République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Faculté des Sciences Exactes
Département d'Informatique

Mémoire

Présenté par

Mohamed ALIOUAT

Pour l'obtention du diplôme de Magister

Filière : Informatique

Option : Cloud Computing

Thème

La programmation parallèle MPI pour les workflows scientifiques dans les infrastructures GRID

Soutenu le : 04/10/2016

Devant le Jury composé de :

Nom et Prénom	Grade		
Mr BOUKERRAM	**Pr**	Univ. de Béjaia	Président
Mr TARI Abdelkamel	**Pr**	Univ. de Béjaia	Rapporteur
Mme EL-MAOUHAB	**CR**	CERIST	Invitée
Mr SLIMANI Hachem	**MCA**	Univ. de Béjaia	Examinateur

Année Universitaire : 2015/2016

Remerciements

Je remercie en premier Mon Dieu pour m'avoir donné la force, le courage et la patience pour terminer ce modeste travail.

Je remercie mon directeur de mémoire, Monsieur Abdelkamel TARI Professeur à l'université de Bejaia. Je lui suis reconnaissant de m'avoir donné l'opportunité de préparer mon diplôme de magistère.

Je remercie tout spécialement Madame Aouaouache EL-MAOUHAB, chef de la Division Réseaux au CERIST, pour la qualité de son encadrement, ses orientations, sa patience et ses encouragements qui m'ont été d'une grande aide.

Je remercie les membres du jury, Monsieur Boukerram d'avoir accépter la charge de président de jury, ainsi que Monsieur Slimani et Monsieur Fouzi qui ont accepté d'être examinateur de ma thèse et qui m'ont fait l'honneur d'évaluer ce travail.

Je remercie ma famille et très particulièrement mes parents pour m'avoir soutenu et encouragé pendant toute la durée de la préparation de cette thèse.

Je remercie tous ceux, qui de près ou de loin, ont contribué par leurs conseils, leurs encouragements ou leur amitié, à mon cher ami Ammar et tout mes camarade de la promotion, à mes collègues du CERIST qui ont toujours été solidaire avec moi et m'ont soutenu pour la terminaison de ce modeste travail.

A la mémoire de ma tante Laldja

Résumé

La recherche scientifique a besoin aujourd'hui de plus en plus de puissance de calculs pour la résolution des problèmes complexes. La grille de calcul présente une solution adéquate aux attentes de la communauté scientifique. Pour une meilleur collaboration et partage de connaissances, les workflows scientifiques ont une place importante et apporte de nouvelle manière de travail pour les chercheurs. L'utilisation des portails scientifiques va simplifier aux chercheurs l'utilisation des workflows sur les infrastructures des grilles.

Notre objectif est de proposer une approche qui tire profit de la puissance de calcul en exploitant le paradigme du parallélisme par envoi de message sans avoir recoure a modifié le workflow des scientifiques à travers l'interface disponible des portails scientifiques.

MOTS-CLES : Workflow scientifique, grille de calcul, portail scientifique, parallélisme, MPI.

ABSTRACT

Scientific research today needs more and more computing resources to solve complex problems. The grid computing presents an adequate solution to the expectations of the scientific community. For better collaboration and knowledge sharing, Scientifics workflows have an important role and bring a new way of working for researchers. The use of science gateway will simplify the use of workflows on grids infrastructures.

Our goal is to offer an approach that takes advantages of the computational resources on the grids by exploiting the paradigm of the message passing interface parallelism without having to change the tasks of the scientific workflow across the interface offered by science gateway that hides all the complexity of the grid infrastructures<

KEY WORDS: Grid, Scientific workflow, Science gateway, parallelism, MPI.

ملخص

البحث العلمي اليوم يحتاج الى طاقة أكثر من الحوسبة من أجل حل المشاكل المعقدة. تقدم الحوسبة الشبكية حل مناسب للمجتمع العلمي. من أجل تحسين التعاون وتبادل المعرفة.اذ تعتبر تدفقات العمل العلمي دورا هاما وتجلب طريقة جديدة في في الحوسبة عمل الباحثين. إن استخدام البوابات العلمية تبسيط استخدام سير العمل في البنية التحتية الشبكية

هدفنا هو اقتراح النهج الذي يستفيد من قوة الحوسبة من خلال استغلال نموذج من رسالة ترسل عن طريق التوازي دون اللجوء الى تغيربرمجيات المطورة من طرف العلماء عبر واجهة البوابات العلمية المتاحة

كلمات رئيسية : الحوسبة الشبكية, البوابة العلمية, تدفقات العمل العلمي

Table des matières

Liste des figures

Chapitre 1 - Workflow Scientifique

Chapitre 2 - Les grilles de calculs

Chapitre 4 - Approche proposée

Chapitre 5 – Implémentation

INTRODUCTION GENERALE

Introduction générale

L'utilisation du calcul informatique dans la recherche scientifique a accéléré l'avancement des travaux. Cette excitation d'aller plus vite afin de résoudre des problèmes plus complexes a engendré des besoins de plus en plus croissants en ressources de calcul et a posé plusieurs obstacles d'ordre de gestion des systèmes et disponibilité d'infrastructures informatiques.

Les grilles de calcul ou « Grids » ont émergé comme une forme de cyber-infrastructure mondiale pour les applications scientifiques en intégrant des ressources hétérogènes et distribuées à grande échelle. Les grilles de calcul sont une solution très sollicitée par la communauté scientifique.

Le calcul scientifique est l'un des piliers de la recherche scientifique sur la base des fondements de la théorie et de l'expérimentation. Son efficacité et sa rapidité portent à la recherche un gain en temps et en argent. Aujourd'hui les chercheurs développent des applications scientifiques complexes pour gérer, traiter et analyser un ensemble important de données. Ceci est un dur et long travail qui vient de s'ajouter aux scientifiques et qui demande du temps, de l'apprentissage et de la collaboration. La collaboration et le partage des données et des résultats demeurent indispensables pour optimiser le temps d'analyses et améliorer le traitement des données volumineuses.

Pour mener à bien de tels travaux scientifiques et afin de tirer profit des ressources de la grille de calcul, les workflows scientifique ont été adopté dans le milieu scientifique afin de permettre l'automatisation des tâches, une flexibilité dans les travaux et une efficacité des actions entreprises par les chercheurs.

Le parallélisme par envoi de message est un paradigme très intéressant à explorer sur les workflows scientifiques. Parallélisé les applications scientifique est une tâche fastidieuse qui requière de l'expérience, du temps et beaucoup d'efforts. Il est intéressant de paralléliser les workflows scientifiques de manière transparente aux chercheurs.

Objectifs

Dans ce mémoire, notre travail consiste à proposer un interpréteur développé en MPI (Message Passing Interface) et d'utilisé ces fonctions de communication pour paralléliser les tâches constituant le workflow scientifique en exploitant les ressources de calcul disponible sur notre grille nationale DZ e-Science Grid.

Organisation du mémoire

Afin de bien mener notre travail, notre mémoire est organisé de la manière suivante :

Chapitre 1 : nous avons commencé avec une étude sur les workflows scientifiques et leurs systèmes de gestion et les différents types de parallélisation dans ces derniers.

Chapitre 2 : nous avons exploré le monde des grilles de calcul, leurs architectures et leurs fonctionnements.

Le *chapitre 3* : est une vue rapide sur les portails scientifiques, le rôle et l'importance qu'ils occupent pour faciliter la tâche aux scientifiques.

Chapitre 4 : nous avons proposé notre approche pour la parallélisation du workflow scientifique dans la grille de calcul à travers la conception d'un interpréteur MPI.

Chapitre 5 : présente la mise en œuvre pour la réalisation de notre interpréteur MPI.

Nous terminons notre modeste travail par une conclusion générale et des perspectives.

Chapitre 1 - Workflows scientifiques

1. Introduction

Les outils de calculs ont été adoptés dans le domaine de recherche scientifique afin de résoudre des problèmes complexes et pouvoir analyser et visualiser les résultats. La résolution de ces problèmes requière une exécution d'un ensemble de tâches de calcul bien définies souvent répétitives engendrant à chaque étape de calcul un ensemble de données. L'automatisation d'exécution des tâches a été prie en charge par les systèmes workflows afin d'aider les chercheurs dans leurs travaux. La réutilisation de l'expérience est une action importante pour les scientifiques; le partage des expériences et des résultats dans la communauté scientifique permettra une avancée rapide de la recherche.

Les scientifiques ont été inspirés par le modèle workflow appliqué dans les entreprises pour apporter tous ses points positifs à la communauté de recherche. Les systèmes de gestions des workflows scientifiques ont été développés pour prendre en charge la définition, la gestion et l'exécution du workflow.

Nous allons présenter dans ce chapitre un état de l'art sur les workflows scientifiques, la différence entre le workflow scientifique et le workflow d'entreprise. Nous découvrirons la structure d'un workflow, son cycle de vie et les différents systèmes de gestion des workflows scientifiques et leurs architectures et nous finirons par la parallélisation des workflows scientifiques.

2. Workflow scientifique

Une avancée importante en matière de recherche scientifique a été atteinte grâce au calcul et l'analyses des volumes importants de données. Ces calculs peuvent être constitués de plusieurs centaines d'étapes, ou chaque étape peut intégrer différentes sources de données et des modèles de calcul développés par différents groupes de recherche. Les applications et les données peuvent être distribuées dans l'environnement d'exécution. La coordination et la gestion d'un tel calcul complexe et distribué pose aux scientifiques de sérieux défis à relever.

Les workflows ont émergé récemment comme un paradigme pour la gestion et la représentation des calculs scientifiques complexes et distribués, accélérant le pas du progrès scientifique.

Définition 1 : Le ‘‘Workflow Management Coalition’’ décrit le workflow scientifique comme une automatisation des processus métier, d'une partie ou bien le tout, dans laquelle documents, informations ou bien tâches sont passés d'un participant a un autre pour une action, selon un ensemble de règles et de procédure [1]. Les workflows

sont des activités qui impliquent la coordination de l'exécution de multiples tâches par différents processus.

Définition 2 : Pour la recherche scientifique, il s'agit de l'ensemble des étapes et des traitements pour une expérience spécifique. Le workflow scientifique est une transposition du terme général de workflow dans le contexte expérimental, c'est à dire relatif uniquement à des enchaînements de traitements par l'intermédiaire de flots de données. Dans le contexte de Grille de calcul, le workflow scientifique est l'automatisation des processus qui sont impliqués dans l'orchestration du flux de données, d'un ensemble des services de grille.

Les workflows scientifiques sont composées d'un ensemble de tâches séquentielles et concurrentielles, dont l'ordre est déterminé par l'interdépendance des données [4].

Définition 3 : Une tâche est l'unité de traitement de donnée la plus basique dans le workflow scientifique, ses sources de données sont soit des fichiers ou bien des données produites par l'exécution des tâches qui le précédent, et ces données générés sont soit reprises par les tâches qui la succède ou écrites dans des fichiers de sortie.

Chaque étape dans le workflow scientifique spécifie un processus ou un calcul à exécuter. Le workflow scientifique relie les étapes par rapport au flux de données et les dépendances entre les étapes. La représentation de ces workflows scientifiques contient des détails nécessaires pour l'analyse à chaque étape, y compris l'utilisation de ressources spécifique pour l'exécution et le stockage des données dans un environnement distribué.

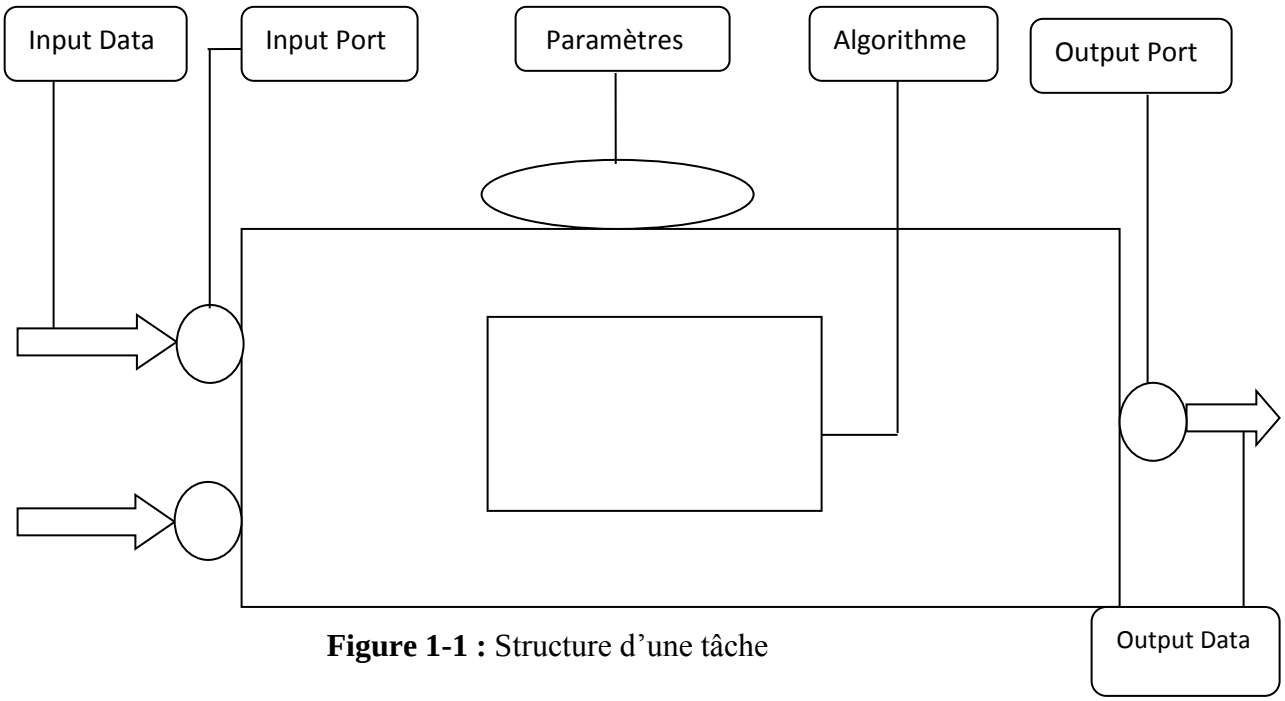

Figure 1-1 : Structure d'une tâche

2.1 Progression des Workflows

Le premier workflow était encodé et géré par un langage complexe de contrôle de job et un script shell qui ont été utilisé pour organiser les données d'entrée et de passer ensuite les données de sortie vers les stations d'enregistrement. Avec l'arrivée des réseaux et des systèmes distribués, les simples scriptes ne pouvaient plus contrôler l'exécution et la coordination des tâches sur les différentes machines.

Le workflow a été conçu pour régler les problèmes comme la synchronisation des taches en concurrence à distance, rétablissement après un crash du système, le loging distribué, la gestion des données à distance afin d'évoluer dans un environnement bâtit sur un système de fichier et de bases de données distribuées.

Proposer le paradigme du workflow pour la composition d'applications sur les grilles de calculs a plusieurs avantages [2] :

- L'habilité à construire une application dynamique qui orchestrera les ressources distribuées.

- Utilisation des ressources qui sont localisées dans un domaine particulier.

- Intégration de multiples équipes impliquées dans différentes parties du projet, ce qui permet la collaboration entre équipes.

- En plus de l'automatisation, le workflow peut procurer l'information nécessaire pour une reproductibilité scientifique, et partage des résultats entre les collaborateurs.

2.2 Cycle de vie du Workflow scientifique

Le cycle de vie d'un workflow est constitué de quatre phases [6] :

- La conception et la composition de workflow concret : la modélisation des flux de données par une concaténation des étapes conceptuel de traitement et l'attribution des méthodes actuelles à des tâches abstraites donnera comme résultat un workflow concret.

- Le mapping du workflow concret avec les ressources physiques sous-jacentes (scheduling) : si le mapping est effectué de manière automatique, il est appelé ‘‘workflow planning’’ [5].

- L'exécution du workflow physique : pour exécuter un workflow concret, les données d'entrée et les traitements nécessaires doivent être désigné aux ressources calculatoires. Dans le workflow scientifique, cette attribution est appelée scheduling et les résultats sont dans le workflow exécutable.

- L'enregistrement des méta-données et la provenance des données dans toutes les étapes du cycle de vie du workflow.

3. Workflow Scientifique et le Workflow Entreprise

Le concept formel des workflows a existé pendant une longue durée dans le domaine de l'entreprise. Le ''Workflow Management Coalition'' a travaillé depuis plusieurs années dans le développement d'un ensemble de modèle de référence, de documents et des standards.

Un processus d'entreprise est une instance de n'importe quelle tâche bien définie qui est souvent répété comme une partie standard des tâches de l'entreprise [3]. Ça peut être des étapes requises pour accomplir des tâches d'entreprise interne comme l'audit interne ou la gestion du système d'information de l'entreprise.

Le *workflow d'entreprise* se caractérise par la sécurisation et l'intégrité des séquences d'actions. Le concept d'intégrité d'actions est incorporé dans le concept transactionnel et est le noyau pour comprendre le workflow d'entreprise. Ces transactions doivent être conforme aux règles dites ACID. Une transaction ACID représente une unité de tâche logique qui est composée d'un ensemble d'opérations.

On définie l'acronyme ACID comme suit :

- **A :** pour *Atomicité*, qui garantie une exécution "tous ou rien".
- **C** : se réfère à l'état de la base de données qui est toujours *Consistante.*
- **I :** l'action de chaque transaction est *Isolée,* c'est à dire qu'elle ne sera pas vue et ne sera pas affectée par d'autres opérations qui ne font pas partie de la transaction.
- **D** : pour la *Durabilité* car une fois la transaction terminée, son résultat sera conservé même si le système tombe en panne.

➢ ***Progression***

Le workflow d'entreprise a largement adopté le concept de l'orienté objet dans des systèmes d'objets distribuées comme le Common Object Request Broker Architecture (CORBA) [3]; et ce afin de gagner dans le temps d'exécution des processus du workflow, avoir une bonne fiabilité des résultats et réussir l'interopérabilité entre différentes plateformes.

Une architecture orientée service SOA (Service Oriented Architectures) dont les composants d'application peuvent être assemblés avec peu d'effort dans un réseau de services pour créer un processus métier.

L'ensemble dominant de ses standards est celui connu sous le nom Web Services. On en trouve le *Web Service Description Language (WSDL)* qui est un service de description, le *Universal Description Discovery and Integration (UDDI)* qui est un protocole pour le service Discovery, le *Simple Object Access Protocol (SOAP)* pour les services de communications, et le *Web Service Business Process Execution Language (WS-BPEL)* pour les workflows.

Le *BPEL* offre un langage riche pour l'orchestration que se soit pour le workflow entreprise ou le workflow scientifique. Le processus BPEL spécifie l'ordre exacte dans le quel les services participants doivent être sollicité dans un ordre séquentiel ou parallèle. Les workflows spécifiés par le BPEL sont entièrement exécutable et portable dans des environnements conformes au BPEL, ce qui est très important pour la réutilisation et l'échange des workflows.

Avec des connaissances limitées dans la programmation et des outils appropriés, le *BPEL* a été utilisé pour construire un workflow afin de réaliser une expérience ou récupérer des données d'un service à distance.

- ***Evolutivité***

Les workflows d'entreprises sont moins dynamiques et évolutifs que les workflows scientifiques. Ces derniers changent fréquemment et peuvent impliquer un volume important de données.

- ***Développement***

Les workflows d'entreprises tendent à être construit par des développeurs et des ingénieurs spécialisés dans le business flow. Le workflow scientifique est en générale construit par les scientifiques eux même, chose qui peut être difficile car ils ne sont pas expert dans le développement ou les réseaux.

- ***Composition***

Les scientifiques ont souvent tendance a intégré des portions existantes d'autres workflows, quand c'est nécessaire. Le workflow d'entreprise ne prend pas en charge le stockage des workflows dans les repositories et de pouvoir les récupérer plus tard durant la composition d'un nouveau workflow.

- ***Flexibilité***

Le domaine scientifique est nettement plus flexible que dans le domaine d'entreprise. Le processus d'entreprise est généralement prédéfini et exécuté dans des routines précises. Le domaine scientifique demande de l'exploration et de la recherche en permanence. Le scientifique est toujours confronté à des expériences qui peuvent aboutir à des erreurs, ce qui nécessite des modifications des étapes ou il peut même décider de filtrer les données récoltées à partir d'un appareil de mesure. Cette habilité d'exécuter, de faire une pause, et de reprendre l'exécution d'un workflow n'est pas disponible dans la plus part des workflows d'entreprise.

4. Conception de workflow scientifique

La conception de workflow est constitué de quatre facteurs qui sont (a) la structure du workflow, (b) modèle de workflow, (c) éditeur de composition du workflow.

Nous allons détailler ses différents facteurs pour bien comprendre la conception des workflows.

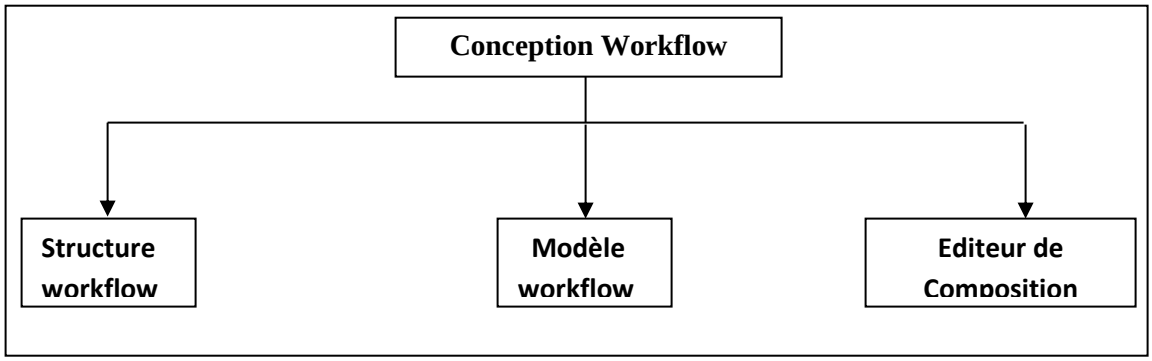

Figure 1-2 : Conception de Workflow

4.1 Structure de workflow

Un workflow est composé par la concaténation de multiples taches en fonction de leurs dépendances. La structure du workflow peut être représenté en deux façon, soit en *Directed Acyclic Graph (DAG)*, ou en *non-DAG*.

Dans les workflows basés sur la représentation DAG, la structure du workflow peut être classifiée en *séquence, en parallélisme ou en choix* [8].

- La structure en *séquence* est définie comme une série de tâches ordonnées. Chaque tâche commence après que la tâche précédente soit terminée.

- Le *parallélisme* représente les tâches qui sont exécutés en concurrence, plutôt qu'en série.
- Dans la structure *en choix*, la tâche est sélection pour s'exécuter quand les conditions associées sont vérifiées.

Comme pour la classification DAG, la classification non-DAG inclus la structure en *séquence*, en *parallélisme*, en *choix* et nous trouvons en plus la structure *itérative* dans la quelle des sections de tâches de workflows sont dans des blocs d'itération ont la permission d'être répéter.

- La structure *itérative* est souvent utilisé dans les applications scientifiques, dans la quelle un ou plusieurs tâches ont besoin d'être exécuter à maintes reprises.

Ces quatre types de structure de workflows peuvent être utilisés pour la construction de workflow complexe.

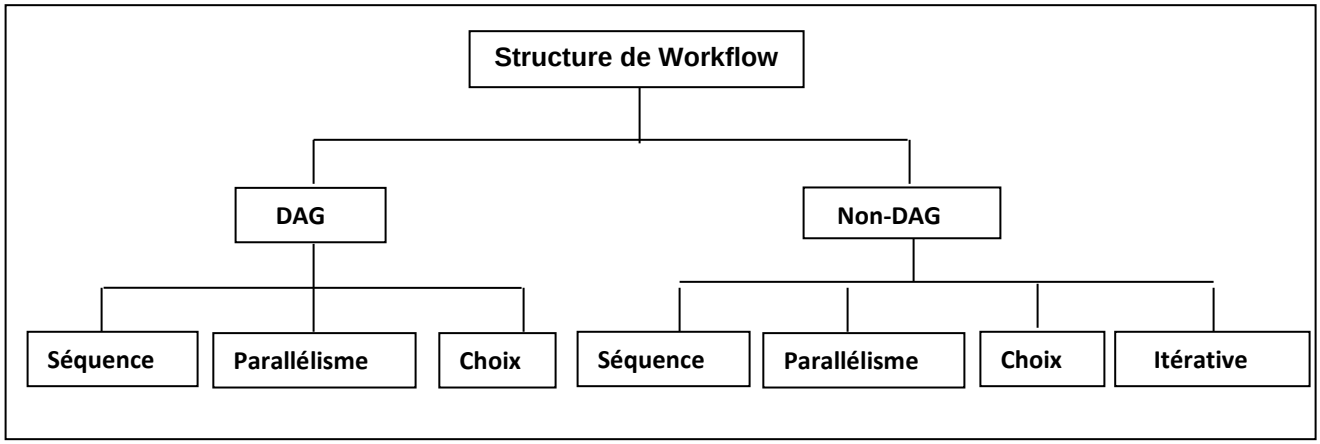

Figure 1-3 : Structure de Workflow

4.2 Modèle de workflow

La modèle de workflow aussi appelé spécification du workflow défini les tâches et la structure constituant le workflow. La modélisation du workflow est basée sur deux modèles qui sont le modèle *abstrait* et le modèle *concret*.

4.2.1 Modèle abstrait

Dans le *modèle abstrait*, un workflow est décrit dans une forme abstraite dans laquelle le workflow est spécifié sans se référer aux ressources de la grille nécessaires à l'exécution des tâches. Un modèle abstrait procure une manière flexible pour les

utilisateurs pour définir les workflows sans être concerné par les détails de l'implémentation de bas niveau [9] [10].

4.2.2 Modèle Concret

Le *modèle concret* lie les tâches du workflow aux ressources spécifiques. Les tâches dans le modèle concret peuvent aussi faire appel à des applications transitaires pour transférer un code d'un nœud de calcul à un autre site de données pour une analyse à grande échelle. Un modèle concret de workflow peut être généré juste avant ou pendant l'exécution du workflow selon l'état actuel des ressources. Dans certain systèmes, chaque tâche dans un workflow est concrétisée seulement au moment de son exécution. Cependant, il y a des utilisateurs qui utilisent le modèle concret pour avoir un contrôle sur la séquence d'exécution [11].

4.3 Editeur de composition de workflow

Les éditeurs de compositions permettent aux utilisateurs d'éditer les workflows directement. Les utilisateurs peuvent utiliser le langage textuel de workflow ou la modélisation basée sur le graphe pour composer les workflows.

4.3.1 Editeur textuel

Avec la modélisation basée sur le langage textuel, les utilisateurs peuvent exprimer leur workflow avec un langage à balise comme le *XML* (Extensible Markup Language), ou d'autres formats comme *Condor DAGman* qui fournit un mécanisme pour décrire les dépendances entre les traitements [13]. Le langage textuel bas niveau répond aux requêtes des utilisateurs via les scripts. Les workflows sont spécifiés dans la forme de fichiers de configuration complexe, qui sont interprétés et exécutés par les systèmes de gestion des workflows scientifiques.

Le langage textuel peut être adapté aux utilisateurs expérimentés, mais ils ont besoin de mémoriser la syntaxe du langage; en d'autres termes, il est difficile d'exprimer un workflow large et complexe avec des scriptes de compositions.

Dans la plus part des systèmes de grilles, les langages textuel de workflow sont conçu pour faire le lien entre l'interface graphique de l'utilisateur et le moteur d'exécution du workflow dans la grille.

4.3.2 Editeur de graphe

La conception basée sur le graphe permet une définition graphique d'un workflow arbitraire à travers des éléments de graphe de base. Elle permet aux utilisateurs de

travailler avec une représentation graphique du workflow. Les utilisateurs peuvent composer et revoir leurs workflows juste en cliquant et en plaçant le composant voulu. Elle évite tous les détails de bas niveau et permettra aux utilisateurs de se concentrer plus sur la couche la plus haute d'abstraction dans leurs applications [14].

Les *réseaux de Petri* sont une classe spéciale de graphes directe qui peuvent modéliser l'exécution des tâches conditionnelles, en boucles, séquentielle et parallèle [18][19].

Le diagramme d'activité d'UML a aussi été étendu et appliqué comme un langage de spécification de workflows.

La modélisation basée sur le graphe est plus préféré par l'ensemble des utilisateurs par rapport à la modélisation basé sur langage textuel.

Plus tôt de suivre la syntaxe et la sémantique des réseaux de Petri et de l'UML, plusieurs outils d'éditeurs de workflows pour les grilles de calculs ont créé leurs propres représentation graphique des composants de workflows. Ces outils sont plus pratique pour les utilisateurs afin de manipuler leurs applications workflows, car ils leurs fournissent un environnement de programmation plus adéquat.

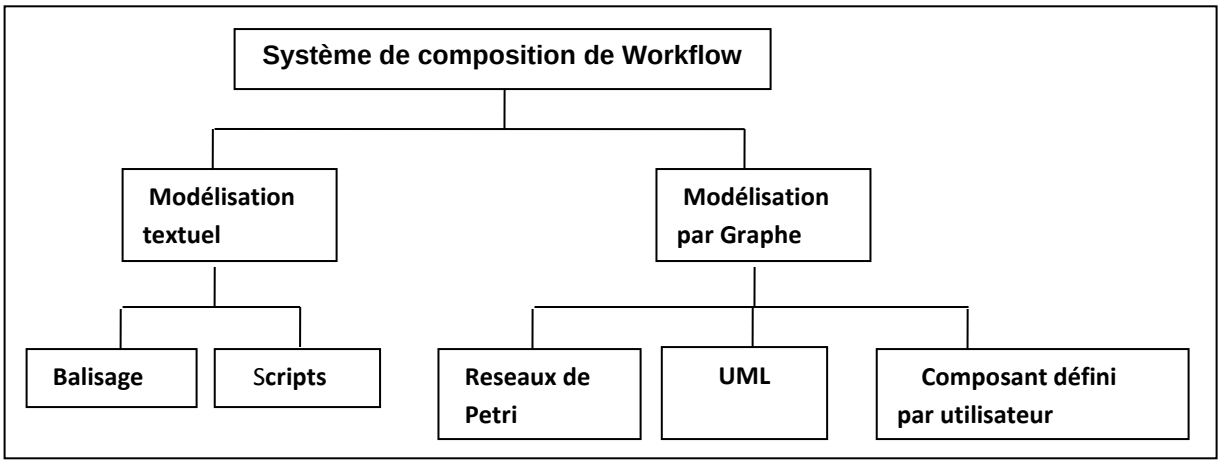

Figure 1-4 : Système de composition de workflows

5. Les systèmes de gestion des Workflows scientifiques

Le système de gestion des workflows a été proposé et développé afin de définir, gérer et exécuter les workflows scientifiques sur les ressources de calculs hétérogènes et distribués tel que les grilles de calcul. Le système de gestion des workflows scientifiques

traite le cycle de vie du workflow en permettant de modéliser, exécuter, gérer et stocker les workflows scientifiques. La plus part des systèmes de gestion fonctionnent sur des workflows concrets en répondant aux demandes des utilisateurs pour des compositions spécifiques de traitement de tâches concrets.

5.1 Architecture des systèmes de gestion des workflows scientifiques

D'après le modèle de référence proposé par la Workflow Management Coalition (WfMC), le niveau le plus haut de la structure est caractérisé par les fonctions build-time et les fonctions run-time :

- Les fonctions build-time sont considérées comme la définition, la modélisation des tâches workflow et leurs dépendances. Les utilisateurs interagissent avec les outils de modélisation du workflow pour la génération d'un workflow spécifique, qui est soumis aux services run-time appelé service de promulgation du workflow pour l'exécution.

- Les fonctions run-time sont concernées par la gestion des exécutions du workflow et l'interaction avec les ressources de la grille pour le traitement d'applications workflows.

La plus part des fonctionnalités proposées par le service promulgation du workflow sont l'*ordonnancement, la gestion des pannes* et le *transfert des données.* Le service de promulgation de workflow peut être construit en haut de la couche inférieure de la structure du workflow dans la grille qui est le *middleware*, invoque les services proposés par les ressources de la grille. Que se soit au niveau du build-time ou bien du run-time, les informations sur les ressources et les applications peuvent avoir besoin d'être récupérées grâce aux informations des services de la grille.

Une classification des différents éléments des systèmes de gestion des Workflows dans les grilles a été proposée par Yu et Buyya [8] afin de comprendre la construction et l'exécution des workflows dans les grilles. L'approche de classification des systèmes de gestion des workflows dans les grille consiste en cinq éléments: (a) conception de workflow, (b) Extraction de l'information, (c) ordonnancement workflow, (d) tolérance aux pannes, (e) transfert de données.

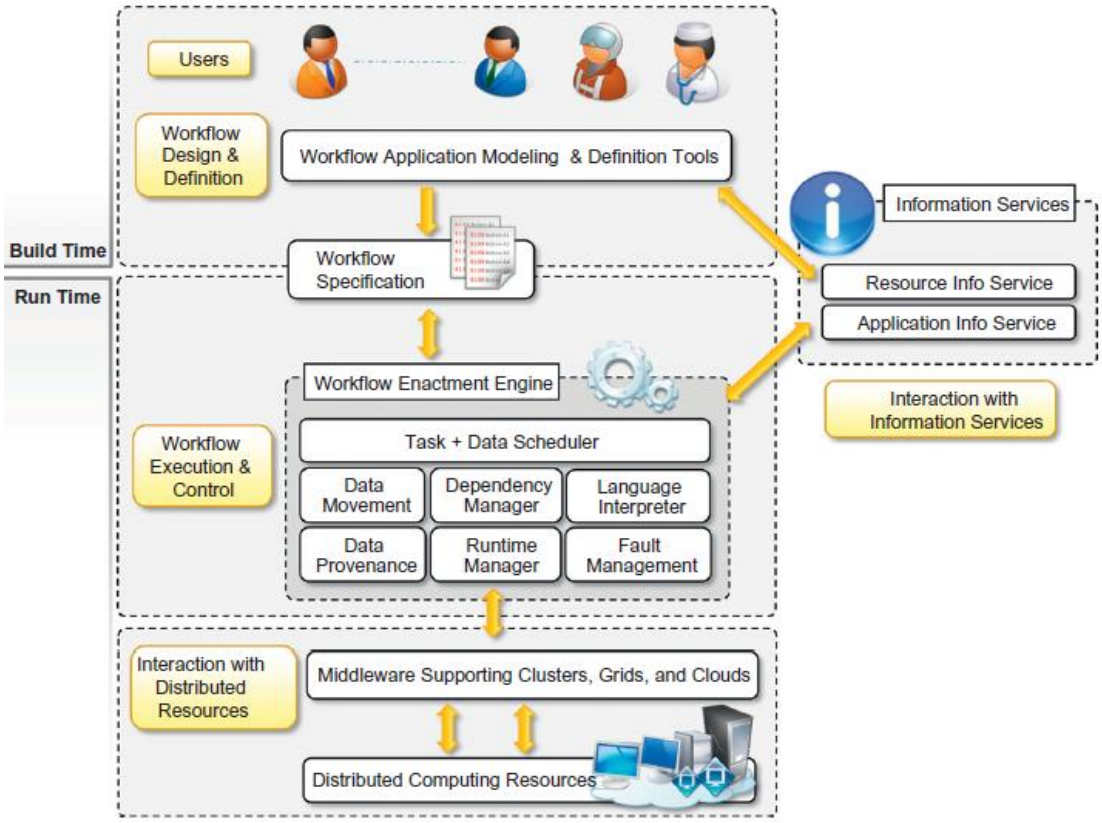

Figure 1-5 : Les systèmes de gestion des workflow dans les Grilles de calcul

5.1.1 Conception de workflow

Cette partie a déjà été abordée dans la section 4.

5.1.2 Extraction de l'information

Le système de gestion des workflows dans les grilles n'a pas la capacité d'exécuter les tâches, il ne fait que coordonner l'exécution des tâches sur les ressources qui se trouvent sur la grille. Afin de distribuer les tâches sur les ressources adequat, l'information sur l'état des ressources doit être extraite [29]. Il est a noté qu'il y a trois types d'extraction d'information : *information statique, information historique et information dynamique.*

- *L'information statique* fait référence à l'information qui ne change pas dans le temps. Ca peut inclure l'infrastructure comme le nombre de processeurs, la configuration système, la qualité de service, l'accès au ressources, l'information utilisateur. L'information statique est utilisée par

le système de gestion des workflows pour une présélection des ressources pendant l'initiation de l'exécution du workflow.

- Contrairement à l'information statique, l'*information dynamique* renvoi l'état des ressources de la grille comme la charge moyenne du cluster, l'espace libre restant dans les disques, l'utilisation du CPU, et les processus actifs. Elle inclut aussi les informations des tâches en exécution.

- L'*information historique* est obtenue à travers divers évènements qui se sont passés comme l'historique des performances et l'historique des exécutions des ressources de la grille et des composants d'applications. Généralement, le système de gestion des workflows peut analyser l'information historique afin de prédire le comportement futur des ressources et des composantes d'applications dans n'importe quel ensemble de ressources. L'information historique peut aussi être utilisée afin d'améliorer la fiabilité dans les futurs exécutions du workflow et éviter les erreurs des problèmes déjà connu.

5.1.3 Ordonnancement du workflow

Dans les systèmes distribués, l'ordonnancement des tâches est catégorisé en ordonnancement de tâche *local* et *global*.

L'ordonnancement *local* implique l'affectation des tâches à tranches de temps sur une seule ressource tandis que l'ordonnancement *global* consiste à décider où une tâche va être exécuté [35].

L'ordonnancement de workflow est une sorte d'ordonnancement global des tâches car elle est axée sur la gestion de l'exécution des tâches interdépendantes sur des ressources partagées qui ne sont pas directement sous son contrôle.

5.1.4 Tolérance aux pannes

Dans un environnement de grille, l'exécution d'un workflow peut être affrontée à des pannes pour divers raisons : la variation dans l'environnement d'exécution, la non disponibilité des composants de services ou des applications nécessaire, surcharge des ressources et la saturation des mémoires, défaillance dans les composants ou le réseau de calculs. Les gestionnaires des systèmes Workflows doivent être capables d'identifier et de traiter les pannes et de relancer l'exécution.

Awang et al ont divisé les techniques de traitement des pannes de workflow en deux niveaux différents : *niveau de la tâche* et *niveau du workflow.*

5.1.5 Transferts de données intermédiaires

Les fichiers en entrée pour des tâches d'applications workflows dans un environnement grille, ou bien encore, les fichiers en sortie qui sont utilisés dans le traitement des tâches ''enfants'' sur d'autres ressources de calcul ont besoin d'être mise sur un site distant avant le traitement de la tâche. Certains systèmes ont besoin de la présence de l'utilisateur pour la gestion de transferts de données intermédiaires dans un workflow spécifique.

1. Différentes approches des mécanismes automatiques ont été proposé; une approche *centralisée a été* proposé pour le transfert des données intermédiaires entre les ressources à travers un point central.

2. La deuxième approche est l'approche *intermédiaire*. Elle se base sur un système de gestion de données distribuées. Les données intermédiaire générés dans chaque étape sont enregistrées dans un service de catalogue de réplication [61], alors les entrés de chaque tâche peuvent être obtenu en consultant ce service.

3. L'approche *peer-to-peer* propose de transférer les données entre les ressources de calculs. Les données sont transférées de la source vers la destination directement sans passer par un autre service intermédiaire, ce qui engendre un gain dans le temps de transmission et une réduction des goulots d'étranglement causés dans les approches *centralisée* et *intermédiaire*.

5.2 Quelques systèmes de gestion de workflows scientifiques

5.2.1 Taverna

Taverna est un outil open source pour la conception et l'exécution des workflows, initialement créé par le projet *myGrid* sous le nom Taverna Workbench. Taverna permet aux utilisateurs d'intégrer différents composants logiciels, en incluant WSDL SOAP ou REST Web services. Le moteur du workflow Taverna est également disponible séparément comme un outil en ligne de commande, API Java ou en tant que serveur. Taverna est utilisé par des utilisateurs de différents domaines, comme la bioinformatique, médecine, astronomie, science social … etc.

5.2.2 Kepler

Kepler est un logiciel libre pour la conception, exécution, réutilisation, archivage et le partage des workflows scientifiques. Basé sur le système Ptolemy 2, Kepler présente une sémantique générique pour tous les types possibles de processus rencontrés dans divers domaine tel l'ingénierie, la physique, l'écologie … etc. Képler fait la séparation du moteur d'exécution à partir du modèle du workflow, et fait assigner un modèle de calcul, appelé « Director » pour chaque workflow.

5.2.3 Pegasus

Pegasus est un système de gestion de workflow, développé à l'institut de science de l'informatique en Californie. Pegasus à la possibilité de mapper le workflow sur l'infrastructure mise a disposition. En plus de la composition et de l'exécution il permet aussi la gestion et le monitoring des tâches composant le workflow.

6. La parallélisation des workflows scientifiques

Les systèmes de gestions des workflows scientifiques ont besoin de solution plus rapide avec une capacité de calculs et de stockage à grande échelle. Il y a toujours le besoin d'exploiter la puissance des machines en s'appuyant sur des algorithmes parallèles s'exécutant sur des ressources de calculs distribuées notamment des grilles de calculs.

Liu et al [67] ont posé les définitions des composants de base d'un workflow scientifique :

Définition 1 : Une activité est une description d'un morceau de processus qui forme une étape logique dans une représentation d'un workflow scientifique. Elle définie l'association entre les formats de données et les méthodes de calculs des données mais elle a besoin des données associées et des ressources de calculs afin d'aboutir a une exécution.

Définition 2 : Les données associées à une activité se basent sur les données d'entrées et les paramètres de configuration. Quand les paramètres de configuration sont fixés et les données d'entrées sont prévus, l'exécution d'une activité de workflow est représenté par plusieurs tâches.

Définition 3 : Une tâche est la représentation d'une activité dans une exécution unique de cette dernière, qui traite un bloc de données. Une activité peut correspondre à un ensemble de tâches pour différents données d'entrées. Parfois une tâche est appelé "job", alors que le terme tâche est utilisé pour représenter le concept d'activité de traitement de données qui compose le workflow scientifique.

Le workflow scientifique décrit le traitement de données par un ensemble de tâches. La parallélisation traite la question comment répartir la charge de travail associé

au workflow scientifique sur un ensemble de ressources de calcul et comment transformer et optimiser l'exécution du workflow séquentielle en un workflow parallèle.

Il y a différentes méthodes afin d'accomplir la parallélisation, qui font intervenir soit une subdivision au niveau des tâches de workflow, soit sur les données d'entrées, ou bien sur les deux en même temps. Un aspect important dans ce contexte est le degré du parallélisme qui est défini par le nombre de machines ou des threads qui sont en concurrence à n'importe quel moment.

Il y a quatre types de parallélisme des workflows [62] à savoir : *parallélisme des tâches, parallélisme des données, parallélisme en pipeline et le parallélisme hybride*. Nous allons voir les caractéristiques et les relations entre ses différentes approches.

6.1 Parallélisme des tâches

Le parallélisme des tâches se base sur la distribution des tâches composant le workflow qui se situent sur des branches parallèles du workflow sur différents nœuds de calcul. Le nombre maximum de tâches en parallèle peut être connu en analysant la structure du workflow. Cette dernière nous procure des informations assez importante pour le bon déroulement de l'étape d'exécution et elle définie de manière explicite des données échangées entres les tâches, ce qui rend le parallélisme des tâche moins difficile à mettre en œuvre.

Deux tâches peuvent être indépendantes si l'exécution de chaque tâche ne dépend pas du résultat d'exécution de l'autre. Les tâches ont besoin de données d'entrées de différents résultant d'exécution des tâches ‘‘parents’’ en concurrence ce qui les oblige à attendre la fin d'exécution de tout les ‘‘parents’’ sources de ses données. Si deux tâches sont dépendantes alors elles seront exécutées de manière séquentielle. Le besoin de mémoire tampon (buffer) et la synchronisation de l'exécution des tâches est nécessaire pour le bon déroulement du parallélisme des tâches. Les tâches indépendantes doivent être exécutées en même temps.

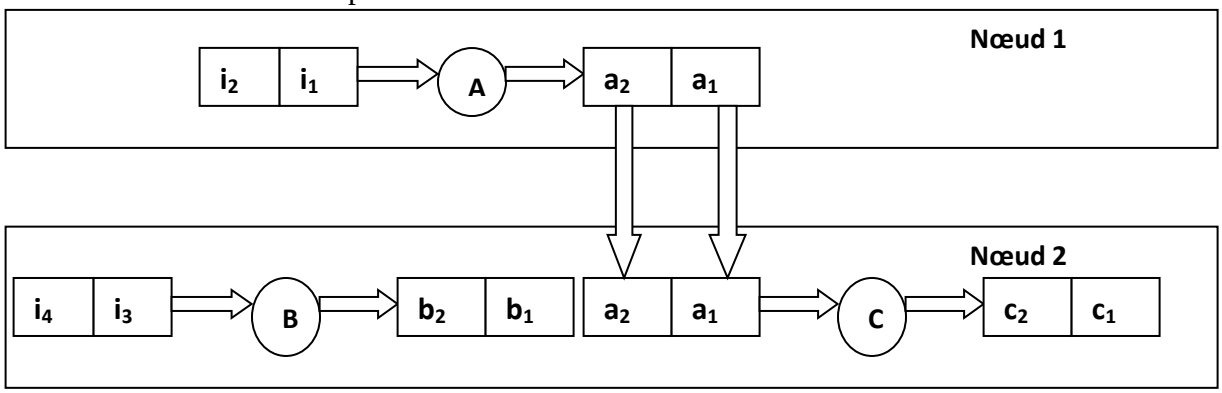

Figure 1-6 : Parallélisation des tâches

6.2 Parallélisme des données

Single Program Multiple Data (SPMD) est une méthode d'exécution parallèle ou un seul programme traite plusieurs données et toutes ses données sont exécutées en parallèle si aucune dépendance n'existe entre eux. Cette technique est utilisé dans la parallélisation des workflows qui ont un large ensemble de données en entrées afin d'accélérer leurs exécution.

Dans le parallélisme des données, le parallélisme est possible si les données en entrée ou intermédiaires peuvent être divisé en différents blocs distincts, chaque bloc de données est distribué et calculé sur un nœud de calcul différent, ce qui implique que le workflow va être répliqué sur les différents nœuds de calcul pour chaque bloc de données qui sera traité. Une combinaison des différents résultats est nécessaire pour produire un résultat final.

Ces techniques demandent une intervention humaine ou fonctionnelle des workflows afin de pouvoir exploiter le parallélisme. Le degré de partitionnement dépend de la taille de l'ensemble de données et le nombre de nœuds de calculs disponibles au moment de l'exécution du workflow.

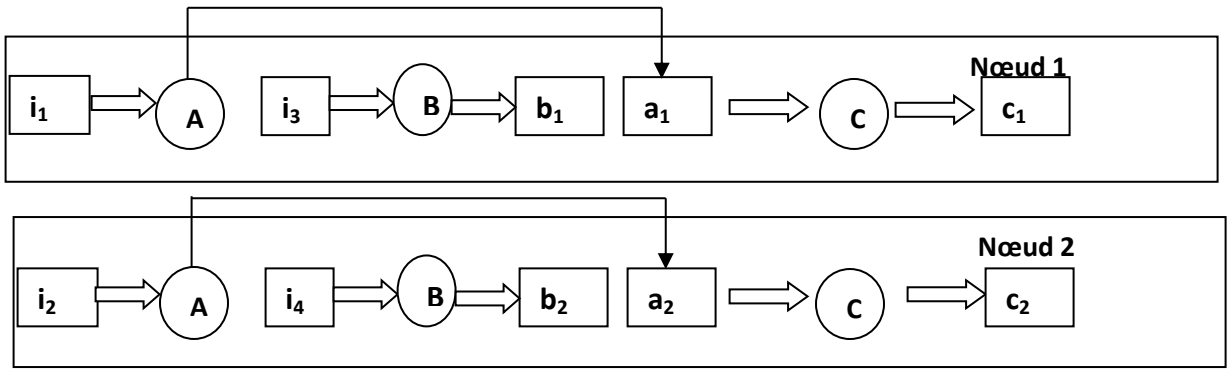

Figure 1-7 : Parallélisation des données

6.3 Parallélisme en pipeline

Dans le parallélisme en pipeline, plusieurs activités dépendantes du type producteur/consommateur sont exécutées en parallèle par différentes tâches. Une part des données en sortie d'une activité est consommée directement par l'activité qui est dépendante d'elle.

Dans le parallélisme en pipeline, les étapes séquentielles de traitement de données sont exécutées simultanément sur différentes parties des données en entrée. Les données

produites par les tâches du workflow en partielle sont passées en entrée pour une consommation immédiate de la part des autres tâches. Le parallélisme en pipeline partage des caractéristiques avec le parallélisme des tâches et le parallélisme des données et peut être considéré comme un sous ensemble des deux. Les tâches composantes le workflow scientifique sont distribuées sur différents nœuds de calcul. Cependant, cette distribution n'est pas limitée aux tâches dans les branches parallèles du workflow, ce qui résulte à un haut degré de parallélisme. Les données en entrée sont fragmentées et traitées dans différents nœuds. Contrairement au parallélisme des données, l'ordre chronologique dans lequel les fragments de données sont traités, ainsi que l'attribution des tâches dans les nœuds de calcul sont déterminées à l'avance et limitée par le concept du pipeline.

L'avantage du parallélisme en pipeline c'est que le résultat de l'activité du producteur n'a pas besoin d'être entièrement concrétisé, des parties de données peuvent être consommés des qu'elles sont prêtes, économisant ainsi la mémoire et les accès aux disques.

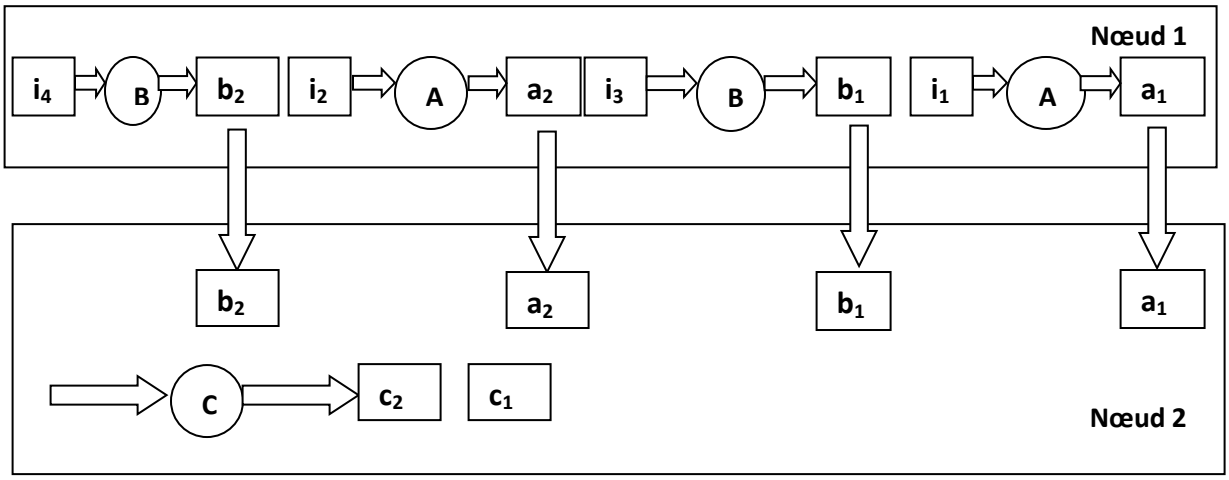

Figure 1-8 : parallélisation en pipeline

6.4 Parallélisme hybride

La combinaison des trois types de parallélisme : tâches, données et pipeline peut avoir un résultat positif sur l'exécution du workflow. Ces différents types de parallélisme ont tous le but de répartir la charge de travail associé avec le workflow scientifique.

Le système de gestion des workflows scientifiques peut commencer par un parallélisme de données au sein de chaque activité, puis il peut partitionner le workflow scientifique sur des parties indépendantes ou des fragments d'activité pour des activités

indépendantes, afin que chaque part ou fragment soit exécuté sur un nœud de calcul. Le parallélisme en pipeline pourra être appliqué pour exécuter les activités en parallèle.

Les stratégies de parallélisme peuvent changer au moment de l'exécution, selon le comportement de l'environnement de calcul parallèle. Les activités qui ont besoin de données d'entrées de plusieurs sources pour commencer leurs traitements, nécessitent une opération de regroupement et de fusion de ses données. Avec la combinaison de ses différents mécanismes, le degré du parallélisme peut être maximisé sur différents niveaux d'exécution.

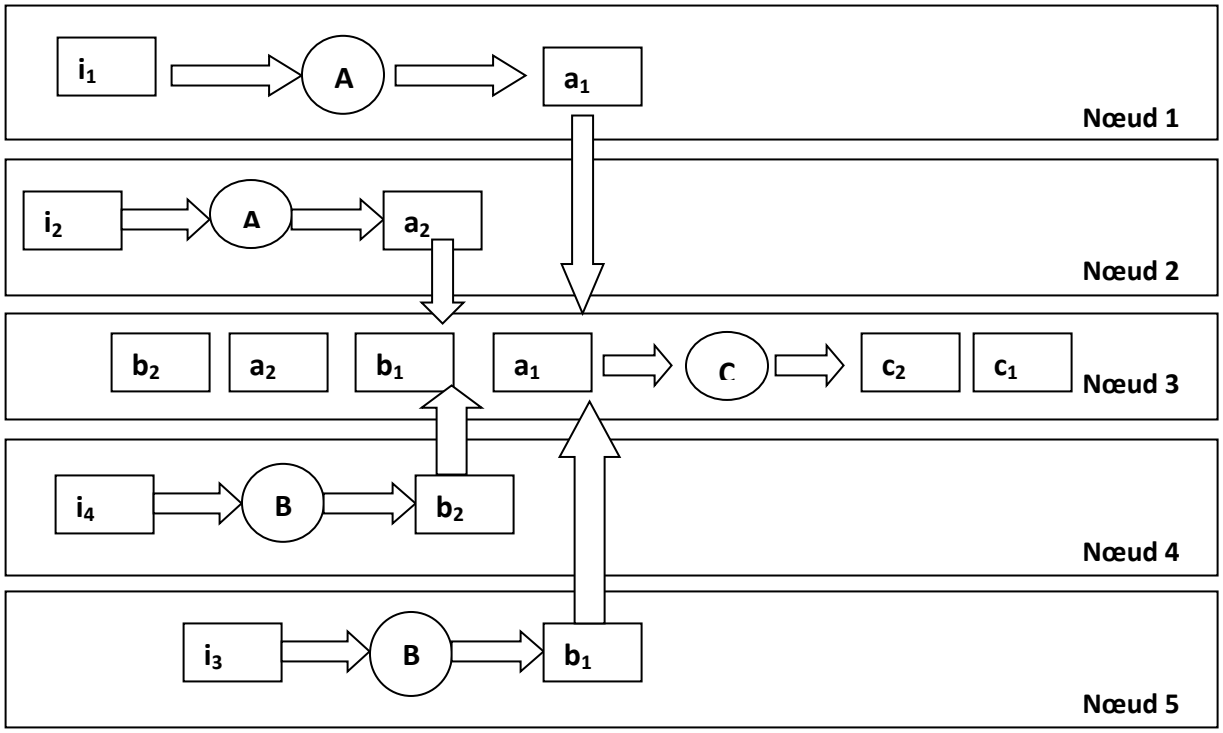

Figure 1-9 : Parallélisation hybride

7. Différents types de calcul parallèle

Selon la nature du problème nous distinguons trois catégories de calcul des tâches qui sont : High Performance Computing (HPC), High Throughput Computing (HTC), et le Many Task Computing (MTC).

Ils n'utilisent pas un modèle d'application spécifique, mais fournissent un point de vue général des caractéristiques du problème et imposent implicitement les exigences sur l'infrastructure et les middlewares.

7.1 High Performance Computing

Dans le modèle *High Perfomance Computing (HPC)* nous utilisons des installations de calcul distribué pour la résolution des problèmes qui ont besoin de puissance de calcul. Historiquement les supercalculateurs et les clusters étaient désignés spécialement pour supporter les applications HPC qui étaient développés pour résoudre des problèmes principalement dans la science et l'ingénierie.

Le profile des applications HPC est constitué par un ensemble de tâches de calcul intensif qui ont besoin d'être traités dans une courte période de temps. C'est récurrent d'avoir des tâches parallèles et étroitement couplé, ce qui demande une interconnexion réseaux avec une faible latence afin de minimiser le temps d'échange de données. Les systèmes HPC sont évalués par la métrique *floating-point operations per second (FLOPS)*.

7.2 High Throughput Computing

Le modèle *High Throughput Computing (HTC)* est l'utilisation des ressources calculatoires distribuées pour des applications qui ont un besoin d'énorme puissance de calculs pour sur une longue période de temps. Les systèmes HTC ont besoin d'être robuste et de fonctionner de manière fiable sur une longue durée. La grille de calculs composée de ressources hétérogènes (clusters, workstations et des machines individuelles bénévoles) a été utilisée pour le traitement HTC. Les applications HTC sont constituées d'un grand nombre de tâches dont l'exécution peut durer pendant un temps considérable comme les simulations scientifiques ou l'analyses statistiques. Il est assez fréquent d'avoir des tâches indépendantes qui peuvent être planifié sur des ressources distribuées car elles n'auront pas besoins de communiquer. Les performances des systèmes HTC sont mesurées par le nombre de jobs accomplis pendant une période de calcul (mois ou année).

7.3 Many Task Computing

Le modèle *Many Task Computing* essaie de remplir le vide qui existe entre le HPC et le HTC. Le MTC est similaire au HTC, mais il se focalise sur l'utilisation de divers ressources de calcul sur une courte période de temps afin d'accomplir plusieurs tâches de calculs. Le MTC a besoin d'une puissance de calcul de haute performance pour plusieurs activités différentes couplé à travers un système de fichiers. Ce qui caractérise le plus le MTC c'est l'hétérogénéité des tâches (petite ou large, mono processeur ou mutliprocesseur, calculs intensifs ou données intensives, statique ou dynamique, ...).

Les applications MTC sont faiblement couplées et sont en communication intensive utilisant l'interface d'envoi de messages que nous trouvons dans le domaine du HPC.

8. Conclusion

Dans ce chapitre nous avons présenté une étude sur les workflows scientifiques et une comparaison avec les workflows d'entreprises et l'influence de ses derniers dans l'émergence des workflows dans le domaine scientifique. Nous avons par la suite étudié la conception et la modélisation des workflows scientifiques.

Nous avons étudié les systèmes de gestion des workflows scientifiques, leurs architectures et le rôle qu'ils occupent dans les projets dans différents domaines de recherches.

Nous avons étudié les différents aspects de la parallélisation dans les workflows scientifiques et comment chaque type de parallélisation est appliqué, parallélisation des tâches, sur les données, en pipeline ou bien hybride.

Dans le chapitre suivant, nous allons présenter les grilles de calculs, leurs architectures, et leurs différents services qui sont présents sur cette infrastructure distribuée.

Chapitre 2 : Grille de calcul

1. Introduction

Le terme “grille” est apparu au milieu des années 1990 afin de distinguer une nouvelle infrastructure des systèmes distribuées destiné pour le domaine scientifique et de l’ingénierie. L’empreinte de distinction des grilles de calcul par rapport aux systèmes distribués est dans le partage des ressources dynamiques à grand échelle grâce au développement des performances des réseaux informatiques, des protocoles, des services, interfaces de programmation d’application et des kits de développement de logiciel. La grille de calcul est vu comme le successeur des systèmes distribuées, certains le voit comme un environnement de calcul de haute performance distribué, d’autre comme un système géographiquement distribué ou comme un système d’unification d’un nombre de ressources.

Avec l’extension des réseaux WAN (Wide Area Network) et l’augmentation sans cesse des besoins en ressources, il y a eu naissance de la deuxième génération qui est le Grid computing ou la grille de calcul, la répartition géographique des nœuds de la grille est complètement différente, elle est beaucoup plus vaste que pour les clusters. Bien souvent, les nœuds sont situés dans des organisations différentes. On se trouve dans un environnement qui a peu de chance d’être homogène. Ce type de système parallèle et distribué permet le partage, la sélection, et l’agrégation des ressources autonomes dynamiquement et distribués géographiquement [9]. Chacune des ces ressources a sa propre disponibilité, capacité, performance, coût, et utilisateurs, avec ses propres contraintes de qualité de service.

L’idée serait venue de trois personnes, du Docteur en mathématiques et en informatique Lan Foster (Directeur du Laboratoire National Argonne à Chicago aux Etats-Unis), de Carl Kesselman (chercheur en informatique à l’université de Californie du Sud) et de Steve Tuecke ingénieur en informatique au Laboratoire National Argonne. Ces trois sont surnommés The fathers of Grid (les pères de la grille), et sont à l’origine de l’une des plus importantes organisations pour la grille "The Globus Alliance" [10].

Le terme de Grid Computing (grille de calcul) fait son apparition au milieu des années 1990 et s’inspire d’électricité (power Grid). Le réseau électrique se borne à distribuer du courant. Le Grid Computing est lui beaucoup plus complexe de par la nature très variée des ressources et des services que l’on peut en extraire. Le but des grilles de calcul est de permettre la disponibilité de grandes quantités de ressources (calcul, stockage, service et application) sur les réseaux informatiques [11].

Cette idée de mutualisation des ressources informatiques s'est développée dans le milieu de la recherche scientifique où depuis quelques années, les besoins de puissance de calcul et de traitement des données augmentent démesurément. Cette augmentation se produit alors que le prix des supercalculateurs continue de grimper au point de rendre leur achat trop onéreux. C'est donc en voulant faire des économies que les chercheurs ont décidé d'exploiter les ressources informatiques délaissées. Ils se sont alors aperçus qu'ils étaient capables de traiter pratiquement 240 Gigaflops, soit l'équivalent de quatre serveurs Sun Entreprise 10000, en reliant en interne, 2000 PC de type Pentium II cadencés à 166Mhz et une centaine de Pentium III. Il est assez rare qu'une personne disposant d'un PC l'utilise constamment, l'idée est de récupérer cette puissance dormante en exploitant les cycles de calcul inutilisés [11].

2. Environnements distribués conventionnel et les grilles de calcul

Afin de tirer profit des performances calculatoires des ressources non disponible localement, les applications distribuées sont composées de plusieurs processus coopératives. Le calcul de haute performance dans les environnements distribués se base sur des environnements de calcul par passage de message comme le PVM ou MPI ou avec des Framework émergeant nommé les grilles de calcul. La différence entre ses deux aspects est dans la façon avec la-quelle ils proposent leurs ressources aux utilisateurs.

Une application dans un environnement distribué conventionnel a un ensemble de nœuds de calcul, qui forme des machines virtuelles en concurrence. L'ensemble est constitué de station de travail, pc et même des supercalculateurs qui procurent des droits d'accès à ses ressources après authentification. En général l'utilisateur se connecte à un nœud et utilise les ressources qui appartiennent ou qui s'attachent à ce dernier localement. Par ailleurs, l'ensemble des nœuds virtuels est considère comme statique, la taille est souvent fixe entre 10 à 100 nœuds rarement ou ça change.

Les grilles de calcul sont basées sur le partage de ressources à grand échelle. Les grilles assument un ensemble de ressources virtuelles, bien que l'aspect de ressource calculatoire est présent comme dans les systèmes distribué, on trouve dans les grilles de calcul d'autres aspects de ressources comme le stockage, le réseau, les données et les applications. Toutes ses ressources sont géographiquement distribuées sur de multiples domaines administratifs. Dans la grille, l'ensemble de ressources virtuelles sont dynamiques et divers, et les ressources peuvent être ajoutées et enlevées à tout moment.

3. Architecture de la grille de calcul

Nous allons présenter l'architecture de la grille proposée par Foster et al [63] en modèle de couche.

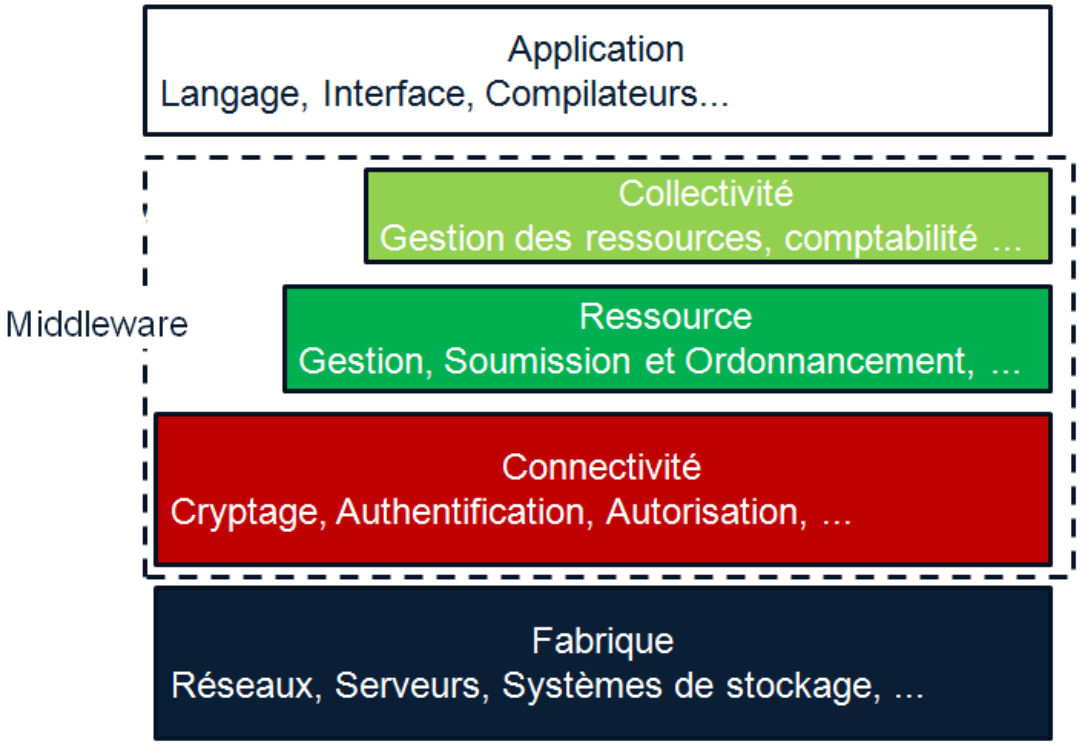

Figure 2-1 : Architecture de la grille de calcul

3.1 La couche fabrique

C'est la couche qui se situe en bas de l'échelle, et qui permet l'accès aux ressources locales comme le réseau, le stockage de données, le cpu et les différents types de ressources disponibles sur la grille.

3.2 La couche connexion

Cette couche regroupe les protocoles d'authentification nécessaires aux transactions ayant besoins d'exploiter les ressources de la grille. Elle a un rôle très important dans l'appartenance aux organisations virtuelles et les ressources qui sont mises à disposition de cette dernière.

3.3 La couche ressources

Cette couche contrôle l'accès aux ressources et au monitoring. Elle accède et contrôle les ressources locales.

3.4 La couche collective

Cette couche assure la gestion et l'interaction des ressources globales de la grille. Cette couche gère les interactions entre les différentes ressources de la grille.

3.5 La couche applicative

Cette couche est située en haut de l'architecture, elle regroupe l'ensemble des interfaces donnant accès aux ressources de la grille.

4. Le Middleware

Le middleware est une couche logicielle intermédiaire qui fait abstraction entre la couche application et le réseau en permettant le dialogue entre les applications hétérogènes. D'après Tan [64], un middleware est une collection d'API, protocoles et logiciels qui permettent l'exploitation des ressources de la grille et offrent une transparence de l'environnement hétérogène et de la distribution du système.

4.1 Le middleware dans la couche protocolaire

La couche fabrique est la couche la plus basse en interaction directe avec les ressources tandis que la couche la plus haute est composé d'applications utilisateurs.

Le middleware se trouve entre ces deux couches. Le middleware est divisé en deux niveaux : le corps du middleware et le middleware niveau utilisateur.

- Le corps du middleware offre les services de base comme l'allocation des ressources, le transfert de fichier, la sécurité et l'exécution des tâches.

- Le niveau utilisateur comprend les outils de programmation, les environnements de développement et les applications de planification de tâches ...etc.

5. Les organisations virtuelles

Le partage des ressources à grande échelle nécessite une gestion claire des droits associés aux ressources et à ceux qui peuvent les utiliser. Les organisations virtuelles ont été adoptées dans le but de regrouper les utilisateurs qui ont des droits et des besoins communs.

Fellensteine et al définissent une organisation virtuelle comme étant : des entités logiques, limitées dans le temps, créées dynamiquement dans le but de résoudre un problème spécifique, en fournissant et allouant les ressources à la demande [65].

L'organisation virtuelle prend en charge les aspects relatifs à la sécurité. Elle définit les conditions d'accès et la politique d'utilisation des ressources disponibles sur la grille tels que les cycles CPU, les logicielle accessibles, les périphériques, les capacités de stockage, etc.

Les organisations virtuelles peuvent être constituées juste le temps nécessaire à la résolution d'un problème spécifique. Un besoin de ressources temporaire par un groupe d'individus peut être résolu par la constitution d'une organisation virtuelle. Cela va permettre d'éviter d'investir dans une infrastructure informatique coûteuse.

6. Fonctionnement de la grille

Afin de comprendre le fonctionnement de la grille, nous allons présenter les différents éléments composants et intervenants dans la prise en charge d'un job sur la grille. La définition de ses acteurs va nous faciliter la compréhension par la suite lors de la soumission d'un job dans la grille.

- **Computing Element "CE"** c'est le point d'accès unifié aux ressources de calcul, aux nœuds travailleurs (WN) qui sont exploités par la grille afin d'exécuter les jobs. Le CE se charge de la gestion des jobs qui lui sont attribués à travers des files d'attente. Le CE peut être vu comme un cluster de ressources de la grille qui sert à exécuter les jobs. Chaque CE est composé de trois éléments de base : notamment le portail de la grille (Grid Gate), le système de gestion des ressources locales (LRMS) et les nœuds travailleurs (WN).

- **Worker Node "WN"** c'est un groupe de machines sur lesquelles les jobs transmis par les CE vont être exécutés.

- **Resource Broker "RB" ou Workload Manager "WM"** il correspond les besoins des utilisateurs avec les ressources disponibles sur la grille.

- **Workload Management System "WMS"** est un ensemble de services ayant la responsabilité de trouver le meilleur élément de calcul disponible où soumettre les jobs utilisateur de manière transparente.

- **Storage Element "SE"** il permet la gestion du stockage des fichiers dans la grille et offre les méthodes de localisation des fichiers.

- **Information System "IS"** il a pour mission de collecter l'ensemble d'information relative aux ressources précisant les caractéristiques et l'état des éléments de calcul et de stockage (CE, SE).

- **User Interface "UI"** c'est l'interface via laquelle l'utilisateur accède aux services de la grille.

- **Logging and BookKeeping "LB"** c'est un service qui collecte et stocke tous les événements relatifs à la gestion des jobs. il pourra ainsi informer l'utilisateur sur l'évolution de son job en lui fournissant l'étape par laquelle il est entrain de transiter; transfert vers le CE, en cours de soumission, terminé, … etc.

La soumission et l'exécution d'un job dans la grille passe par plusieurs étapes basique qu'on peut qualifier de cycle de vie qui sont comme suit :

1. Afin de pouvoir utiliser la grille de calcul, l'utilisateur est tenu d'acquérir un certificat auprès de l'autorité de certificat (CA). Après il faut adhérer a une organisation virtuelle (VO) et obtenir un compte sur l'UI. L'utilisateur est tenu de se connecter à travers son UI et crée un proxy qui lui permettra d'interagir en toute sécurité avec le système;
2. A l'aide d'un script (JDL) compréhensible par le logiciel de la grille (middleware), l'utilisateur va pouvoir soumettre ses jobs sur la grille;
3. L'utilisateur soumet le job au WMS via l'UI. Le WM va analyser le fichier (.JDL) et voir quelles sont les ressources dont le job en a besoin. Il va trouver un élément de calcul CE qui est libre et qui satisfait les exigences du job en terme de temps d'exécution, de taille de mémoire, d'application installées,..... Si le job demande que des fichiers soient accessibles sur le CE, le WM va interroger les éléments de stockage SE pour déterminer le CE qui est connecté avec le SE correspondant. Il transmet alors le job au CE.
4. Le job ainsi que l'InputSandBox sont transférés au CE qui prend en charge le job dans sa file d'attente;
5. Le CE envoie le job sur un ou plusieurs machines ou WN(s) libres;
6. Lorsque le job est terminé, les fichiers produits par celui-ci sont disponibles sur le LRMS (Local Resource Management System). A la fin du job le WN va recopier les fichiers de données ayant été traités sur un SE. Il va généralement renvoyer le fichier log au CE qui le transmettre au WM. Le WM est averti que le job s'est terminé;
7. Le WM va stocker le statut du job pendant tout la durée de vie du job et stocke également le fichier log, il récupère les fichiers de sortie dans l'OutputSandBox;

8. Le WM envoie les résultats à l'utilisateur via l'UI;
9. L'utilisateur peut interroger le WM à tout moment pour savoir l'état de son job par l'intermédiaire du Logging and Bookkeeping Service (LB). LE LB conserve une trace de l'exécution des jobs.

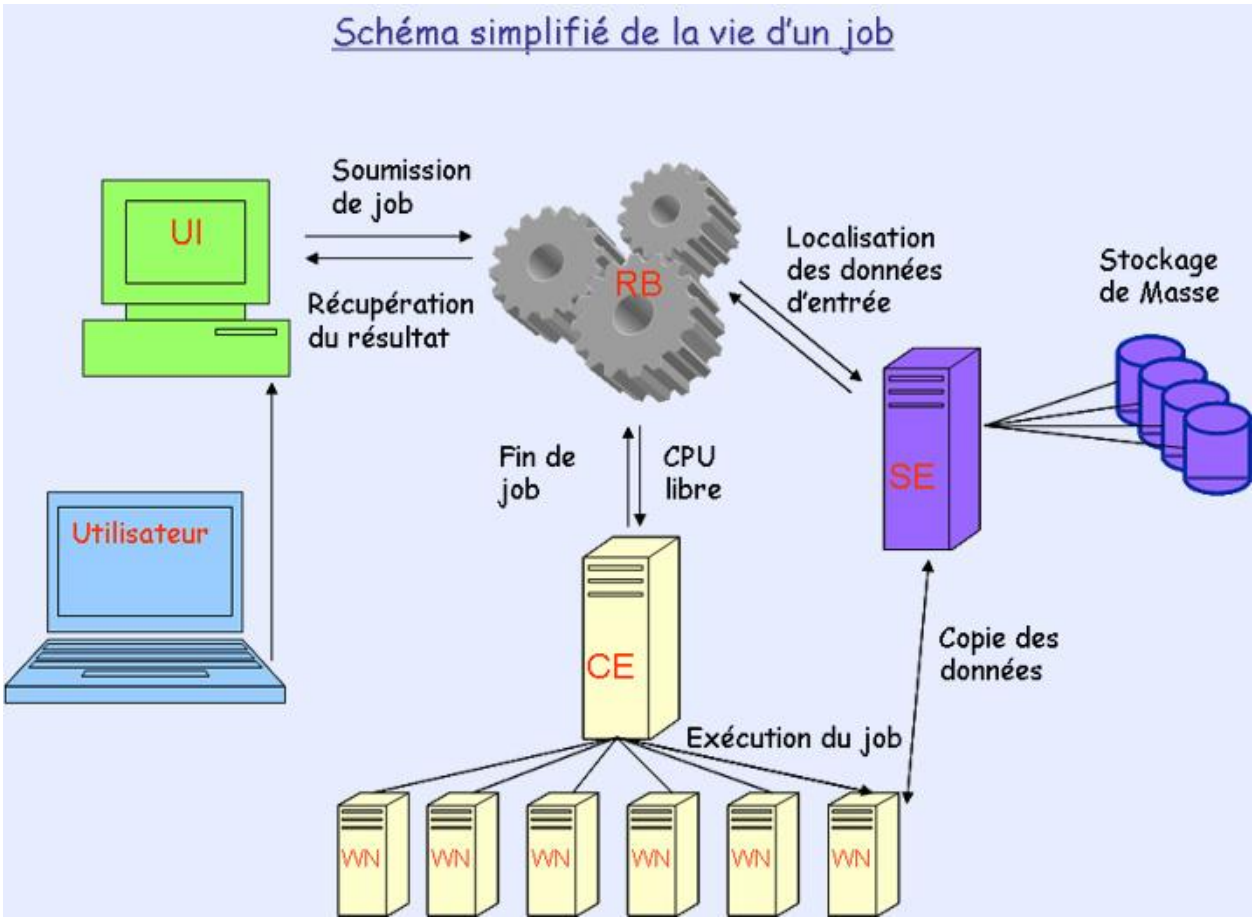

Figure 2-2 : Cycle de vie d'un job dans la grille de calcul

7. Les différents états d'un job soumis à la grille

Un job soumis à la grille peut passer par différents état suivants :

- ➔ **SUBMITTED** : le job a été soumis pour exécution par l'utilisateur mais il n'a pas encore été traité par le serveur ;
- ➔ **WAITING** : le job a été accepté par le serveur réseau mais n'a pas encore été pris en charge par le Workload Manager WM ;
- ➔ **READY** : le job a été assigné à un Computing Element CE mais il n'a pas encore été transmis a celui-ci ;
- ➔ **SCHEDULED** : le job est en liste d'attente sur le Computing Element ;

- ➔ **RUNNING** : le job est en cours d'exécution sur le Worker Node WN ;
- ➔ **DONE** : l'exécution du job est terminée ;
- ➔ **ABORTED** : le job a été arrêté par le WMS faute de temps d'exécution trop long ou expiration du certificat du proxy, etc.
- ➔ **CANCELED** : le job a été arrêté par l'utilisateur ;
- ➔ **CLEARED** : l'OutputSandBox a été transféré vers l'interface utilisateur.

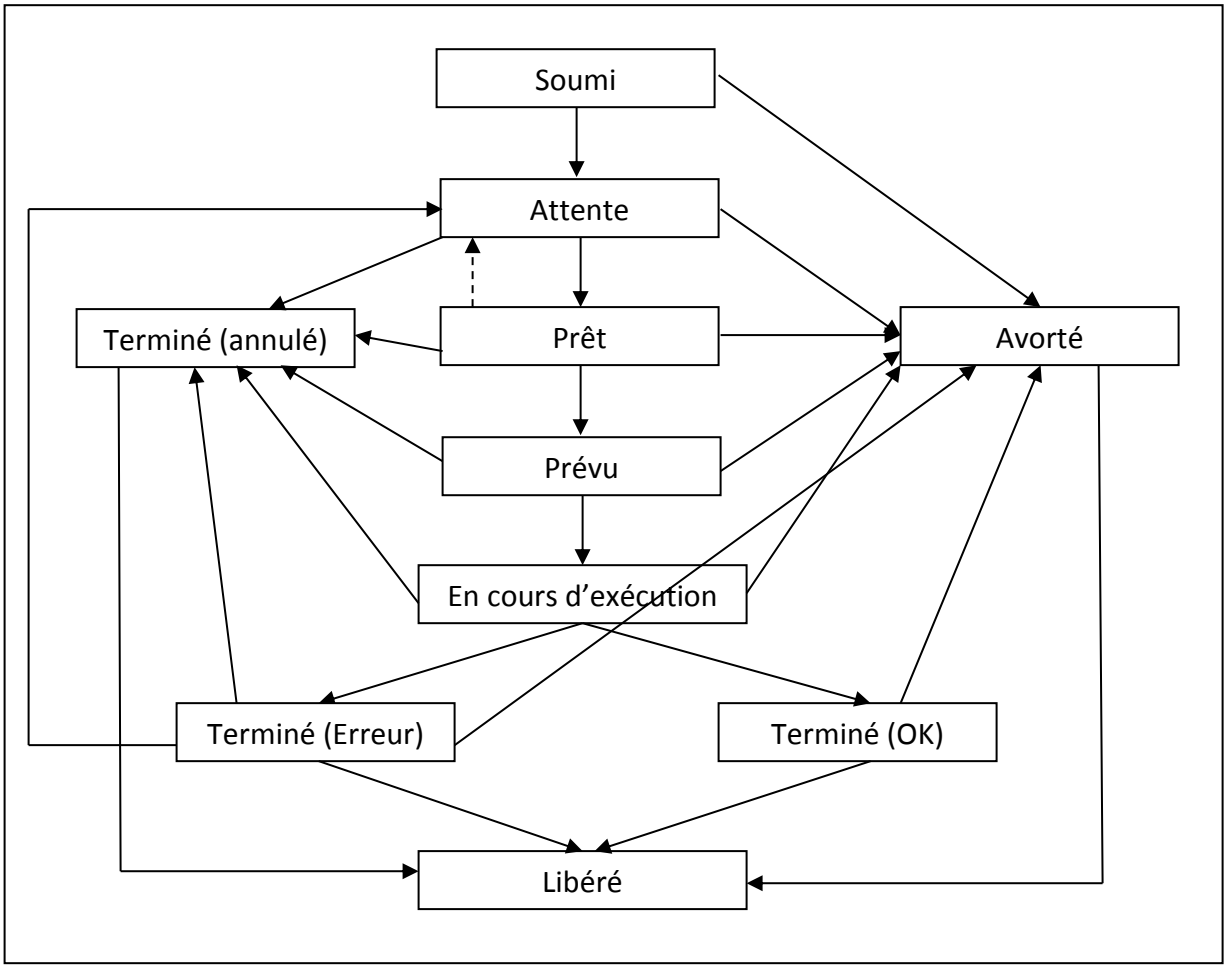

Figure 2-3 : Etats d'un job

8. Le langage de description de job (JDL)

Le JDL est un langage de haut niveau utilisé pour paramétrer un job avec des paramètres spécifiques à l'utilisateur. Le JDL est utilisé par le WMS pour comprendre la spécification du job afin de trouver les ressources de la grille qui correspondent le mieux [66].

Le fichier JDL est une description du job qui est essentiellement composé de lignes ayant un format semblable à : **Attribut = Expression.**

Les expressions peuvent être de longueur variable, mais doivent se terminer par un point-virgule (;). les caractères comme les guillemets doivent être précédé par une barre oblique inverse (\") car le caractère apostrophe n'est pas valide dans le JDL. Les commentaires doivent être précédée par // ou par \# ou par le couple /* et */ pour les commentaires sur plusieurs lignes. Les espaces vides après les points-virgules rendent le JDL invalide.

- **Simple Job**

Un job simple est de type Normal. Pour exécuter un programme sur la grille, on peut écrire uneghier JDL de la manière suivante :

```
[
        JobType = "Normal";
        RetryCount = 0;
        ShallowRetryCount = 3;
        Executable = "compress \_hgdemo.sh";
        Arguments = "argA argB";
        InputSandBox = {"fille:///home/compress_hgdemo.sh"}
        StdOutput = "std.out";
        StdError = "std.err";
        OutputSandBox ={"std.out","std.err", "fileoutput"};
        DataAccessProtocol = {"gsiftp", "http"};
]
```

Le *RetryCount* et *ShallowRetryCount* spécifient le nombre maximum de fois qu'un job est soumis à nouveau avant d'être abandonné. S'il n'est pas défini, il prend les valeurs par défeaut dans le WMS défini par *MaxRetryCount* et *MaxShallowRetryCount*.

L'attribut *Executable* spécifie la commande à exécuter par un job, seul le nom de l'exécutable est nécessaire.

Le *InputSandBox* est l'attribut pour stocker le chemin de commande sur l'interface utilisateur de sorte que le système de grille sait où chercher le fichier exécutable avec précision.

Le *StdOutput* et *StdError* représentent les noms des fichiers contenant la sortie standard et l'erreur standard une fois le job terminé.

Le *OutputSandBox* contient les fichiers qui sont retournés par la grille une fois l'exécution du job est terminé.

La *DataAccessProtocol* définit le protocole utilisé pour le transfert de fichiers et la récupération des sorties de job.

- **Job Paramétrique**

Un job paramétrique est un job de type "Paramétric" où un ou plusieurs de ses attributs sont paramétrés.

```
[
        JobType = "Parametric";
        Parameters = (N); (ou bien: Parameters = {EARTH, MOON, MARS} ;)
        ParameterStart = 1;
        ParameterStep = 1;
        RetryCount = 0;
        ShallowRetryCount = 3;
        Executable = "compress\_hgdemo.sh";
        Arguments = "argA_PARAM_ argB_PARAM_";
        InputSandbox = {"file:/// home/compress_hgdemo.sh"};
        StdOutput = "std .out ";
        StdError = "std.err ";
        OutputSandbox = {"std.out", "std.err", "fileoutput"};
        DataAccessProtocol = {" gsiftp", "https "};
]
```

Dans un job paramétrique, le fichier JDL a trois attributs supplémentaires: Parameters, ParameterStart et ParameterStep. Les paramètres ParameterStart et ParameterStep sont mis dans le JDL que si Parameters est un nombre. _PARAM_ sera une série de nombres ou de chaines de caractères selon le type de Parameters.
Parameters définit le nombre maximum de l'attribut _PARAM_, tandis que ParameterStart définit le nombre de départ pour _PARAM_ et ParameterStep définit l'incrément de _PARAM_.

- **Job DAG**

Un job DAG est un ensemble de jobs où les inputs, les outputs ou l'exécution d'un ou de plusieurs jobs dépendent d'autres jobs. Les dépendances sont représentées par des graphes orientés acycliques (Directed Acyclic Graphs : DAG), où les nœuds sont des

jobs, et les arcs identifient les dépendances. L'attribut ―nodes― est le noyau du job DAG.

```
[
    Type = "dag";
    max_nodes_running = 4;
    nodes = [
        nodeA = [
            file ="nodes/nodeA.jdl" ;
                ];
        nodeB = [
            file ="nodes/nodeB.jdl" ;
                ];
        nodeC = [
            file ="nodes/nodeC.jdl" ;
                ];
        nodeD = [
            file ="nodes/nodeD.jdl";
                ];
    Dependencies = {
        {nodeA, nodeB},
        {nodeA, nodeC},
        {{nodeB,nodeC}, nodeD},
                }
         ];
]
```

- **Job Collection**

Un job collection est un ensemble de jobs indépendants que l'utilisateur veut soumettre et monitorer comme une unique requête. Les jobs d'une collection sont soumis comme des nœuds DAG sans dépendances. Tous les nœuds partagent le même InputSandbox. Le JDL est une liste de classes, qui décrit les sous-jobs.

```
[
    type = "collection";
    InputSandbox = {"date.sh"};
    RetryCount = 3;
    nodes = {
        [
        file ="jobs/job1.jdl" ;
         ],
```

```
        [
            [
                Executable = "/bin/sh";
                Arguments = "date.sh";
                StdOutput = "date.out";
                StdError = "date.err";
                OutputSandbox ={"date.out", "date.err"};
            ]
        ],
    [
        file ="jobs/job3.jdl" ;
        ]
    };
]
```

- **Job MPI**

L'exécution des jobs parallèles est un point essentiel pour l'informatique et les applications modernes. La librairie la plus utilisée pour supporter les jobs parallèles est la MPI (Message Passing Interface). A ce jour, les jobs parallèles peuvent être exécutés que seulement sur un même CE. Cependant, plusieurs projets sont impliqués dans les études concernant la possibilité d'exécuter des jobs parallèles sur des WNs appartenant à des CE différents.

```
[
    Type = "Job";
    JobType = "OPENMPI";
    NodeNumber = 2; # Le nombre de CPU qui seront utilisés
    Executable = "cpi";
    StdOutput = "cpi.out";
    StdError = "cpi.err";
    InputSandbox = {"cpi"};
    OutputSandbox = {"cpi.err","cpi.out"};
    RetryCount = 3;
]
```

9. Avantages et inconvénients de la grille

Les grilles de calcul sont des infrastructures qui ont leurs points forts et leurs points faibles, nous allons brièvement voir ses différents points :

9.1 Avantages

➢ La résolution des problèmes scientifiques complexes en allant toujours vers un temps de plus en plus réduit

➢ La collaboration dans l'accomplissement des tâches et l'orchestration dans la division des différentes tâches sur les intervenants et la collecte des résultats

➢ L'optimisation dans l'utilisation des ressources et la tendance a l'exploitation maximale de ses derniers

➢ La flexibilité de l'infrastructure ou les composants de celle-ci sont modulaires

➢ Les politiques sont gérées par les middlewares de la grille

9.2 Inconvénients

Comme pour toute chose y a des inconvénients dans l'infrastructure de grille qui sont les suivants :

➢ La difficulté de maîtriser et d'apprentissage sur les grilles pose une certaine crainte pour les scientifiques

➢ Les middlewares sont toujours en cour de développement et d'évolution

➢ La latence dans certaines infrastructures réseaux

➢ Le partage des ressources et la non équitabilitée dans l'accès aux ressources

➢ Un des gros problèmes de ce système est l'impossibilité de prévoir à l'avance les ressources réelles qui seront présentes et disponibles à un instant T. On ne peut donc savoir exactement combien de temps le calcul va prendre.

10. Conclusion

Les grilles de calcul est une technologie qui est maintenant mature mais qui ne cesse de gagner encore plus de maturité avec les avancés percées notamment dans l'infrastructure réseaux et systèmes et la collaboration entre les différentes équipes de recherches.

Les grilles de calcul contribuent à satisfaire la demande toujours croissante en ressources de calcul de la communauté de recherche scientifiques pour la résolution des problèmes complexes.

Par la suite nous allons aborder les technologies qui permettent aux chercheurs scientifiques d'exploiter toute la puissance de calcul de la grille de façon la plus simple.

Chapitre 3 : Les portails scientifiques

1. Introduction

La besoin de collaborer entres les scientifiques dans divers projets a poussé à utiliser les solutions informatiques où leurs puissances résident dans le partage des ressources informatiques.

Les portails scientifiques ou les "Science Gateway" ont apporté leurs contributions en facilitant l'accès sécurisé, l'exécution des scripts et les interfaces de lignes de commandes qui sont en générale difficile à maitriser par la communauté scientifique. Les portails scientifiques apportent une puissance dans l'authentification des utilisateurs, le transfert des données et la facilité dans la gestion de la charge de travail à des utilisateurs non-expert.

2. Portail scientifique

Le portail scientifique est un ensemble d'outils, d'applications et des données qui sont intégrés par l'intermédiaire d'un portail ou une suite d'application, généralement via une interface graphique utilisateur pour répondre aux besoins d'une communauté spécifique. Il permet à une communauté entière d'utilisateurs associés à une discipline commune d'utiliser les ressources nationales par le biais d'une interface commune qui est configuré pour une utilisation optimale.

Les chercheurs peuvent se concentrer sur leurs objectifs scientifiques et ne plus se soucier sur les problèmes liés à l'infrastructure informatique. Le portail scientifique procure un environnement de travail qui favorise la collaboration et l'échange d'idée entre les chercheurs.

Les portails scientifiques se sont imposés par la facilité avec la-quelle ils gèrent les utilisateurs, la disponibilité d'outils performants qui aide au développement et le déploiement d'applications web qui s'exécutent sur les infrastructures de la grille.

Les portails scientifiques cachent touts les détails des systèmes de la grille avec des interfaces de haut niveau qui peuvent être intégrés au middleware. Ils proposent des interfaces basées sur l'accès aux services suivants :

- La définition d'un environnement de grille;
- La gestion des accès sécurisés à la grille;
- Le contrôle et l'exécution des applications sur les ressources de la grille;
- La création et la modification des applications simples et basées sur les workflows;

- Le monitoring et le suivi des applications.

2.1 Fonctions des portails scientifiques

Les fonctionnalités des portails scientifiques « Science Gateway » sur les infrastructures grille et leurs interfaces utilisateurs consistent à:

- définition des requêtes des utilisateurs
- initiation des workflows de la grille en fonctions des demandes des utilisateurs
- le contrôle et l'inspection du fonctionnement des workflows
- l'interaction avec les éléments de composition du workflow et du processus d'exécution
- téléchargement et envoie de fichiers de/vers la grille, et fournir une assistance lors de l'orchestration des workflows.

3. Exemple de portails scientifiques

Les portails scientifiques ou "Science Gateway" ont émergé dans le domaine de la recherche scientifique est ce grâce à leurs simplicités et facilités qu'ils offrent aux chercheurs pour accéder aux différents ressources mises à leurs dispositions afin de gagner plus de temps dans la résolution des problèmes rencontrés.

Nous allons maintenant discuter brièvement les différents portails scientifiques utilisés par la communauté de recherche :

3.1 GISELA

Le portail scientifique GISELA est un environnement qui comprend un ensemble d'outils, des données et des applications informatiques avancées disponibles pour les communautés de recherche en Amérique latine. A travers une interface simple et flexible pour répondre aux besoins des communautés de recherche en Amérique latine à collaborer les uns avec les autres et avec le reste du monde. Son infrastructure informatique comprenant les centres de données distribués et interconnectés par des réseaux de recherche et d'enseignement nationaux (NREN). Les chercheurs utilisent l'e-infrastructure GISELA d'une manière plus facile sans certificats numériques personnels. Avec juste un clic, les chercheurs peuvent inscrire à un compte personnel.

3.2 VIP (Virtual Imaging Platform)

Le portail VIP/Gate-Lab39 est une plateforme ouvertement accessible pour les applications encapsulées comme des workflows qui s'exécutent de manière transparentes sur les ressources de calcul distribué de l'organisation virtuelle (VO) biomed dans l'infrastructure EGI (European Grid Infrastructure). En VIP, les utilisateurs ont la possibilité de s'authentifier, lancer l'exécution d'une application, la suivre ou bien accéder à l'historique et aux résultats des exécutions précédentes. Le transfert de données vers et depuis les éléments de stockage est géré aussi par la plateforme.

Le système de gestion de tâches utilise des tâches pilotes gérées par le service DIRAC40, qui permet de répartir la charge de travail sur plusieurs sites et d'obtenir un meilleur résultat en termes de support aux utilisateurs. De nouvelles fonctionnalités, telles que des catalogues de modèles et des ontologies, ont été récemment ajoutées pour faciliter l'intégration et l'échange de modèles et de nouvelles applications.

3.3 WS-PGRADE

Le Portail WS-Pgrade/gUSE est également une solution de portail basée sur un Framework offrant une interface générique pour créer et exécuter des Workflows sur différentes infrastructures, y compris les clusters, les grilles, et les Clouds. Il prend également en charge l'étude des paramètres des Workflows. Les applications sont décrites comme des Workflows en utilisant son propre langage de workflow basé sur XML. L'authentification est basée sur login et mot de passe (ou issue de fédérations d'identité) et mappé à un certificat robot lors de l'exécution dans l'infrastructure et est basé sur des clés en mode SSH pour le cas du Cluster. C'est est un portail qui tente de résoudre le problème d'interopérabilité au niveau du workflow avec un grand succès. Cela signifie que les composants d'un workflow peuvent être exécutés simultanément dans plusieurs grilles

4. Conclusion

Les portails scientifiques jouent un rôle important dans l'avancée de la recherche en simplifiant l'utilisation des ressources proposées par l'infrastructure de la grille de calcul et en cachant toute la complexité du développement des applications sur cette dernière.

Chapitre 4 : Conception

1. Introduction

Nous avons commencé par comprendre les workflows scientifiques et l'importance qu'ils occupent dans le domaine de la recherche scientifique; nous avons ensuite enchaîné avec l'étude des infrastructures des grilles de calcul et toute la puissance qu'elles apportent.

Nous avons aussi vu que ces infrastructures sont exploitées à travers les portails scientifiques afin de permettre aux utilisateurs non expérimentés d'exploiter des ressources multiples, rapidement et de manière intuitive.

Notre travail va se focaliser sur l'implémentation d'une solution sur un portail scientifique qui va assurer l'exécution d'un workflow scientifique sur une architecture de grille de calcul en exploitant le paradigme par passage de messages pour l'exécution d'un workflow scientifique.

2. Conception de l'approche proposée

Nous présentons dans la section suivante notre approche proposée qui définie trois couches de notre architecture qui sont : la couche représentation, la couche métier et la couche infrastructure.

2.1 Couche présentation

C'est l'interface graphique du portail scientifique à partir de laquelle les utilisateurs vont pouvoir exécuter, suivre et récupérer les résultats de leurs applications. La couche présentation envoie les requêtes des utilisateurs vers la couche métier et elle retourne les résultats d'exécution à l'utilisateur. L'utilisation du portail scientifique via l'interface graphique ne sera possible qu'après authentification et avoir l'autorisation d'exploiter les ressources calculatoires.

2.1.2 Authentification et autorisation

Le besoin le plus important dans un portail scientifique est de faciliter l'accès aux ressources de calcul et de stockages distribués pour le plus grand nombre possible d'utilisateurs non experts du domaine informatique à travers des applications spécifiques et bien définis. Afin de répondre à cette exigence, les mécanismes d'authentification et d'autorisation ont été conçus pour offrir un accès facile aux applications tout en préservant le degré de sécurité exigé.

Les données sont transférées à partir de l'interface web vers les ressources physiques de la grille, limitant ainsi l'interaction de l'utilisateur à la couche présentation.

Le composant le plus haut de la hiérarchie doit être intégré dans le portail scientifique de manière à supporter le mécanisme de "Single Sign On" qui permet à l'utilisateur d'accéder à plusieurs applications informatiques (ou sites web sécurisés) en ne procédant qu'à une seule authentification.

Lorsqu'un utilisateur tente d'utiliser l'une des applications disponibles sur le portail, il est redirigé vers un service de découverte (DS) dont il contient la liste de tous les fédérateurs d'identités (IdF) et fournisseurs d'identités (IdP) ou il peut choisir celui dans le-quel il est membre. Le IdP identifie l'utilisateur, généralement à travers une paire de nom d'utilisateur et mot de passe. Si l'authentification par l'IdP a réussie, le contrôle revient au portail scientifique où l'autorisation de l'utilisateur est vérifiée.

Le gestionnaire des ressources de la grille accorde les autorisations à travers un registre (basé sur un LDAP centralisé) connecté au portail scientifique pour stocker et gérer les rôles et les privilèges. Les rôles des utilisateurs sont ensuite mappés sur ceux qui effectuent des transactions sur la grille en utilisant les fonctionnalités de Virtual Organisation Membership Service (VOMS).

Le VOMS stocke uniquement les certificats de robots qui sont utilisés pour signer des transactions grille avec leurs procurations. Les certificats de robot et leur utilisation dans le portail scientifique sont décrits dans la section suivante.

2.2 Couche métier

C'est la partie fonctionnelle du portail scientifique, elle décrit les traitements opérés sur les données en fonction des requêtes des utilisateurs effectués à travers la couche présentation. La couche métier offre des services applicatifs et métier à la couche présentation. Pour fournir ces services, elle s'appuie sur les données du système accessible via les services de la couche inférieur. En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés. Les différents services proposés afin d'accéder aux fonctionnalités de la grille de manière facile et transparente se basant sur les interfaces de programmation des applications (API). Le portail scientifique est ouvertement accessible aux utilisateurs pour exécuter leurs applications à base de job simple ou a base de workflow sur une infrastructure de grille de calcul.

2.2.1 Framework du portail scientifique

Le Framework est le noyau du portail scientifique, il permet grâce à des fonctionnalités et des outils l'accès plus facile aux utilisateurs sur différents niveaux.

Le Framework du portail scientifique se base sur la technologie du web et adopte des standards et des protocoles à travers leurs implémentations. Nous allons citer les plus utilisés :

- Standards "JSR 168" and "JSR 286" (connu aussi sous les standards: "portlet 1.0" et "portlet 2.0");
- Standards OASIS41 Security Assertion Markup Language (SAML) et ces implémentations Shibboleth et SimpleSAMLphp;
- Le protocole LDAP "Lightweight Direct Access Protocol" basée sur l'implémentation OpenLDAP42;

- Standards basée sur l'interface « Cryptographic Token » qui est le PKCS#11 et son implémentation Cryptoki, ou PKCS#11 est l'une des standards de la famille appelé: «Public-Key Cryptography Standards».
- Standards basée sur le Open Grid Forum (OGF) pour créer des APIs simples des applications Grid (SAGA) et leurs implémentations JSAGA.

2.2.2 Portlet

Une ‘‘Portlet’’ est un composant web unitaire s'intégrant au sein d'un conteneur (portail). Chaque Portlet est indépendante des autres et peut être imaginé comme une application WEB. Les Portlets se basent sur une API définie par la spécification JSR 168 ou JSR286 qui a pour objectif de standardiser les Portlets et d'assurer une interopérabilité entre les différents portails.

L'utilisation de standards de portlet permet de développer chaque application en tant que portlet et la déployer dans d'autres Science Gateway d'une manière simple. Une portlet est basée sur le modèle « MVC » (Model View Controller), un paradigme qui aide à garder le code de la portlet bien ordonnée.

2.2.3 Grid Engine

Le moteur de grille est utilisé pour effectuer les transactions de la grille. Le Grid Engine est un module générique de logiciel capable de relier la couche de présentation du portail scientifique avec les infrastructures de grille sous-jacente en utilisant des technologies standard. Il permet la création rapide de nouveaux portails

scientifiques en fournissant aux développeurs une interface simple et sans se soucier des spécificités du middleware.

Le Grid Engine représente le noyau du Framework incluant les aspects importants et fondamentaux pour l'exécution des applications (jobs). Il adopte a la base une API's de haut niveau orientée applications de grille qui est SAGA (Simple API for Grid Application) qui a été spécifié par le OGF (Open Grid Forum). La mise en œuvre de la JSAGA permet de créer une interface unique pour différents middleware et rend les portails scientifiques en mesure d'exploiter les ressources provenant de différentes infrastructures de grille.

Le Grid Engine est constitué des modules suivants :

- **L'interface du portail scientifique :** une interface cachant toute la complexité de l'infrastructure de la grille à la couche de présentation offrant des capacités supplémentaires convenablement conçus pour le web. Il permet aux développeurs de créer des portlets pour des applications dans un temps très court.

- **Job Engine :** Le job Engine reçoit des demandes pour soumettre des jobs à partir de l'interface du portail scientifique et prend soin des jobs jusqu'à ce qu'ils soient exécutés correctement. Il offre toutes les opérations préliminaires nécessaires pour exécuter un job tel que l'association d'un proxy pour un job, la recherche d'un gestionnaire de ressources de job, la satisfaction des besoins des utilisateurs particuliers, et ainsi de suite.

Le job Engine dispose d'une interface au serveur eToken pour créer des procurations des certificats de robot pour être associés à un job. Il offre également une capacité de tolérance aux pannes en soumettant un job à l'infrastructure de grille jusqu'à ce qu'il soit correctement exécuté protégeant l'utilisateur des défaillances de l'infrastructure.

- **Data Engine :** C'est un module permettant aux utilisateurs de tirer pleinement parti des services de gestion de données de la grille. Il mappe les opérations de données à des fonctions de JSAGA. Il est responsable de la collecte des requêtes issues de portail scientifique, afin de réaliser les transferts directs avec les éléments de stockage de la grille. Il offre un système de gestion de fichiers comme une interface web à travers des services de gestion de données qui comprennent des modules supplémentaires, tels que le service Builder.

2.3 Couche infrastructure

La couche infrastructure englobe tous les éléments qui vont interagir entre les services de la couche métier et l'infrastructure matérielle déployée pour la grille. Cette couche est représentée par plusieurs éléments qui sont :

2.3.1 L'infrastructure de calcul

L'infrastructure de calcul comporte les services nécessaires au fonctionnement de la grille comme les : VOMS, WMS, SE, CE, WNs, Myproxy ... etc.

2.3.2 Certificats Robot

La gestion des certificats personnels pour accéder aux infrastructures est difficile pour un utilisateur non expert et représente un facteur limitant à la propagation rapide de cette technologie dans de nouveaux domaines scientifiques où l'informatique n'est pas une connaissance de base.

Les avantages apportés par ce nouveau type de certificats numériques sont multiples. Pour des soucis de sécurité, les certificats de robots sont généralement stockés à bord de dispositifs inviolables tels que les cartes à puce. Cela améliore la sécurité et évite toute utilisation frauduleuse des clés privées. Un serveur multi-thread, appelé serveur eToken, est installé et configuré pour gérer une liste de certificats de robot (un certificat par application) stockés dans différentes cartes à puce USB eToken PRO de 32/64 KB.

Le portail scientifique envoi une requête d'ID au serveur eToken. Si le portail scientifique est autorisé, et en tenant compte des informations disponibles dans le HashMap, le serveur eToken envoi au portail scientifique un nouveau certificat de procuration, si la requete ID est introuvable ou la durée de vie de l'ancien proxy a expirée, il envoi aussi un nouveau certificat de procuration.

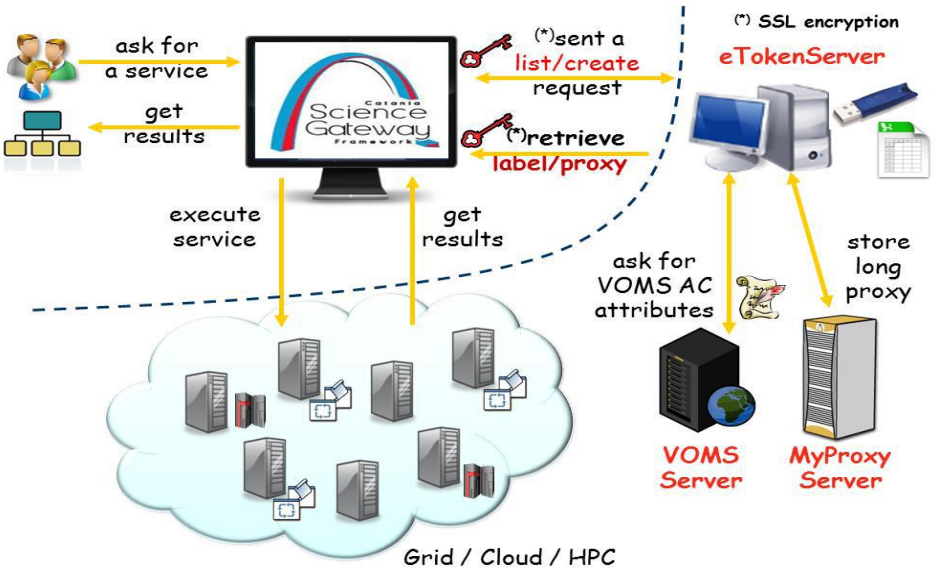

Figure 4-1 : Usage du serveur eToken avec la grille

2.3.3 La base de données

La base de données « UsersTrackingDB » permet le suivi des utilisateurs du portail scientifique et conserve les informations de chaque transaction sur la grille. Elle est chargée de contrôler le taux des interactions avec la grille (au total et par utilisateur) exécuté via le portail en implémentant des politiques de contrôle de flux (par exemple, le nombre maximum de Job soumis par les utilisateurs, etc.).

Une entrée d'un utilisateur sur la base de données "UsersTrackingDB" contient les éléments suivants :

- ➢ **Common Name** : le nom de l'utilisateur stocké dans la base du serveur LDAP
- ➢ **IP + Port** : l'adresse et le numéro du port TCP utilisé
- ➢ **Timestamp** : identificateur de l'opération de la grille avec la date et le temps.
- ➢ **Grid Interaction** : L'identificateur de l'interaction avec la Grid (soumission du Job "X" ; fichier upload/download).
- ➢ **Grid ID** : stocke le ID Interaction de la grille (Job Id pour le Job soumis et autres informations relatives aux transfères de données)
- ➢ **Robot Certificat** : Identifie le certificat Robot utilisé pour les opérations de la grille/

3. Soumission des jobs

A travers les API's du Grid Engine, nous allons voir les étapes nécessaires pour la soumission des jobs sur l'infrastructure de grille.

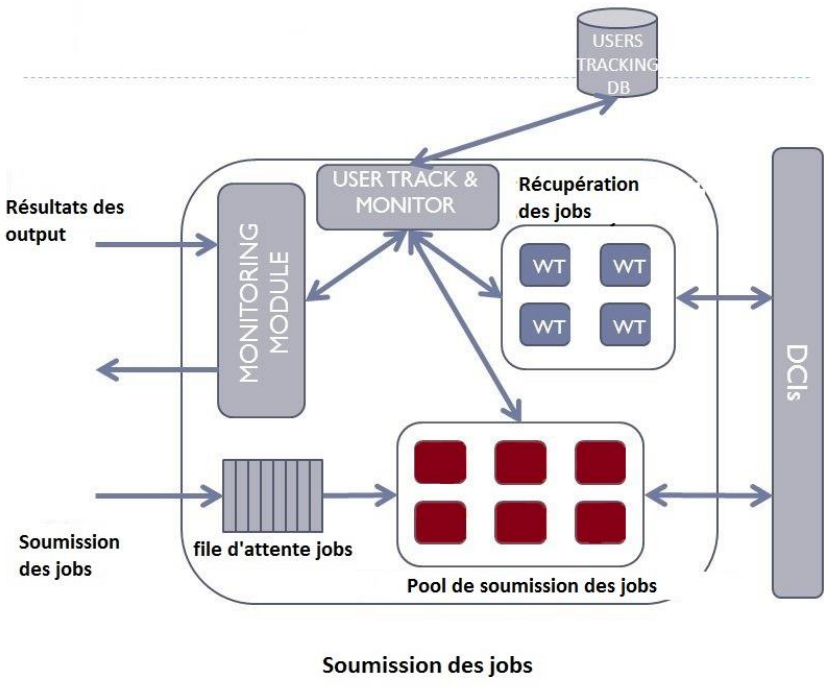

Figure 4-2 : Soumission des jobs dans le job engine

3.1 Définition des ressources

La définition des ressources mise à disposition du Workload Management System (WMS) utilisé par le Grid Engine pendant l'exécution des jobs est une étape très importante, elle est déclarée sous forme d'une liste de ressources allouée

3.2 Description d'un job simple

Dans la description d'un job simple nous spécifions cinq éléments nécessaire pour son exécution qui sont : le chemin vers l'exécutable, les arguments nécessaires aux jobs, les fichiers inputs et les outputs et le fichier des erreurs.

3.3 Soumission d'un job simple

La soumission du job sur la grille à travers le Grid Engine se fait de la manière suivante :

```
JsagaJobSubmission tmpJSaga = new JsagaJobSubmission(description);

tmpJSaga.useRobotProxy("etoken_server", "port", "
bc681e2bd4c3ace2a4c54907ea0c379b ", true);

tmpJSaga.setUserEmail("m.aliouat@grid.arn.dz");

tmpJSaga.setResourceManagerList(wmsList);

tmpJSaga.submitJobAsync("Mohamed", "@ip_server:port", 1, "gLite Test job");
```

La soumission d'un job MPI sur l'infrastructure de la grille est réalisée de la même manière qu'un job simple en rajoutant des paramètres comme le nombre de CPU utilisé dans l'exécution du job mpi et les différents scripts nécessaires pour la compilation et le lancement du job.

4. Message Passing Interface

Le Message Passing Interface (MPI) est un standard qui est né au début des années 1990, développé par MPI Forum pour faire fonctionner une large variété d'ordinateurs parallèles. En effet, à cette époque, la seule façon de faire du calcul parallèle efficace consistait à acheter une grosse machine parallèle propriétaire. Le but de MPI est de fournir un standard qui soit portable, efficace et flexible pour l'écriture de programmes qui utilisent la communication par envois de messages. MPI est une spécification pour les développeurs et les utilisateurs de bibliothèque d'envois de messages, en soit même ce n'est pas une bibliothèque mais plutôt une spécification de ce que doit être une bibliothèque. Les designers de MPI ont essayer de réunir les fonctionnalités les plus intéressantes et les plus utiles d'autres systèmes à

communication par messages existants. MPI a été influencée par Zipcode, Chimp, PVM, Chameleon, et PICL. Les lacunes rencontrées dans les précédents systèmes a communication par messages ont permis l'introduction de nouvelles fonctionnalités dans MPI.

Les points forts de MPI sont :

- *La standardisation :* MPI est la seule bibliothèque qui peut être considérée comme un standard, elle est supportée dans les plateformes HPC, et elle a remplacé toutes les anciennes bibliothèques de communication par envois de messages.

- *La portabilité :* nous n'avons pas besoin de modifier son code source quand on transporte notre application à une autre plateforme qui supporte MPI.

- *La performance :* nous exploitons un matériel basique pour optimiser la performance.

- *La fonctionnalité :* Plus de 500 routines rien que dans MPI-2.

- *La disponibilité :* une multitude d'implémentations sont disponibles.

4.1 Programmation avec MPI

Message Passing Interface est une spécification pour la programmation parallèle. Comparé à d'autre modèle, MPI introduit la contrainte de communication entres les tâches MPI qui s'exécutent en même temps. MPI peut être utilisé dans une programmation à mémoire partagé ou distribué. MPI procure aux développeurs un ensemble de routines qui :

- Gère l'environnement distribué ou les programmes MPI sont exécutés
- Procure une facilité dans les communications point à point
- Procure une facilité dans les communications entre groupe
- Procure un support pour la définition des structures de données et l'allocation de la mémoire
- Procure un support de base pour la synchronisation de communications bloquantes

Une application distribuée écrite en MPI est composée d'une collection de processus MPI qui s'exécutent en parallèle dans des infrastructures distribuées qui support MPI.

Les applications MPI qui partagent le même environnement Runtime MPI font parti par défaut du même groupe appelé *MPI_COMM_WORLD*. Dans ce groupe, tous les processus distribués ont un identifiant unique qui permet au Runtime MPI de les localiser. Dans la figure 4-4 nous constatons qu'il est possible de créer des sous-ensembles de ce groupe, si nous voulons isoler certains processus. Chaque processus MPI a un identifiant unique appelé *rank* dans le groupe auquel il appartient, il permet l'établissement des communications entres les différents processus au sein du même group.

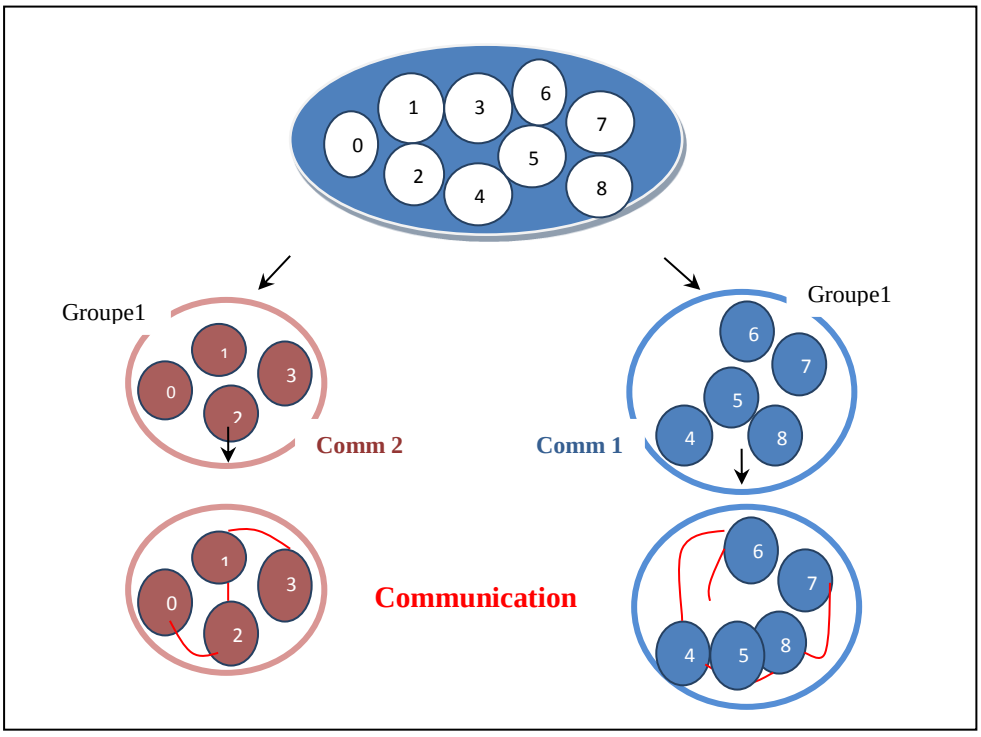

Figure 4-3 : Communication en groupe dans MPI

4.2 Structure d'un programme MPI

Pour créer une application MPI il est nécessaire de définir la partie qui est exécutée en parallèle. Cette partie est située entre deux fonctions d'initialisation et de finalisation. La structure du programme MPI est représenté dans la figure 4-5. Entre ses deux fonctions, nous pouvons appeler toutes les fonctions d'envoi et de réception de messages que se soit dans le mode synchrone ou asynchrone ou d'autres fonctions de communication MPI.

Figure 4-5 : Structure d'un programme MPI

5. Parallélisation d'un workflow avec MPI

Nous allons proposer une approche qui gère l'exécution d'un workflow scientifique composé d'applications non parallèle développées avec différents langages de programmation, dans un environnement parallèle distribué en utilisant le paradigme de Message Passing Interface. Les applications faisant partie du workflow scientifique ne sont pas réécrites en MPI vu la difficulté et la complexité du travail.

Dans notre approche le workflow scientifique est décrit dans un fichier plat et traité par l'interpréteur MPI qui a été développé. Ce dernier est soumis comme un seul job MPI sur la grille de calcul DZ e-Science GRID à travers une portlet MPI développé sur le portail scientifique « Science Gateway ».

Nous avons vu dans l'état de l'art les différents types de parallélisation des workflows scientifiques. Dans notre approche nous allons appliquer la parallélisation des tâches qui composent le workflow scientifique.

Un workflow simple peut être schématisé comme dans la figure 4-6 :

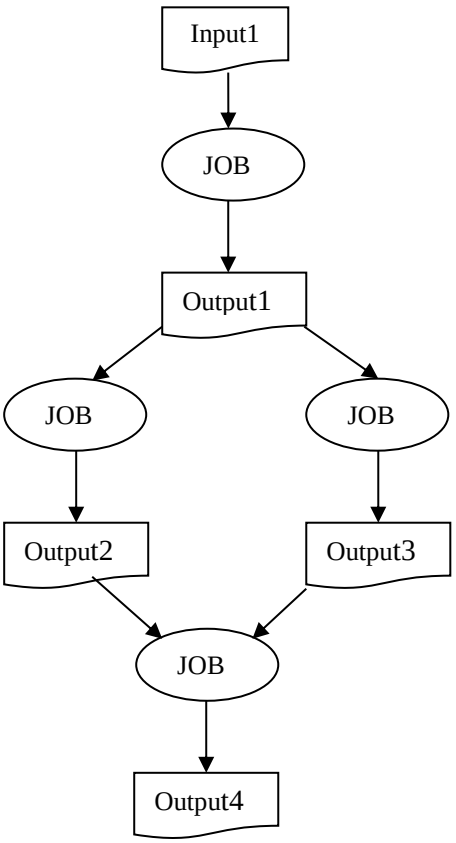

Figure 4-6 : Workflow simple

Le fichier de description de ce workflow peut avoir la forme suivante :

```
/chemin/vers/job1    –i  Input1      -o Output1
/chemin/vers/job2    -i Output1      -o Output2
/chemin/vers/job3    -i Output1      -o Output3
/chemin/vers/job4    -i Output2 –i Output3        -o Output4
```

Dans le cas où le workflow est complexe, nous décomposons le workflow en plusieurs sous graphe indépendants les uns des autres. Comme nous le montrons dans la figure 4-7, chaque sous graphe est composé par un ensemble de tâches, ce dernier (sous graphe) est par la suite envoyé par le maitre à un nœud esclave. Les tâches composantes le sous graphe sont regroupées sous un seul script.

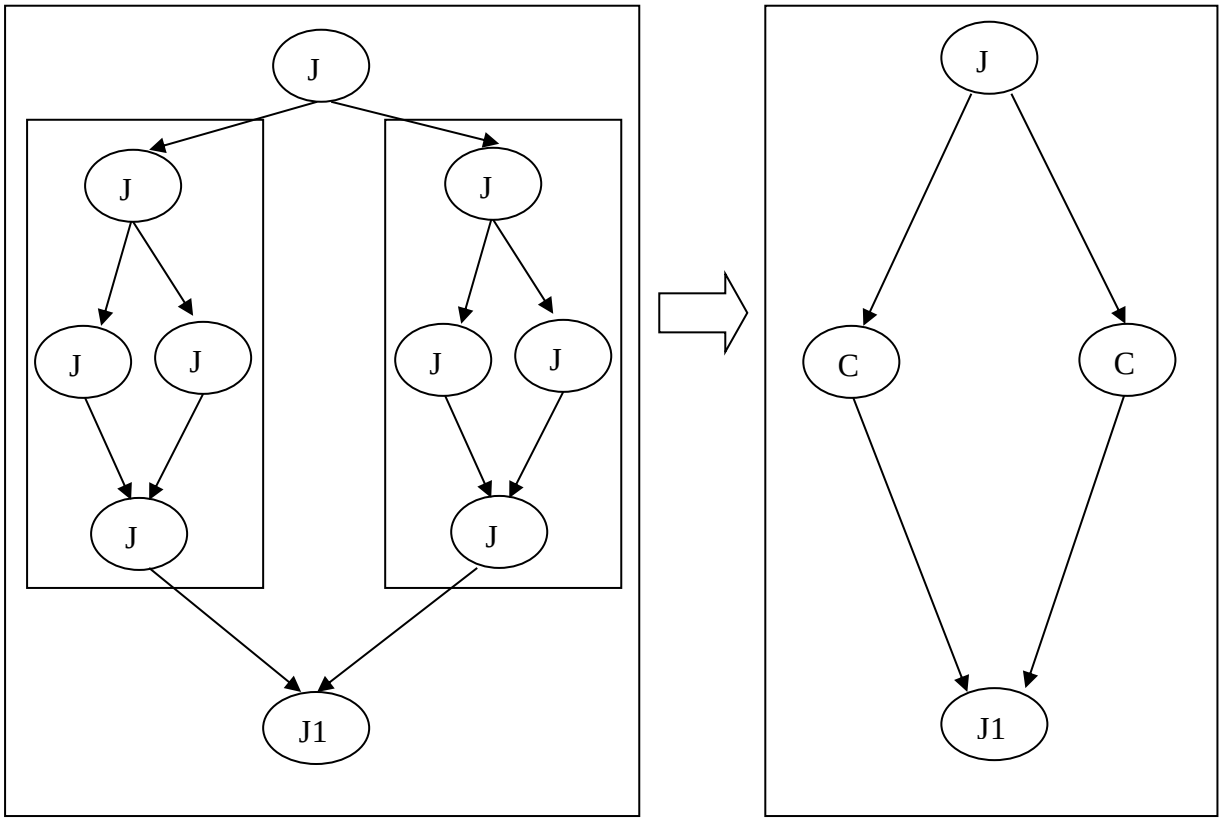

Figure 4-7 : Réduction d'un workflow complexe

5.1 Description du diagramme d'état de l'interpréteur MPI

Après la composition du workflow, nous présentons dans la figure 4-8 le diagramme d'état de notre programme d'interprétation et d'exécution du workflow scientifique en utilisant MPI comme approche de programmation parallèle suivant le modèle maitre/esclave. Le nœud maitre (en générale le processus d'Id = 0), va gérer l'analyse et l'exécution du workflow scientifique en attribuant les tâches aux différents nœuds esclaves à travers les fonctions de communication de MPI.

Suivant le diagramme de séquence représenté dans la figure 4-8, les étapes de notre algorithme sont les suivantes :

- Initialisation du programme MPI avec un appel à la fonction MPI_Init;

Si le nœud est le maitre :

- Récupérer le chemin du fichier descripteur du workflow scientifique ;
- Lecture des tâches à partir du fichier et les mettre dans une file d'attente, cette file est géré suivant le modèle FIFO (First In First Out);

- Initialisation du vecteur d'information des états des nœuds;
- Recherche et sélection d'un nœud esclave vacant;
- Avertis le nœud esclave pour se préparer à la réception de la tâche;
- Soumission de la tâche qui se trouve en tête de file au nœud esclave sélectionné ;
- Marquer le nœud esclave comme étant occupé dans le vecteur d'information des états des nœuds.
- Incrémentation du compteur des tâches lancées et on boucle jusqu'à exécution de tous les tâches;
- Si toutes les tâches sont exécutées, le nœud maitre fait appel à la fonction MPI_Finalize pour mettre fin au programme parallèle MPI.

Sinon, si c'est un nœud esclave :

- Si le nœud est vacant (libre), il envoie un message au nœud maitre pour l'informer de son statut;
- il attend la réception d'un message de la part du maitre l'informant qu'il est désigné pour l'exécution de la prochaine tâche ;
- Le nœud esclave se met à écouter pour intercepter le prochain message du root;
- En interceptant le message du root, le nœud esclave se met en mode réception de la tâche qui lui a été envoyée;
- Lancement d'exécution de la tâche reçue;
- Après la fin de la tâche, le nœud esclave informe le nœud maitre qu'il a terminé et qu'il est de nouveau vacant.

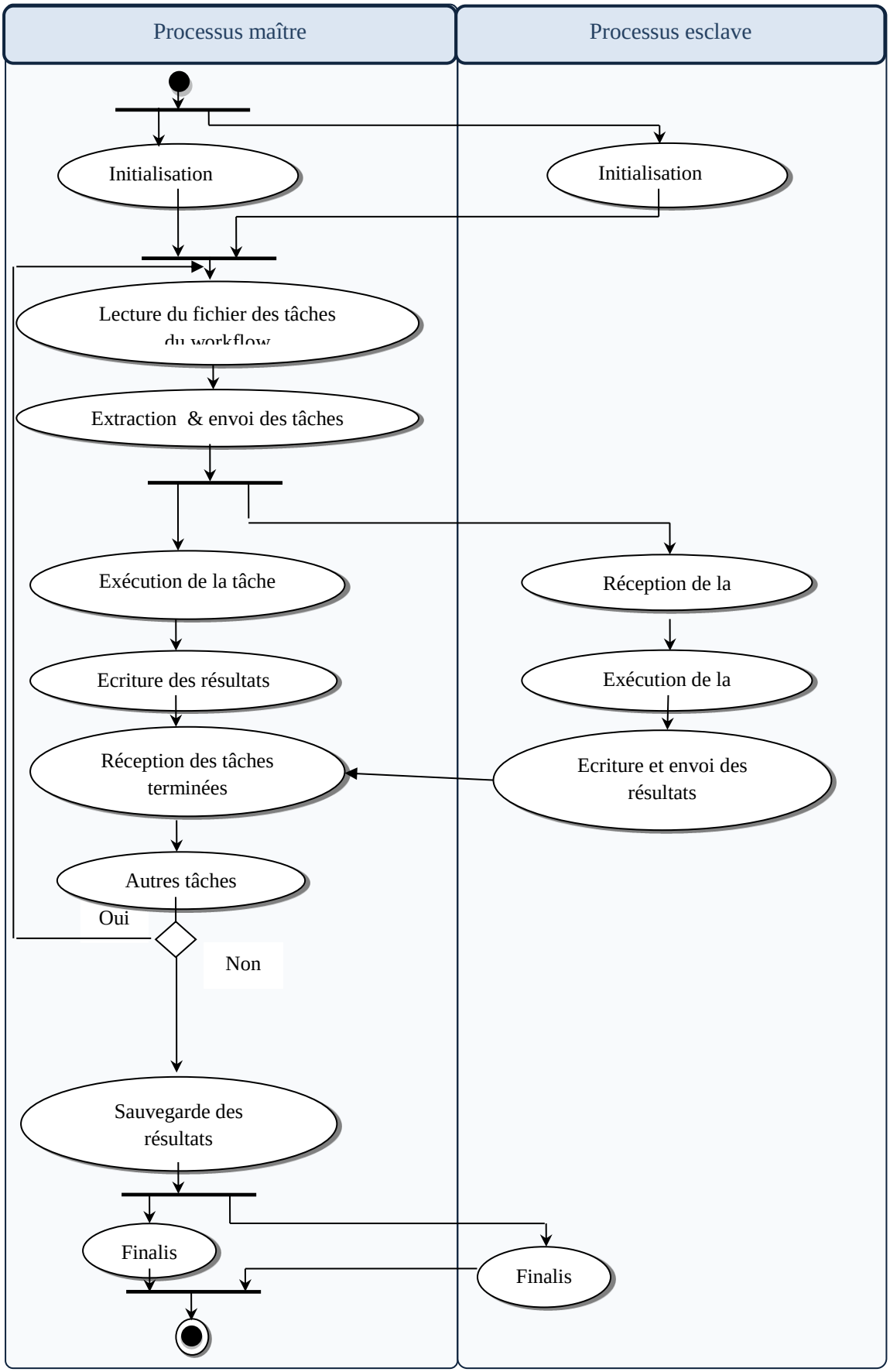

Figure 4-8 : Diagramme d'état de l'interpréteur MPI

6. Conclusion

Nous avons présenté dans ce chapitre notre approche qui consiste à la parallélisation d'un workflow avec les fonctions MPI sans modification des tâches du workflow sur une plateforme de grille de calcul. Dans le chapitre suivant nous allons présenter l'implémentation et la validation de notre approche.

Chapitre 5 : Implémentation

1 Introduction

Dans ce chapitre, nous allons suivre les étapes pour la mise en œuvre de la parallélisation du workflows scientifique avec MPI. Nous allons commencer par citer les composants et l'environnement de travail déployés sur notre plateforme, par la suite nous verrons la mise en place de la portlet d'exécution du job MPI en spécifiant les préférences et les caractéristiques d'automatisation et comment et comment notre interpréteur de workflow scientifique programmé avec MPI a été intégré.

2. Les composants de la plateforme déployée

2.1 Portail Liferay

Liferay est l'un des portails qui offre la possibilité de travailler dans un environnement unifié et paramétrable. Il inclut les standards (JSR168 e t JSR286) garantissant l'intégration d'application métier. Il permet de choisir le serveur d'application JEE et d'exploiter les EJB. Liferay s'interface avec plusieurs types de bases de données comme : MySQL, Oracle, PostgreSQL, ... etc.

Liferay est notamment compatible avec la plupart des systèmes d'exploitation, serveurs d'applications et bases de données. Il embarque des fonctionnalités de gestion de contenu qui permettent de restituer des contenus formatés sur les pages du portail à travers des portlets dédiés. Des portlets internes permettent de gérer le portail de manière rapide et facile.

2.2 Serveur d'application

Le serveur d'application est un Framework conçu pour héberger les applications qui s'exécutent dans un environnement web. Il contient un ensemble d'API disponible pour la gestion de l'environnement d'exécution. Le serveur d'application utilisé dans notre solution et le serveur GlassFish.

2.3 SAGA

SAGA est un standard qui offre une interface de haut niveau aux fonctionnalités de calculs distribués, définis par l'Open Grid Forum (OGF). SAGA est constitué de bibliothèques qui contiennent le système de base de SAGA, l'exécution et le package de l'API (gestion des tâches, gestion des données, etc ...), constitué aussi des adaptateurs qui fournis un accès à l'infrastructure GRID.

2.4 JSAGA

JSAGA est une bibliothèque en java qui implémente la spécification de l'Open Grid Forum54 SAGA. Elle propose des adaptateurs (plugins) pour divers environnements de calcul et de stockage. Ainsi elle permet une utilisation uniforme de différents systèmes de soumission et suivi de calculs distribués (gLite, Globus, Unicore, ARC) et de différents systèmes de gestion de données (LFC, SRM, GsiFTP) à partir de simples URLs.

2.5 OpenMPI

OpenMPI est une librairie open source de Message Passage Interface qui combine plusieurs technologies et ressources de plusieurs autres projets comme FT-MPI, LA-MPI, LAM/MPI et PACX-MPI ... etc. Nous avons utilisé dans notre développement la version Openmpi-1.10.

3. Environnement de mise en œuvre de l'approche

3.1 DZ e-Science GRID

DZ e-Science GRID est une infrastructure nationale de grille de calcul du réseau Algérien ARN (Algerian Research Network). L'utilisation de la grille DZ e-Science GRID est accéssible à tout les chercheurs algériens apartenant aux différents centres et laboratoires de recherches algérien pour leurs besoins en puissance de calcul.

DZ e-science GRID est composée de deux clusters géographiquement distribués (Alger et Batna) et est connecté à EGEE (Enabling Grids fo E-sciencE) via le réseau Pan-Européen de recherche GEANT. La connexion du réseau ARN au réseau GEANT a été réalisé dans le cadre du projet de coopération EUMEDCONNECT avec l'Union Européenne à travers une liaison en fibre optique de 2,5 Gbps.

3.2 Le portail DZ e-Science Gateway

Le portail scientifique est basé sur le Framework "Liferay", qui est un outil open source qui supporte les portlets. Liferay permet une gestion facile des utilisateurs et met en avant l'aspect collaboratif entre eux. Il facilite la fusion des applications en un unique portail, et accélère le développement d'application web. Notre portail scientifique " DZ e-Science Grid Science Gateway " est enregistré sous le nom de domaine suivant : https ://sgw.grid.arn.dz. Le portail permet un accès aux utilisateurs pour pouvoir interagir avec l'infrastructure mise à leurs disposition afin d'exécuter leurs expérience.

Notre portail DZ e-Science Grid Science Gateway est configuré en tant que fournisseur de services Service Provider "SP" enregistrés dans les différentes fédérations d'identité.

3.3 L'authentification et l'autorisation

L'accès aux différents services de notre plateforme DZ e-Science Grid Science Gateway se fait après l'authentification et la vérification des droit d'accès a ses derniers.

L'authentification est vérifiée travers le service Identite Provider "IdP. L'accès au service IdP est très facile; il intègre la fonction de "Single Sign-On" et de l'étendre aux ressources extérieures. C'est le module "Shibboleth" qui est responsable de lire le SSO provenant de l'IdP et de le transmettre au portail.

L'autorisation d'accès aux ressources est assuré par le serveur LDAP. Les utilisateurs sont enregistrés sur le serveur LDAP a travers le protail DZ e-Science Grid Science Gateway et grâce à un module de mis en œuvre basé sur Shibboleth.

3.4 Framework Catania Science Gateway

Notre portail scientifique DZ e-Science Gateway repose sur le Framework Catania Science Gateway qui est basé sur des technologies standardisées. Le Framework est basé su le modèle de référence, qui est capable de relier la couche présentation de notre plateforme DZ e-science Gateway avec notre infrastructure de grille DZ e-Science GRID. C'est un élément fondamental car il fournit des outils et des fonctionnalités qui permettent d'exécuter les applications à travers une interface unifiée sans se soucié du middleware qui tourne derrière. Le Framework Catania permet le déploiement des portlets qui permettent la création, la gestion et l'exécution des jobs a travers le Grid Engine ainsi la gestion des utilisateurs à travers une base de données "UserTrackingDB".

3.4.1 Portlets du Framework Catania Science Gateway

Le framework a une portlet principale appelé "MyWorkSpace". Cette portlet à été déployé sur notre plateforme de façon a être l'espace de travail de l'utilisateur. Cette portlet est composée de plusieurs autre portlets qui fournissant les fonctionnalités nécessaires à l'utilisateur afin de pouvoir interagir avec les services de la grille via notre plateforme. Les portlets peuvent aussi contrôler et gérer l'état de leurs Jobs et aussi gérer les données et les fichiers en upload ou download.

3.4.2 Catania Grid Engine

Catania Grid Engine est un standard indépendant de bibliothèque JAVA basé middleware qui procure plusieurs API afin de soumettre et de gérer des jobs sur les infrastructures informatiques distribuées. Il est compatible avec le standard Open Grid Forum (OGF) Simple API for Grid Applications (SAGA).

Catania Grid Engine fournit un moyen standard pour interagir avec différents middlewares, de sorte que les développeurs peuvent développer leurs applications sans se soucier des détails sur les infrastructures où les applications vont être exécutées.

L'architecture du framework Catania Grid Engine se compose de :

- **Deux interfaces :**
    - **Interface Science Gateway :** envers les applications;
    - **Interface Infrastructure distribuées SAGA/JSAGA API :** envers les middlewares d'infrastructures distribuées basés sur le standard SAGA ;
- **Trois modules :**
    - **Le module Job Engine :** pour la gestion des jobs ;
    - **Le module Data Engine :** pour transférer les données vers / à partir des infrastructures informatiques distribuées ;
    - **Le module de Monitoring et User Track :** pour stocker les informations sur les interactions des utilisateurs dans la base de données UserTracking ;

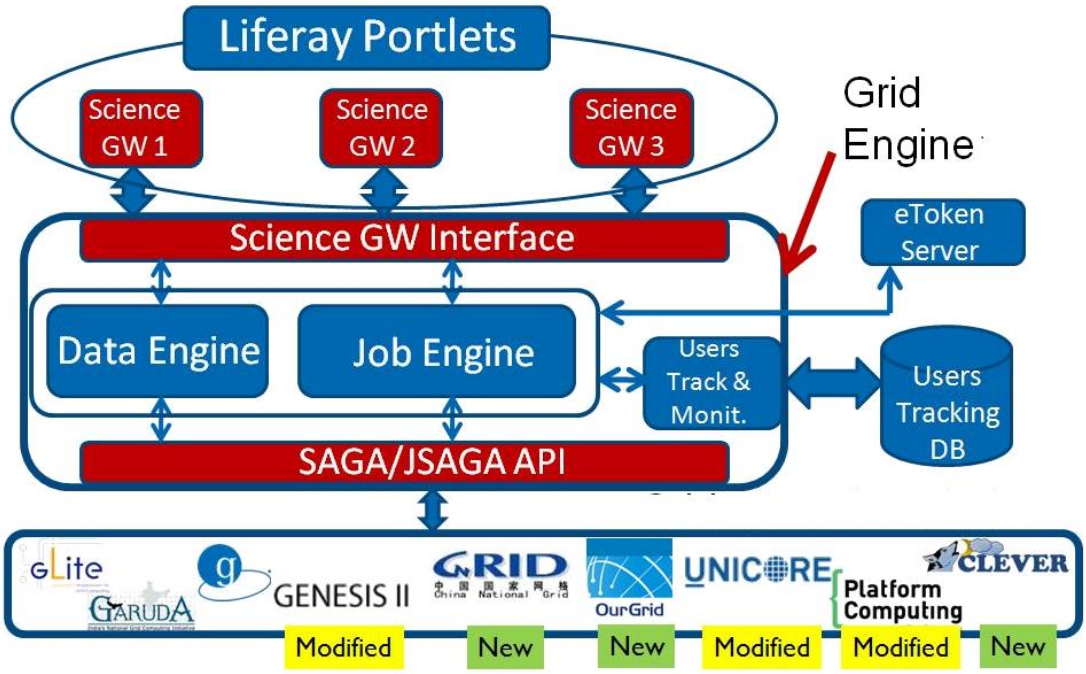

Figure 5-1 : Architecture de Catania Grid Engine

3.4.3 La base de données

La base de données "UserTrackingDB" implémenté sur notre infrastructure sert à la gestion des utilisateurs de notre plateforme DZ e-Science GRID, qui comporte les tables suivantes :

- La table *ActiveGridInteractions* contient les entrées des utilisateurs et leurs interactions avec la grille. Chaque élément possède les informations relatives au Jobs soumis à travers la plateforme et qui ne sont pas encore terminés.

- La table *GridInteractions* contient les entrées Jobs qui ont été terminés leurs exécutions.

- La table *GridOperations* contient les entrées des applications déployées sur la plateforme avec leurs identificateurs. L'identificateur de chaque application doit être utilisé sur les préférences de la portlet de l'application.

- La table *all_ces* contien les entrées de l'ensemble des éléments de calcul disponible sur les infrastructures de la grille.

4. L'interpréteur des workflows scientifique MPI sur l'infrastructure grille de calcul

Un workflow scientifique est un ensemble de tâches ou d'activités qui sont reliées à travers des flux de données. Le workflow est vu comme un seul job MPI qui est soumis sur la grille de calcul DZ e-Science GRID.

Pour les workflow scientifiques itératifs, nous avons choisi d'appliquer la parallélisation des tâches. Le workflow scientifique est décrit dans un fichier plat et traité par l'interpréteur MPI développé qui exécute les tâches indépendantes en parallèle.

4.1 Interpréteur MPI des workflows scientifiques

Notre interpréteur des workflows scientifiques à été développé en langage de programmation C++. Ce dernier offre des fonctionnalités intéressantes pour la programmation avec MPI. Comme est montré dans la figure 5-2, l'interpréteur va traiter le fichier plat reçu en input des tâches constituant le workflow scientifique à exécuter sur l'infrastructure de notre grille de calcul la DZ e-Science GRID et les mettre dans la file d'attente suivant une politique de gestion FIFO (First In First Out).

Les communications MPI sont des communications point-à-point, C'est le nœud maitre qui orchestre la communication avec ses nœuds esclaves. Dans la figure 5-3, nous avons montré le scénario de déroulement avec des fonctions de communication MPI.

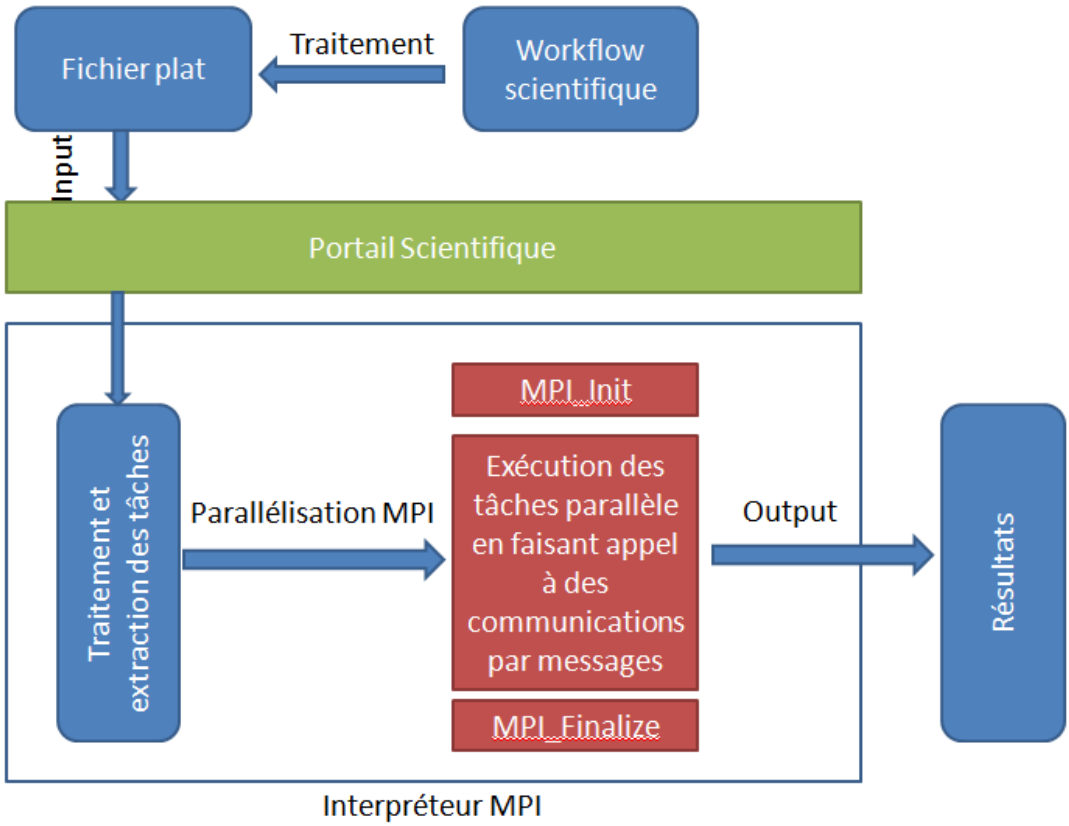

Figure 5-2 : Interpréteur MPI des workflows scientifiques

4.2 MPI-Start

Pour cacher tout la complexité des l'environnement de grille aux cours de la soumission de Jobs, nous avons eu recours mpi-start. Le but de ce dernier est de rendre plus que possible la soumission des Jobs transparente par rapport aux détails de l'infrastructure des grilles.

MPI-Start est un ensemble de scripts qui permettent l'exécution de programmes MPI sur la grille. Le principal avantage de MPI-Start est la possibilité de détecter et d'utiliser les fonctions de configuration spécifiques au site - comme le planificateur de lot et le système de fichiers sur le site. En outre, différentes implémentations de MPI sur un site sont prises en charge.

L'utilisation de MPI-Start exige la définition d'un script wrapper et d'un ensemble de hooks. Le système MPI-Start prend alors en charge la plupart des détails de bas niveau de l'exécution de la tâche MPI sur un site particulier.

Pour soumettre un job MPI, il est nécessairement de soumettre les fichiers suivants:

1. *mpi-start-wrapper.sh* : Le script configure en premier l'environnement pour les variables MPI. Il définit par la suite l'exécutable, les arguments, MPI

flavor et la location du script hook pour le mpi-start. L'utilisateur peut demander des informations de log avec les variables d'environnement et de débogage.

2. *mpi-hooks.sh* : sa fonction est d'être appelée avant que l'exécutable mpi démarre et après qu'il termine. Le pre-hook peut être utilisé pour compiler l'exécutable lui-même ou pour télécharger des données. Le post-hook peut être utilisé pour analyser les résultats ou pour enregistrer les résultats sur la grille.

   Avec ses scripts les commandes MPI seront transparents à l'utilisateur ; pas besoin d'exécuter les commandes mpiexec/mpirun et les scripts seront lancer à travers les LRMS directement.

4.3 Développement de la portlet MPI

- La classe **mpi_portlet :** Le contrôle va se faire dans la classe mpi_portlet. Dans cette dernière elle reçoit le nombre de cpu utilisé lors de l'exécution de l'interpréteur mpi. Elle est récupérée de la page input.jsp.

Le chemin vers les scripts mpi-start-wrapper et mpi-hooks et l'application mpi est indiqué dans la méthode submitJob( ) :

```
public void submitJob() {
. . .
JSagaJobSubmission tmpJSaga = new JSagaJobSubmission();
. . .
tmpJSaga.setTotalCPUCount(cpunumber);
. . .

String executable="/bin/sh";
String arguments="mpi-start-wrapper.sh Interpreteur Openmpi";
String outputPath=”/tmp/”;
String outputFile=”mpi-Output.txt”
String errorFile=”mpi-Error.txt”
. . .
String inputSandbox= appServerPath+"/WEB-INF/job/



pilot_script.sh”
+","+appServerPath+"/WEB-INF/job/mpiInterpretor.cpp"
+","+appServerPath+"/WEB-INF/job/mpi-hooks.sh"
+","+appServerPath+"/WEB-INF/job/mpi-start-wrapper.sh"
+","+inputSandbox_inputFile;
. . .
tmpJSaga.submitJobAsync(username, hostUTDB, applicationId, wmsHost,
jobIdentifier);
. . .
}
```

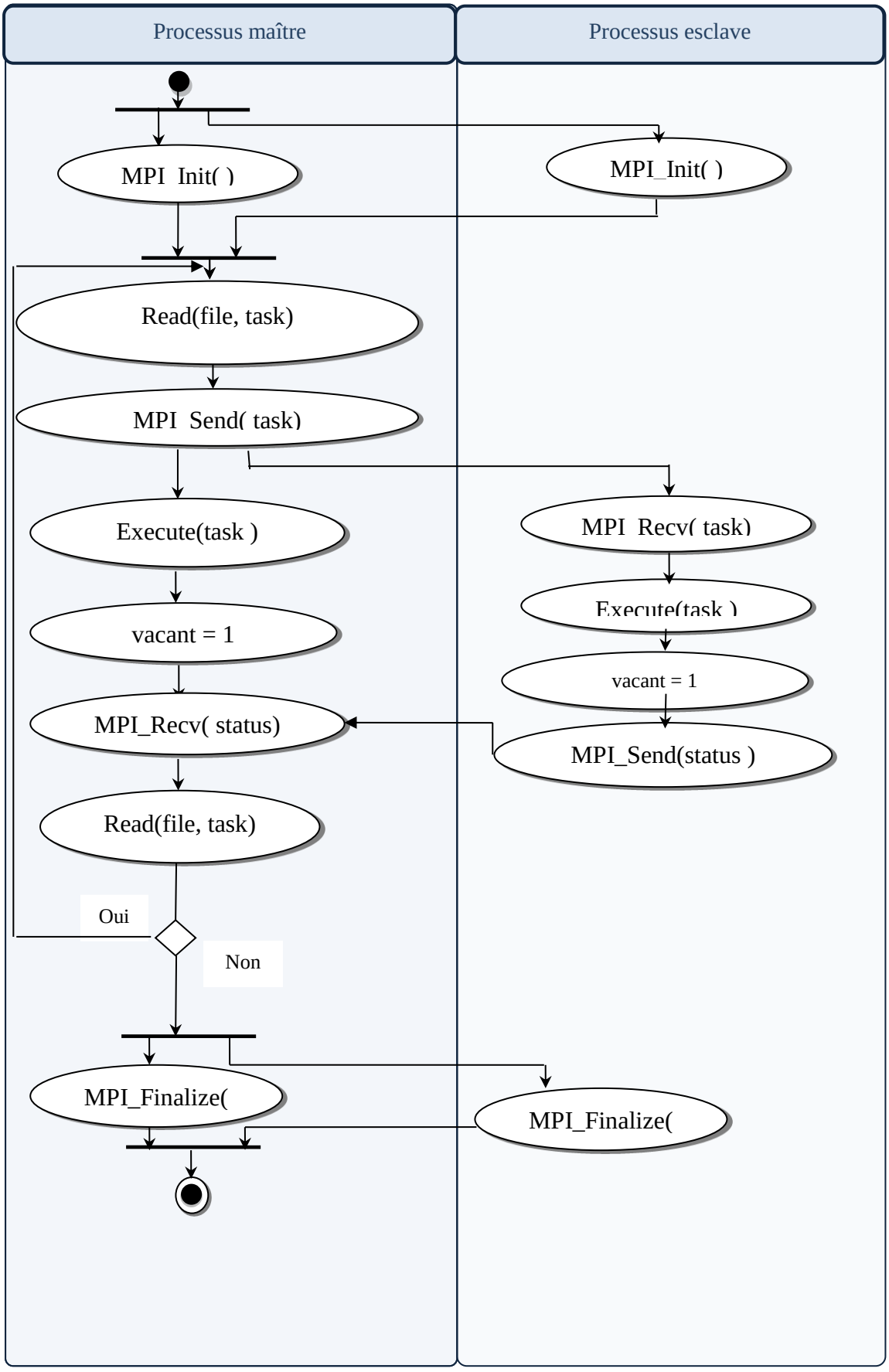

Figure 5-3 : Diagramme des routines MPI de l'interpréteur des workflows scientifique

5. Teste et validation

Après le développement de notre interpréteur de workflows, nous avons procédé à tester notre programme. Dans le cas d'un workflow simple :

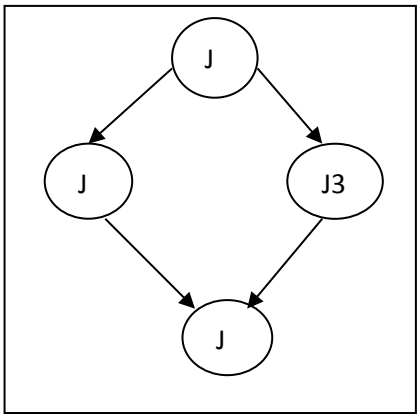

Figure 5-4 : workflow simple

Nous avons commencé par la constitution d'un fichier JDL (mpi-interpréteur.jdl) nécessaire pour la soumission du job MPI sur l'infrastructure de la grille de calcul :

```
JobType ="MpiCh";
CPUNumber = 4;
Executable = "mpi-start-wrapper.sh";
Arguments = "interpreteurMpi MPICH task.txt";
StdOutput = "std.out";
StdError = "std.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh","interpreteurMpi.cpp","workflow.txt"};
OutputSandbox = {"std.err","std.out"};
Requirements =
   Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment)
   && Member("MPICH", other.GlueHostApplicationSoftwareRunTimeEnvironment);
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    7,86          All
```

Après soumission du job mpi, le résultat du teste est satisfaisant

```
[mohamed@ui ~]$ glite-wms-job-output https://wms01.grid.arn.dz:9000/4UuYXvsHF8ar
vI6KQ3-kGg

Connecting to the service https://wms01.grid.arn.dz:7443/glite_wms_wmproxy_serve
r


================================================================================

                        JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://wms01.grid.arn.dz:9000/4UuYXvsHF8arvI6KQ3-kGg
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/mohamed_4UuYXvsHF8arvI6KQ3-kGg

================================================================================
```

6. Conclusion

Dans ce chapitre, nous avons déployé une solution de parallélisation de workflow scientifique à base de MPI sur notre plateforme de grille de calculs DZ e-Science GRID.

L'exploitation de notre portail scientifique DZ e-Science Gateway nous a énormément facilité la tâche d'utilisation de l'infrastructure de la grille de calcul, en se focalisant sur le développement de l'interpréteur MPI.

Pour une première étape nous avons utilisé MPI pour la parallélisation des tâches composant le workflow scientifique. Avec notre travail, la voie est ouverte pour de nouvelles approches dans l'utilisation de MPI dans les workflows scientifiques.

CONCLUSION GENERALE ET PERSPECTIVE

Conclusion générale

La recherche scientifique de nos jours est devenu attaché au avancé acquis dans le milieu informatique. Toute innovation est exploitée pour la résolution des problèmes complexes que rencontrent les chercheurs.

Nous avons vu l'importance des workflows scientifiques dans le travail collaboratif et comment des problèmes dans différents domaines sont modélisé dans des diagrammes DAG. Cette modélisation est importante pour la parallélisation du workflow scientifique.

L'infrastructure de la grille est environnement idéal pour la parallélisation des workflows scientifique. L'étude sur cet environnement nous a permis de comprendre son architecture et son mode de fonctionnement.

Pour épargner les chercheurs de toute la complexité de l'infrastructure de la grille de calcul, l'utilisation des portails scientifiques est devenue plus qu'une nécessité afin de leurs facilités et qu'ils puissent se concentrer sur leurs recherches.

Le standard MPI Message Passing Interface a toujours été utilisé dans le domaine de calcul de haute performance (HPC), afin de résoudre des problèmes de calcul numérique complexe. Dans notre travail nous avons exploré MPI d'une manière non traditionnelle mais en exploitant ses fonctions d'envoi de messages pour la parallélisation des tâches constituant le workflows scientifiques. L'objectif de notre travail a été atteint de manière globale.

Perspectives

Pour compléter le travail que nous avons commencé, nous présentons quelques points comme perspective pour l'avenir :

- La conception d'une interface graphique pour la composition du workflow et l'automatisation par la suite de la création des dépendances entre les tâches.

- Exploitation des autres types de parallélisation des workflows la plus intéressante la parallélisation des données.

- Adaptation des portails scientifiques avec la technologie du Cloud Computing afin de bénéficier des avantages de ce dernier.

Bibliographie

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

[2] D.P Spooner, J. Cao, S.A. Jarvis, L. He and G.R Nudd, "Performance-aware Workflow Management for Grid Computing", The Computer Journal, Oxford University Press, London, UK, 2004.

[3] F. Leymann and D. Roller., "Production Workflow: Concepts and Techniques." Prentice Hall, Englewood Cliffs, NJ, 1999.

[4] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, J. Myers , "Examining the Challenges of Scientic Workflows", IEEE Computer (2007)

[5] U. Radetzki, U. Leser, S. C. Schulze-Rauschenbach, J. Zimmermann, J. L□ussem, T. Bode, A. B. Cremers, "Adapters, shims, and glue service interoperability for in silico experiments", Bioinformatics 22(9):1137-1143, (2006)

[6] E. Deelman, D. Gannon, M. Shields, I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities", Future Generation Computer Systems 25(5):528–540, (2009).

[7] P. Romano, "Automation of in-silico data analysis processes through workflow management systems", Briefings in Bioinformatics 9(1):57–68, (2008).

[8] Jia Yu, Rajkumar Buyya, "A Taxonomy of Workflow Management Systems For Grid Computing", Journal of Grid Computing 3:171-200, (2006).

[9] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputing Applications, Vol. 15, No. 3, 2001

[10] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S.Patil, M.H. Su, K. Vahi and M. Livny, "Pegasus: Mapping Scientific Workflow onto the Grid", in Across Grids Conference 2004, Nicosia, Cyprus, 2004.

[11] J. Yu and R. Buyya, "A Novel Architecture for Realizing Grid Workflow using Tuple Spaces", in 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh, USA, IEEE CS, Los Alamitos, CA, USA, Nov. 8, 2004.

[12] G. von Laszewski, K. Amin, M. Hategan, N.J. Zaluzec, S.Hampton and A. Rossi, "GridAnt: A Client-Controllable Grid Workflow System", in 37th Annual Hawaii

International Conference on System Sciences (HICSS'04), Big Island, Hawaii: IEEE CS, Los Alamitos, CA, USA, January 5Y8, 2004.

[13] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor - A Distributed Job Scheduler", Beowulf Cluster Computing with Linux, The MIT Press, MA, USA, 2002.

[14] F. Herna´ndez, P. Bangalore, J. Gray and K. Reilly, "A Graphical Modeling Environment for the Generation of Workflows for the Globus Toolkit", in Workshop on
Component Models and Systems for Grid Applications, 18th Annual ACM International Conference on Super-computing (ICS 2004), Saint-Malo, France, ACM, NewYork, NY, USA, June 2004.

[15] T. Murata, "Temporal Uncertainty and Fuzzy-Timing High-Level Petri Nets, in Application and Theory of Petri Nets", Lecture Notes in Computer Science (LNCS), Vol. 1091, pp. 11Y28, Springer, Berlin, Heidelberg, NewYork, 1996.

[16] C. A. Petri, "Communication with Automata", PhD Thesis, Department of instrumental Mathematics, Bonn., 1962

[17] Object Management Group. Unified Modeling Language (UML), http://www.uml.org/ [Feb 2005].

[18] Z. Guan, F. Hernandez, P. Bangalore, J. Gray, A. Skjellum, V. Velusamy and Y. Liu, "Grid-Flow: A Grid-Enabled Scientific Workflow System with a Petri Net-based Interface", Technical Report, [December 2004].

[19] A. Hoheisel. "User Tools and Languages for Graph-based Grid Workflows", Grid Workflow Workshop, GGF10, Berlin, March 9, 2004.

[20] A. Lerina, C. Aniello, G. Pierpaolo and V.M. Luisa, "FlowManager: A Workflow Management System Based on Petri Nets", in 26th Annual International Computer
Software and Applications Conference, Oxford, England, IEEE CS, Los Alamitos, CA, USA, pp. 1054Y1059, August 2002.

[21] H.M.W. Verbeek, A. Hirnschall and W.M.P. van der Aalst, "XRL/Flower: Supporting Inter-Organizational Workflows Using XML/Petri-nets Technology", in Workshop on Web
Services, e-Business, and the Semantic Web (WES): Foundations, Models, Architecture, Engineering and Applications, The Fourteenth International Conference on Advanced Information Systems Engineering (CAiSE 2002), Toronto, Ontario,

Canada, Lecture Notes in Computer Science (LNCS), Springer, Berlin, Heidelberg, New York,
pp. 535Y552, May 27Y28, 2002.

[22] W.M.P. van der Aalst, K.M. van Hee and G.J. Houben, "Modelling and Analysing Workflow using a Petri-net Based Approach", in 2nd Workshop on Computer-supported Cooperative Work, Petri Nets Related Formalisms, pp. 31Y50, 1994.

[23] R. Eshuis and R. Wieringa, "Comparing Petri Net and Activity Diagram Variants for Workflow Modelling Y A Quest for Reactive Petri Nets", Advances in Petri Nets: Petri Net Technology for Communication Based Systems; Lecture Notes in Computer Science (LNCS), Vol. 2472, pp.321Y351, Springer, Berlin, Heidelberg, New York, 2003.

[24] I. Taylor, M. Shields and I. Wang," Resource Management of Triana P2P Services", Grid Resource Management, Kluwer, Netherlands, June 2003.

[25] I. Altintas, A. Birnbaum, K. Baldridge, W. Sudholt, M. Miller, C. Amoreira, Y. Potier and B. Ludaescher, "A Framework for the Design and Reuse of Grid Workflows",
in International Workshop on Scientific Applications on Grid Computing (SAG'04), LNCS 3458, Springer, Berlin, Heidelberg, New York, 2005.

[26] J. Cardoso, J. Miller, A. Sheth and J. Arnold, "Modeling Quality of Service for Workflows and Web Service Processes", Web Semantics Journal: Science, Services and
Agents on the World Wide Web, Vol. 1, No. 3, pp. 281Y308, Elsevier Inc., Massachussetts, USA, 2004.

[27] J. Cardoso, "Stochastic Workflow Reduction Algorithm", Technical Report, LSDIS Lab, Department of Computer Science University of Georgia, 2002.

[28] B. Kao and H. Garcia-Molina, "Deadline Assignment in a Distributed Soft Real-Time System, IEEE Transactions on Parallel and Distributed Systems", Vol. 8, No. 12, pp. 1268Y1274, IEEE CS, Los Alamitos, CA, USA, 1997.

[29] R. Yahyapour, P. Wieder, A. Pugliese, D. Talia and J. Hahm, "Grid Scheduling Use Cases", White Paper, Global Grid Forum, 19 July, 2004.

[30] S. Fitzgerald, I. Foster, C. Kesselman, G. Von Laszewski, W.Smith and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Computations", in 6th IEEE

[31] R. Wolski, N.T. Spring and J. Hayes, “The NetworkWeather Service: A Distributed Resource PerformanceForecasting Service for Metacomputing”, Future Generation Computer Systems, Vol. 15, No. 5Y6, pp. 757Y768, 1999.

[32] V. Hamscher, U. Schwiegelshohn, A. Streit and R.Yahyapour, “Evaluation of Job-Scheduling Strategies for Grid Computing”, in 1st IEEE/ACM International Work-shop on Grid Computing (Grid 2000), Berlin, LectureNotes in Computer Science (LNCS), Springer, Berlin, Heidelberg, New York, pp. 191Y202, 2000.

[33] G. Mateescu, “Quality of Service on the Grid via Metascheduling with Resource Co-scheduling and Coreservation”, International Journal of High Performance Computing Applications, Vol. 17, No. 3, pp. 209Y218, SAGE Publications Inc, London, UK, August 2003.

[34] V. Hamscher, U. Schwiegelshohn, A. Streit and R.Yahyapour, “Evaluation of Job-Scheduling Strategies for Grid Computing”, in 1st IEEE/ACM International Work-shop on Grid Computing (Grid 2000), Berlin, Lecture Notes in Computer Science (LNCS), Springer, Berlin, Heidelberg, New York, pp. 191Y202, 2000.

[35] T.L. Casavant and J.G. Kuhl, “A Taxonomy of Scheduling in General-purpose Distributed Computing Systems”, IEEE Transactions on Software Engineering, Vol. 14, No. 2, pp.141Y154, IEEE CS, Los Alamitos, Feb. 1988.

[36] E. Deelman, J. Blythe, Y. Gil and C. Kesselman, “Workflow Management in GriPhyN”, The Grid Resource Management, Kluwer, Netherlands, 2003.

[37] D.C. Li and N. Ishii, “Scheduling Task Graphs onto Heterogeneous Multiprocessors”, TENCON’94, IEEE Region 10’s Ninth Annual International Conference, Theme: Frontiers of Computer Technology, IEEE CS, Los Alamitos, CA, USA, 1994.

[38] Xiaorong Xiang, Gregory Madey, "Improving the Reuse of ScientificWorkflows and Their By-products", *ICWS*, 2007, 2007 IEEE International Conference on Web Services, pp. 792-799, doi:10.1109/ICWS.2007.107.

[39] R. Sakellariou and H. Zhao, “A Low-Cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems”, Scientific Programming, Vol. 12, No. 4, pp.253Y262, IOS, Netherlands, December 2004.

[40] A. Sulistio and R. Buyya, “A Grid Simulation Infrastructure Supporting Advance Reservation”, in 16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004), MIT Cambridge, Boston, USA, ACTA, CA, USA, November 9Y11, 2004

[41] D. Fernandez-Baca, “Allocating Modules to Processors in a Distributed System”, IEEE Transactions on Software Engineering, Vol. 15, No. 11, pp. 1427Y1436, November
1989.

[42] R.A. Moreno, “Job Scheduling and Resource Management Techniques in Dynamic Grid Environment”, in 1st European Across Grids Conference, Spain, Lecture Notes in Computer Science (LNCS), Springer, Berlin, Heidelberg, New York, February 2003.

[43] R. Buyya, D., Abramson and J. Giddy, “Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid”, HPC Asia 2000,
Beijing, China, IEEE CS, Los Alamitos, CA, USA, pp. 283Y289, May 14Y17, 2000.

[44] S. Venugopal, R. Buyya and L. Winton, “A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids”, in 2nd International Workshop
on Middleware for Grid Computing, Middleware 2004, Toronto, Ontario-Canada, ACM, New York, NY, USA, October 18, 2004.

[45] A. Geppert, M. Kradolfer and D. Tombros, “Market-based Workflow Management”, International Journal of Cooperative Information Systems, World Scientific, New Jersey,
USA, 1998.

[46] I. Brandic, S. Benkner, G. Engelbrecht and R. Schmidt, “Towards Quality of Service Support for Grid Workflows”, First European Grid Conference (EGC 2005), Amsterdam, The Netherlands, Feb 2005.

[47] S.Y. Zhao and V. Lo, “Result Verification and Trust-based Scheduling in Open Peer-to-Peer Cycle Sharing Systems”, Technical Report, University of Oregon, USA, 2005.

[48] P.A. Dinda, “Online Prediction of the Running Time of Tasks”, Cluster Computing, Vol. 5, No. 3, pp. 225Y236, Kluwer, Netherlands, 2002.

[49] G. Zheng, T. Wilmarth, P. Jagadishprasad and L.V. Kale, “Simulation-based Performance Prediction for Large Parallel Machines”, International Journal of Parallel Programming,Vol. 33, No. 2Y3, pp. 183Y207, Springer Academic Publishers, The Netherlands, 2005.

[50] K. Cooper, A. Dasgupata, K. Kennedy, C. Koelbel, A. Mandal, G. Marin, M. Mazina, J. Mellor-Crummey, F.Berman, H. Casanova, A. Chien, H. Dail, X. Liu,

A.Olugbile, O. Sievert, H. Xia, L. Johnsson, B. Liu, M. Patel,D. Reed, W. Deng, C. Mendes, Z. Shi, A. YarKhan and J.Dongarra, "New Grid Scheduling and Rescheduling Methods in the GrADS Project", NSF Next Generation Software Workshop, International Parallel and Distributed Processing Symposium, Santa Fe, IEEE CS, Los Alamitos, CA, USA, April 2004.

[52] H.J. Dail, "A Modular Framework for Adaptive Scheduling in Grid Application Development Environments", Master's Thesis, UCSD Technical Report CS2002-0698, University of California at San Diego, March 2002.

[53] G.R. Nudd, D.J. Kerbyson, E. Papaefstathiou, S.C. Perry, J.S. Harper and D.V. Wilcox, "PACE-A Toolset for the Performance Prediction of Parallel and Distributed Systems", International Journal of High Performance Computing Applications (JHPCA), Special Issues on Performance Modelling-Part I, Vol. 14, No. 3, pp. 228Y251, SAGE,
London, UK, 2000.

[54] S. Jang, X. Wu, V. Taylor, G. Mehta, K. Vahi and E.Deelman, "Using Performance Prediction to Allocate Grid Resources", Technical Report 2004-25, GriPhyN Project,
USA.

[55] A. Mayer, S. McGough, N. Furmento, W. Lee, S. Newhouse and J. Darlington, "ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time", in UK e-Science All Hands Meeting, Nottingham, UK, IOP, Bristol,UK, pp. 627Y634, September 2003.

[56] W. Smith, I. Foster and V. Taylor, "Predicting Application Run Times Using Historical Information", in Workshop on Job Scheduling Strategies for Parallel Processing, 12th International Parallel Processing Symposium & 9th Symposium on Parallel and Distributed Processing (IPPS/SPDP '98), IEEE CS, Los Alamitos, CA, USA, 1998.

[57] Taverna User Manual. http://taverna.sourceforge.net/manual

[58] DAGMan Application. http://www.cs.wisc.edu/condor/manual

[59] J.H. Abawajy, "Fault-Tolerant Scheduling Policy for Grid Computing Systems", in 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico, IEEE Computer Society (CS), Los Alamitos, CA, USA, pp. 238Y244, April 26Y30, 2004.

[60] S. Hwang and C. Kesselman, "Grid Workflow: A Flexible Failure Handling Framework for the Grid", in 12th IEEE International Symposium on High

Performance Distributed Computing (HPDC'03), Seattle, Washington, USA, IEEE CS, Los Alamitos, CA, USA, June 22Y24, 2003.

[61] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Lamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger and B. Tierney, "Giggle: A Framework for Constructing Scalable Replica Location Services", in Supercomputing (SC2002), Baltimore, USA, IEEE Computer Society, Washington, DC,
USA, November 16Y22, 2002.

[62]] M. Bux and U. Leser. "Parallelization in scientific workflow management systems", The Computing Research Repository (CoRR), abs/1303.7195, 2013.

[63] I. Foster, C. Kesselman, S. Tuecke. "The Anatomy of the Grid, enabling Scalable virtual organizations". Supercomputer Application, 2001.

[64] K, Tan, F. Magoulès, J. Pan, A. Kumar. "Introduction to Grid Computing", CRC Press 2009, ISBN: 978-1-4200-7406-2, 2009.
[65] C. Felenstein, J. Joseph, "Grid Computing". Pentice Hall/IBM Press, ISBN-10:0-13-145660, 2004.

[66] F. Magoulès, J. Pan, K. Tan and A. Kumer. "Introduction to Grid Computing". CRC Press, Inc. Boca Raton, FL, USA, 2009. ISBN : 1420074067
334 p.

[67] J.Liu, E.Pacitti, P.Valduriez, M.Mattoso, "Parallelization of Scienti_c Workows in the Cloud", RR-8565, 2014. hal-01024101v1.