si e partilhar resultados, já no modo **não-cooperativo** não existe colaboração entre operações, uma vez que todas as decisões são efetuadas independentemente dos resultados de outras operações.

Em [16] são mencionadas outras características dos sistemas de escalonamento. A saber:

- **Escalonamento adaptativo** em oposição ao **escalonamento não adaptativo**. No escalonamento adaptativo os parâmetros do escalonador podem ser alterados conforme o estado dinâmico do ambiente *Grid* ou conforme as necessidades da tarefa a executar. Por exemplo, se for submetida uma tarefa ao ambiente *Grid* que necessite principalmente de memória RAM, o peso da memória RAM no cálculo de disponibilidade pode ser alterado e ajustado de forma a encontrar os recursos que melhor respeitem este critério.
- **Balanceamento de Carga**, no qual é realizado uma tentativa de manter a carga de todos os recursos ao mesmo nível, criando assim um equilíbrio na utilização global do ambiente *Grid*.

Fases do escalonamento

O escalonamento de tarefas pode ser definido em três fases [17]: 1) descoberta de recursos; 2) seleção do recurso; e 3) execução da operação. A Figura 9 ilustra as diferentes fases existentes no processo de escalonamento de tarefas.

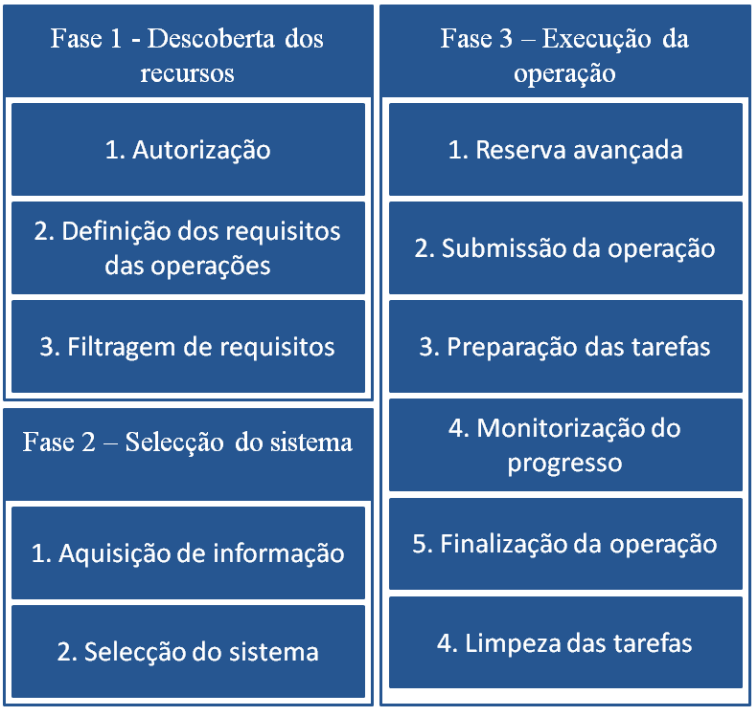


Figura 9 - Fases do processo de escalonamento de tarefas

A primeira fase é responsável por realizar a descoberta de recursos disponíveis para um determinado utilizador, estando esta dividida em três etapas:

- **Autorização**, na qual se verifica quais os recursos que podem ser disponibilizados ao utilizador para submissão de um *workflow*.
- **Definição de requisitos das operações**, em que se verifica quais os recursos que cumprem os requisitos necessários (velocidade de processamento, memória RAM disponível, carga de processamento atual, entre outros) para execução das operações.
- **Filtragem de requisitos**, que é a etapa em que se retorna os recursos que cumprem as etapas de autorização e definição de requisitos das operações.

A segunda fase seleciona um ou mais recursos para submissão do *workflow*, que por sua vez está dividida em duas etapas:

- **Aquisição da informação dos recursos**, na qual se consulta toda a informação disponível sobre os recursos.
- **Seleção do sistema**, em que se escolhe um ou mais recursos para submissão do *workflow*, esta escolha é realizada através de critérios aplicados à informação consultada.

A terceira e última fase é a Execução da Operação. Esta fase está organizada em seis etapas:

- **Reserva avançada**, um passo de carácter opcional que é responsável pela reserva antecipada de um recurso.
- **Submissão da operação**, em que é realizada a submissão da operação ao recurso selecionado nas etapas anteriores.
- **Preparação das tarefas**, na qual se integram todos os procedimentos necessários para executar a operação.
- **Monitorização do progresso**, no qual é efetuada a monitorização do progresso da execução da operação, de modo a que se possam detetar erros ou inconsistências, sendo assim possível resubmeter a operação a outro recurso.
- **Finalização da operação**, com as informações recolhidas na etapa anterior é possível alertar o utilizador, avisando-o do fim da tarefa.
- **Limpeza das tarefas**, passo em que é realizada a transferência dos resultados da operação para o utilizador e é efetuada uma limpeza de todos os ficheiros e variáveis temporárias criadas com a execução da operação.

Workflow

Neste trabalho o processo de ETL é modelado recorrendo a um *workflow*. O *workflow* representa a ordem de precedências das operações de ETL. A dependência entre tarefas pode ser resolvida recorrendo a um Gráfico Acíclico Direto (DAG), de forma a determinar a ordem de precedência das operações. O principal desafio, evidenciado em [18], na utilização de um *workflow* em ambientes *Grid*, é o seu escalonamento, ou seja, a forma como se submete as operações respeitando as suas precedências no *workflow*.

No âmbito do trabalho desenvolvido, cada nodo do *workflow* representa uma operação elementar, sendo que as suas dependências são determinadas pelas arestas que se conectam com outros nodos. Desta forma, é possível determinar facilmente as precedências de cada nodo, sendo deste modo possível determinar um conjunto de operações que podem ser executadas em simultâneo. Após modelação do processo ETL, recorrendo a um motor de *workflow*, é realizada a sua conversão para um ficheiro XML que contém todas as informações relativas ao *workflow*, dando-se relevo, principalmente, à definição do conjunto de atributos (executável, *query* e tabelas de *input* independentes). Todas as restantes definições (tabelas de *output*, e tabelas de *input* dependentes) são criadas dinamicamente recorrendo a informação existente sobre as precedências entre os nodos do *workflow*. Após a conversão para um ficheiro XML, este é submetido ao escalonador do ambiente Grid. A Figura 10 apresenta o esquema utilizado para efetuar todo o processo de submissão de um *workflow* representativo do processo de ETL em causa.

Figura 10 - Esquema de submissão do workflow no ambiente Grid selecionado

Para minimizar o impacto na transferência de ficheiros entre máquinas foi implementado um sistema de distribuição de ramos de operações recorrendo ao esquema de *workflow* apresentado na Figura 10. Este sistema baseia-se nos seguintes critérios:

- Se uma operação possuir apenas um antecedente, deve-se verificar se este apenas possui um nodo filho e se está disponível. Caso estas condições se verifiquem, a operação é submetida ao nodo.
- Se uma operação possuir mais que um antecedente, então é submetida ao nodo pai que possuir maior performance e disponibilidade.

Seguindo estes critérios, o motor de escalonamento consegue poupar largura de banda no ambiente *Grid*, uma vez que grande parte dos ficheiros de dependências já se encontram disponíveis na máquina onde a operação vai ser executada.

Escalonamento baseado em previsões

Após ter sido realizada a previsão de disponibilidade e de largura de banda para um conjunto de máquinas num determinado instante é gerado um ficheiro XRFF com os seguintes dados:

- Nome da máquina (se a previsão for de largura de banda contém o nome da máquina de envio e da máquina de receção).
- Informação relativa a data e hora da previsão.
- Confiança da previsão para cada classe.
- Previsão da classe (de performance ou de largura de banda, consoante a previsão efetuada).

A Figura 11 apresenta um exemplo de um ficheiro relativo a uma previsão de largura de banda.

```
<?xml version="1.0" encoding="windows-1252"?>
<dataset name="RapidMinerData" version="3.5.4">
  <header>
    <attributes>
      <attribute name="From" type="nominal">
        <labels>
          <label>debian1.ciicesi.intra.estgf.ipp.pt</label>
        </labels>
      </attribute>
      <attribute name="To" type="nominal">
        <labels>
          <label>debian2.ciicesi.intra.estgf.ipp.pt</label>
          <label>debian3.ciicesi.intra.estgf.ipp.pt</label>
          <label>debian4.ciicesi.intra.estgf.ipp.pt</label>
        </labels>
      </attribute>
      <attribute name="Year" type="numeric"/>
      <attribute name="Month" type="numeric"/>
      <attribute name="Day" type="numeric"/>
      <attribute name="DayWeek" type="numeric"/>
      <attribute name="Hours" type="numeric"/>
      <attribute name="Minutes" type="numeric"/>
      <attribute name="confidence(4)" type="numeric"/>
      <attribute name="confidence(5)" type="numeric"/>
      <attribute name="prediction(Class)" type="nominal">
        <labels>
          <label>4</label>
          <label>5</label>
        </labels>
      </attribute>
    </attributes>
  </header>
  <body>
```

```
<instances>
  <instance>
    <value>debian1.ciicesi.intra.estgf.ipp.pt</value>
    <value>debian2.ciicesi.intra.estgf.ipp.pt</value>
    <value>111.0</value>
    <value>9.0</value>
    <value>10.0</value>
    <value>1.0</value>
    <value>15.0</value>
    <value>39.0</value>
    <value>1.0</value>
    <value>0.0</value>
    <value>4</value>
  </instance>
  <instance>
    <value>debian1.ciicesi.intra.estgf.ipp.pt</value>
    <value>debian3.ciicesi.intra.estgf.ipp.pt</value>
    <value>111.0</value>
    <value>9.0</value>
    <value>10.0</value>
    <value>1.0</value>
    <value>15.0</value>
    <value>39.0</value>
    <value>1.0</value>
    <value>0.0</value>
    <value>4</value>
  </instance>

  (...)

    </instances>
  </body>
</dataset>
```

Figura 11 - Ficheiro exemplo de previsão de largura de banda

No processo de tomada de decisão sobre o escalonamento das operações elementares foi observado que existem algumas operações que necessitam de grande largura de banda, pois envolvem transformações que aumentam o tamanho das tabelas resultantes. Desta forma foi realizada uma lista com o peso de impacto da largura de banda para cada uma das operações elementares especificadas no processo de *workflow* (Figura 12).

```
<OperationBandwidthImportance>
        <Operation>
                <Name>StaxAddColumn.jar</Name>
                <BandwidthImportance>0.35</BandwidthImportance>
        </Operation>
        <Operation>
                <Name>StaxLeftJoin.jar</Name>
                <BandwidthImportance>0.5</BandwidthImportance>
        </Operation>
        <Operation>
                <Name>StaxProjection.jar</Name>
                <BandwidthImportance>0.2</BandwidthImportance>
        </Operation>
        <Operation>
                <Name>StaxRenameColumn.jar</Name>
                <BandwidthImportance>0.3</BandwidthImportance>
        </Operation>
        <Operation>
                <Name>StaxSelection.jar</Name>
                <BandwidthImportance>0.2</BandwidthImportance>
        </Operation>
        <Operation>
                <Name>StaxThetaJoin.jar</Name>
                <BandwidthImportance>0.4</BandwidthImportance>
        </Operation>
        <Operation>
                <Name>StaxUnion.jar</Name>
                <BandwidthImportance>0.55</BandwidthImportance>
        </Operation>
</OperationBandwidthImportance>
```

Figura 12 - Ficheiro de configuração dos pesos da importância da largura de banda para cada operação elementar

Após definida a localização dos ficheiros de previsão e do ficheiro de configuração dos pesos, o escalonador aplica a fórmula (5) para calcular a classe de performance global de um recurso, considerando que os ficheiros de previsão contêm uma classe que varia entre 0 e 5, definida quando conjugando a sua performance com a largura de banda disponível entre o recurso que pode executar a operação e o que aloja os ficheiros.

$$ClasseGlobal\ [i]=ClassePerformance[i] \times (1-ImportanciaLarguraBanda[x]) + ClasseLarguraBanda[j,i] \times ImportanciaLarguraBanda[x] \quad (\mathbf{5})$$

As variáveis que aparecem nesta fórmula têm o seguinte significado:

- **ClasseGlobal[i]**, que representa um valor entre 0 e 5 para a classe global da máquina ***i***, considerando a performance do recurso e a sua largura de banda.
- **ClassePerformance[i]**, que representa um valor entre 0 e 5 para a classe de performance da máquina ***i***. Este valor é obtido utilizando o modelo de previsão de classe de performance, baseado em árvores de decisão, que pode ser consultado pelo escalonador recorrendo ao ficheiro XRFF criado.
- **ImportânciaLarguraBanda[x]**, que representa um valor entre 0 e 1 da importância que a largura de banda tem sobre a operação ***x***. Este valor pode ser consultado pelo escalonador recorrendo ao ficheiro XML que contém a configuração dos pesos das influências da largura de banda para cada operação elementares.
- **ClasseLarguraBanda[j,i]**, que representa um valor entre 0 e 5 da classe de largura de banda da máquina ***j*** para a máquina ***i***. Este valor é previsto utilizando o modelo de previsão de largura de banda, baseado em árvores de decisão, que pode ser consultado pelo escalonador recorrendo ao ficheiro XRFF criado.

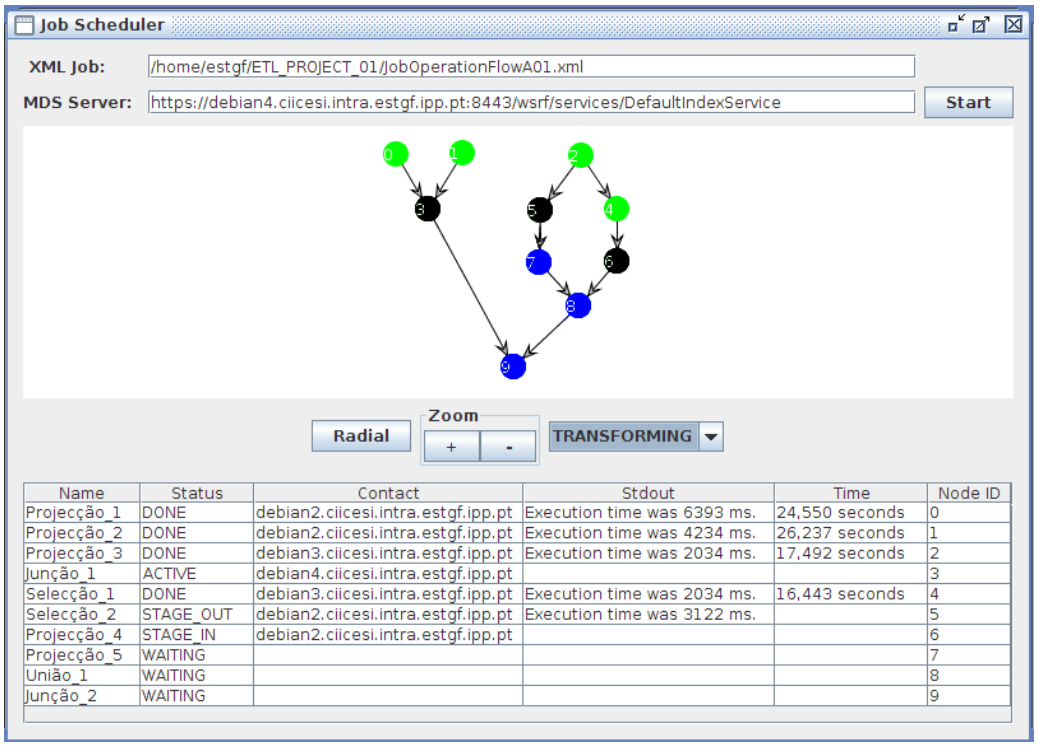
Adicionalmente, foi introduzido um fator de probabilidade de execução a cada uma destas classes, para que fosse possível introduzir diversidade no escalonamento de tarefas e maximizar a taxa de utilização global do ambiente *Grid* (6)(7).

$$ProbC_i < ProbC_{i+1} . \quad (\mathbf{6})$$

$$\sum_{i=0..5} ProbC_i = 1. \quad (\mathbf{7})$$

Após a conclusão da modelação lógica do processo ETL, o modelo obtido é submetida na *Grid* através da sua codificação em RSL. A RSL é uma linguagem que providencia uma sintaxe constituída por

pares <*atributo, valor*> sendo utilizada para descrever os recursos requeridos para uma tarefa e a sua definição. Através da sintaxe disponibilizada pela RSL é possível definir qual a operação ETL a realizar e quais os seus argumentos. Adicionalmente, podemos também incluir a localização dos ficheiros necessários à execução da tarefa especificada, bem como identificar o local onde será colocado o resultado (eventualmente um outro nodo da *Grid*). Após a definição do ficheiro RSL, a operação é submetida ao nodo que foi selecionado pelo escalonador. Ao longo do processo de ETL é monitorizado o estado de execução de cada uma das operações (Figura13), tornando-se possível lançar operações aquando da conclusão da execução das suas precedentes ou relançar operações quando for detetado um erro durante a execução. A representação gráfica do *workflow* foi realizada recorrendo à API Jung[14], que foi implementada de forma a adaptar-se aos requisitos exigidos pela representação do *workflow* de um processo ETL.



Name	Status	Contact	Stdout	Time	Node ID
Projecção_1	DONE	debian2.ciicesi.intra.estgf.ipp.pt	Execution time was 6393 ms.	24,550 seconds	0
Projecção_2	DONE	debian2.ciicesi.intra.estgf.ipp.pt	Execution time was 4234 ms.	26,237 seconds	1
Projecção_3	DONE	debian3.ciicesi.intra.estgf.ipp.pt	Execution time was 2034 ms.	17,492 seconds	2
Junção_1	ACTIVE	debian4.ciicesi.intra.estgf.ipp.pt			3
Selecção_1	DONE	debian3.ciicesi.intra.estgf.ipp.pt	Execution time was 2034 ms.	16,443 seconds	4
Selecção_2	STAGE_OUT	debian2.ciicesi.intra.estgf.ipp.pt	Execution time was 3122 ms.		5
Projecção_4	STAGE_IN	debian2.ciicesi.intra.estgf.ipp.pt			6
Projecção_5	WAITING				7
União_1	WAITING				8
Junção_2	WAITING				9

Figura 13 - Monitorização do estado de execução das operações ETL em curso

3.3 Configuração de um sistema ETL numa GRID

Para configurar um sistema de ETL num ambiente *Grid* foi definido um esquema XSD que descreve pormenorizadamente um *workflow* de operações. O principal objetivo para a criação deste esquema é simplificar o processo de definição e configuração na submissão de operações ETL ao ambiente *Grid*,

[14] http://jung.sourceforge.net/

uma vez que grande parte dos *middlewares Grid* atuais possuem uma linguagem de especificação de tarefas complexa e bastante morosa. Assim, é necessário definir atributos e características específicas da arquitetura do ambiente *Grid* no momento de definição e configuração da tarefa. A Figura 14 apresenta o esquema XSD criado.

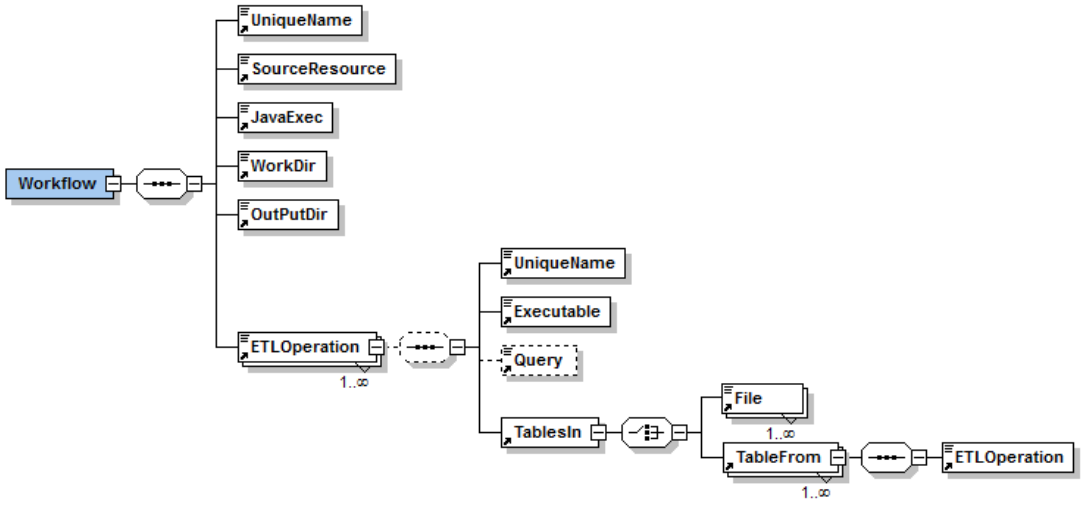


Figura 14 - Esquema XSD de definição de um workflow de operações ETL

O *Workflow* de operações ETL é caracterizado pelos seguintes atributos:

- ***UniqueName***: um identificador único utilizado para identificar um *workflow*.
- ***SourceResource***: a identificação do recurso que possui os ficheiros fonte necessários à execução do *workflow*.
- ***JavaExec***: o diretório onde se encontra o executável Java.
- ***WorkDir***: o diretório criado temporariamente para armazenar todos os ficheiros originados durante a execução do *workflow*.
- ***OutPutDir***: o diretório no qual irá ser armazenado o resultado final da execução do *workflow*.
- ***ETLOperation***: o conjunto de operações ETL que vão ser executadas neste *workflow*.

O atributo *ETLOperation* é composto pelos seguintes campos:

- ***UniqueName***: um identificador único utilizado para identificar uma operação ETL.
- ***Executable***: o nome do ficheiro executável Java que aplica uma determinada transformação ETL.
- ***Query***: o nome do ficheiro XML que possui os parâmetros da operação ETL. Este atributo é opcional, pois nem todas as operações necessitam de parâmetros, como por exemplo a operação de União.

- ***TablesIn***: o conjunto de ficheiros XML, representativos de tabelas, que vão servir de ficheiros de entrada para esta operação. Os ficheiros de entrada podem ser um *File* ou um *TableFrom*. Um *File* representa um ficheiro XML fonte que existe previamente antes da execução do *workflow*. Um *TableFrom* é composto por um conjunto de *ETLOperation*, sendo que cada *ETLOperation* possui um nome único de identificação. Desta forma é possível representar a dependência de uma tabela resultante de outra operação ETL executada anteriormente.

Quando o *workflow* é submetido ao escalonador do ambiente *Grid*, é realizada a sua transformação e tradução para a linguagem de especificação de tarefas (RSL) do *Globus Toolkit*. Uma vez que o escalonador do ambiente *Grid* conhece pormenorizadamente a arquitetura do ambiente *Grid*, é possível adicionar mais informação, em tempo de execução, sobre a definição de cada tarefa ETL. Quando o escalonador terminar a formulação do RSL o trabalho é submetido à *Grid*.

4 Caso de Estudo

4.1 Caracterização do ambiente GRID

Para criar o protótipo do ambiente *Grid* foram configuradas quatro máquinas com características heterogéneas, de forma a poder-se simular um ambiente *Grid* real. Na Tabela 3 são apresentadas as características de cada máquina presente no ambiente *Grid* configurado. Todas as máquinas estão ligadas a um *router* com uma velocidade de 100 Mbit/s.

Tabela 3 - Características das máquinas de suporte à Grid

Máquina	Processador	Memória RAM	Disco	Sistema Operativo
Debian1	Pentium 4 2.4 GHz	1.2 GB	70.3 GB	Ubuntu 10.10
Debian2	Pentium 4 1.5 GHz	376.7 MB	35.6 GB	Debian 5
Debian3	Pentium 4 2.4 GHz	494.9 MB	36.2 GB	Ubuntu 10.10
Debian4	Pentium 4 2.8 GHz	1.5 GB	35.1 GB	Debian 5

4.1.1 Middleware Grid

A escolha de um *middleware Grid* constitui um dos passos mais importantes na concepção e implementação de um ambiente *Grid*. A análise dos diferentes *middlewares Grid* exige um grande volume de tempo e é, normalmente, uma etapa crítica no futuro sucesso deste tipo de ambiente. Para escolher adequadamente um *middleware Grid* foram especificados um conjunto de requisitos bastante diversos. A saber:

1. Um sistema de informação que seja capaz de agregar a informação proveniente de diferentes nodos, permitindo assim organizar a *Grid* de forma hierárquica.
2. Um sistema de informação que possua a capacidade de disponibilizar informação sobre o *hardware* presente na *Grid* e sobre o estado dos serviços que são disponibilizados pela *Grid*.

3. Um conjunto de ferramentas de desenvolvimento que possuam um conjunto de abstrações sobre a *Grid*, facilitando assim o desenvolvimento de aplicações para a *Grid*.
4. O tipo de suporte e a documentação disponível.

Comparativamente aos *middleware Grid* analisados, nomeadamente o *Globus Toolkit*, *Virtual Data Toolkit*, gLite e UNICORE, o *Globus Toolkit* é o que responde melhor aos requisitos definidos, destacando-se principalmente no ponto 1) e 3). A instalação do *Globus Toolkit* foi realizada em todas as máquinas, sendo instalado o gestor de trabalhos *fork* ao invés de um sistema de gestão de *cluster*, dado que o número reduzido de máquinas presente no ambiente *Grid* criado não justifica a utilização de um sistema de gestão de *clusters*.

Contudo, para utilizar o *Globus Toolkit*, é necessária a configuração do componente de segurança, que será o responsável pela proteção de toda a comunicação entre os diversos recursos da Grid e pela gestão de identidades de utilizadores e serviços. Para o estabelecimento e gestão de identidades, recorre a certificados X.509, gerados através do módulo *SimpleCA* presente no *toolkit*, sendo que o DN (*Distinguished Name*) neles presente tem como principal objetivo a validação da identidade de uma autenticação efetuada no ambiente Grid. Para configurar o DN de cada certificado, utiliza-se, para o caso dos utilizadores, o *fullname* correspondente e, para o caso dos recursos/serviços, o respetivo *hostname*.

Para armazenar os certificados foi configurado um servidor *myProxy* na máquina *Debian4*, já que esta era a máquina com melhores recursos computacionais, podendo assim alojar serviços extra sem comprometer em demasia o seu desempenho computacional.

Para realizar a transferência de ficheiros entre máquinas foi configurado o *Grid FTP*, que é uma modificação do protocolo FTP que se encontra presente no pacote de instalação do *Globus Toolkit*. De modo a submeter e gerir um conjunto de transferências de ficheiros ou diretórios, utilizando o *Grid FPT*, foi configurado o serviço RFT.

Para finalizar, foi configurado o gestor de trabalhos GRAM4 que é responsável por realizar a gestão dos recursos existentes no ambiente *Grid*, tornando-se assim possível a submissão de trabalhos entre os diversos recursos existentes. Os restantes serviços disponibilizados pelo *Globus* não necessitam de configurações extras, uma vez que se encontram automaticamente ativados e configurados por defeito, na finalização da instalação. A Figura 15 apresenta a arquitetura do ambiente *Grid* implementado.

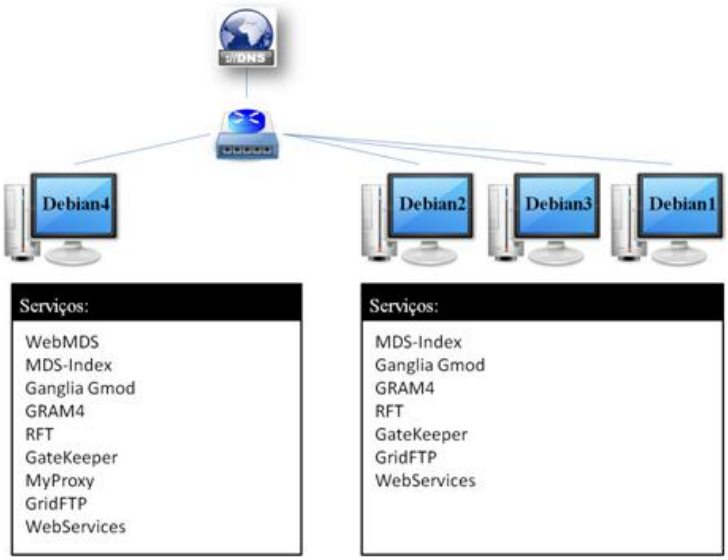


Figura 15 - Ilustração da arquitectura do ambiente Grid implementado

4.1.2 Serviço de Informação

O serviço de informação que melhor se adapta ao ambiente *Grid* com *Globus Toolkit* é o MDS4, pois é o único que recolhe informação de serviços específicos da *Grid*, tais como o PBS, GRAM, RFT, RLS, entre outros. Com o MDS4 é ainda possível organizar a informação disponibilizada por organizações virtuais, criando-se assim um serviço de monitorização de grande escalabilidade. No entanto, o MDS4 não é capaz de monitorizar as características de *hardware* de um recurso, embora possa ser configurado para recolher informação disponibilizada por outros serviços de monitorização, tais como o Ganglia e o Nagios.

Comparativamente, o Ganglia tem algumas vantagens sobre o Nagios, principalmente porque está dividido em três partes distintas, que podem operar individualmente. Para operar em conjunto com o MDS4 só é necessário executar o Gmond, que é o serviço responsável pela recolha da informação sobre os recursos. Todos os restantes serviços podem ser desativados para não ocupar recursos na máquina. Outra vantagem do Ganglia face ao Nagios, reside no facto de este ser um sistema de monitorização especializado em *Grids* e *clusters* de alto desempenho, o que possibilita um modo de consulta com grande detalhe das características de *hardware* de um recurso, informação esta indispensável num ambiente *Grid*.

No ambiente de testes, o Ganglia foi instalado e configurado em todas máquinas, para operar em conjunto com o MDS4. A máquina Debian4 foi a escolhida para hospedar o serviço de agregação de informação do MDS4, que permite recolher informação disponibilizada pelas restantes máquinas. Adicionalmente, foi configurado o serviço WebMDS que permite a publicação de toda a informação recolhida através da *Web* e que facilita a sua consulta. Tendo em conta as características do ambiente

de testes no que diz respeito à quantidade de máquinas e a sua localização geográfica, optou-se por não dividir a *Grid* em diferentes VOs. A Figura 16 apresenta a arquitetura do serviço de informação implementado.

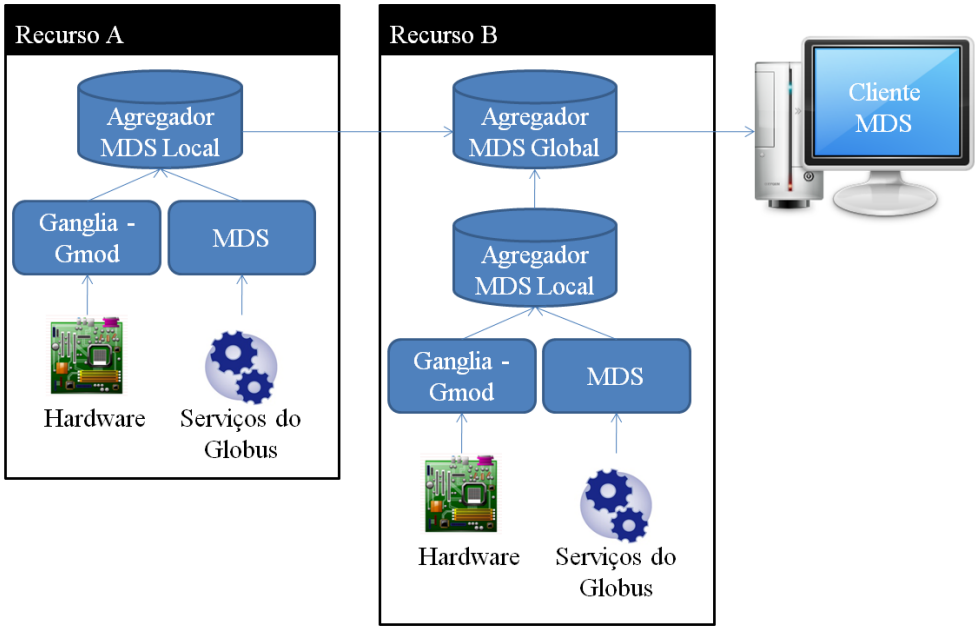


Figura 16 - Ilustração da arquitetura do serviço de informação implementado

4.1.3 Framework de desenvolvimento

Para a criação de aplicativos para a *Grid* a linguagem de programação escolhida foi a linguagem Java devido, essencialmente, à sua portabilidade. A linguagem Java atualmente executa na maior parte dos sistemas operativos e *hardware* existentes, sendo assim uma linguagem com grande potencial de aplicação num ambiente *Grid* com características heterogéneas. Foi necessário ainda escolher uma plataforma de desenvolvimento capaz de criar uma camada de abstração sobre o ambiente *Grid*, tendo sido selecionada a plataforma Java CoG Kit e JGlobus [19]. Estas ferramentas fornecem aos programadores de aplicações para ambiente *Grid* uma *framework* de alto nível, que permite o desenvolvimento rápido e fácil de aplicações, providenciando a implementação em Java dos seguintes serviços: GSI, *Grid FTP*, *myProxy* e cliente GRAM. Com estas duas *frameworks* é possível controlar e programar todos os serviços do ambiente *Grid*, sendo assim possível criar um escalonador totalmente funcional e com total controlo sobre o ambiente.

4.2 Preparação e configuração do processo ETL para ser executado na Grid

O *workflow* apresentado na Figura 17 representa um processo de ETL que é constituído por um conjunto de operações de álgebra relacional. É possível observar através deste *workflow* que existe um conjunto de operações que dependem do resultado de outras, sendo possível fazer a execução em paralelo de diferentes ramos de operações.

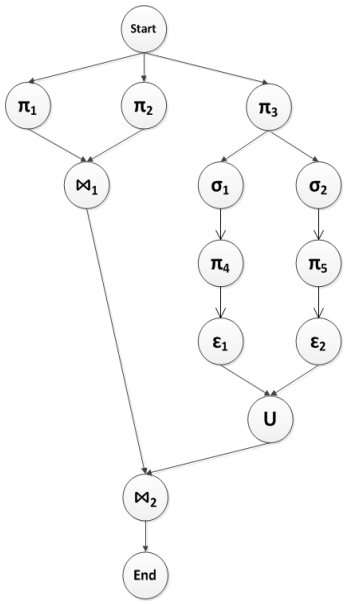


Figura 17 - Exemplo de um workflow ETL

Para submeter o *workflow* ao ambiente *Grid* deve-se ter em conta um conjunto de atributos necessários para executar cada uma das operações de ETL. De forma a simplificar a demonstração da configuração do processo de ETL apenas vão ser consideras três operações: *Projection_1*, *Projection_2* e *Join_1,* cujos atributos se encontram apresentados e descritos na Tabela 4. Cada operação possui um nome único que permite identificar a operação ETL, o executável que representa o programa que vai aplicar transformações ao documento de entrada, o ficheiro de definição que indica ao executável um conjunto de atributos sobre as transformações a efetuar e uma ou mais tabelas de entrada que podem ter origem num ficheiro XML ou no resultado de uma outra operação ETL.

Tabela 4 - Descrição dos atributos das operações de ETL

Nome único	Executável	Ficheiro de definição	Tabelas de entrada
Projection_1	Jars/StaxProjection.jar	DefinitionFiles/CSTUDY/projection1.xml	Ficheiro Files/PersonAddress.xml
Projection_2	Jars/StaxProjection.jar	DefinitionFiles/CSTUDY/projection2.xml	Ficheiro Files/HumanResourcesEmployeeAddress.xml
Join_1	Jars/StaxThetaJoin.jar	DefinitionFiles/CSTUDY/InputFileJoin.xml	Tabela resultante Projection_1 Tabela resultante Projection_2

Através da análise da Tabela 4 é possível criar o ficheiro de configuração do processo de ETL ilustrado na Figura 18. De notar que o cabeçalho do ficheiro de configuração do *workflow* não necessita de ser definido pelo utilizador, sendo os seguintes campos opcionais: *SourceResource*, *JavaExec*, *WorkDir* e *OutPutDir*. Este cabeçalho possui informação sobre critérios da arquitetura do ambiente *Grid*, que não necessitam ser preenchidos pelo utilizador ficando apenas disponíveis para depuração do sistema. Se nenhum dos campos anteriores for definido, o escalonador opta por preencher estes campos com valores padrão. Este exemplo de um processo de ETL segue o modelo XSD apresentado no capítulo 3.3.

```
<Workflow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="file:///E:/GridFiles/ETL_PROJECT_01/esquema2.xsd">
  <UniqueName>Job_1</UniqueName>
  <SourceResource>debian1.ciicesi.intra.estgf.ipp.pt</SourceResource>
  <JavaExec>/usr/lib/jdk1.6.0_23/bin/java</JavaExec>
  <WorkDir>/home/estgf/ETL_PROJECT_01/</WorkDir>
  <OutPutDir>OUTPUT/</OutPutDir>
  <ETLOperation>
    <UniqueName>Projection_1</UniqueName>
    <Executable>Jars/StaxProjection.jar</Executable>
    <Query>DefinitionFiles/CSTUDY/projection1.xml</Query>
    <TablesIn>
      <File>Files/PersonAddress.xml</File>
    </TablesIn>
  </ETLOperation>
  <ETLOperation>
    <UniqueName>Projection_2</UniqueName>
    <Executable>Jars/StaxProjection.jar</Executable>
    <Query>DefinitionFiles/CSTUDY/projection2.xml</Query>
    <TablesIn>
      <File>Files/HumanResourcesEmployeeAddress.xml</File>
    </TablesIn>
  </ETLOperation>
  <ETLOperation>
    <UniqueName>Join_1</UniqueName>
    <Executable>Jars/StaxThetaJoin.jar</Executable>
    <Query>DefinitionFiles/CSTUDY/InputFileJoin.xml</Query>
    <TablesIn>
      <TableFrom>
        <ETLOperation>Projection_1</ETLOperation>
      </TableFrom>
      <TableFrom>
        <ETLOperation>Projection_2</ETLOperation>
      </TableFrom>
    </TablesIn>
  </ETLOperation>
</Workflow>
```

Figura 18 - Exemplo do ficheiro de configuração do workflow ETL

4.3 Resultados

Após a submissão do processo ETL (Figura 17) foram obtidos alguns resultados interessantes, sobre os quais se falará aquando da descrição de cada um dos cenários de teste executados. O escalonador do ambiente *Grid* foi executado na máquina *Debian1*, uma vez que esta era a máquina detentora de todos os ficheiros fonte necessários para a execução do *workflow*. Sendo assim, a máquina *Debian1* foi configurada para não executar trabalhos, de forma a não afetar a performance geral do ambiente *Grid*. O tamanho dos ficheiros fonte estão apresentados na Tabela 5.

Tabela 5 - Tamanho dos ficheiros fonte usados na execução do processo de ETL

Ficheiro	Operação	Tamanho
PersonAddress.xml	Projection_1	6,378 MB
HumanResourcesEmployAddress.xml	Projection_2	0,070 MB
HumanResourcesEmployee.xml	Projection_3	0,203 MB

O algoritmo de escalonamento, apresentado em todos os cenários de teste efetuados, apresenta um comportamento de escolha e seleção de nodos baseado em probabilidades, de forma a não sobrecarregar os melhores recursos do ambiente *Grid*, mantendo assim um balanceamento de carga entre todos os nodos. Em cada cenário de testes foi executado um conjunto total de cinco testes de forma a obter resultados mais rigorosos e exatos. Embora o ambiente *Grid* esteja atualmente dedicado somente à execução de operações provenientes do ambiente *Grid*, existem ainda alguns processos e serviços que podem interferir com a capacidade de processamento de uma máquina, podendo-se verificar algumas diferenças nos tempos de execução e de envio de ficheiros.

Todas as tarefas do *workflow* podem ser observadas através da interface gráfica do escalonador, permitindo observar a evolução do estado da *Grid* à medida que o escalonamento das tarefas é efetuado (Figura 19).

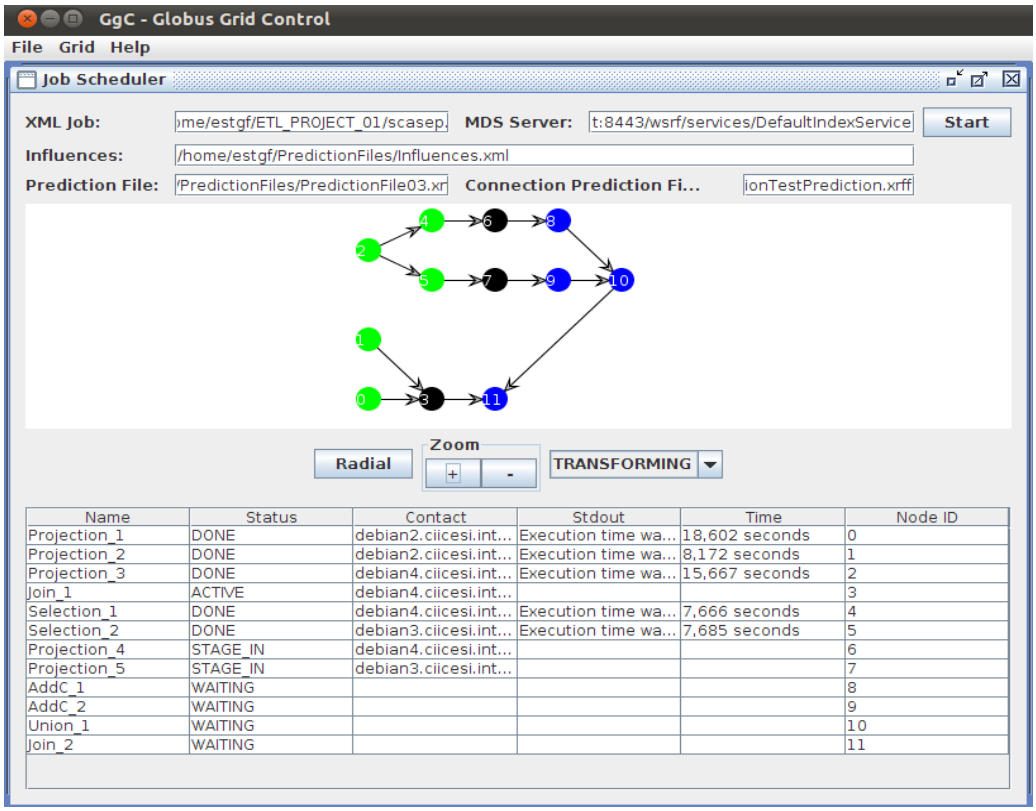


Figura 19 - Ambiente gráfico do escalonador desenvolvido

Agora passemos à descrição de cada um dos cenários de teste levados a cabo. No primeiro cenário de testes, o escalonador foi configurado para utilizar como critério de escolha e seleção a disponibilidade computacional real de um nodo, ou seja, o estado de um nodo é consultado em tempo real recorrendo ao serviço de monitorização sem a realização da previsão de performance. A fórmula utilizada neste cenário para medir a disponibilidade computacional foi apresentada anteriormente no secção 3.2.1. Após calculada a disponibilidade computacional de um nodo, é efetuada a divisão dos nodos em seis classes (C0-C5), baseadas na performance de cada um, recorrendo-se à fórmula já apresentada no secção 3.2.3. Foi ainda definido que cada classe possui uma importância 20% superior relativamente à sua classe antecedente (Tabela 6).

Tabela 6 - Probabilidades atribuídas às classes

Classe	Probabilidade
C0	10%
C1	12%
C2	15%
C3	17%
C4	21%
C5	25%

Os resultados obtidos no primeiro cenário de testes estão apresentados na Figura 20.

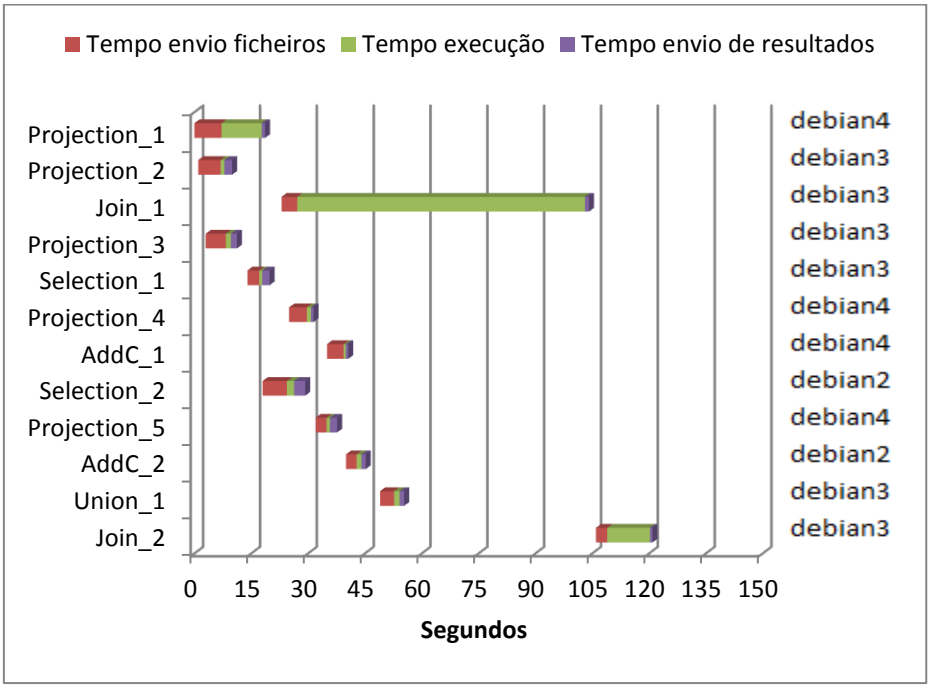


Figura 20 - Resultados obtidos no primeiro cenário de testes

O segundo cenário de testes envolveu um modelo de distribuição de tarefas baseado na previsão de performance, apresentado anteriormente na secção 3.2.3. Todos os nodos, do ambiente *Grid*, foram avaliados considerando a sua disponibilidade computacional ao longo do tempo, sendo essa informação armazenada num ficheiro estatístico e utilizada para prever a classe de cada nodo num determinado período de tempo. Para realizar a previsão de performance foi utilizado o algoritmo de

árvores de decisão disponibilizado pela ferramenta *RapidMiner*. Neste cenário, a previsão de performance apresenta-se como o fator de tomada de decisão mais importante para o escalonador. Os resultados obtidos no segundo cenário de testes podem ser observados na Figura 21.

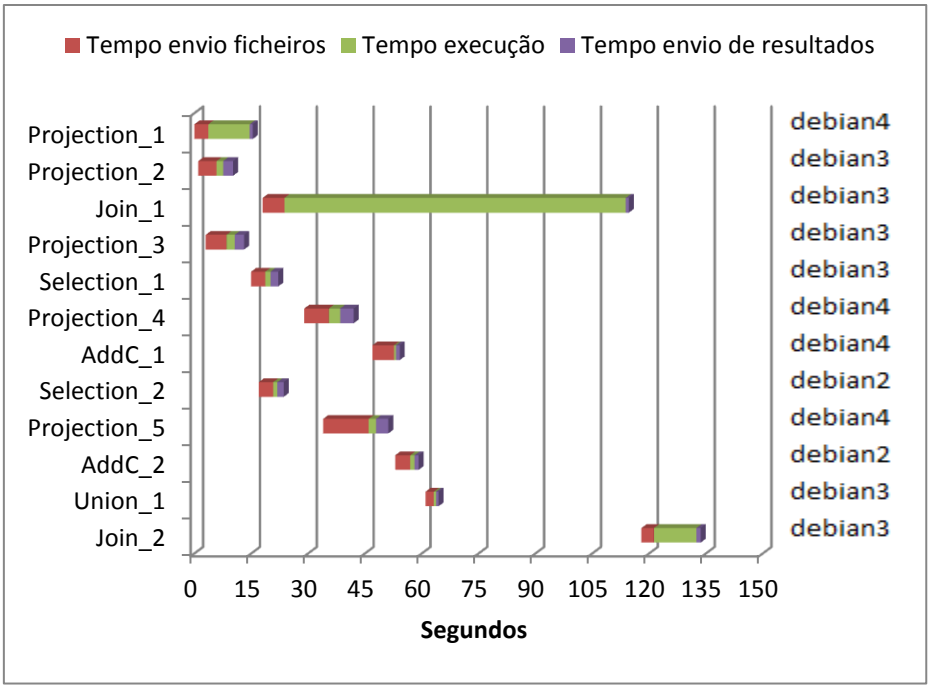


Figura 21 - Resultados obtidos no segundo cenário de testes

Com o objetivo de reduzir o tempo de *stage-out* (envio da tabela de resultados) e *stage-in* (receção do trabalho a executar e respetivas tabelas) de uma tarefa e o custo de comunicação entre os nodos, criou-se um terceiro cenário de testes. O modelo de escalonamento utilizado neste cenário baseia-se na distribuição de ramos de operações (secção 3.2.4). Com esta simulação pretende-se reduzir o custo de comunicação com o nodo que realiza o escalonamento de tarefas, bem como os tempos de *stage-in* e *stage-out*. Para atingir este objetivo, o comportamento do escalonador foi alterado de forma a não realizar o *stage-out* das operações para o nodo de origem. Deste modo, as tarefas seguintes podem extrair os ficheiros fonte necessários diretamente do nodo antecedente. Quando uma tarefa é atribuída a um nodo, o escalonador verifica se a tarefa antecedente apenas possui um nodo filho. Se essa condição se verificar, a tarefa é atribuída ao mesmo nodo que a tarefa pai, de forma a diminuir o tempo de envio de ficheiros fontes, visto que os ficheiros já estão presentes no nodo. Se o nodo pai tiver mais que um nodo filho, o comportamento do escalonador é igual às simulações descritas anteriormente. Neste cenário os ficheiros fonte não ficam localizados no nodo que possui o escalonador, mas sim no nodo onde a tarefa pai foi executada, constituindo a principal diferença relativamente a outros testes. Assim, a transferência de ficheiros fonte só ocorre entre esses dois nodos. Os resultados obtidos no terceiro cenário de testes estão apresentados na Figura 22.

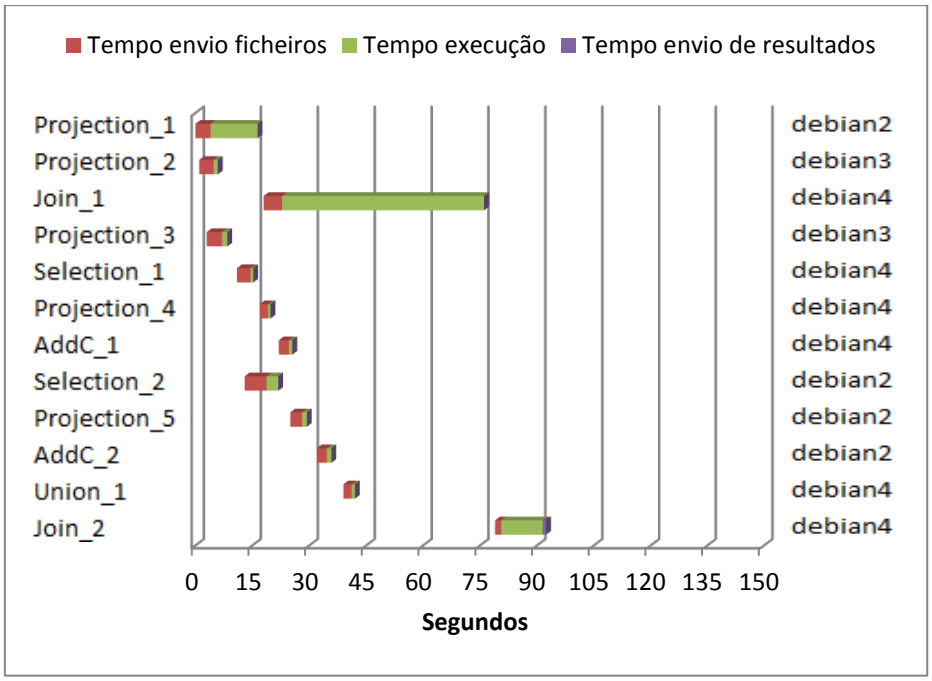


Figura 22 - Resultados obtidos no terceiro cenário de testes

Os próximos cenários de teste foram baseados na premissa de que a arquitetura de um ambiente *Grid* possui vários nodos com características heterogéneas, em termos de disponibilidade computacional e largura de banda. Como tal, o ambiente *Grid* foi alterado de forma a limitar a largura de banda do nodo Debian3 para 20Mbit/s em vez dos originais 100Mbit/s. Com esta alteração pretendeu-se avaliar o comportamento dos algoritmos de escalonamento num cenário em que a largura de banda se apresenta como um fator heterogéneo.

O quarto cenário de testes apresenta as mesmas diretrizes enunciadas no segundo cenário, isto é, apenas suportámos o teste com base na previsão de performance. Os resultados obtidos no quarto cenário de testes podem-se observar na Figura 23.

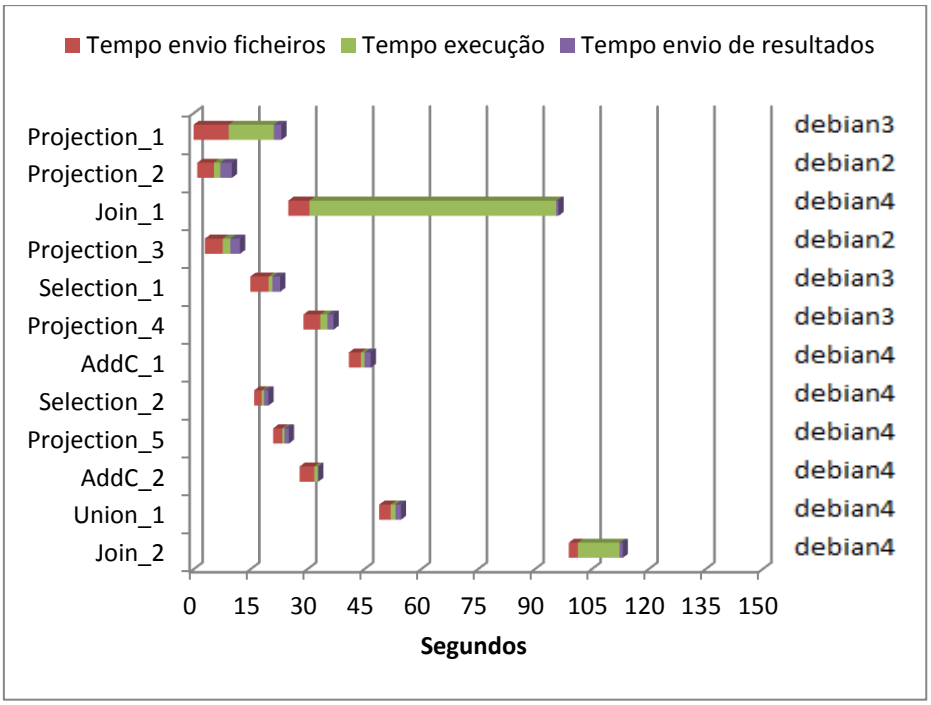


Figura 23 - Resultados obtidos no quarto cenário de testes

Já no quinto cenário de testes, o escalonador foi configurado para utilizar otimização na distribuição de ramos de operações, tal como tinha sido efetuado anteriormente no terceiro cenário. Os resultados obtidos estão apresentados na Figura 24.

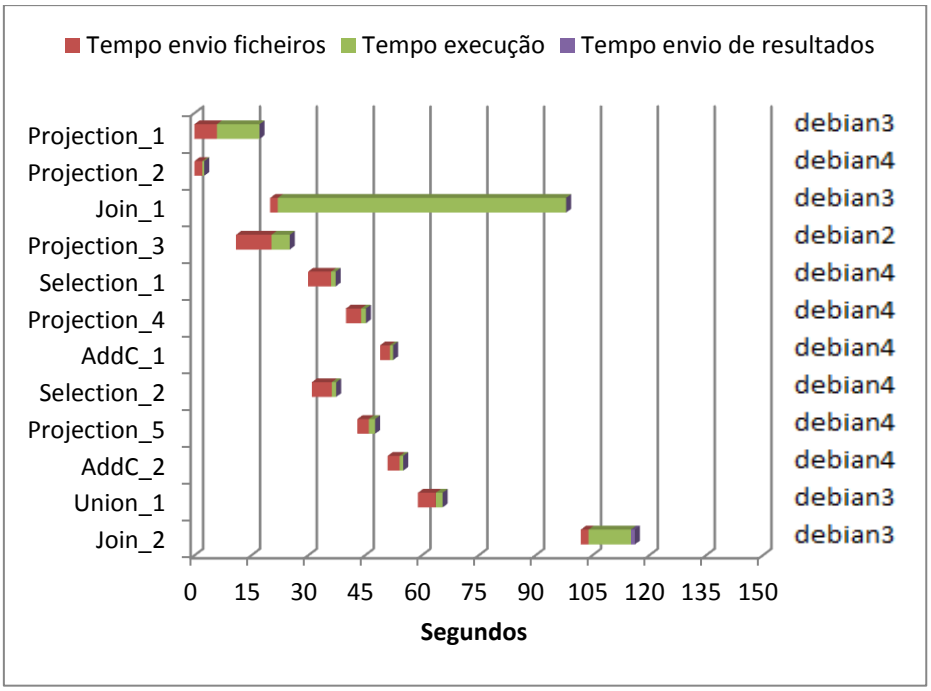


Figura 24 - Resultados obtidos no quinto cenário de testes

Para finalizar os testes efetuados foi criado um sexto cenário de testes, que tem como base a análise da largura de banda dos nodos no processo de tomada de decisão efetuado pelo escalonador (secções 3.2.2, 3.2.3 e 3.2.4). Neste cenário, o escalonador considera não só a disponibilidade computacional, como também a largura de banda de um nodo. A informação sobre a largura de banda é armazenada e atualizada periodicamente num ficheiro estatístico que é utilizado posteriormente pela ferramenta *RapidMiner* para prever a largura de banda de um nodo, num determinado período de tempo. De seguida é calculada a média entre as classes de performance e largura de banda de cada nodo, para se poder obter uma classe geral (secção 3.2.4). Os resultados obtidos no sexto cenário de testes estão apresentados na Figura 25.

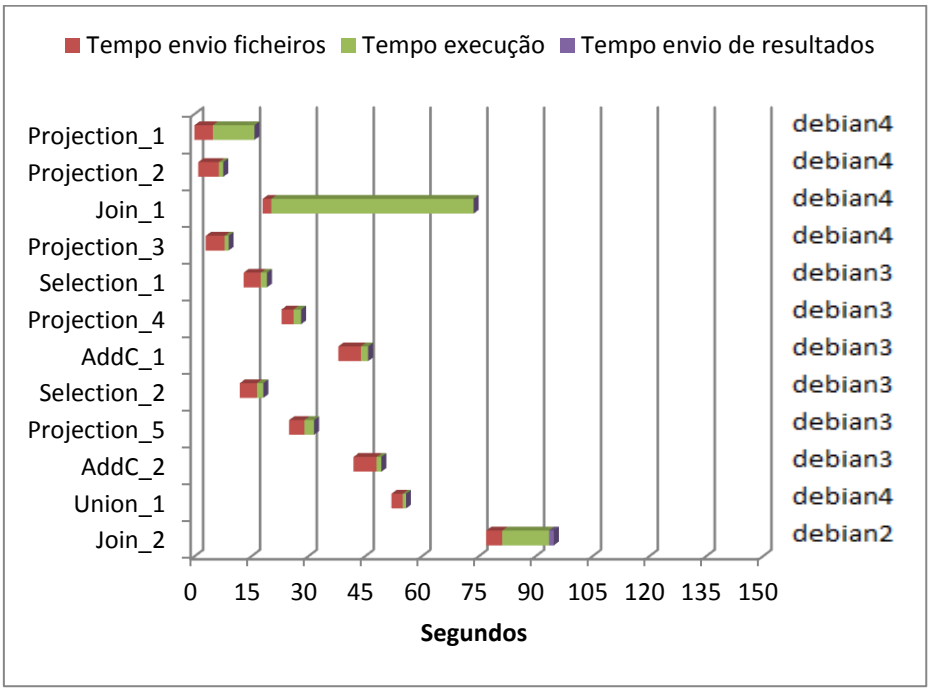


Figura 25 - Resultados obtidos no sexto cenário de testes

5 Conclusões e Trabalho Futuro

5.1 Discussão de resultados

Como foi visto anteriormente, foram obtidos alguns resultados recorrendo a diferentes tipos de cenários de teste, nos quais se alterou as características do ambiente Grid simulado e do escalonador aplicado. Tendo em consideração o *workflow* de execução aplicado nos testes realizados anteriormente, foi possível observar a existência de três ramos de operações que terminam com o início de uma operação de junção. Esta operação só pode ter início quando todas as operações antecedentes terminarem, uma vez que depende de resultados provenientes de operações anteriores. A operação de junção requer um grande poder de processamento, o que faz com que o tempo total de execução do *workflow* seja longo.

O segundo cenário de testes, quando comparado com o primeiro cenário desenvolvido, apresentou um aumento no tempo total de execução do *workflow*. Este aumento pode ser justificado pelo facto de o ambiente *Grid* implementado utilizar poucos nodos para executar tarefas. Como ambos os modelos apresentados em cada cenário utilizam um fator aleatório (utilizado para selecionar o nodo), e considerando que a diferença de importância de cada classe tem um valor reduzido, o resultado final pode nem sempre ser o esperado. De facto, uma tarefa complexa pode ser atribuída a um nodo com menor performance (como foi o caso da operação de Junção) aumentando desta forma o tempo total de execução. Além disso, como é utilizado um algoritmo de previsão para determinar a disponibilidade computacional de um nodo num determinado período de tempo, a performance real do nodo pode ser diferente. Desta forma, foi possível constatar que a previsão de disponibilidade computacional tem um fator de erro atribuído. Contudo, o escalonador apresentou melhor performance quando utilizou uma previsão de disponibilidade, uma vez que, assim, não necessitou de consultar todas as máquinas de cada vez que seja necessário executar uma tarefa.

Comparativamente aos cenários de testes anteriores, o terceiro cenário apresentou tempos de *stage-in* e *stage-out* inferiores, o que era esperado devido à utilização do algoritmo de otimização de distribuição de ramos de operações.

Face ao quarto cenário de testes foi possível observar que a primeira tarefa foi atribuída ao nodo mais limitado (Debian3). Neste teste, a fase de *stage-in* demorou, em média, duas vezes mais que quando se utilizou a largura de banda original de 100Mbit/s. Este resultado é justificado principalmente pelo facto de nesta fase ser efetuada a autenticação entre os nodos, independentemente da largura de banda disponível, o que, como sabemos, consome algum tempo. Também outras operações foram atribuídas ao nodo limitado. No entanto, o tamanho dos ficheiros transferidos não foi relevante para afetar o resultado final.

Com a execução do *workflow* no quinto cenário de testes foi possível constatar que a distribuição de operações, recorrendo ao modelo de otimização de distribuição de ramos de operações, num ambiente de largura de banda limitada, foi um pouco problemática. No entanto, como um ramo de operações foi atribuído a um único nodo, os dados apenas foram transferidos no início e no fim da execução do ramo de operações, minimizando assim o efeito da limitação da largura de banda no resultado final. No entanto, para cada tarefa, o escalonador manteve a transferência da tarefa (e alguma informação adicional) para o nodo que ia vai executar. De relevar que, se o nodo limitado fosse o escolhido para executar uma tarefa, que utilizasse ou produzisse um grande volume de dados, o impacto da limitação da largura de banda seria exponencial. Quando comparado com o terceiro cenário de testes, foi possível observar que os resultados pioraram, uma vez que o tempo de execução total foi aproximadamente 20% mais longo.

O último cenário de testes apresentou como resultado uma melhoria no processo, se considerarmos o tempo total de execução do *workflow*. Isto porque embora o nodo limitado tenha apresentado uma boa performance computacional no presente cenário, verificou-se uma redução na sua classe global levando, por sua vez, a uma diminuição da sua probabilidade de executar tarefas, pois a disponibilidade da largura de banda é um fator de grande impacto na determinação da classe global de um nodo. No entanto, dado a pequena dimensão do ambiente de testes, o nodo limitado acabou por ser escolhido para executar um ramo de operações. Melhorar o balanceamento entre a largura de banda e a disponibilidade computacional, atribuindo pesos mais equilibrados a cada classe, faria com que os resultados finais melhorassem. Eventualmente, cada tarefa do *workflow* poderia ter um rácio atribuído de forma a balancear a largura de banda e a disponibilidade computacional, tendo em consideração o seu volume de *inputs*, *outputs* e poder computacional exigido.

5.2 Considerações finais

A escolha e implementação do *middleware Grid* exigiu um grande esforço no estudo e análise de requisitos para atingir níveis de operacionalidade desejáveis, uma vez que a sua incorreta definição poderia originar a troca de *middleware Grid* num estado avançado de desenvolvimento, o que seria, com certeza, um contratempo decisivo no desenvolvimento deste trabalho.

Com a realização deste trabalho foi observado que muitos dos principais desafios na implementação de um ambiente *Grid* continuam em aberto e que uma grande parte dos aspetos teóricos deste tipo de ambientes ainda não são totalmente aplicados na prática. No presente trabalho, foi construído um sistema de recolha de informação de recursos, recorrendo ao serviço de informação MDS. Com os dados recolhidos foi possível aplicar uma fórmula para calcular a disponibilidade dos recursos de uma *Grid*. Além disso, foi ainda aplicado um algoritmo de previsão que permite antecipar valores de disponibilidade num determinado período de tempo, melhorando deste modo a performance do processo de escalonamento de tarefas. No contexto do processo de ETL, que lida com grandes volumes de dados, era indispensável uma análise de desempenho da largura de banda existente entre os diferentes recursos do ambiente *Grid*, tendo assim sido criado um procedimento que permite recolher informação acerca da largura de banda disponível entre os diversos recursos do ambiente *Grid*. Após a recolha de informação da largura de banda, foi aplicado um modelo de previsão de largura de banda entre nodos computacionais, para aferir a classe que esses mesmos nodos possuem num determinado período de tempo, o que pode ser um fator crucial para o melhor aproveitamento dos recursos presentes na *Grid*.

Através da decomposição de operações de transformação de dados realizados no processo de ETL em operações elementares, foi possível otimizar o escalonamento das operações, considerando os respetivos pesos, tentando desta forma classificar globalmente o desempenho do recurso especificamente para cada uma das diferentes operações. Para melhorar a qualidade de escalonamento foram definidos pesos de importância entre as duas principais métricas para comparação entre recursos (disponibilidade e largura de banda).

Tendo em consideração que a transferência de ficheiros entre nodos tem um grande impacto sobre o tempo total de processamento do processo de ETL, foi também implementado no escalonador um sistema de submissão de ramos de operações que visa tentar submeter, se possível, um bloco de operações sequenciais ao mesmo nodo, com o objetivo de manter os ficheiros intermédios armazenados localmente. Desta forma, consegue-se evitar a realização de transferências com outros nodos computacionais, o que originaria um grande impacto no tempo final de processamento do processo ETL.

O escalonamento de *workflow* que envolve um grande volume de dados não apresenta apenas um problema de processamento computacional, mas também um problema na transferência de dados, tendo em consideração que os ambientes *Grid* podem conter nodos com limitação de velocidade de *download* e *upload*.

Noutra vertente, também foi estudado o impacto no escalonamento de um *workflow* representativo de um processo ETL utilizando previsão de performance, otimização da distribuição de ramos de

operações e por último a previsão de largura de banda. Como foi apresentado anteriormente, os primeiros três cenários de teste desenvolvidos não consideraram a largura de banda disponível. Assumindo que normalmente um ambiente *Grid* não possui apenas nodos locais, as soluções obtidas não se apresentavam como muito credíveis para acomodar um sistema ETL convencional. No entanto, os cenários de teste seguintes consideraram a problemática da largura de banda em ambientes distribuídos. A previsão de performance e de largura de banda pode produzir piores resultados comparativamente com os resultados sem previsão de performance, mas reduzem o trabalho do escalonador do ambiente *Grid.* Caso contrário, o escalonador teria que consultara largura de banda e a disponibilidade computacional de cada nodo antes de submeter cada trabalho, o que se traduziria num aumento do tempo despendido no processo de escalonamento. Adicionalmente, o modelo de submissão de ramos de operações reduziu significativamente o tempo de comunicação entre tarefas, simplesmente através da utilização de resultados já armazenados no nodo de execução, ao invés de os retornar ao nodo do escalonador para futura transferência.

Os resultados apresentados permitem apresentar algumas vantagens na utilização de ambientes *Grid* no processamento em paralelo das diferentes operações de ETL, tais como aproveitamento de recursos, redução de custos, entre outras. No entanto, não foi possível comparar estes valores com um sistema ETL real, uma vez que os testes realizados apenas incidem no processo de transformação/limpeza do processo de ETL, reduzindo o número de operações a executar.

5.3 Trabalho futuro

Como trabalho futuro é proposta a implementação de novos algoritmos de escalonamento, de forma a obter resultados mais variados para posterior análise e comparação, sendo recomendada a utilização de métodos heurísticos que permitam obter soluções de boa qualidade tomando em conta a performance do algoritmo de escalonamento. Sugere-se também a incorporação de novas funções com diferentes objetivos de escalonamento, uma vez que diferentes organizações possuem diferentes políticas e diferentes níveis desejados de qualidade de serviço. Isto justifica-se porque uma organização pode optar por minimizar o tempo de execução do processo ETL, enquanto que outras podem decidir por obter melhor qualidade de serviço e evitar o aparecimento de erros no seu processo de ETL. Dessa forma utilizarão a duplicação de operações e posterior verificação e comparação dos seus resultados, recorrendo para isto a uma função que vise maximizar a taxa de utilização do ambiente *Grid*.

Neste trabalho foi pressuposto que todas as operações têm por base o mesmo cálculo de disponibilidade, no entanto, na realidade existem operações que podem ter critérios de disponibilidade diferentes daqueles que foram definidos no contexto em estudo. Nesse âmbito, seria pertinente realizar um estudo sobre diferentes parâmetros para o cálculo de disponibilidade, considerando os objectivos das diferentes operações elementares. A aplicação de diferentes pesos às classes de performance

utilizadas no processo de tomada de decisão, sobre a escolha do nodo a executar a tarefa, apresenta-se como um estudo futuro de grande valia, tendo em vista a afinação deste parâmetro de forma a obter melhores resultados. Por fim, seria interessante aplicar modelos de previsão ao tempo de execução das diferentes operações que constituem um processo de ETL. Dessa maneira seria possível reservar recursos antecipadamente, mesmo antes do processo ETL ser submetido aos nodos de uma *Grid*, facilitando e acelerando o processo de tomada de decisão do nodo computacional a utilizar.

Bibliografia

[1] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, "Conceptual Modeling for ETL Processes," presented at the Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, McLean, Virginia, USA, 2002.

[2] R. Kimball, *The Data Warehouse Lifecycle Toolkit*: Wiley Pub., 2008.

[3] V. Santos, B. Oliveira, R. Silva, and O. Belo, "Configuring and Executing ETL Tasks on GRID Environments," *Procedia-Computer Science Journal*, 2011.

[4] I. Foster, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations

Euro-Par 2001 Parallel Processing." vol. 2150, R. Sakellariou, J. Gurd, L. Freeman, and J. Keane, Eds., ed: Springer Berlin / Heidelberg, 2001, pp. 1-4.

[5] P. Plaszczak and R. Wellner, *Grid Computing: The Savvy Manager's Guide*: Morgan Kaufmann Publishers Inc. , 2005.

[6] I. Foster, "What is the Grid? A Three Point Checklist," 2002.

[7] G. Kaur and I. Chopra, "Grid Computing-Challenges Confronted and Opportunities Offered," *Challenges & Opportunities in Information Technology*, 2007.

[8] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," presented at the Proceedings of the 5th ACM conference on Computer and communications security, San Francisco, California, United States, 1998.

[9] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, *et al.*, "A Resource Management Architecture for Metacomputing Systems," presented at the Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[10] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, "GASS: A Data Movement and Access Service for Wide Area Computing Systems," in *IOPADS*, ed, 1999, pp. 78-88.

[11] Laszewski, W. Smith, and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations," presented at the Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing, 1997.

[12] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, *et al.*, "Programming the Grid with gLite," *Computational Methods in Science and Technology*, 2006.

[13] D. Erwin and D. Snelling, "UNICORE: A Grid Computing Environment," presented at the Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing, 2001.

[14] S. Fitzgerald, "Grid Information Services for Distributed Resource Sharing," presented at the Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, 2001.

[15] N. Guerreiro and O. Belo, "Predicting the Performance of a GRID Environment: An Initial Effort to Increase Scheduling Efficiency," in *Future Generation Information Technology*. vol. 5899, Y.-h. Lee, T.-h. Kim, W.-c. Fang, and D. Slezak, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 112-119.

[16] T. Casavant and J. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Trans. Softw. Eng.*, vol. 14, pp. 141-154, 1988.

[17] J. Schopf, "A General Architecture for Scheduling on the Grid," *Specila Issue on Grid Computing, Parallel and Distributed Computing*, 2002.

[18] F. Dong, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems," School of Computing, Queen's University, Kingston, Ontário, Canada2006.

[19] G. v. Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java Commodity Grid Kit," *Journal Name: Concurrency: Pract. Exper.; Journal Volume: 13; Journal Issue: 8-9 ; Jul./Aug. 2001*, pp. Medium: X; Size: 645-62, 2001.