GRILLES DE CALCUL
GRID COMPUTING

Polycopié du cours

Dr. Hayat BENDOUKHA

Département d'Informatique
Faculté des mathématiques et Informatique
USTO-MB - Oran
Mai 2023

Table des matières

Table des figures

I. Introduction générale

I.1. Présentation du cours

De nos jours, les applications scientifiques, industrielles et commerciales connaissent une constante évolution et requièrent de plus en plus de ressources en termes de stockage et de calcul. Les systèmes distribués à grande échelle, tels que les grilles de calcul offrent une variété de ressources pour satisfaire les exigences de telles applications. Ces systèmes fournissent des plateformes et des outils efficaces pour le partage de ressources et la distribution des données et du calcul.

Au cours de ces dernières décénies, le paradigme calcul orienté service (SOC pour Service Oriented Computing) a révolutionné non seulement l'implémentation, mais aussi le déploiement et la gestion des middlewares de grille. Ce paradigme a considérablement marqué l'évolution des systèmes distribués et en particulier les systèmes de grille qui ont fortement bénéficié des concepts de services afin d'améliorer leurs middlewares.

Le plus grand avantage du paradigme SOC est sa flexibilité et sa facilité d'intégration. En effet, le calcul orienté service n'a pas essayé de remplacer des paradigmes existants, tels que l'orienté objet, l'orienté aspect, l'orienté composant, etc., mais il a cherché à intégrer les technologies existantes pour les améliorer en introduisant une plus grande fiabilité, évolutivité, performance et passage à l'échelle à leurs environnements respectifs . Ainsi, ce paradigme est de plus en plus adopté par de nombreux fournisseurs de solutions logicielles dans différents domaines de l'informatique.

Comme exemple illustratif de l'utilisation de ce paradigme, nous pouvons citer le projet collaboratif international "Globus Alliance" qui a développé le premier middleware de grille orienté-service , appelé Globus Toolkit . La dernière version stable de ce middleware est le GT6 (Globus Toolkit 6) . Depuis ses premières versions, Globus constitue la référence de multiples fournisseurs de services et développeurs de middlewares de grille.

Ce polycopié de cours s'inscrit dans ce domaine de l'informatique et s'interesse spécialement au GRID COMPUTING

I.2. Public cible

Ce cours s'adresse à des étudiants en Master 1 en Informatique, notamment dans les spécialités autour des systèmes informatiques, réseaux et bases de données distribuées.

L'étudiant a besoin de ses acquis de la licence en Informatique avec un focus sur les cours de systèmes distribués. L'objectif de ce cours et de présenter la technologie Grid Computing (Grilles de Calcul) comme une alternative de plates formes efficaces pour le

développement et l'exécution des applications gourmandes en ressources de calcul et de stockage.

Pour une excellente assimilation de tous les concepts liés au grid computing, ce cours vise à situer cette technologie chronologiquement dans les différentes générations de systèmes et d'infrastructures à hautes performances, à savoir démarrer d'une présentation du concept du « distribué » avec toues ses dimensions et faire le pont avec « l'orienté service » jusqu'à aboutir aux nouvelles technologies inspirées et basées sur les grilles de calcul telles que le « Cloud computing ».

Le contenu de ce cours est parfaitement adapté à la matière "Grilles de Calcul" adréssé aux étudiants de Master RSID (Réseaux et Systémes Informatiques distribués) en semestre 2. Il peut être également adopté pour la matière "Grilles Informatiques" pour le Master SID (Système Informtiques et Données) en Semestre 2 avec l'ajout d'un chapitre dédié à la manipulation des données dans un environnement de type Grid via des technoiques telles que la réplication de données.

I.3. Prérequis

Licence en Informatique

I.4. Structure du polycopié du cours

Ce polycopié dédié au Grid computing est composé, outre de cette introduction, de 5 chapitres et d'une conclusion générale.

Le chapitre II, intitulé "Calcul & Systèmes Distribués", présente le paradigme du distribué incluant les définitions, les architectures et les concepts de chacune des technologies liées au calcul distribué, ainsi que l'évolution des différents systèmes à travers les générations donnant naissance au Grid Computing.

Le chapitre III, intitulé "Grid Computing", présente le coeur de notre polycopié à savoir le Grid Computing. La technologie est d'abord définie ensuite décrite à travers ses caractéristiques, ses concepts et son architecture générique.

Dans le chapitre IV, intitulé "Développement & Architectures Orientées Service", tous les concepts et définitions liés au développement orienté service sont présentés.

Le chapitres V, intitulé "Déploiement des Services dans les Grilles de Calcul" vient démontrer toute la force du développement orienté service à travers la présentation du fonctionnement des middelwares de grilles orientées service.

Le chapitre VI, intitulé "Administration : Ordonnancement via les Outils Workflow", s'interesse au volet le plus important dans le fonctionnement des middleware des grilles, à savoir l'ordonnacement. Après la présentation de l'ordonnancement , ce chapitre décrit comment d'autres technologies de l'informatique actuelle servent efficacement l'ordonnacement de grilles, notamment le BPM (Business Process Management) et le Workflow.

Enfin, ce polycopié termine par une conclusion générale.

II. Calcul & Systèmes Distribués

II.1. Introduction

Ce chapitre a pour objectif de présenter le calcul et les systèmes distribués, incluant définitions, concepts et architectures. Ce chapitre présente aussi l'évolution qu'ont connue les middlewares depuis les supercalculateurs, clusters jusqu'aux systèmes de grille et cloud.

II.2. Calcul distribué

A l'opposé d'une configuration séquentielle où la plupart des modèles de processeurs exécutent fidèlement un seul modèle de calcul (calcul séquentiel ou concurrent), le modèle de calcul distribué a permis d'avoir une grande variété d'architectures distribuées. Cette section présente quelques définitions, concepts et propriétés liés au modèle de calcul distribué.

II.3. Le Distribué au coeur du calcul réparti

Souvent, les deux paradigmes parallèle et distribué sont confondus ou utilisés conjointement pour représenter les architectures et les systèmes à calcul intensif. Le parallélisme reste néanmoins plus répandu comme appellation de ce type de calcul.

Une machine parallèle est essentiellement un ensemble de processeurs qui coopèrent et communiquent. Historiquement, les premières machines parallèles sont des réseaux d'ordinateurs, et des machines vectorielles faiblement parallèles (années 70 - IBM 360-90 vectoriel, IRIS 80 triprocesseurs, CRAY 1 vectoriel. . .).

Il existe en littérature une classification des architectures parallèles qui indique la nuance de la distribution et du partage de mémoire pour la distribution des données et calculs.

On distingue classiquement quatre types principaux de parallélisme selon la Taxonomie de Flynn Tanenbaum : **SISD, SIMD, MISD et MIMD**. Cette classification est basée sur les notions de flot de contrôle (deux premières lettres, I voulant dire —Instruction‖) et flot de données (deux dernières lettres, D voulant dire —Data‖).

II.3.1. Machine SISD

Une machine SISD (Single Instruction Single Data) est ce que l'on appelle d'habitude une machine séquentielle, ou machine de Von Neuman. Une seule instruction est exécutée à un moment donné et une seule donnée (simple, non-structurée) est traitée à un moment donné.

```
int A[100];
... for (i=1;100>i;i++)
A[i]=A[i]+A[i+1];
```

Le petit programme ci-dessus s'exécute sur une machine séquentielle en faisant les additions A[1]+A[2], A[2]+ A[3], etc., A[99]+A[100] à la suite les unes des autres.

II.3.2. Machine SIMD

Une machine SIMD (Single Instruction Multiple Data) est une machine qui exécute à tout instant une seule instruction, mais qui agit en parallèle sur plusieurs données, on parle en général de —parallélisme de données‖. Les machines SIMD peuvent être de plusieurs types, par exemple, parallèles ou systoliques. Les machines systoliques sont des machines SIMD particulières dans lesquelles le calcul se déplace sur une topologie de processeurs, comme un front d'onde, et acquiert des données locales différentes à chaque déplacement du front d'onde (comportant plusieurs processeurs). En général dans les deux cas, l'exécution en parallèle de la même instruction se fait en même temps sur des processeurs différents (parallélisme de donnée synchrone).

II.3.3. Machine MISD

Une machine MISD (Multiple Instruction Single Data) peut exécuter plusieurs instructions en même temps sur la même donnée. Cela peut paraître paradoxal mais cela recouvre en fait un type très ordinaire de micro-parallélisme dans les micro-processeurs de type processeurs vectoriels et les architectures pipelines.

II.3.4. Machine MIMD

Le MIMD (Multiple Instruction Multiple Data) est le plus intuitif et est celui qui va nous intéresser le plus dans ce cours. Ici, chaque processeur peut exécuter un programme différent sur des données différentes. On a plusieurs types d'architecture possibles :

(1) Mémoire partagée (Sequent etc.)

(2) Mémoire locale + réseau de communication (Transputer, Connection Machine) - Système réparti

Parce qu'il n'est en général pas nécessaire d'utiliser des programmes différents pour chaque processeur, on exécute souvent le même code sur tous les nœuds d'une machine MIMD mais ceux ci ne sont pas forcément synchronisés. On parle alors de modèle SPMD (Single Program Multiple Data).

II.4. Définition du calcul distribué

Le calcul distribué englobe les architectures, les modèles et les techniques employées pour gérer un système distribué. La définition la plus commune d'un système distribué est celle donnée par Tanenbaum et al. dans [23] :

> *"Un système distribué est une collection d'ordinateurs indépendants qui apparaissent aux utilisateurs comme un seul système cohérent. "*

Cette définition est assez générale pour prendre en considération les différents types de systèmes distribués.

Le calcul distribué est considéré comme le processus d'agrégation de différentes entités de calcul. Ces entités sont logiquement distribuées et peuvent être géographiquement distribuées afin d'exécuter, sous forme collaborative, un ensemble de tâches d'une manière transparente et uniforme et apparaissent comme un seul système centralisé. Tout système distribué est composé d'au moins deux ordinateurs connectés à travers un réseau. De ce fait, la communication devient un aspect important en calcul distribué. La communication concerne l'échange de données et d'informations entre ordinateurs reliés par un réseau. Coulouris et al. présente cette vision dans [8] par la définition suivante :

> *"Un système distribué est un système dans lequel un ensemble de composants situés sur des ordinateurs reliés en réseau communiquent et coordonnent leurs actions seulement par envoi de messages."*

II.5. Composants et architecture d'un système distribué

Lors de la conception d'un système distribué, il est important de distinguer entre les responsabilités des composants du système (applications, serveurs, et processus) et leur placements dans le réseau. Dans cette section, nous présentons une vue en couches de l'architecture générique des systèmes distribués comme le montre la Figure II.1.

Trois principales couches sont impliquées dans la fourniture de services à un système distribué.

— L'ordinateur et le matériel réseau constituent l'infrastructure physique à la couche inférieure. Tous les deux, ils sont gérés par le système d'exploitation. Ensemble, l'infrastructure physique et le système d'exploitation sont la plate-forme sur laquelle le logiciel est déployé pour faire fonctionner le réseau des ordinateurs. Au niveau de cette couche, le sytème d'exploitation utilise un ensemble de normes, protocoles et services bien connues, tels que : les services de la IPC (InterProcesses Communication Services), TCP/IP (Transmission Control Protocol/Internet Protocol) et UDP (User Datagram Protocol).

— Le middleware arrive à la deuxième couche. Son rôle principal est de construire un environnement uniforme pour le développement et le déploiement d'applications distribuées. Ceci est soutenu par des protocoles, langages et formats propres au

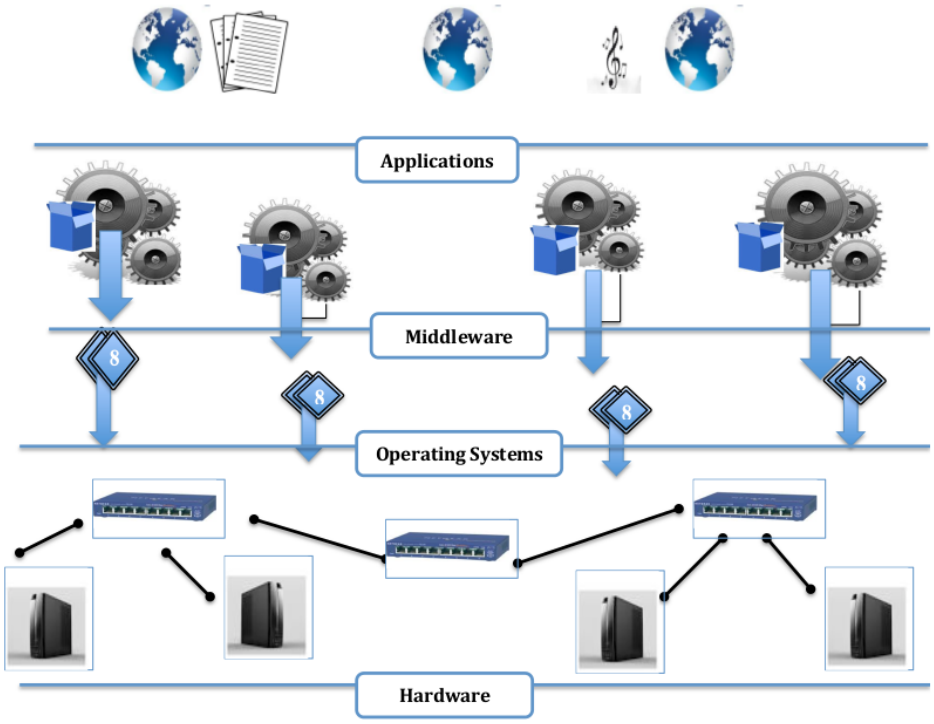

FIGURE II.1. – Vue en couches de l'architecture générique des systèmes distribués

middleware constituant ensemble une interface uniforme complètement indépendante du système d'exploitation sous-jacent.
— La couche supérieure comprend toutes les applications et services développés pour utiliser le système distribué via une interface utilisateur graphique (GUI) accessible localement ou via un navigateur web.

II.6. Propriétés du calcul distribué

Dans cette section, nous présentons une liste non exhaustive de propriétés du calcul distribué. Ces propriétés ont été choisies selon les concepts invoqués le plus souvent lors de l'évaluation d'un système distribué.

— **Partage de ressources**. Cette concerne la capacité d'utiliser tout composant (matériel et/ou logiciel) n'importe où dans le système. Cela nécessite l'existence d'un *modèle de partage de ressources* tel que le modèle Client/Serveur.
— **Ouverture**. L'ouverture concerne toutes les extensions et les améliorations que peut subir un système distribué durant sa durée de vie.
— **Concurrence**. Les ressources d'un système distribué sont accessibles de manière concurrente pour permettre d'assurer leur partage entre les applications des utilisateurs. Pour assurer l'intégrité du système, il est nécessaire de coordonner les mises à jour concurrentes aux ressources.
— **Passage à l'échelle**. Le passage à l'échelle d'un système distribué est sa capacité à pouvoir changer de taille et/ou de volume pour s'adapter aux besoins des utilisateurs.
— **Transparence**. Les systèmes distribués doivent être perçus, à la fois par les utilisateurs et par les développeurs d'applications, comme une entité indivisible et non comme des composants coopératifs. Les différentes dimensions de la transparence ont été identifiées par l'ANSA (Advanced Network Services Architecture) [23]. Ces dimensions incluent la transparence de l'évolutivité, la transparence de la performance, la transparence de la migration, la transparence de la réplication, de l'accès et de la localisation.
— **Tolérance aux fautes**. Les composants d'un système physique ou logique peuvent faire l'objet de pannes durant leur utilisation. Ces pannes revêtent un aspect particulier dans les systèmes distribués (à cause de leur distribution et de leur partage) et ces systèmes doivent être capables de maintenir la disponibilité des services composants. La tolérance aux pannes est assurée par différentes techniques, telles que la redondance (réplication), le recouvrement, etc.

II.7. Middlewares et technologies du calcul distribué

Comme nous l'avons mentionné dans la définition du calcul distribué, un middleware est le composant clé de tout système distribué. Les middlewares fournissent des environnements pour le développement de systèmes pouvant être distribués sur une variété de

topologies, applications, services, etc. Dans cette section, nous présentons les types les plus connus de middlewares pour le calcul distribué.

II.7.1. Middlewares orientés objet

Les middlewares les plus populaires sont CORBA [1], Java RMI [2] et SOAP [3] qui fournissent une base de support pour les objets pouvant être dispersés dans un réseau. Ils permettent aux utilisateurs d'invoquer des objets via des opérations à distance.

L'OMG (Object Management Group) ne produit pas de middlewares à proprement parler, mais fournit uniquement des spécifications à respecter pour le développement de middlewares. Ces spécifications ont pour objectif principal de standardiser les normes de développement de middlewares. En 1991, l'OMG a publié la révision 1.1 de la spécification de Common Object Request Broker Architecture (CORBA) , une description concrète des interfaces et des services qui doivent être fournis par un middleware.

II.7.2. Middlewares orientés composant

Ce type de middlewares est apparu au milieu des années 90's afin de remédier aux limites des middlewares orientés objets tels que : les limites au niveau des fonctionnalités et les limites liées aux standards de déploiement et de configuration. Les middlewares orientés composant les plus connus sont le Enterprise Java Beans, le CORBA et .NET, qui sont les successeurs d'approches distribuées orientées objet. Ils se sont basés sur une composition relativement autonome. Aussi, ils ont fusionné les fonctionnalités logicielles des éléments qui peuvent être distribuées ou co-localisées à travers un large éventail de réseaux et d'interconnexions.

II.7.3. Middlewares orientés service

L'architecture orientée service (SOA) est un style d'organisation et d'utilisation de capacités distribuées pouvant être contrôlées par différents organismes ou propriétaires. Cette architecture fournit une façon uniforme d'offrir, de découvrir, d'interagir et d'utiliser les capacités de services logiciels à faible couplage. L'équitabilité du Web Wild World (WWW) et les leçons tirées des middlewares antérieurs ont formé la première version du protocole SOAP dans les années 1990. SOAP est un protocole d'échange de messages XML sur un réseau informatique, normalement en utilisant le protocole HTTP. Il permet l'interopérabilité avec les différents types de middlewares y compris Corba etc.

Au lieu de remplacer les approches antérieures, les technologies orientées services mettent l'accent sur l'intégration de middleware, en ajoutant de la valeur à des plates-formes existantes. Les langages dédiés au web tels que WSDL (Web Service Description Language) sont les clés de ces technologies permettant aux développeurs de décrire les interfaces abstraites des services web. D'autres mécanismes sont également impliqués dans

1. `http://www.corba.org/`
2. `http://docs.oracle.com/javase/7/docs/technotes/guides/rmi/`
3. `http://www.w3.org/TR/soap/`

cette classe de middlewares comme les serveurs web, les protocoles HTTP ainsi que les WSRF (Web Service Resource Frameworks) qui permettent d'établir des connexions faciles entre les navigateurs web et les pages web qui peuvent être conçues comme des portails vers des systèmes d'information complexes.

Aujourd'hui, nous ne pouvons pas parler de SOA sans évoquer les grilles de calcul et inversement. Il y a deux connexions entre SOA et les grilles de calcul. L'une est l'application de l'architecture SOA dans les grilles de calcul, à travers les architectures de grille comme l'architecture OGSA. La deuxième est l'utilisation d'une grille de calcul sous forme d'architecture SOA. De cette manière, les services sont considérés comme des ressources de la grille pour soutenir le partage, la gestion et l'accès aux services. Du point de vue de l'orientation service, les services web se focalisent sur la description de l'interface et la messagerie entre services. Or, les grilles de calcul mettent en phase les ressources d'un calcul distribué incluant l'accès transparent, la tolérance aux fautes, l'équilibrage de charge, etc.

II.8. Premières générations des systèmes distribués des machines mono processeur aux grilles de calcul

Le besoin de puissance de calcul et de stockage de plus en plus importante a guidé et guide encore l'évolution des architectures matérielles pour le calcul scientifique. Il s'agit de l'exigence des applications scientifiques en termes de rapidité et d'efficacité de calcul ainsi que la capacité de stockage de grands volumes de données. La rapidité et l'efficacité ne se traduisent pas uniquement par la fréquence de traitement des instructions d'un programme par un ordinateur mais aussi par la capacité d'exécuter en parallèle (simultanément) des codes d'une même application.

Cependant, en plus des critères de rapidité de calcul, il s'agit aussi de la possibilité de faire communiquer des codes scientifiques qui s'exécutent sur des infrastructures géographiquement distribuées. C'est le cas par exemple, des applications de couplage de codes appartenant à différentes institutions contraintes de préserver la confidentialité de leurs codes respectifs.

II.8.1. Machines mono-processeurs

Caractérisé à la base par une unique unité de calcul, un mono-processeur ne peut exécuter qu'un seul processus à la fois. Cependant, les systèmes d'exploitation comme UNIX et Windows permettent de gérer l'exécution concurrente par permutation de plusieurs processus d'un même programme à travers l'exécution multi-fils.

Récemment, les mono-processeurs ont pris un autre tournant avec l'apparition des puces « bi-cœurs » (d'AMD, IBM, Intel, etc.). Un processeur a intégré ainsi deux instances d'unité de calcul sur une même puce.

Il s'agit des premiers exemplaires des processeurs «multi-cœurs» qui connaissent aujourd'hui un grand essor. L'approche de la technologie «multi-cœurs» consiste à démultiplier la puissance sur une architecture parallèle, de manière à pouvoir augmenter le nombre d'opérations exécutées simultanément en un cycle d'horloge.

Outre les processeurs «bi-cœurs», il existe aujourd'hui des processeurs à quatre cœurs et plus : quadri-cœurs (Core 2 Quad d'Intel), octocoeurs (UltraSparc T1 et T2 de Sun), etc.

II.8.2. Les machines parallèles

Il s'agit d'un ensemble de processeurs identiques ou quasi identiques. Ces processeurs sont reliés par un réseau d'interconnexion et forment un tout logé dans une pièce au sein d'une même organisation. Les ordinateurs multi-processeurs sont dotés de plusieurs processeurs identiques. Ces processeurs participent au traitement parallèle de plusieurs flots d'exécution d'une même ou de plusieurs applications. Deux catégories d'architectures matérielles sont rattachées aux machines parallèles : les ordinateurs multi-processeurs et les grappes. Cette section présente un bref descriptif de ces deux architectures.

II.8.3. Les ordinateurs multi-processeurs

Cette catégorie de matériel inclut les ordinateurs à plusieurs processeurs identiques. Ces processeurs participent au traitement parallèle de plusieurs flots d'exécution d'une même ou de plusieurs applications. Le nombre de processeurs au sein d'un ordinateur peut varier de quelques dizaines à quelques centaines de milliers de processeurs.

Plusieurs types et nominations de machines peuvent être classées dans la catégorie ordinateurs multi-processeurs. On trouve par exemple, les machines vectorielles spécialisées pour le traitement parallèle sur des vecteurs (machines CRAY, Fujitsu, etc.) ou les machines SMP pour Symmetric Multi-Processor (Sun SPARC Enterprise, etc.). On trouve aussi les super-calculateurs comme le BlueGene/L, parue en 2005 avec ses 65 536 processeurs bi-cœurs et un classement premier dans le top 500 des machines les plus performantes dans le monde.

Malgré l'importante puissance offerte par les ordinateurs multi-processeurs, ceux-ci présentent quelques limitations. Leur inconvénient principal est leur coût très élevé. Pour faire face à ce problème, une autre infrastructure matérielle a vu le jour au milieu des années80, nommée « grappe », avec la sortie du « VAXCluster » réalisé par Digital Equipment Corporation.

II.8.4. Les grappes

Une grappe ou cluster en anglais, est tout d'abord un système distribué. Avant de décrire plus en détail les propriétés des grappes, rappelons la définition d'un système distribué telle qu'elle est énoncée par Andrew Tanenbaum dans [23] :

«Un ensemble d'ordinateurs indépendants qui apparaissent à l'utilisateur comme un seul et unique système cohérent. Une grappe est un groupe de PCS (ou stations de travail)

homogènes ou peu hétérogènes. Ces PCs sont interconnectés entre eux par un réseau local (Local Area Network, LAN) tel que Gigabit Ethernet, ou bien par un réseau haute performance (System Area Network, SAN) tel que Quadrics, Myrinet ou Infini Band.»

Une grappe peut être vue comme un super-calculateur virtuel. La différence principale avec les ordinateurs multi-processeurs est la rapidité de communication entre les processeurs. L'augmentation des latences de communication due aux réseaux d'interconnexion dans une grappe est le compromis à accepter contre un moindre coût de l'infrastructure. Cependant l'évolution importante des latences et débits réseaux offre aujourd'hui des performances très satisfaisantes pour le calcul scientifique.

Une grappe est aussi un ensemble de machines indépendantes pouvant être utilisées partiellement ou complètement par un ou plusieurs utilisateurs en même temps. Comparé à un super-calculateur, le partage des ressources dans une grappe n'est pas restreint à quelques utilisateurs et la granularité d'une partition est de l'ordre du processeur.

La gestion du partage des ressources dans une grappe se fait à l'aide de gestionnaires de tâches ou de systèmes SSI (Single System Image). Des exemples de gestionnaires de tâches sont OAR et PBS, etc. Ils ont un rôle et une utilisation similaires à ceux des gestionnaires de tâches utilisés sur les super-calculateurs. Les systèmes SSI quant à eux, tel que Mosix ou Kerrighed, offrent à l'utilisateur la vision d'une ressource unique lui permettant de soumettre des tâches sans se préoccuper de la multiplicité des ressources dans une grappe.

II.8.5. Les grilles de calcul

Le concept de grilles de calcul consiste à exploiter pleinement les ressources de l'intégralité d'un parc informatique (serveurs et PC). C'est une forme d'informatique distribuée, basée sur le partage dynamique des ressources entre des participants, des organisations et des entreprises dans le but de pouvoir les mutualiser, et faire ainsi exécuter des applications de calcul intensif ou des traitements de très gros volumes de données.

II.8.6. Middlewares dédiés aux grilles de calcul

La distribution est le concept clé du grid compting (calcul de grille) [11]. L'objectif du grid compting est de permettre le partage coordonné de ressources et la résolution des problèmes au sein d'organisations dynamiques, virtuelles et multi-institutionnelles. Une grille de calcul offre une infrastructure qui associe ordinateurs, logiciels/middlewares, instruments spéciaux, personnes et capteurs. Une grille est souvent construite à travers un LAN, WAN ou des réseaux dorsaux d'Internet à une échelle régionale, nationale ou mondiale. Les ordinateurs utilisés sont des stations de travail, des serveurs, des clusters et/ou des super ordinateurs.

Comme d'autres systèmes distribués, dans une architecture de type grille le composant le plus important est le middleware. Ce dernier est souvent vu comme un ensemble de composants. Par exemple, le middleware CoreGrid [4] offre des services tels que la gestion

4. `http://coregrid.ercim.eu/mambo/`

de processus à distance, la co-allocation de ressources, l'enregistrement et la découverte d'information, la sécurité et les aspects de qualité de services (QoS) tels que la réservation des ressources. Ces services font abstraction de la complexité et l'hétérogénéité du niveau physique en fournissant une méthode rigoureuse pour l'accès aux ressources distribuées [3]. La couche utilisateur d'un middleware de grilles utilise des interfaces fournies par un middleware de bas niveau pour offrir des services et des abstractions de haut niveau.

II.9. Conclusion

Ce chapitre a été dédié au calcul distribué étant à la base de la naissance du concept du grid computing qui sera introduit dans le chapitre suivant.

III. Grid Computing

III.1. Introduction

Le grid compting est une technologie permettant la collaboration en ligne, la découverte et l'accès à des ressources distribuées. Un mécanisme de grille est essentiellement un middleware ; il s'agit d'une technologie de calcul distribué. Les réseaux haut débit jouent un rôle fondamental afin de rendre le concept de grid compting possible et viable du coté utilisateur.

III.2. Présentation des grilles de calcul

Le terme « grille de calcul » est la traduction de grid computing. Ce dernier vient de l'analogie avec le réseau de distribution d'électricité appelé electrical power grid en Anglais. En effet, on peut rapprocher la puissance de calcul actuelle avec ce qu'était l'électricité au début du XXe siècle. On maitrisait l'électricité et disposait de matériel pour l'exploiter mais chaque personne devait produire sa propre électricité et la consommer sur place. Ce n'est qu'avec le développement du réseau de distribution électrique que la vraie révolution a pu s'opérer en offrant, au plus grand nombre, la possibilité de recevoir et utiliser de l'électricité sans se soucier de la façon ou de l'endroit où elle a été produite. La figure III.1 montre un réseau de distribution électrique, avec plusieurs source de production d'énergie et un consommateur d'énergie, ce consommateur étant physiquement relié à ce réseau va pouvoir utiliser l'énergie (les ressources) mise à disposition sans se soucier de quelle source provient exactement cette énergie.

Cette idée est utilisée dans le calcul de grille informatique. Un utilisateur possédant un ordinateur et nécessitant une énorme puissance de calcul peut, avec le système de grille informatique, soumettre ses tâches à plusieurs "supercalculateurs" enregistrés sur la grille. Actuellement, chaque ordinateur ou unité de calcul utilise la puissance de calcul qu'elle a généré localement. L'idée du grid computing est de mettre à disposition ses ressources en échange d'un accès aux ressources des autres utilisateurs.

III.3. Définition du grid computing

Le concept de grille de calcul étant encore relativement récent en informatique, on peut en trouver de nombreuses définitions que ce soit dans la littérature scientifique ou sur Internet.

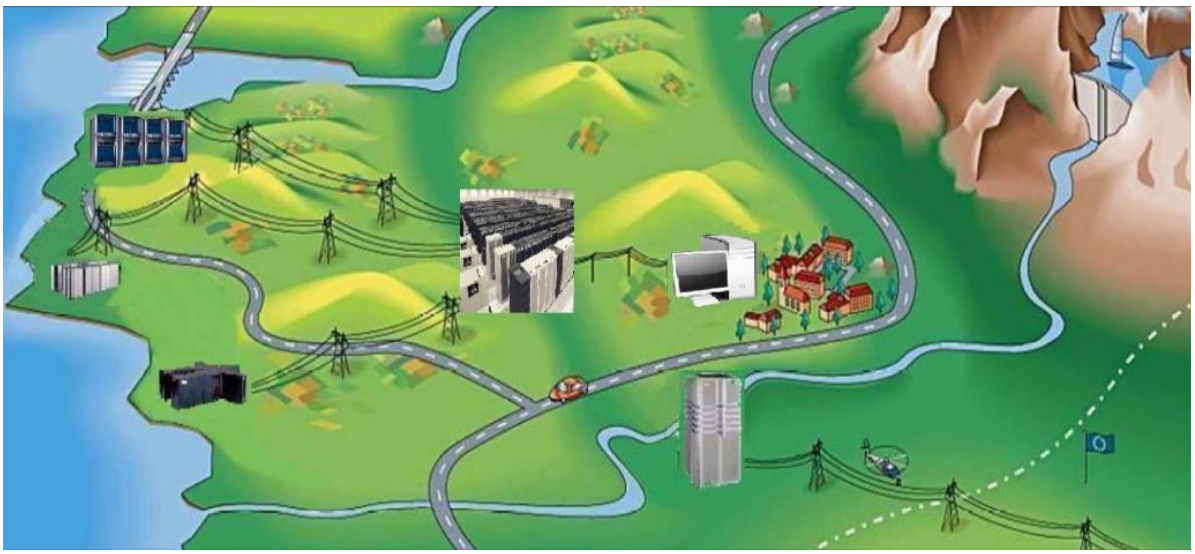

FIGURE III.1. – Le réseau électrique

— **Plaszczak/Wellner** définissent la grille de calcul comme la technologie qui permet la virtualisation de ressource à la demande, et le partage entre plusieurs organisations.
— **IBM** définit le calcul en grille comme une méthode utilisant une panoplie de protocoles standards pour avoir l'accès aux applications et aux données, à la puissance de calcul, à la capacité de stockage entre un vaste réseau de ressources informatiques à travers Internet.
— **Buyya** définit la grille comme un type de système parallèle et distribué qui permet le partage, la sélection, et l'agrégation de ressources autonomes géographiquement distribués dynamiquement. Chacune de ces ressources ont leur propre disponibilité, capacité, performance, coût, et utilisateurs, avec leurs propres contraintes de qualité de service.
— **Le CERN (European Organization for Nuclear Research)**, un des plus gros consommateurs de puissance de calcul à travers la technologie du Grid Computing, la définit comme un service pour le partage de puissance informatique et de capacité de stockage à travers l'Internet.

Mais la définition la plus complète est celle donnée en par **Ian Foster et Carl Kesselman** dans leur livre **"The Grid : Blueprint for a New Computing Infrastructure"** [11]. Cette définition présente le grid compting comme suit :

> *"Une grille de calcul est une infrastructure matérielle et logicielle qui fournit un accès fiable, cohérent, en ligne et peu coûteux avec une grande capacités de calcul."*

ou encore :

> *«Une grille de calcul est un environnement persistant qui permet à des applications logicielles d'intégrer des instruments, des affichages, des données,*

des ressources de calcul et de stockage, gérées par différentes organisations dispersées à travers le monde.»

Il s'agit de fournir, en utilisant les réseaux informatiques, un accès "uniforme et transparent" à un ensemble de ressources : puissance de calcul, logiciels, espaces de stockage, bases de données, etc. A travers cette définition, Foster traite trois points essentiels, à savoir : des ressources informatiques dont leur administration n'est pas centralisée, des méthodes utilisées qui sont standardisées ainsi que des ressources dont la qualité de service n'est pas assurée.

III.4. Ressources des grilles

Une grille est une collection de ressources dont pourra bénéficier l'utilisateur. Une ressource est définie comme étant "une entité partagée qui n'est pas obligatoirement physique et qui est définie en terme d'interface et non en terme de composant matériel".

En d'autres termes, une ressource est tout élément qui permet l'exécution d'une tâche ou le stockage d'une donnée numérique. Cette définition inclut bien sûr les ordinateurs personnels, mais également les supercalculateurs et tout objet qui présente un composant informatique.

III.4.1. Types de ressources des grilles

Les ressources d'une grille sont en général réparties en plusieurs types :

— **Cycles processeur :** c'est l'une des ressources les plus importantes d'une grille. Il existe plusieurs façons d'exploiter les cycles processeurs non utilisés des machines qui composent une grille. La première est de mettre en œuvre une application sur une machine distante au lieu d'utiliser un processeur local (surcharge, performances réduites, etc.).
La seconde consiste à décomposer une application en plusieurs tâches qui seront exécutées, en parallèle, sur différents processeurs. La troisième permet de mettre en œuvre plusieurs occurrences d'une même application sur différentes machines, afin de traiter différentes données plus rapidement.

— **Capacité de stockage :** le second type de ressources disponible dans une grille est la capacité de stockage. En effet, les machines participantes à une grille pourront fournir une partie de la capacité de stockage nécessaire au fonctionnement de la grille. Ainsi, les capacités de stockage de chaque machine dans une grille pourront être utilisées (agrégées) afin d'augmenter la capacité offerte, la performance, l'efficacité de partage et la fiabilité.

— **Equipements spéciaux et logiciels à licence élevée :** certains logiciels dont les prix de licence sont élevés seront présents en quelques exemplaires seulement dans une grille. Ils pourront ainsi être partagés par plusieurs utilisateurs, plutôt que d'acquérir une licence pour chaque utilisateur. Il en sera de même pour certains équipements spéciaux comme les microscopes électroniques et les appareils médicaux, dont les prix sont parfois hors de portée d'organisations individuelles.

III.4.2. Caractéristiques des ressources des grilles

Les ressources sont potentiellement qualifiées de :

•

— Partagées : Elles sont mises à la disposition des différents consommateurs de la grille et éventuellement pour différents usages applicatifs.
— Distribuées : Elles sont situées dans des lieux géographiques différents.
— Hétérogènes : Elles sont de toute nature, différant par exemple par le système d'exploitation ou le système de gestion des fichiers.
— Coordonnées : Les ressources sont organisées, connectées et gérées en fonction de besoins (objectifs) et contraintes (environnements). Ces dispositions sont souvent assurées par un ou plusieurs ordonnanceurs.
— Externalisées : Les ressources sont accessibles à la demande chez un fournisseur externe.
— Non-contrôlées (ou autonomes) : Les ressources ne sont pas contrôlées par une unité commune. Contrairement à un cluster, les ressources sont hors de la portée d'un moniteur de contrôle.

III.5. Architecture des grilles

Le concept de départ des premiers systèmes de grille était d'utiliser les ressources qui pourraient comprendre les ressources de calcul, de stockage et de réseau à partir de plusieurs institutions réparties géographiquement. Ces ressources sont généralement hétérogènes et dynamiques. Les grilles ont focalisé sur l'intégration des ressources existantes avec leurs infrastructures, leurs systèmes d'exploitation, leur gestion des ressources locales ainsi que leur matériels de sécurité. A cet effet, les réseaux fournissent des protocoles et des services à cinq couches différentes que nous présentons dans la Figure III.2. La *couche Fabric* est la couche inférieure de la pile incluant différents types de ressources telles que des ressources de calcul, stockage, réseaux, etc. La *couche connectivité* définit les noyaux de communication et protocoles d'authentification pour les transactions. La *couche ressources* définit les protocoles de publication, de découverte, de négociation, de pilotage et de payement des opérations de partage des ressources individuelles tandis que la *couche collectivité* capte les interactions à travers des collections de ressources et services répertoires. La *couche application* comprend toutes les applications des utilisateurs construites à partir de protocoles et d'APIs et opère dans des environnement d'organisations virtuelles [5].

III.5.1. Types de grilles

Plusieurs classifications des environnements basés sur les grilles peuvent être adoptées selon différents critères. Dans cette section, nous présentons une typologie des grilles basée sur l'utilisation des grilles. Cette classification divise les grilles en cinq types [12] :

— **Grille de calcul :** Ce type de grilles fournit un accès sécurisé à des ressources de calcul capables de résoudre des problèmes de calcul qui autrement auraient requis

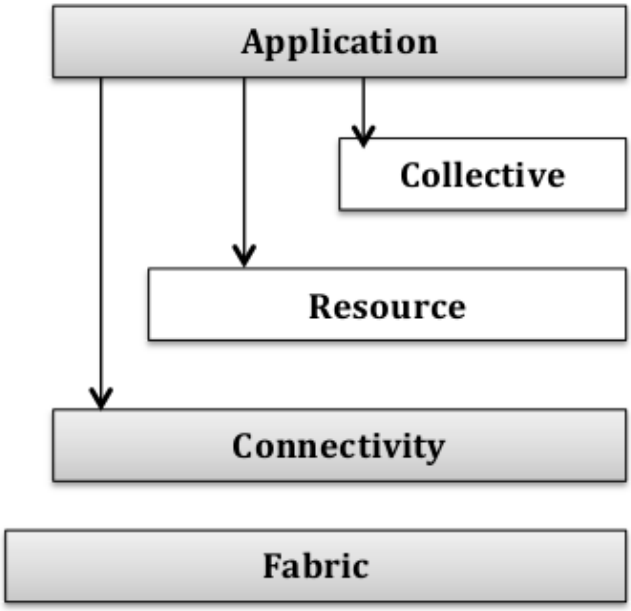

FIGURE III.2. – Architecture des grilles [11]

des capacités de calcul qui sont impossibles à obtenir avec des machines classiques, incluant les machines parallèles et distribuées.

— **Grille de collaboration :** Avec le progrès dans technologies matérielles et logicielles et dans les services d'Internet, la demande d'une meilleure collaboration a augmenté. Une telle collaboration peut être obtenue par ce type de grilles
— **Grille d'utilité :** Dans ce type de grilles, pas seulement les cycles CPU sont partagés mais aussi les logiciels et les périphériques spéciaux comme les capteurs.
— **Grille du réseau :** Si les réseaux de communication ont des débits faibles, les grilles ne peuvent pas être utilisées de façon optimale malgré la grande capacité de calcul et la robustesse des machines. Les grilles de réseau fournissent une communication à haute performance en utilisant la capture de données entre les noeuds d'une grille.
— **Grille de donnée :** Deux notions sont importantes dans ce type de grilles : données et calcul sur ces données. Les grilles de données fournissent un support à très forte capacité de stockage de données ainsi que les services liés aux données tels que : découverte, prise en charge, publication, etc.

III.6. Technologies associées au calcul de grille

La plupart des chercheurs considèrent le calcul de grille comme une évolution et non une révolution. Les grilles de calcul représentent une évolution récente des notions de et complète de développements communs, y compris (mais pas seulement) le calcul distribué, du calcul pair-à-pair et des technologies de metacomputing [17]. Dans ce qui

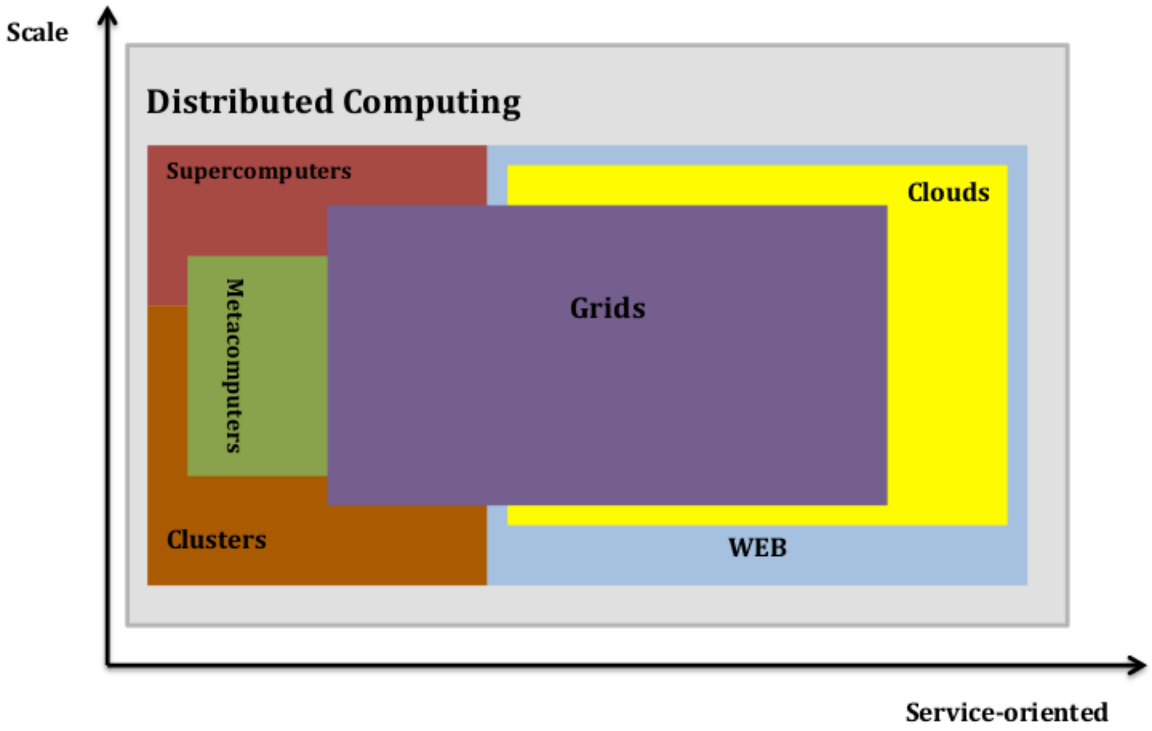

FIGURE III.3. – Les grilles de calcul au coeur des technologies du calcul distribué

suit, nous présentons les étapes les plus signifiantes dans l'émergence du concept de grilles, à savoir, cluster et metacomputing (voir Figure III.3) [25].

III.6.1. Clusters

Les clusters permettent à un ensemble d'ordinateurs autonomes indépendants interconnectés à travers un réseau, de travailler ensemble comme une seule ressource informatique intégrée. Un cluster est local et tous ses composants et sous-systèmes sont supervisés comme un seul système informatique. Les clusters ont donné des résultats impressionnants dans la prise en charge d'applications gourmandes avec de grands ensembles de données grâce à leurs composants [16]. L'architecture d'un cluster est illustrée par la Figure III.4.

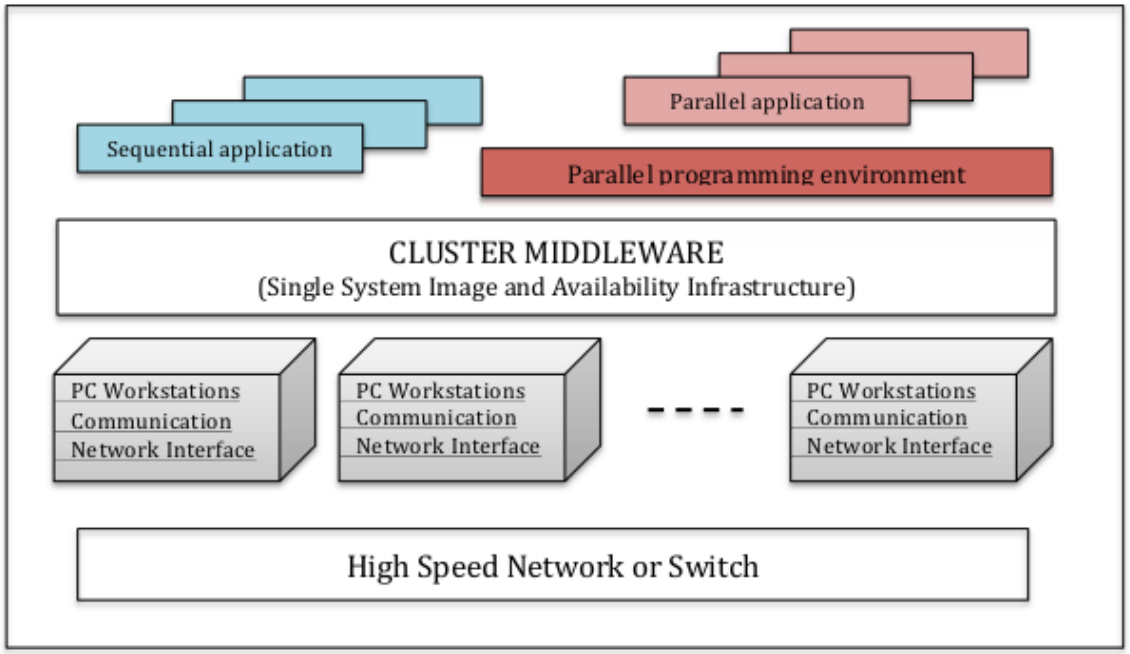

FIGURE III.4. – Achitecture d'un Cluster

III.6.2. Metacomputing

La motivation pour le metacomputing était différente de celle pour les clusters. L'idée du metacomputing est de pouvoir interconnecter une collection d'ordinateurs maintenus ensemble de telle sorte que pour un utilisateur individuel, cette collection parait et agit comme un seul ordinateur. Dans un environnement de métacomputing, les superordinateurs et les grappes jouent le même rôle que les microprocesseurs dans un superordinateur.

Le concept clé du metacomputing n'est défini par ses composants individuels mais plutôt par la manière ces composants collaborent pour réaliser une tâche. Nous présentons néanmoins, à un niveau abstrait, les composants de base de tout système basé sur le concept de métacomputing :

- Processeurs et mémoires aui sont les composants de tout système informatique et qui lui fournissent ses capacités de calcul et de stockage.
- Réseaux et logiciels de communication représentant les connections physiques entre ordinateurs ;
- Environnements virtuels, comme les systèmes d'exploitation sont utilisés pour configurer, gérer et maintenir un meta-ordinateur.
- L'accès et l'extraction de données à distance permettant aux ordinateurs d'interagir les uns avec les autres via des réseaux de communications nationaux et/ou internationaux .

III.6.3. Cloud computing

Le paradigme Cloud Computing a émergé suite à la convergence de nombreux modèles, technologies et concepts associés à la notion de service. Un cloud peut être mis en œuvre sur un centre de données, une collection de clusters ou un système distribué hétérogène composé de PC, stations de travail et serveurs.

Dans [13], les auteurs ont fait une comparaison exhaustive entre le grid et le cloud computing et ont abordé la question de savoir si le cloud n'était pas un nouveau terme pour désigner le grid computing. Les auteurs considèrent que les deux technologies ont la même vision en augmentant la fiabilité et la flexibilité tout en fournissant des offres aux mêmes besoins, mais les problèmes qui sont abordés sont très différents. La définition d'un Cloud comme donnée dans [13] est la suivante :

> *"Un paradigme de calcul distribué à grande échelle porté par les économies actuelles, dans lequel sont gérées des puissances de calcul et de stockage abstraites, virtualisées et dynamiquement évolutives et où des services sont livrés à la demande à des clients externes via Internet."*

Dans [4], les auteurs donnent autre définition qui capture principalement les aspects du cloud computing comme suit :

> *"Le Cloud compting est une manière orientée utilité et centrée sur Internet pour délivrer, à la demande, des services. Ces services couvrent tous les aspects liés à un calcul informatique : de l'infrastructure matérielle emballée comme un ensemble de machines virtuelles aux services logiciels comme des plateformes de développement ou des applications distribuées. "*

En d'autres termes, le Cloud Computing est un paradigme de calcul qui implique l'externalisation des ressources informatiques avec des capacités de ressources extensibles et évolutives et un approvisionnement à la demande avec peu ou pas de coûts initiaux d'investissement.

III.7. Conclusion

Dans ce chapitre, les principales technologies et architectures dédiées au Grid Computing ont été présentées. Ce chapitre présente l'idée et l'historique des grilles de calcul, leur définition, leur architecture, leures caractéristiques ainsi que leures technologies associées.

IV. Développement & Architectures Orientés Service

IV.1. Introduction

Depuis leurs premières années, les architectures orientées service notamment les grilles de calcul sont en constante évolution marquée essentiellement par le développement des middlewares.

Ce cours aborde ce type d'architectures car elles ont révolutionné les environnements de type Grid. En effet, dès les 1ères générations des Middlewares de Grille, la convergence entre les deux architectures a eu lieu et des grilles orientées service sont nées. ce chapitre décrit tous les concepts lié au développement et architectures orientées service.

IV.2. Calcul orienté service

Le paradigme du calcul orienté service SOC (Service-Oriented Computing) fait référence à un ensemble de concepts, principes et méthodes qui représentent le calcul au sein d'une architecture orientée service, appelée SOA (Service-Oriented Architecture), dans laquelle des applications logicielles sont construites à la base avec des composants (sous forme de services) indépendants via des interfaces standards. L'idée principle du SOC/-SOA est de séparer explicitement entre l'ingénierie des logiciels et la programmation.

IV.2.1. Propriétés du calcul orienté service

De nos jours, les services web fournissent les technologies les plus adéquates pour rendre possible la création d'une architecture orientée service. Le développement d'applications utilisant les SOA requiert l'adoption d'un projet orienté service qui focalise sur les exigences déterminées en termes de stratégies et processus métier. le SOC représente les architectures et systèmes distribués basés sur le concept de services caractérisé par les propriétés suivantes :

- **Vue logique :** Un service est une vue logique d'un système.
- **Orienté message :** La communication entre un fournisseur et un agent est définie en terme d'échanges de messages.
- **Orienté description :** Un service est défini par des métadonnées.
- **Granularité :** Un service communique en utilisant un nombre réduit de messages.
- **Orienté réseau :** Les services sont souvent utilisés à travers un réseau. Cependant, ce n'est pas une exigence absolue.

— **Plateforme neutre :** Les services communiquent en utilisant des messages codés en une représentation indépendante de la plateforme. C'est le cas par exemple CORBA qui utilise CDR (Common Data Representation) comme représentation de données ou XML.

IV.2.2. Eléments du calcul orienté service

Le SOC divise le développement de logiciels en trois parties indépendantes : Concepteurs d'applications (ingénieurs de logiciels), fournisseurs de services (programmeurs) et ordonnateurs de services (organisations, sociétés,etc.).

— **Fournisseurs de services :** Ils utilisent un langage de programmation traditionnel tel que Java, C++ ou C afin de développer les composants d'un programme. Tous les composants seront enveloppés dans des interfaces standards, appelées services ou services web si ces derniers sont disponibles sur internet, de telle sorte que les développeurs d'applications pourraient simplement utiliser les services sans communication préalable avec les fournisseurs de services. Les mêmes services peuvent être utilisés dans plusieurs applications.
— **Courtiers de services :** Ils permettent aux services d'être enregistrés et publiés. Ils aident les constructeurs d'applications à trouver les services dont ils ont besoin.
— **Constructeurs d'applications :** Au lieu de construire des logiciels à partir des constructions de base des langages de programmation.

IV.3. Service comme paradigme

À l'origine, la technologie des services web a été lancée par IBM et Microsoft [21], puis normalisée par le W3C (World Wide Web Consortium) [1], l'organisme responsable de la normalisation des évolutions du web. Aujourd'hui, cette technologie est acceptée par les acteurs de l'industrie de l'informatique. Les services web peuvent être utilisés pour mettre en œuvre une architecture orientée service, mais seulement en respectant les propriétés décrites ci-dessus (vue logique, message orienté service, etc.)

Les organismes qui ont contribué à la normalisation des services web sont le W3C, l'OASIS (The Organization for the Advancement of Structured Information Standards) et le WS-I (Web Services Interoperability Organization). L'organisation WS-I prépare des outils et des directives pour aider les développeurs à créer des logiciels correspondant aux normes de services web. WS-I fournit de nombreux produits, tels que les profils, les logiciels de démonstration et les outils de test. Le WS-I fait maintenant partie de l'OASIS. Le W3C est responsable du développement de nombreuses normes, tels que XML, WSDL, SOAP et WSA. OASIS est responsable de l'élaboration de normes comme UDDI, WS-Security et, plus récemment, WSBPEL (Web Services Business Process Execution Language) [22].

Beaucoup de langages et protocoles sont impliqués dans la gestion des services. L'ensemble XML, SOAP, WSDL et UDDI est le plus adapté pour être utilisé avec les SOA.

1. `http://www.w3.org/`

— **XML (eXtensible Markup Language)**, un langage à balises pour formater et échanger des données structurées.
— **SOAP (originally Simple Object Access Protocol** (mais techniquement ce n'est plus un acronyme), un langage basé sur XML pour spécifier l'enveloppe et le contenu d'un message. Même si un service web peut supporter n'importe quel protocole de communication et peut offrir à ses clients un choix, le plus commun est SOAP par rapport aux protocoles HTTP ou HTTPS.
— **WSDL (Web Services Description Language)**, un language basé sur XML, utilisé pour décrire les attributs, les interfaces et d'autres propriétés des services web. Un document WSDL peut être lu par un client potentiel afin d'obtenir certaines informations sur un service.
— **UDDI (Universal Description, Discovery and integration)**, il définit une méthode universelle pour les entreprises pour découvrir dynamiquement et invoquer les services web.

IV.4. Architecture orientée service

L'architecture orientée services (SOA) est une approche architecturale permettant la création de systèmes basés sur un ensemble de services développés dans différents langages de programmation, hébergés sur différentes plates-formes, avec des modèles de sécurité différents et des processus métier. Chaque service représente une unité autonome pour le calcul et la gestion des données ainsi que la communication avec ses environnements par le biais de messages.

L'idée principale d'une architecture orientée service est que chaque élément du système d'information doit devenir un service. Ce service est identifiable, documenté, fiable, indépendant des autres services, accessible et capable d'effectuer un ensemble de tâches bien définies.

Une architecture SOA peut alors être considérée comme un style architectural qui met l'accent sur la mise en œuvre de composants que les services modulaires qui peuvent être découverts et utilisés par les clients. Dans [19], les auteurs proposent une vue étendue de la SOA qui répartit la fonctionnalité sur trois plans : les fondations de service au fond, le service composition dans le milieu, et la gestion des services et suivi au dessus. Comme le montre la Figure IV.1, les auteurs estiment que cette séparation logique est basée sur la nécessité de distinguer :

— Les capacités des services basiques fournis par des middlewares, infrastructures et des SOA à partir de fonctionnalités avancées des services nécessaires à la composition dynamique des services.
— Les processus métier des services centrés système.
— La composition de services de la gestion des services.

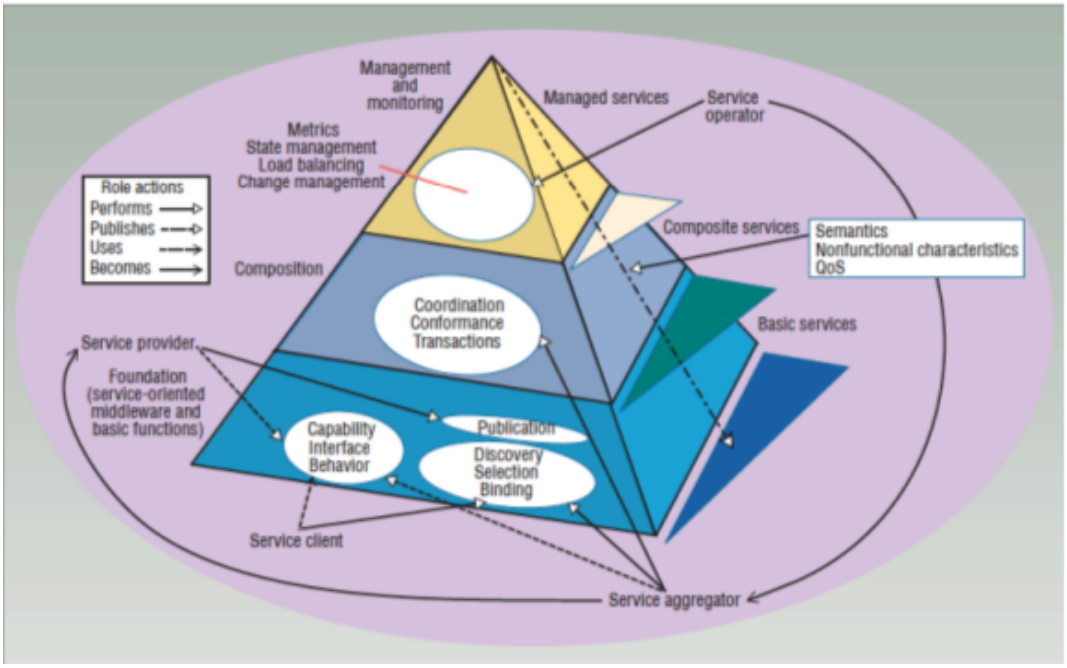

FIGURE IV.1. – Architecture orientée service [19]

IV.5. Les services web

Les services web sont des applications auto descriptives, modulaires et faiblement couplées qui fournissent un modèle simple de programmation et de déploiement d'applications, basé sur des normes s'exécutant à travers l'infrastructure web. Ils sont définis par un ensemble de standards qui permettent de décrire des interfaces logicielles et d'accéder aux fonctions correspondantes sur un réseau utilisant des messages XML.

Les motivations pour conduire à la création des services web ont été variées, parmi ces motivations : l'intégration d'applications "à gros grain» et la définition d'une unité d'intégration (le service). L'apport principal des services web est la solution au problème d'hétérogénéité par rapport aux plates-formes et langages des applications. Les éléments qui constituent la mise en oeuvre d'un service web sont :

— La description du service.
— Le protocole d'accès.
— L'annuaire des services.

IV.5.1. Définition d'un service

Un service web est concrètement un ensemble de fonctionnalités affichées sur un réseau. Ces fonctionnalités sont bien définies et ne dépendent pas du contexte ou des états des services. En d'autres termes, il s'agit d'un composant logiciel accessible à travers intranet, extranet et internet via des technologies web et des systèmes standards de messageries

basés sur XML. Selon la W3C [2], un service est défini comme suit :

> *"Les services web sont complémentaires à d'autres programmes et applications existantes, développés dans des langages de programmation différents et servant comme un pont permettant à ces programmes de communiquer entre eux."*

Les services possèdent généralement les caractéristiques suivantes :

- Les services peuvent être utiles individuellement, comme ils peuvent être intégrés (composés) pour fournir des services de niveau supérieur. Entre autres avantages, ceci favorise la réutilisation de fonctionnalités existantes.
- Les services communiquent avec leurs clients par échange de messages. Ils sont définis par les messages qu'ils peuvent accepter et les réponses qu'ils peuvent donner.
- Les services peuvent participer à un workflow, où l'ordre dans lequel les messages sont envoyés et reçus affecte le résultat des opérations effectuées par un service. Cette notion est définie comme *la chorégraphie des services*.
- Les services peuvent être complètement autonomes ou ils peuvent dépendre de la disponibilité d'autres services, ou sur l'existence d'une ressource comme une base de données. Dans le cas le plus simple, un service pourrait effectuer un calcul sans avoir besoin de se référer à une ressource externe. Inversement, un service qui effectue la conversion de devises aurait besoin d'accéder en temps réel aux informations de taux de change pour obtenir des conversions correctes.
- Les services annoncent des détails tels que leurs capacités, les interfaces, les politiques et les protocoles de communication pris en charge. Les détails de mise en œuvre tels que langage de programmation et la plateforme d'hébergement sont d'aucun intérêt pour les clients et ne sont pas révélés.

IV.5.2. Interactions des services dans une SOA

Généralement, les services ne sont pas exécutés de manière isolée, mais en coopération avec d'autres services (par exemple en échangeant des messages). A cet effet, le service d'interaction est organisé par la SOA.

Au sein d'une SOA, la communication est facilitée grâce à un *courtier de services*. Ce dernier Par conséquent, le fournisseur, demandeur et le courtier exécutent les trois commandes suivantes : Un fournisseur de services envoie des informations au courtier de service, indiquant comment un demandeur de services peut utiliser son service. Le courtier de service, par la suite stocke cette information avec l'ID de l'opérateur dans un référentiel. Cette opération est appelée *publie* l'information reçue d'un fournisseur de services, puis *trouve* le service adéquat à la requête du demandeur de services. Enfin, le demandeur et fournisseur *se lient* et exécutent conjointement leurs services respectifs.

La Figure IV.2 illustre un cycle d'interactions simples au sein d'un environnement orienté service selon les opérations décrites précédemment.

2. `http://www.w3.org/w3c`

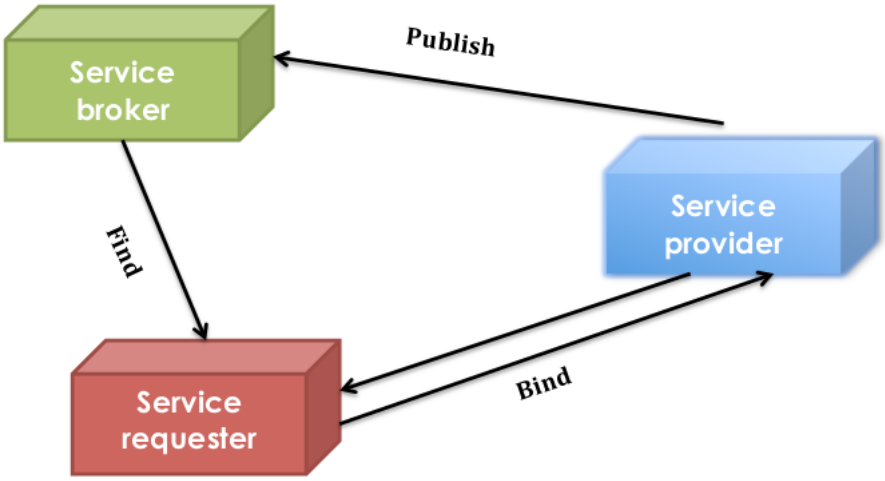

FIGURE IV.2. – Les services et leurs interactions

IV.5.3. Description du service

Le langage utilisé pour décrire un service web est le WSDL (en anglais, Web Services Description Language) qui est la notation standard basée sur XML pour construire la description de l'interface d'un service [6]. Cette spécification définit une grammaire XML pour décrire les services web comme des ensembles de points finaux de communication (ports) à travers lesquels on effectue l'échange de messages. La description des messages et des opérations est faite d'une manière abstraite.

La structure d'un fichier WSDL est présentée sur la figure IV.3 ci-dessous. Un fichier WSDL contient donc 5 éléments [9] :

- **Types :** fournissent la définition de type de données utilisées pour décrire les messages échangés.
- **Messages :** représentent une définition abstraite (noms et types) des données en cours de transmission.
- **PortTypes :** décrivent un ensemble d'opérations abstraites. Chaque opération a zéro ou un message en entrée ; zéro ou plusieurs messages de sortie ou de fautes.
- **Binding :** spécifie une liaison entre un portType et un protocole concret (SOAP, HTTP, MIME).
- **Service :** contient un ensemble de ports. Chacun de ces ports a un type portType

IV.5.4. Protocole d'accès aux service web

Pour accéder aux services web, l'architecture est basée sur le protocole SOAP, le standard qui permet la communication entre applications pour l'accès aux services web. Il permet la normalisation des échanges de données (décrits en XML) et les transferts de données en utilisant des protocoles au niveau application (HTTP, FTP, SMTP) [26].

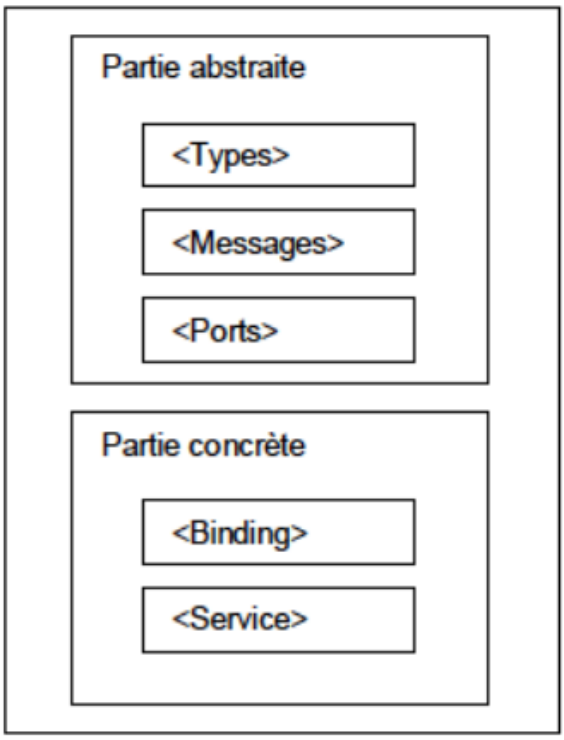

FIGURE IV.3. – La structure d'un document WSDL

Les éléments de SOAP sont présentés dans la Figure IV.4

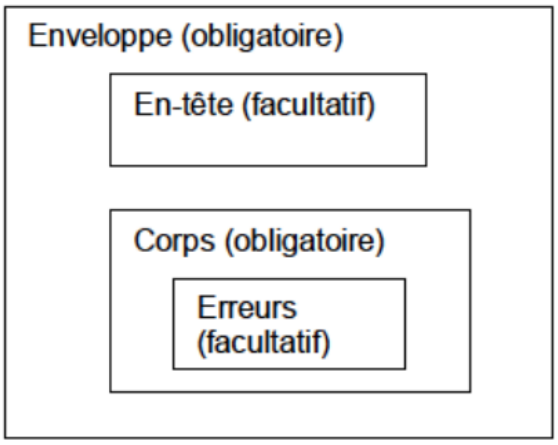

FIGURE IV.4. – Les éléments de SOAP

— Enveloppe (*envelope*) : contient la spécification des espaces de désignation (namespace), du codage de données (version du schéma XML-SOAP supporté), etc.
— En-tête (*header*) : contient une signature pour la sécurité et des informations de facturations. L'en-tête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.
— Corps (*body*) : contient les éléments principaux du service : les méthodes et les paramètres qui seront exécutés par le destinataire final.
— Erreurs (*fault*) : réactions en cas d'erreur (ayant des causes différentes).

Un message SOAP voyage du SOAP sender au SOAP receiver, en passant par un groupe de SOAP intermédiaires, définit comme une entité qui est capable de recevoir et transmettre les messages SOAP. Les noeuds intermédiaires aussi bien que le SOAP receiver sont identifiés par un URI (Uniform Resource Identifier) [2]. Le protocole SOAP permet d'invoquer un objet distant en transmettant via HTTP les informations nécessaires à l'appel (nom de la méthode et sa signature) dans un message au format XML. La réponse à la demande est aussi renvoyée au format XML. Après la mise en forme de la demande SOAP, elle est envoyée au serveur web. Le parseur XML du serveur vérifie la cohérence du message avant de le transmettre sur http. De même, le parseur XML du serveur destinataire vérifie la cohérence du message reçu et le refuse s'il n'est pas valide [27].

IV.5.5. Annuaire des services

L'annuaire de services correspond au lieu où les services web sont enregistrés. L'architecture utilise le standard UDDI [24] (en Anglais, Universal Description, Discovery and Integration) qui représente le protocole pour l'enregistrement et la découverte des

services web. Il fournit une spécification permettant de faciliter la collaboration entre des partenaires dans le cadre des échanges commerciaux.

UDDI fonctionne à partir de pages [14] :

— Blanches : information sur les fournisseurs de services (par nom)
— Jaunes : information sur les fournisseurs de services (par catégorie)
— Vertes (spécificité de UDDI) : définition des services fournis en WSDL.

Les entreprises publient les descriptions de leurs services web en UDDI sous la forme de fichiers WSDL. Ainsi, les clients peuvent plus facilement rechercher les services web dont ils ont besoin en interrogeant le registre UDDI.

Lorsqu'un client trouve dans UDDI une description du service web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI. Ensuite, à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service web, le client peut invoquer le service web et lui demander d'exécuter certaines de ses fonctionnalités.

La Figure IV.5 présente le schéma général du protocole UDDI. L'entreprise B a publié le service web S et l'entreprise A est client de ce service [24].

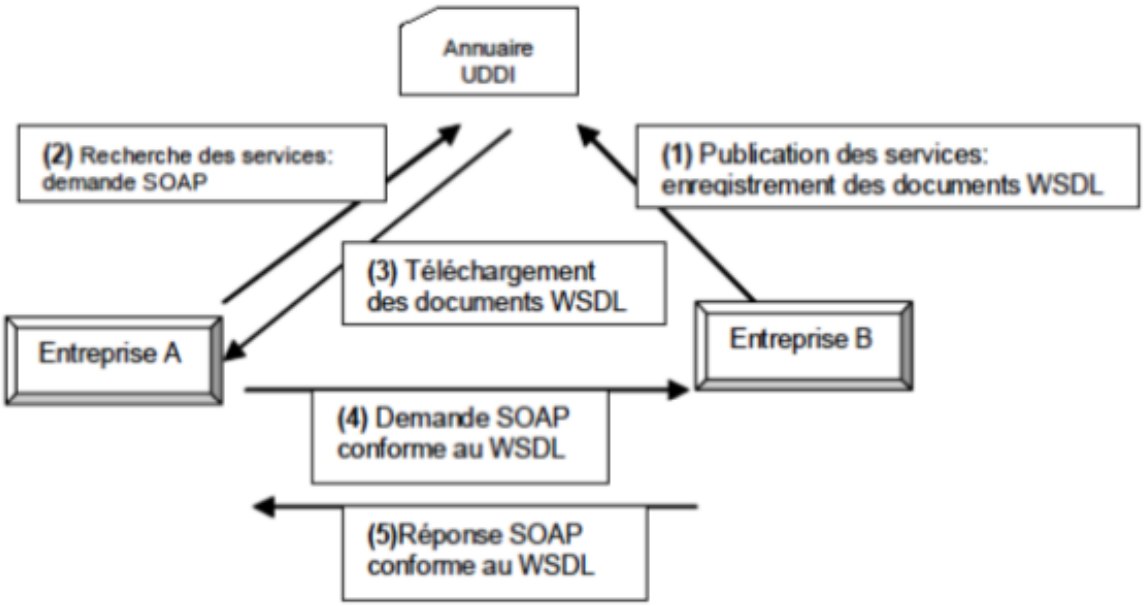

FIGURE IV.5. – Le schéma général de UDDI

IV.6. Conclusion

ce chapitre s'adresse à l'un des concepts les plus importants qui ont fait brillé les grilles de claucl et par la suite un semble de technologies assiciés telles que le cloud. En effet, le

grid computing a connu ses meilleurs jours grace à la aconvergence avec le calcul orienté service.

V. Déploiement des Services dans les Grilles de Calcul

V.1. Introduction

La force de la technologie grid computing réside dans l'ouverture de ses environnements aux autres technologies de l'informatique actuelle. Ceci rend les grilles sujettes d'amélioration sans cesse croissante. Ce chapitre présente les principales générations largement inspirées et nées des architectures orientées service et qui ont marqué l'évolution des grilles de calcul.

V.2. Evolution des grilles de calcul

Depuis leur naissance, les grilles de calcul ont connu une évolution très importante. Cette dernière est caractérisée essentiellement par l'intégration des Middleware dans la technologie du grid computing. Néanmoins d'autres facteurs ont marqué cette évolution faisant apparaitre trois générations principales des grilles de calcul :

V.2.1. La première génération

La première génération permettait de soumettre des travaux en traitement différé sur une grille, souvent après avoir placé manuellement les programmes et données sur les nœuds disponibles. Cette architecture est toujours utilisée, ne serait-ce que pour bâtir des grilles dites légères, cet adjectif soulignant leur facilité d'implantation et d'administration.

Deux grands projets ont marqué cette génération de grilles, à savoir FAFNER (Factoring via Network-Enabled Recursion) et I-WAY (Information Wide Area Year). Le projet FAFNER a gagné un prix dans le « TeraFlop challenge at Super-computing 95 (SC95) » et a ouvert la piste vers les projets de méta-computing basé sur le Web. Tandis que I-WAY était orienté application. Il en a défini plusieurs types (super-computing, l'accès distant aux ressources, le web, etc.) Plus tard, les applications et logiciels développés par I-Way ont été intégrés dans le projet GLOBUS.

V.2.2. La deuxième génération

De nouveaux défis ont apparu avec cette 2ème génération, notamment :

— Le passage à l'échelle : une grille pourra passer de quelques dizaines de ressources à des millions voire des dizaines de millions de ressources de nature très diversifiées. Cette dimension relative à l'échelle d'une grille pose de nouvelles contraintes sur les applications et les algorithmes de gestion des ressources, qui n'existent pas pour les systèmes distribués (échelle plus réduite).
— L'hétérogénéité des ressources : les ressources dans une grille sont de nature hétérogène en termes de matériels et de logiciels.
— L'existence de plusieurs domaines administratifs : les ressources sont géographiquement distribuées et appartiennent à différentes organisations, chacune ayant ses propres politiques de gestion et de sécurité. Ainsi, il est indispensable de respecter les politiques de chacune de ces organisations.

Cette génération apporte un outil essentiel d'automatisation dans l'usage de grilles, le gestionnaire de ressources (Resources Broker). Celui-ci dispose de l'état de l'ensemble des ressources offertes par la grille, ce qui lui permet d'allouer celles demandées par les scripts de lancement des travaux et disponibles, voire de les réserver préalablement, avant ordonnancement puis exécution. Le projet Globus a vu le jour pendant cette période et a donné naissance au Globus Toolkit 1 (GT1) et GT 2. Légion est, également, un système de cette génération.

V.2.3. La troisième génération

La troisième génération se met en place. Les points majeurs caractérisant cette génération sont :

— la généralisation de la notion d'organisation virtuelle permettant de définir des sous-ensembles de ressources et leurs modalités d'usage par des communautés d'utilisateurs, et ceci indépendamment des notions de comptes et d'appartenance juridique.
— le passage d'un mode accès à des ressources d'une grille vers la fourniture de services de grilles (web et grid services)
— L'intégration des services réseaux : protocoles plus performants pour le multicast, les transferts massifs de données pour les messages type MPI (Message Passing Interface) ainsi que l'allocation de bande passante à la demande.
— l'interopérabilité via une normalisation des services web appliqués aux grilles : OGSA (Open Grid Service Architecture) et WS-RF (Web Services Resource Framework).

V.3. Standards pour le déploiement de services dans des grilles de calcul

Depuis les premières générations des grilles, il s'averait nécessaire d'établir des normes ouvertes qui définissent l'interaction entre les ressources hétérogènes dans un environnement de grille et encouragent l'interopérabilité entre les composants. Cela permet

l'interopérabilité au sein d'un environnement hétérogène, en supportant le partage des ressources entre les organisations.

V.3.1. Open Grid Service Architecture.

L'Open Grid Service Architecture (OGSA) a été spécifiée par le groupe de travail du forum Grid Forum mondial (GGF) en Juin 2002. OGSA régit les concepts et définit la capacité du grid computing. OGSA a précisé les exigences sur lesquelles un environnement de grille comme la virtualisation des ressources et de la gestion, la découverte de ressources, les services à état et organisations virtuelles. Les objectifs de l'OGSA se résument dans : la gestion des ressources, la qualité de services (QoS) ainsi que la définition d'interfaces publiques pour l'interoperabilité et la tolérence aux pannes [10].

V.3.2. Open Grid Service Infrastructure.

L'Open Grid Service Infrastructure (OGSI) a été développé par l'OGF (Open Grid Forum) afin de fournir une couche *Infrastructure* pour l'OGSA. OGSI est un ensemble de spécifications avec une extension spécifique des interfaces WSDL d'un service tel que GWSDL (Grid WSDL). En particulier, ces spécifications sont des interfaces de service pour un accès virtuel aux ressources. OGSI prévoit également des mécanismes pour définir les services à état et certaines fonctionnalités de gestion. Le Globus Toolkit 3 a été le premier environnement de développement de grille qui implèmente les spécifications OGSI. L'OGSI a introduit quelques interfaces et conventions qui peuvent être résumées en [7] : plateforme, cycle de vie, gestion de l'état, groupes de services, etc.

V.3.3. Web Service Resource Framework.

Le Web Service Resource Framework (WSRF) a été présenté en 2004, comme un framework ouvert pour la modélisation et l'accès aux ressources à état utilisant les services web. Le WSRF définit comment les normes de service web évoluent pour se rapprocher des éléments et exigences des services de grille. WSRF représente une refactorisation de l'OGSI à travers six spécifications : WS-Resource Properties, WS-Resource Lifetime, WS-Renewable References, WS-Service Group, WS-BaseFault and the WS-Notification.

V.4. Grid service & Web service

Dans ce qui suit, le concept de grid servic est introduit à partir du web service.

V.4.1. La gestion des états des ressources en utilisant les Grid services

Dans le contexte des grilles, une ressource est supposée représenter quelques états ou données et fournit des capacités utiles par l'intermédiaire d'une interface.

Une interface liée à une ressource définit un groupe d'opérations logiques qui peuvent être appelées par ses clients et grâce aux avantages des architectures orientées services, les ressources de la grille peuvent exposer leurs services à travers une interface standard.

Cependant, comme ceux qui sont familiers avec les web services le savent, les web services sont en général **sans état**. C'est-à-dire, il n'y a aucune mémoire entre les transactions séparées appelées sur la même instance du service. Alors que pour la grille de calcul, l'état d'une ressource ou le service est souvent importante.

L'OGSI décrit les Grid services en prenant les web services comme base. Cela signifie d'exploiter :

- Les mécanismes pour l'encodage des protocoles de transmission de messages qui sont décrits comme des liaisons dans les Web Services. Il ya plusieurs façons de coder des messages, et le Web Services sépare ces préoccupations à partir des définitions XML des interfaces d'applications : OGSI standardise des interfaces au niveau des applications, indépendamment des liaisons.
- Les conventions utilisées dans les services Web afin de séparer les interfaces d'application primaires (appels de fonctions et leurs paramètres) à partir de fonctions qui peuvent être gérées par normalisés, les intergiciels génériques. Cela inclut des questions telles que l'authentification et contrôle d'accès.
- Les techniques utilisées dans les services Web pour séparer les fonctions de gestion du réseau de l'interface de l'application. Les questions de gestion comprennent la charge de travail, l'équilibre entre performances et diagnostic du problème.

V.4.2. Qu'est-ce qu'un Grid service ?

Une interface de service liée à une ressource de grille est connue comme Grid service. Ce dernier peut commander, contrôler une ressource et son état. Un Grid service peut exiger l'accès à plus d'une ressource ou vice versa. Il est également possible que multiples Grid services accèdent à la même ressource ou peuvent créer une nouvelle instance d'une ressource chaque fois qu'elle est appelée.

Les diverses ressources d'une grille peuvent nécessiter afin d'agir et s'intégrer entre elles. Souvent, ces ressources sont hébergées dans un environnement technologiquement hétérogène. Par conséquent, un Framework est requis sont gérés pour faire l'abstraction des détails de l'implémentation des ressources d'un environnement spécifique. L'architecture orientée service fournit un tel Framework.

Une telle architecture nous aiderait à partager des ressources distribuées à travers des organisations virtuelles hétérogènes et dynamiques, qui est la grille de calcul. Le GGF (Global Grid Forum) a adopté l'OGSA qui fournit un Framework pour mettre en œuvre une grille basée les standards de la SOA (architecture orientée service). Toutes les ressources (physiques ou logiques) dans une grille conforme à OGSA sont modélisées comme étant des Grid services.

V.4.3. Comparaison entre Grid service et web service

Bien que les grid services sont implémentés en utilisant la technologie « web service », il y'a une différence fondamentale entre un Grid service et un web service.

Un Web service a pour rôle la découverte et l'invocation des services persistants. Un document WSDL conforme pointe sur l'emplacement qui héberge le web service.Un Grid service s'intéresse aux ressources virtuelles et la gestion de leurs états. Une grille est un environnement dynamique. Par conséquent, un Grid service peut être transitoire (provisoire), plutôt que persistant. Il peut être dynamiquement créé et détruit, contrairement à un Web service, qui est souvent présumé disponible si le fichier WSDL correspondant est accessible à ses clients.

Ceci a des implications importantes sur la façon dont les Grid service sont gérés, nommés, découverts et utilisés. Le modèle OGSA adopte un modèle de conception pour créer des services transitoires de la grille. Ainsi, un Grid service est un service Web potentiellement transitoire basée sur les protocoles de la grille en utilisant WSDL.

V.5. Globus Toolkit : Le grand standard des grilles de calcul orientées services

Dans le mond du Grid computing, Globus constitue sans doute le leader. Il s'agit de la grille qui a révolutionné le calcul distribué à grande échelle et a constitué le noyau de développement d'un grand nombre de grilles par la suite.

V.5.1. La Globus Aliance et le Globus Toolkit

La Globus Alliance est une "Organisation virtuelle" qui a pour but d'aider le monde scientifique, la recherche et les entreprises à créer des applications basées sur le Grid Computing.

L'architecture de grille permet à plusieurs entités de collaborer, de mettre en commun leurs ressources informatiques. Etant une organisation virtuelle, la Globus Alliance s'étend du laboratoire national d'Argonne(Etats-Unis, Illinois) en passant par l'université de l'institut des sciences de l'information de la Californie méridionale(Etats-Unis), l'université de Chicago(Etats-Unis), le centre national pour les applications de "Super computing" (NCSA, Arlington en Virginie, Etats-Unis), l'université d'Edimbourg (Ecosse), et enfin vers le centre suédois pour les ordinateurs mise en parallèles(Linköping, Suède).

Elle développe et essaie de standardiser la technologie qu'est le Grid Computing. Par son application vedette le "Globus Toolkit". Comme son nom l'indique c'est une boîte à outils (modules, bibliothèques) développée en C et en Java, qui permet de créer une architecture de grille et de développer des applications basées sur la technologie du Grid Computing.

Le Globus Toolkit est à la base de nombreux projets dans le monde scientifique et des entreprises ce qui représente un budget de un demi-milliard de dollars (en tout pour tous les projets). Comme nous le savons tous, le Web a révolutionné l'accès à l'information.

La Globus Alliance tente de réaliser la même chose en ce qui concerne les applications et les calculs distribués. De nouveaux types d'applications seront mis en œuvre quand l'accès des supercalculateurs, à l'imagerie des satellites, à la mémoire de masse et d'autres ressources en ligne deviendra aussi simple et courant que l'utilisation du Web.

V.5.2. Globus Toolkit

Afin d'acquérir une idée sur les middlewares de grilles, leurs concepts et fonctionnalités, nous présentons dans cette section le projet Globus ainsi que le Globus Toolkit, qui constitue le framework le plus populaire basé sur des standards. Le Globus Toolkit a été développé au laboratoire national d'Argonne au sein de l'université de Chigago par l'équipe de recherche dirigée par Ian Foster. En plus d'être au coeur de nombreux projets scientifiques et d'ingénierie.

Les éléments clé de l'architecture de la Globus Toolkit 4 (la version stable de Globus sortie en 2005) sont présentés dans la Figure V.1.

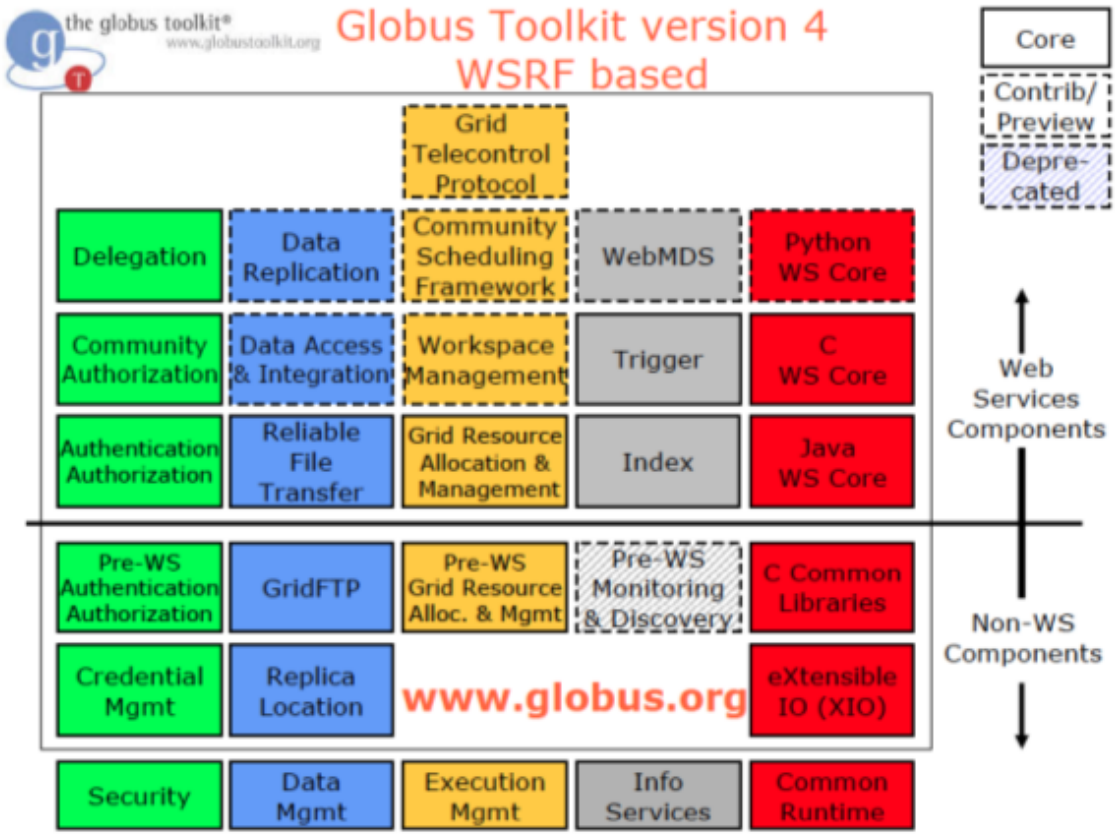

FIGURE V.1. – Architecture du Globus Toolkit 4

Le Globus Toolkit est une architecture ouverte à base communautaire, un ensemble

de services et de bibliothèques de logiciels qui supportent les grilles et les applications grilles. Le toolkit aborde les questions liées à la sécurité, la découverte d'informations, la gestion des ressources, la gestion des données, la communication, la tolérance aux fautes et la portabilité. Les mécanismes du Globus Toolkit sont en cours d'utilisation sur des centaines de sites et par des dizaines de grands projets de grille à travers le monde.

En plus d'être un élément central de projets scientifiques et techniques qui totalisent près d'un demi-milliard de dollars à l'échelle internationale, le Globus Toolkit est une base sur laquelle reposent des sociétés leaders dans la construction de produits importants pour les grilles commerciales. Il a été largement adopté comme une solution de technologie Grid pour le calcul scientifique et technique ainsi que les services web. Ses composants sont souvent au noyau pour de nombreux middlewares de type grid.

V.6. Conclusion

Dans ce chapitre,l'interêt est porté sur le déploiement des services des grilles de calcul. Le déploiement a été expliqué à travers le middleware pionnier des grilles à savoir Globus Toolkit.

VI. Administration : Ordonnancement via les Outils Workflow

VI.1. Introduction

L'objectif de ce chapitre est d'explorer un volet important de l'administration ds grilles de calcul, à savoir l'***Ordonnacement***. Il s'agit principalement de la problématique liée à la fois à l'affectation optimale des ressources de la grille aux diféfrentes applications soumises par les utilisateurs et aussi à la planification de leur exécution tenant compte d'un ensemble de contaraintes souvent classées en contraintes de tolérences aux pannées et à l'équiilibrage de charges dans le monde du Grid Computing. Ce chapitre présente l'ordonancement et le décrit à travers l'utilisation de la technologie Workflow la plus utilisée par les fournisseurs de services de Grid.

VI.2. Limites des grilles sans workflow

La puissance de calcul, la large bonde passante, et la grande capacité de stockage offertes par les environnements de calcul à haut débit comme les grilles, ne seront plus utiles sans

l'existence d'un mécanisme qui gère l'exécution de chaque tâche d'une manière optimale et de façon qui ne freine pas l'exécution des tâches qui dépendent d'elle. Les chercheurs ont développé des algorithmes d'ordonnancement pour atteindre cet objectif. Dans ce chapitre,

nous allons présenter l'ordonnancement des jobs et taches à exécuter sur une grille via les outils workflow.

La première génération des grilles avait comme objectif principal de déterminer les concepts et les architectures de base pour définir des standards dans le domaine des grilles.

Plus tard, les concepteurs des grilles ont tenté de rendre l'utilisation de ces dernières plus faciles et plus transparentes. La dernière génération a été caractérisée surtout par le travail de Foster, qui s'intéresse essentiellement à faciliter la communication entre l'environnement d'exécution de la grille et les utilisateurs.

Parmi les chemins menés par les chercheurs dans ce sens, le développement des middlewares est justement un des thèmes les plus abordés dans cet axe de recherche. Les

applications middlewares sont utilisées comme moyen d'intégration des applications logicielles exécutées sur des environnements distribués hétérogènes.

Dans une grille, le middleware est particulièrement utilisé pour cacher la nature hétérogène et fournir un ensemble d'interfaces standards sous formes de couches logicielles entre les utilisateurs et l'environnement d'exécution de la grille. Ceci a donné naissance à une nouvelle génération de grilles basées sur les middlewares, tels que GLOBUS, LEGION ou UNICORE, les grilles réussissent à satisfaire les besoins de la communauté scientifique d'aujourd'hui en permettant une migration facile des applications des ingénieurs dans le monde des grilles de calcul capables d'offrir une variété de services pour accéder à des ressources hétérogènes et isolés.

Ci-dessous, les principaux services assurés par les middlewares :

— Services d'information, qui permettent la découverte et le pilotage des ressources. L'information fournie peut être utilisée pour trouver les ressources disponibles et sélectionner celles qui sont les plus appropriées pour la tâche à exécuter.
— Services de sécurité, qui permettent l'authentification mutuelle des utilisateurs et des ressources et permettent aux utilisateurs d'accéder aux ressources en se basant sur des politiques locales.
— Gestion des ressources qui établit un ordonnancement des jobs et se charge de l'affectation de ces derniers à des ressources particulières.
— Service de gestion de données qui aident les utilisateurs et les applications à gérer des ensembles de données larges, dupliquées et distribuées.

Néanmoins, et malgré cette importante évolution des grilles en particulier grâce à la convergence des technologies, concepts et outils des grilles vers les standards du Web, ces dernières restent inaccessibles aux simples utilisateurs.

En effet, les grilles de calcul actuelles offrent des plates-formes d'exécution pour répondre aux besoins des applications les plus gourmandes. Mais, pour atteindre un tel degré de performance, les grilles de calcul font appel aux technologies et outils de l'informatiques les plus récents et les plus sophistiqués. Ceci rend les grilles difficiles à manipuler par des utilisateurs simples.

Aussi la grille est caractérisée principalement par la dynamicité et l'hétérogénéité à la fois de l'architecture et des applications. Une hétérogénéité souvent difficile voire impossible à cacher malgré tous les efforts fournis actuellement pour atteindre un haut degré de transparence.

VI.3. Ordonnancement

L'ordonnancement (également connu sous le nom de planification des processus) est une méthode utilisée pour distribuer des ressources informatiques précieuses, généralement le temps du processeur, la bande passante et la mémoire, aux différents processus, threads, flux de données et applications qui en ont besoin.
L'ordonnancement est effectué pour équilibrer la charge sur le système et assurer une

répartition égale des ressources et donner une certaine priorité selon les règles établies. Cela garantit qu'un système informatique puisse répondre à toutes les demandes et atteindre une certaine qualité de service.

VI.3.1. Bases de l'ordonnancement

L'ordonnancement dans un système est fait par un planificateur bien nommé, qui est principalement concerné par trois aspects :

— Le débit, ou la vitesse à laquelle il peut finir un certain nombre de tâches du début à la fin par unité de temps
— La latence, qui est le délai d'exécution ou le temps qu'il faut pour terminer la tâche à partir du moment de la demande ou la soumission jusqu'à la fin, ce qui comprend le temps d'attente avant qu'il puisse être servi
— Le temps de réponse, c'est-à-dire le temps qu'il faut pour que le processus ou la demande soit servi, en gros le temps d'attente

L'ordonnancement est largement basé sur les facteurs mentionnés ci-dessus et varie selon le système et la programmation des préférences et des objectifs du système ou de l'utilisateur. Dans les ordinateurs modernes tels que les ordinateurs avec de grandes quantités de puissance de traitement et d'autres ressources et avec la capacité de multitâche en exécutant plusieurs threads ou pipelines à la fois, la planification n'est plus un gros problème et la plupart du temps les processus et les applications sont donnés librement régner avec des ressources supplémentaires, Mais le planificateur est encore difficile à gérer les demandes.

Les types d'ordonnancement comprennent :

— Premier arrivé, premier servi - L'approche la plus simple et peut être désigné comme premier entré, premier sorti ; Il fait simplement ce que le nom suggère.
— Round robin - aussi connu comme le temps de tranchage, puisque chaque tâche est donnée un certain temps pour utiliser les ressources. C'est toujours sur la base du premier arrivé, premier servi.
— Temps restant le plus court d'abord - La tâche qui requiert le moins de temps pour terminer est prioritaire.
— Priorité - Les tâches sont assignées prioritaires et sont servies en fonction de cette priorité. Cela peut conduire à la famine des tâches les moins importantes car ils sont toujours préemptés par les plus importants.

VI.3.2. Définition de l'ordonnanceur

Un ordonnanceur ou un planificateur, est l'une des principales composantes de l'infrastructure informatique est un produit logiciel qui permet à une entreprise de planifier et de suivre les tâches de traitement par lots informatiques. Ces unités de travail comprennent l'exécution d'un programme de sécurité ou la mise à jour du logiciel. Les planificateurs de tâches peuvent également gérer la file d'attente de travaux pour un cluster d'ordinateurs.

Un planificateur démarre et gère les tâches automatiquement en manipulant un algorithme de langage de contrôle de tâche ou en communiquant avec un utilisateur humain. Les planificateurs de tâches d'aujourd'hui offrent souvent une interface utilisateur graphique (GUI) et un point de contrôle unique pour toutes les tâches d'un réseau PC distribué.

Certains attributs qui peuvent être trouvés dans un planificateur de tâches comprennent :

- Suivi constant et automatique des travaux et notification d'achèvement
- Planification des tâches événementielles
- Surveillance des opérations
- Planification des rapports

VI.3.3. Le Workflow comme ordonnanceur

L'ordonnancement est une méthode d'affectation de tâches à des ressources informatiques pour son exécution. Dans un environnement distribué comme les grilles ou le cloud, différents travaux sont programmés pour différentes ressources d'informatique virtuelle et ces tâches sont exécutées à distance.

Le problème de planification peut être classé en deux types comme problème d'optimisation et problème de décision basé sur l'objectif. Le problème d'optimisation trouve la meilleure solution parmi toutes les solutions possibles alors que le but du problème de décision est simple, il faut un résultat pour analyser si l'objectif est atteint. Clairement, le problème d'optimisation est plus difficile que le problème de décision. Les problèmes de planification appartiennent à une large classe d'optimisation problème vise à trouver une correspondance optimale des tâches à différents ensembles de ressources informatiques.

La planification du workflow est un algorithme qui choisit la ressource appropriée pour l'exécution du lot de tâches de façon efficace à minimiser la durée de réalisation d'un workflow et à satisfaire la qualité de service (QOS) de l'utilisateur.

VI.4. Les techniques de l'ordonnancement du workflow

L'objectif de cette section est de brasser un ensemble de techniques utilisées dans l'ordonnancement via les outils de workflow. Il faut noter que la liste n'est pas exhaustive mais il s'agit d'une sélection d'algorithmes.

VI.4.1. Algorithmes d'ordonnancement heuristiques en mode batch (BMHA) :

Dans BMHA, les tâches sont mises en file d'attente et recueillis dans un ensemble allé ils arrivent dans le système. L'algorithme d'ordonnancement commencera après une bonne période d'idée de l'ère. Les principaux exemples d'algorithmes basés sur BMHA

sont : Algorithme d'ordonnancement First Come First Served (FCFS), algorithme d'ordonnancement Round-Robin (RR), algorithme Min-Min, algorithme Max-Min et algorithme Most-fit.

VI.4.2. Algorithme d'ordonnancement heuristique en mode en ligne :

Les travaux sont ordonnancés en arrière-plan avant de les mettre dans le system. Etant donné que le cloud est un système hétérogène et que la vitesse de chaque processeur varie rapidement, les algorithmes d'ordonnancement heuristique du mode surstock sont plus appropriés à une atmosphère de nuage. La plupart des algorithmes de planification des tâches (MFTF) sont suffisants pour illustrer l'algorithme d'ordonnancement heuristique du mode On-stock.

VI.4.3. Un modèle de planification de tâches guidées par la QoS :

Ce modèle se compose de quelques stratégies d'ordonnancement et d'un algorithme heuristique de Suffrage-min guidé par ordonnancement QoS. Ce modèle inclut le niveau de QoS des ressources et des tâches. La stratégie de ce modèle est basée sur le partitionnement. Les tâches et les ressources ne parlent pas en charité de deux niveaux, d'abord est QoS niveau élevé et le second niveau de QoS est faible. Ce modèle a la porte d'ordonnancement de remplacement pour les deux niveaux. Ce modèle réduit la valeur Makespan et équilibre la charge de travail.

VI.4.4. Une planification de tâches multi-files efficace :

Un algorithme de planification multi-file d'attente [MQS] réduit le coût de la réservation et de la poursuite concernant les plans de demande à l'aide du planificateur global. La méthodologie proposée est basée sur le concept de regroupement des tâches en fonction de leur éclatement leur temps de rupture. La privation de vie et la fragilité se retrouvent dans les méthodes habituelles, en tenant compte de FCFS, SJF. Pour surmonter ces problèmes, la planification des files d'attente multiples est introduite. Pour combiner la poursuite judiciaire de l'algorithme d'ordonnancement MQS utiliser l'ensemble errant l'aspect inutilisé. Dans cet algorithme, la sélection des tâches est effectuée dynamiquement pour obtenir l'encombrement optimal de la planification et, par conséquent, elle résout la meule de fragmentation.

VI.4.5. Algorithme d'ordonnancement des tâches basé sur la priorité :

Dans cet algorithme, la priorité est affectée à chaque tâche introduite dans le système. La priorité est fondée sur la référence à la théorie du processus analytique. Modèle de

décision Multicritères [MCDM] et un modèle de décision Multi-Attributs [MADM] sont les modèles de base pour définir la priorité pour les tâches. La comparaison des tâches est terminée et les rideaux au milieu de la technique de matrice de comparaison. Puisque beaucoup de comparaisons sont faites le long ainsi que des tâches de la profondeur de l'algorithme est élevé. La priorité est basée durement les trois niveaux, le premier est le niveau objectif, le deuxième est le niveau d'attribut et le troisième niveau est le niveau alternatif.

VI.4.6. Algorithme de planification basé sur le modèle Berger :

La planification des tâches est terminée et s'est terminée au cours de l'examen des caractéristiques de la communication et de la virtualisation. Deux contraintes sont appliquées dans cet algorithme. La première contrainte est la classification des tâches utilisateur par les préférences QoS, et établit l'attente générale prendre des mesures. Cette classification améliore l'équité dans le processus de sélection des ressources. La deuxième contrainte est de décrire l'équité des ressources que la justice accomplit pour juger de l'équité de l'allocation des ressources. Ainsi, sur la base de Berger modèle de planification est faite grossièrement le système de justice et la justice des tâches contraintes d'équité. Le résultat expérimental de cet algorithme montre la meilleure équité.

VI.5. Composition des services workflow

Un des défis des SOA est l'intégration de services ou de systèmes distribués pour la fourniture de nouveaux services personnalisés, plus riches et plus intéressants aussi bien pour l'application actuelle que pour d'autres applications. Si une application ou un client requièrent des fonctionnalités et qu'aucun service n'est apte à les fournir tout seul, il devrait être possible de combiner ou de composer des services existants afin de répondre aux besoins de cette application ou de ce client. C'est ce que l'on appelle la *composition de services*. On peut composer des services entre eux pour ensuite exposer ce "méta-service" sous forme d'un nouveau service, et ainsi de suite récursivement.

La composition de service inclut deux paradigmes. D'une part, l'*orchestration* qui consiste à avoir un service qui joue le rôle de *"chef d'orchestrer"* et qui connaît seul la logique de composition ; les services auxquels il fait appel n'ont pas besoin de savoir qu'ils font partie d'un processus plus gros.

D'autre part, la *chorégraphie* dispense de ce rôle de chef d'orchestre. Dans ce 2 ème paradigme, chaque service doit être au courant de la logique du (ou des) processus au(x)quel(s) il appartient, et doit par exemple savoir que : "quand il reçoit un message du service x, il doit attendre 50 secondes puis envoyer un message au service y". La chorégraphie de service n'est pas facile à mettre en oeuvre ainsi l'orchestration est plus répandue [6]. Ces deux concepts sont présentés dans la figure VI.1

Nous allons présenter dans ce qui suit la définition de ces paradigmes. Nous précisons que dans le cadre de notre travail nous nous sommes basées sur l'orchestration pour composer les services.

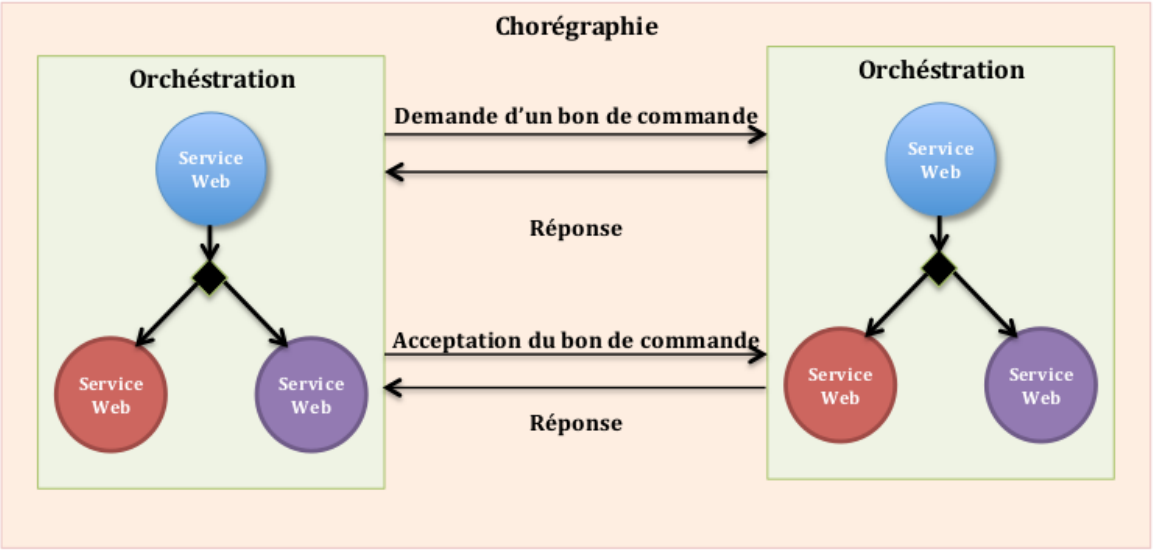

FIGURE VI.1. – La composition des services

VI.5.1. Chorégraphie

La chorégraphie décrit la collaboration entre une collection de services dont le but est d'atteindre un objectif donné via des échanges ordonnés de messages [1]. Elle décrit deux aspects de la composition : (i) un ensemble d'interactions qui peuvent ou doivent avoir lieu entre un ensemble de services (représentés de façon abstraite par des règles), et (ii) les dépendances entre ces interactions.

La chorégraphie modélise la séquence des échanges de messages entre les services web et conditions dans lesquelles ces messages sont échangés entre des clients, des fournisseurs et des partenaires. La chorégraphie est typiquement associée à l'échange de messages publics entre les services web, alors qu'un procédé métier est exécuté de manière centralisé [18].

VI.5.2. Orchestration

L'orchestration de services permet de définir l'enchaînement des services selon un canevas prédéfin, et de les exécuter à travers des "scripts d'orchestration". Ces scripts sont souvent représentés par des procédés métier ou des workflows inter/intra-entreprise. Ils décrivent les interactions entre applications en identifiant les messages et en branchant la logique et les séquences d'invocation [20].

L'orchestration décrit la manière dans laquelle les services web peuvent interagir ensemble au niveau des messages, incluant la logique métier et l'ordre d'exécution des interactions. Ces interactions peuvent couvrir des applications et/ou des organisations et le résultat peut être un modèle de procédé de longue durée, transactionnel, et multi-

étapes [28].

VI.5.3. Chorégraphie et orchestration : La Comparaison

Une différence importante entre l'orchestration et la chorégraphie est que l'orchestration offre une vision centralisée, c'est-à-dire que le procédé est toujours contrôlé de la perspective des partenaires métier. En revanche, la chorégraphie offre une vision globale et plus collaborative de la coordination. Elle décrit le rôle que joue chaque participant impliqué dans l'application.

En orchestration, les services web impliqués sont sous contrôle d'un seul point central (un service web particulier). Ce processus coordonne l'exécution des différentes opérations sur le service web participant dans ce processus. Les services web impliqués ignorent complètement dans quelle composition de processus ils sont impliqués ou qu'ils jouent un rôle dans la définition d'un processus métier. Seulement le service web central (le coordinateur de la composition) est sensé avoir ces informations. Par conséquent, l'orchestration est centralisée autour de définitions et opérations explicites et un ordre d'invocation de service web (voir la Figure VI.2).

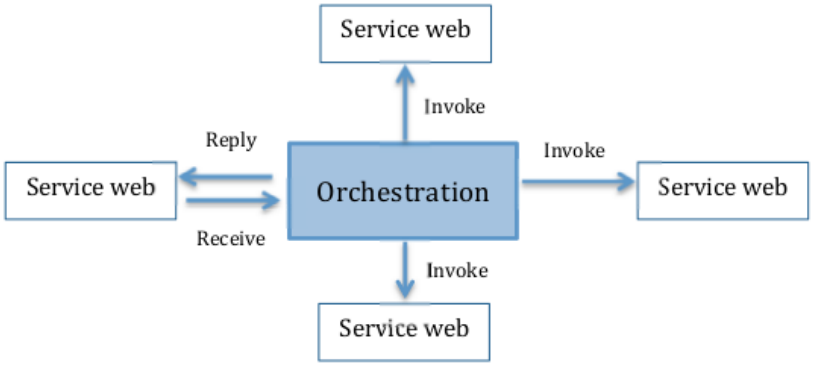

FIGURE VI.2. – L'orchestration

A l'opposé, la chorégraphie ne dépend pas d'un chef d'orchestre central. Chaque service web participant dans la chorégraphie doit savoir exactement quand il doit devenir actif et avec qui inter-opérer.

La chorégraphie est basée sur la collaboration et elle est principalement utilisée pour l'échange de messages dans des processus métier publiques. Tous les services web impliqués doivent être consciencieux du processus métier, des opérations à exécuter, des messages à échanger ainsi que du moment d'échanger ces messages (voir Figure VI.3). Comparée à la chorégraphie, l'orchestration est plus sécurisée et plus efficace quand il s'agit de composer des processus métier ou scientifiques. En effet, elle présente les avantages suivants :

— La coordination du processus est gérée par un composant spécifique.

— Les services web peuvent être incorporés sans se soucier de la possibilité de faire partie d'un processus plus grand.
— Des scénarios alternatifs peuvent être mis en place en cas de panne.

VI.6. Langages de Composition de service

Le langage WSDL est destiné à la description d'un service web simple ou service web basique. Alors que l'évolution des échanges de types B2B (Business to Business) et B2C (Business to Consumer) a entrainé la nécessité de concevoir des services web plus intelligents et plus appropriés pour la prise en charge de cet aspect de composition. Un certain nombre de langages métier ou langages de composition a été développé dans cette perspective. Parmi ceux-ci, on peut citer : ebXML, XLANG, WSFL ainsi que le langage BPEL [15] et ses variantes. Dans ce qui suit, nous présentons ces langages qui ont marqué l'évolution de la composition des services web.

VI.6.1. ebXML

Electronic Business using eXtensible Markup Language (ebXML) a été développé par le United Nation Center for Trade Facilitation and Electronic Business (UN/Cefact) et l'organisation OASIS (Organization for the Advancement of Structured Information Standard). Son objectif était de créer un marché électronique unique et global.

ebXML est une norme d'infrastructure qui vise à fournir aux entreprises non seulement un référentiel commun (structure d'annuaires répartis) recouvrant d'ailleurs une partie des fonctionnalités de UDDI, mais aussi une méthode de description des processus métiers. Par contre, il ne spécifie pas le type ou le contenu des transactions des processus métiers.

Les concepts clés de ebXML sont :

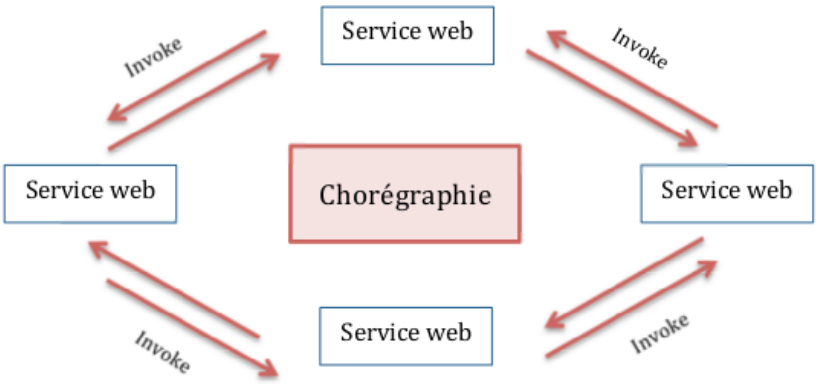

FIGURE VI.3. – La chorégraphie

— **Collaboration Métier :** c'est un ensemble de transactions d'affaires. ebXML autorise les collaborations binaires (par exemple un acheteur - un vendeur) et multiparties (par exemple : un vendeur - plusieurs acheteurs).
— **Transaction Métier :** elle représente une unité de travail dans un arrangement entre deux partenaires commerciaux (le demandeur et le répondeur).
— **Chorégraphie :** elle décrit l'ordonnancement des transactions et les transitions qui permettent le passage d'un état à un autre.
— **Patrons :** ebXML fournit un ensemble d'éléments sémantiques non ambigus réutilisables et pouvant être intégrés dans la description des transactions et des collaborations.

Le langage ebXML présente des qualités comme : sa spécification dans le domaine des affaires (échanges commerciaux), l'implémentation de la chorégraphie ainsi que l'utilisation du standard XML. Malgré ses avantages, ebXML présente des inconvénients qui sont : la non-adaptabilité, la non-extensibilité et le bas niveau d'expressivité l'empéchant de devenir un standard

VI.6.2. XLANG

XML Business Process Language (XLANG) est un langage créé par Microsoft, utilisé principalement dans BizTalk Server. Il fournit un modèle d'orchestration des services et des contrats de collaboration entre services. Pour la représentation du workflow entre les participants, XLANG utilise des actions, lors de la définition des processus. Ces actions incluent les types d'opérations WSDL (requête/réponse, sollicitation, et la notification) ainsi que d'autres actions : arrêts (date limite et durée) et exceptions.

XLANG couvre les points suivants :
— flux de contrôle séquentiel et parallèle ;
— transactions longues ;
— corrélations des messages entre eux ;
— gestion des défaillances et des erreurs ;
— découverte dynamique des services ;
— contrats multipartites.

Les avantages de XLANG se résument dans son utilisation dans l'environnement BizTalk, le fait qu'il soit édité par un concepteur reconnu (Microsoft) ainsi que sa capacité d'orchestration des flux nécessaires à l'accomplissement d'un processus de type workflow (travail collaboratif). Néanmoins, la principale limite de XLANG est qu'il s'agit plus d'un format de définition de processus que d'un format d'interface de processus. Ce manque d'interface rend XLANG difficile à exploiter.

VI.6.3. WSFL

Web Services Flow Language (WSFL) est une spécification basée sur XML pour la description de la composition de services web en tant qu'élément d'une définition de processus métier. WSFL a été conçu par IBM pour faire partie du cadre des techniques de services web et complète les outils existants comme SOAP, WSDL, XML et UDDI. Il

décrit deux manières de composer des services web : (i) un processus métier exécutable nommé *Flow Model* qui est la description explicite de la succession des étapes et de l'enchaînement des appels aux opérations des services web et (ii) une collaboration métier nommée *Global Model* qui est le modèle d'interactions de services web pris deux à deux (le contrat).

Les avantages de WSFL se résument dans son utilisation dans les processus de type workflow (travail collaboratif) et dans le fait d'être édité par un concepteur reconnu (IBM). Cependant il présente les mêmes inconvénients que XLANG et n'a pas pu être standardisé bien qu'il était portable. Sa non extensibilité a abouti à la création de BPEL.

VI.6.4. BPEL

Avant que les deux derniers langages (XLANG et WSFL) soient normalisés par le W3C (World Wide Web Consortium), leurs concepteurs ont créé le langage *BPEL4WS (Business Process Execution Language for Web Services)* ou tout simplement BPEL [15]. Le langage BPEL est le dernier né du domaine. Il représente une référence du domaine. BPEL représente un langage de programmation standard aidant les entreprises à définir la manière de combiner des services web (services web basiques) en vue de mener à bien certaines tâches (services web agrégés ou complexes). La section suivante décrit rigoureusement le langage BPEL étant le langage de composition que nous avons adopté pour la composition des service workflows dans JASMIN.

VI.7. BPEL pour la composition des service

Le Business Process Execution Language 4 Web Service (BPEL4WS), tout simplement BPEL ou encore WS-BPEL est un langage de programmation impératif pour le domaine spécifique de la définition des processus métier exécutables.

Contrairement à la programmation conventionnelle, les processus métier sont généralement de longue durée, des programmes concurrents dont le contrôle et les flux de données sont différemment spécifiés et interprétés par un moteur de workflow.

L'accent est mis sur la programmation à deux niveaux : Ici, les services sont d'abord mis en œuvre dans un langage de programmation classique, comme Java et mis à la disposition d'autres utilisateurs par le biais d'une interface. Ces services peuvent être à un niveau d'abstraction plus élevé d'applications d'entreprise complexes "orchestrées".

BPEL est également classé dans le groupe de langages d'orchestration de services. BPEL est principalement utilisé pour le processus de back-end tels que les processus métier qui peuvent être effectués de manière automatique sans intervention manuelle (traitement de fond).

Avec les dernières extensions BPEL4People et WS-Human Task, les actions manuelles peuvent être insérées dans le processus, dans lequel le traitement en amont est possible. Les deux approches peuvent aussi être mélangées.

Lors de la conception du langage, le consortium du développement et de la standardisation des services Web a défini BPEL comme la technologie de base pour l'identification

et l'intégration des services.

Il s'agit d'une technologie très flexible qui interagit avec des standards tels que : SOAP et WSDL. BPEL n'est pas un langage de programmation mais un langage de modélisation qui supporte la configuration des services web et la logique des processus métier.

La spécification BPEL distingue explicitement entre deux niveaux de conception de processus : processus abstraits et processus exécutables.

VI.7.1. Structure d'une spécification BPEL

La description BPEL d'un processus métier se fait sur trois grands aspects représentés en trois blocs constituants un document BPEL, tel que le montre la Figure VI.4.

FIGURE VI.4. – La structure d'une spécification BPEL

Ces blocs sont les suivants [15] :

— **Bloc B1 :** Ce bloc permet de déclarer des variables, utilisant les types importés de descriptions WSDL associées à la description BPEL. Ces variables seront utilisées dans la partie de description comportementale du processus métier pour maintenir un état au niveau du service et ainsi assurer des liaisons de données entre deux opérations. En d'autres termes, le processus BPEL a un état. Cet état est maintenu par des variables contenant des données. Ces données sont combinées afin de contrôler le comportement du processus. Elles sont utilisées dans les expressions et les opérations d'affectation. Les expressions permettent d'ajouter des conditions

de transition ou de jointure au flux de contrôle. L'affectation (*assignment*) permet de mettre à jour l'état du processus, en copiant les données d'une variable à une autre ou en introduisant de nouvelles données en utilisant les expressions.

— **Bloc B2 :** Ce bloc permet la déclaration des *partnerlinks* : ces derniers définissent une liaison entre les ensembles d'opérations des services invoqués par le service BPEL et un nom de liaison bien précis du service BPEL. Ces liaisons sont appelées *partner* et permettent ainsi d'identifier facilement les différents services utilisés. Un partnerlink correspond au service avec lequel le procédé échange des informations. Le partnerlink représente la relation de conversation entre deux procédés partenaires. Chaque partnerlink est typé par un *partnerLinkType*, il est chargé de définir le rôle que joue chacun des deux partners dans une conversation.

— **Bloc B3 :** Le bloc B3 décrit le comportement du processus métier lui-même en utilisant différents opérateurs dits de programmation. Afin de supporter ce comportement, BPEL est basé sur deux types d'activités : les activités de base et les activités structurées.

VI.7.2. Activités dans une spécification BPEL

Les activités décrites dans une spécification donnée représentent la partie la plus importante dans une structure BPEL. Ces activités représentent le flux de travail dans un processus. Elles sont classées en deux catégories : les activités basiques et les activités structures.

Activités basiques de BPEL

Les activités de base sont :

— **L'activité vide *<empty>* :** elle permet d'insérer une opération vide dans un processus. Ceci signifie que le processus ne fait rien, mais ne se termine pas complètement : l'instruction suivante sera exécutée. Ce processus est très utilisé dans le cadre de la synchronisation des activités parallèles.

— **L'activité de terminaison *<terminate>* :** elle force le processus à terminer immédiatement.

— **L'activité d'affectation *<assign>* :** elle permet d'affecter un nouveau contenu à une variable.

— **L'activité déclenchement d'une exception *<throw>* :** elle permet de lancer une erreur d'exécution.

— **L'activité de délai *<wait>* :** elle permet d'attendre un temps donné (for) ou jusqu'à une certaine heure (until).

— **L'activité de reception *<receive>* :** elle permet au processus métier d'attendre qu'un message de jointure arrive. Elle se termine quand ce message arrive.

— **L'activité réponse *<reply>* :** elle permet à un processus de répondre à un message qu'il aurait reçu par un receive. La combinaison d'un receive et d'un reply permet de réaliser une opération de question/réponse tel que le définit WSDL, en mode synchrone.

— **L'activité d'invocation d'un service *<invoke>* :** permet d'invoquer un web service partenaire (défini dans la partie partners vue précédemment). Cette invocation peut être réalisée de manière asynchrone.

Activités structurées de BPEL

Les activités structurées sont constituées d'activités de base liées entre elles pour donner naissances à des traitements plus complexes. Un des points forts du langage BPEL est la prise en charge de ces activités complexes qui ne sont rien d'autre que les règles de routage des applications workflow. Ces activités peuvent être :

— **Une séquence :** Elle représente l'activité permettant d'exécuter de façon séquentielle (dans l'ordre de lecture de la séquence) une ou plusieurs instructions BPEL (mot clé activity). La séquence se termine elle-même lorsque le dernier processus la constituant se termine.

— **Un choix :** Il permet, comme en programmation structurée, d'effectuer un branchement conditionnel : suivant l'évaluation d'une condition ou d'un critère (attribut condition), le processus de la première branche (noeud case) répondant à ce critère sera exécuté. Dans le cas où aucune condition ne serait vraie, il est possible de définir une branche par défaut (noeud otherwise) qui sera alors exécutée.

— **Une boucle :** Elle permet de réaliser une boucle while traditionnelle : la boucle (composée des instructions représentées par le mot clé activity) est exécutée tant que la condition (attribut condition) est évaluée à vrai. Tout comme dans le cas du switch, l'évaluation se passe de manière silencieuse pour les clients et/ou partenaires du service.

— **Un parallélisme :** permet d'exécuter en parallèle plusieurs processus (mot clé *activity*). Ces différents processus peuvent être indépendants, et alors, le processus englobant l'exécution parallèle se termine lorsque tous ses processus sont terminés. Ou alors, les processus peuvent être dépendants les uns des autres, et l'utilisation de liens (*links*) permet de synchroniser où d'ajouter des dépendances temporelles entre les différents processus.

VI.7.3. Infrastructure de BPEL

L'infrastructure typique de BPEL (*Bpel infrastructure*) consiste en plusieurs composants. La Figure VI.5 montre une architecture référentielle d'une telle infrastructure. Sur le coté gauche, sont placés les outils dédiés aux développeurs mais qui servent aussi pour l'interaction avec les utilisateurs. Au début, le processus est modélisé en utilisant un outil de modélisation qui cache au maximum les détails sur le service web et sur la plate-forme d'exécution. Le résultat de cette étape est un ensemble de fichiers BPEL, WSDL et XSD. Les fichiers BPEL contiennent les modèles de processus, les fichiers WSDL contiennent l'interface et les fichiers XSD décrivent les types de données. Ce paquetage est ensuite envoyé au moteur de BPEL qui exécute les processus. Cette étape est nommée déploiement. A ce niveau, quelques spécificités du système et de l'environnement d'exécution sont introduites.

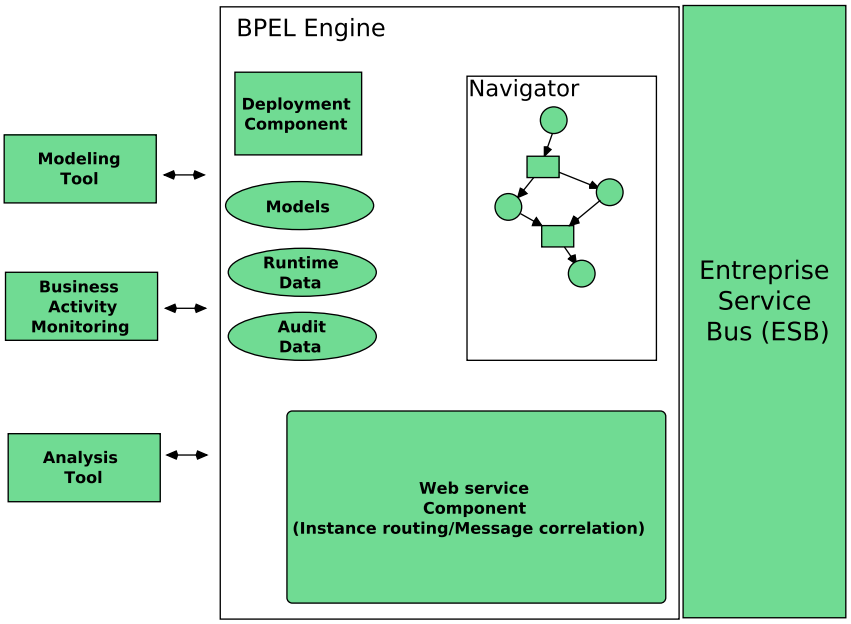

FIGURE VI.5. – L'Infrastructure de BPEL

Le moteur de BPEL vérifie si tous les éléments requis sont valides ; sauvegarde le paquetage de déploiement dans la base de modèles où plusieurs versions d'un modèle peuvent être stockées quand celui-ci est très long et un nombre important d'instances de modèles du processus sont déployées.

A la fin du déploiement, le moteur attend l'arrivée des messages de lancement de l'instance ; le navigateur démarre les activités du processus. L'état actuel est sauvegardé dans la base d'exécution et chaque étape de navigation est écrite comme un événement dans l'audit. Ces événements contiennent par exemple : la création d'une instance, la lecture/écriture d'une variable ou la documentation sur les entrées/sorties d'une activité,

Le moteur interagit avec le monde extérieur via par exemple le *Enterprise Service Bus* (ESB) qui assume l'implémentation des appels des services via des protocoles de transport et qui transmet tous les messages de et vers le moteur. Les données de la base d'exécution représentent toujours l'état actuel de l'instance du processus.

Les outils de pilotage peuvent y accéder et créer des rapports d'exécution qui seront traités par les outils d'analyse et de l'audit afin de tirer des conclusions sur la performance. .

VI.8. Conclusion

Ce chapitre s'interesse à l'administation des grilles de calcul et en particulier à l'ordonnancement de services en utilisant la technologie workflow.

Conclusion générale

Ce support de cours s'inscrit dans le domaine des systèmes distribués à large échelle et les systèmes orientés service. Il s'agit d'un support de cours sur le Grid Computing. Ce cours décrit de façon exaustive tous les concepts, architectures, définitions et sytèmes de type Grid mais aussi met l'accent sur les architutures et technologies connexes comme l'orienté service ou le workflow.

VII. Installation et Déploiement de Service de Globus

Nous rappelons que toutes nos expérimentations ont été lancées à partir de machines munies avec le système d'exploitation Linux (distribution Fedora 10).

1. Les pré-requis

— Globus Toolkit installer. Il est disponible sur (`http://toolkit.globus.org/toolkit/`).
— J2SE 1.5.0+ SDK.
— Ant 1.6.2 et plus (`http://jakarta.apache.org/ant`).
— C compiler.
— GNU Tar (`http://www.gnu.org/software/tar/tar.html`).
— GNU Sed (`http://www.gnu.org/software/sed/sed.html`).
— zlib.1.4.1+ (`http://www.gzip.org/zlib/`).
— GNU Make (`http://www.gnu.org/software/make/`).
— Sudo (`http://www.courtesan.com/sudo/`) pour les fonctionalités basiques de GRAM4.
— Openssl 0.9.7.
— gpt-3.2autotools2004.

2. Ajout de l'utlisateur "globus"

```
$ adduser globus
```

3. Installation de GT4

— Créer un répertoire d'installation

```
$ mkdir /usr/local/globus-4.2.1.1
$ chownglobus:globus /usr/local/globus-4.2.1.1
```

— En tant que l'utilisateur globus, exécuter la commande suivante

```
$ export GLOBUS_LOCATION=/usr/local/globus-4.2.1.1
$ ./configure --prefix=$GLOBUS_LOCATION
```

— Ensuite, executer la commande suivante :

```
$ make
```

— Et enfin

```
$ make install (ceci complete l'installation)
```

4. Configurer le chemin des variables d'environnement

— Modifier le fichier /etc/profile

```
$ vi /etc/profile
```

— Ensuite,ajouter ces lignes

```
export ANT_HOME=/usr/local/apache-ant-1.6.5
export JAVA_HOME=/usr/local/jdk1.5
export GLOBUS_LOCATION=/usr/local/globus-4.2.1
export
PATH=$ANT_HOME/bin:$JAVA_HOME/bin:$GLOBUS_LOCATION
    /bin:$PATH/bin:$PATH
```

5. L'autorité de certification CA

— En ce qui concerne la sécurité, il faut installer l'autorité de certification.

```
$ export GLOBUS_LOCATION=/usr/local/globus/
$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
```

— Ensuite exécuter cette commande :

```
$ $GLOBUS_LOCATION/setup/globus/setup-simple-ca
```

— Et pour obtenir un certificat, effectuer une demande au près de l'autorité de certification installée par la commande :

```
$ grid-cert-request -host `hostname`
```

6. Configuration du GridFTP

— Modifier le fichier /etc/xinetd.d/gridftp

```
$ vi /etc/xinetd.d/gridftp
```

— et écrire les informations suivantes :

```
servicegsiftp
{
instances = 100
socket_type = stream
wait = no
user = root
env += GLOBUS_LOCATION=/usr/local/globus
env += LD_LIBRARY_PATH=/usr/local/globus/lib
server = /usr/local/globus/sbin/globus-gridftp-server
server_args = -i
log_on_success += DURATION
nice = 10
disable = no
}
```

7. Configuration du RFT

— On ouvre le fichier *$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml*, et on change le *userName* par le nom d'utilisateur qui a créé la base de données (dans notre cas c'est globus) et mettre le mot de passe qu'on a indiqué lors de la création de l'utilisateur globus dans *postgresSQL*.

— Pour le GRAM, on ajoute ces deux lignes dans le fichier /etc/sudoers

```
- globusALL=(username1,username2) NOPASSWD: /usr/local/
  globus-4.2.1/libexec/globus-
- gridmap-and-execute -g /etc/grid-security/grid-mapfile /
  usr/local/globus-4.2.1/libexec/globus-job-manager-script
  .pl
- globusALL=(username1,username2) NOPASSWD: /usr/local/
  globus-4.2.1/libexec/globus-
- gridmap-and-execute -g /etc/grid-security/grid-mapfile
```

VIII. Bibliographie

[1] Abdaldhem Albreshne, Patrik Fuhrer, and Jacques Pasquier. Web services technologies : state of the art. University of Fribourg, Department of Informatics, 2009.

[2] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform resource identifiers (uri) : Generic syntax. RFC Editor, United States, 1998.

[3] Jinjun Chen BLizhe Wang, Wei Jie. *Grid Computing : Infrastructure, Service, and Applications*. CRC Press, Taylor & Francis Group, 2009.

[4] Rajkumar Buyya, Christian Vecchiola, and Thamarai Selvi. *Mastering Cloud Computing : Foundations and Applications Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.

[5] Ann Chervenak, Ewa Deelman, Miron Livny, Mei-Hui Su, Rob Schuler, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Data placement for scientific applications in distributed environments. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*, Austin, TX, September 2007.

[6] Word Wide Web Consortium. w3 Official Web Site : `http://www.w3.org`, 2023.

[7] Ewa Deelman, Dennis Gannon, Matthew S. Shields, and Ian Taylor. Workflows and e-science : An overview of workflow system features and capabilities. *Future Generation Comp. Syst.*, 25(5) :528–540, 2009.

[8] George F. Coulouris Jean Dollimore and Tim Kindberg. *Distributed systems : concepts and design*. International computer science series. Addison-Wesley, Harlow, England, New York, 2005.

[9] Jürgen Dorn, Peter Hrastnik, and Albert Rainer. Web service discovery and composition for virtual enterprises. *Int. J. Web Service Res.*, 4(1) :23–39, 2007.

[10] Ian Foster. The physiology of the grid : An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.

[11] Ian Foster and Carl Kesselman. *The Grid : Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[12] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid - enabling scalable virtual organizations. *International Journal of Supercomputer Applications*, 15 :200–222, 2001.

[13] Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. *CoRR*, pages –1–11, 2009.

[14] Deke Guo, Honghui Chen, Xueshan Luo, and Wei Ming Zhang. Enhance uddi and design peer-to-peer network for uddi to realize decentralized web service discovery. In Hamid R. Arabnia, editor, *Proceedings of The 2005 International Symposium on Web Services and Applications, ISWS 2005, Las Vegas, Nevada, USA, June 27-30, 2005*, pages 59–65. CSREA Press, 2005.

[15] Matjaz B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2Nd Edition*. Packt Publishing, 2006.

[16] Brijender Kahanwal and Tejinder Pa Singh. The distributed computing paradigms : P2p, grid, cluster, cloud, and jungle. *CoRR*, abs/1311.3070, 2013.

[17] Tevfik Kosar, George Kola, and Miron Livny. A framework for self-optimising, fault-tolerant, high performance bulk data transfers in a heterogeneous grid environment. In *Proceedings of the Second International Symposium on Parallel and Distributed Computing (ISPDC2003)*, Ljubljana, Slovenia, October 2003.

[18] Guadalupe Ortiz, Juan Hernández, and Pedro J. Clemente. Preparing and re-using web services for choreography. *Int. J. Web Eng. Technol.*, 2(4) :307–334, 2006.

[19] Michael P. Papazoglou and Paolo Traverso. Service-oriented computing : State of the art and research challenges. *IEEE Computer*, 40 :2007, 2007.

[20] Cesare Pautasso. Model-driven service composition with jopera. 29.6.2006 2006.

[21] Marcello Formica Ranieri Baraglia, Gianluca Faieta and Domenico Laforenza. Experiences with a wide area network metacomputing management tool using ibm sp2 parallel systems. *Concurrency : Practice and Experience*, 9 (3) :223–239, March 1997.

[22] David Schumm, Frank Leymann, and Alexander Streule. Process Views to Support Compliance Management in Business Processes. In Francesco Buccafurri and Giovanni Semeraro, editors, *Proceedings of the 11th International Conference on Electronic Commerce and Web Technologies (EC-Web 2010)*, volume 61 of *Lecture Notes in Business Information Processing*, pages 131–142, Bilbao, Spain, September 2010. Springer-Verlag.

[23] Andrew Tanenbaum and Maarten van Steen. *Distributed systems : principles and paradigms*. Pearson Prentice Hall. Pearson Education International, Upper Saddle River, NJ, 2007.

[24] Discovery The Official online community for the Universal Description and Integration Oasis Standard. Universal description, discovery and integration (uddi). `http://uddi.xml.org/`.

[25] Peter Troger, Daniel Templeton, Roger Brobst, Andreas Haas, and Hrabri Rajic. Distributed Resource Management Application API 1.0 - IDL Specification (GFD-R-P.130), April 2008.

[26] W3C. Standard soap. `http://www.w3.org/TR/soap`.

[27] Christian Werner, Carsten Buschmann, and Stefan Fischer. Wsdl-driven SOAP compression. *Int. J. Web Service Res.*, 2(1) :18–35, 2005.

[28] Jia Yu, Srikumar Venugopal, and Rajkumar Buyya. Grid market directory : A web services based grid service publication directory. *CoRR*, cs.DC/0302006, 2003.