TABLE DES MATIERES

INTRODUCTION 1

PARTIE I : ETAT DE L'ART 3

I. Le HTC 4

II. Les grilles informatiques 6

a. Pourquoi les grilles informatiques ? 6

b. Définition 7

c. Applications des grilles 10

d. Architecture générale d'une grille informatique 11

e. Le paysage des grilles 14

f. Impacts sur les réseaux 15

g. Utilisation en entreprise 15

III. Les agents mobiles 17

a. Pourquoi la mobilité ? 17

b. Définition d'un agent mobile 18

c. Particularités des agents mobiles 19

d. Quelques systèmes d'agents mobiles 20

IV. Conclusion 22

PARTIE II : LE SYSTEME CONDOR 23

I. Le système Condor 24

II. Fonctionnalités 25

a. Les classAds 26

b. Matchmaking 27

c. Checkpointing et Migration 29

d. DAGMan 30

III. Architecture 31

a. Les démons du Condor 32

b. Le demon Condor en action 33

IV. Les univers de Condor 34

PARTIE III : NOTRE ETUDE 36

I. Mécanismes de tolérances aux pannes dans Condor 37

a. Le checkpoint 37
b. DAGMan 37
II. Etude des défaillances à gérer dans Condor 38
a. Définitions 38
b. Les défaillances dans Condor 39
III. Utilisations des agents mobiles 41
a. Les agents mobiles dans les grilles de calcul 41
b. Nos agents mobiles 42
CONCLUSION 44
REFERENCES 45

TABLES DES FIGURES

Figure 1 : Architecture en couches des grilles informatiques 11
Figure 2 : Architecture simplifiée d'une grille 13
Figure 3 : Quelques fournisseurs et produits à considérer 14
Figure 4 : Exemple de pool Condor 24
Figure 5 : Matchmaking dans Condor 28
Figure 6 : Graphe acyclique dirigé (DAG) 30
Figure 7 : un fichier DAG 31
Figure 8 : les démons de Condor 33
Figure 9 : Origine des défaillances 39
Figure 10 : Saturation mémoire 40

Introduction

Des applications géographiquement réparties ont été traitées dès le début de l'informatique : il s'agissait alors essentiellement de relier à un site central des terminaux d'accès ou des organes de mesure munis ou non de possibilités de traitement local. Dans une étape ultérieure, des soucis d'économie (mise en commun de ressources ou d'information) amènent à connecter par des réseaux de communication des installations existantes capables de fournir des services généraux ou spécialisés. La majorité des applications informatiques sont maintenant réparties, c'est-à-dire mettent en jeu un ensemble d'ordinateurs ou d'organes divers reliés par un réseau de communication. La répartition est un état de fait pour un nombre important d'applications.

L'évolution de la technologie se manifeste essentiellement par : la baisse du coût et l'accroissement des performances des processeurs et des mémoires résultent des progrès des composants intégrés ; le développement de moyens de communication permet l'échange d'information sur une gamme étendue de distances et de débits.

De nouveaux défis au domaine des systèmes répartis sont amenés par le développement de l'informatique mobile, des systèmes pair à pair, des grilles informatiques (GRID).

Le concept de grille informatique a fait ses preuves, des grilles de production sont maintenant opérationnelles en entreprises et entre organismes. Au delà d'une solution pour les calculs intensifs, l'approche grille permet la constitution d'organisations virtuelles (VO) regroupant des communautés d'utilisateurs partageant des ressources globalisées de traitement, de stockage, des instruments et des services largement distribuées, quelles qu'en soient les organisations propriétaires. Le déploiement des grilles a donc un impact fort sur le plan organisationnel, les réseaux, la sécurité et l'expérimentation du paradigme HTC dans la construction des plates-formes de calcul et de services informatiques à haute performance, en direction de la disponibilité et de la continuité de service que doivent offrir ce type de plate-forme pour une utilisation dans un environnement de production.

Le système Condor qui est un système distribué très utilisé pour les traitements à haut débit, est un excellent moyen d'introduction à la recherche sur les grilles informatiques et les infrastructures répartis hautement disponibles, ainsi que les technologies d'agents mobiles. Ces derniers ayant des caractéristiques de déplacement, d'adaptation qui vont permettre de tirer partie de ces environnements répartis de calcul haut débit, particulièrement pour la gestion des défaillances.

C'est ainsi que nous avons axés nos recherches sur la gestion répartie des défaillances dans un pool Condor par agents mobiles. Ce manuscrit est organisé en trois parties :

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

- une première partie sur l’état de l’art du paradigme HTC ou Calcul haut débit, sur les grilles informatiques et la technologie des agents mobiles.
- Dans la seconde partie nous présenterons le système Condor et ses fonctionnalités
- La troisième présentera la recherche faite sur la gestion des défaillances dans Condor avec les agents mobiles

PARTIE I : ETAT DE L'ART

- **High Throughtput Computing (HTC)**
- **Grilles Informatiques**
- **Agents Mobiles**

I. High Throughput Computing (HTC)

Pour beaucoup de scientifiques, le progrès et la qualité scientifiques de la recherche sont fortement liés au débit de calcul. En d'autres termes, la plupart des scientifiques sont concernés par combien d'opération à virgule flottante par mois ou par année ils peuvent extraire à partir de leur environnement de calcul plutôt que le nombre de telles opérations l'environnement peut fournir par seconde ou minute.

Les opérations à virgule flottante par seconde (FLOPS) ont été la mesure employée pour l'évaluation des environnements de Calcul Haute Performance (HPC).

Peu d'attention a été consacrée par la communauté informatique aux environnements qui peuvent fournir de grandes quantités de capacité de traitement sur de longues périodes. Nous faisons référence aux environnements de Calcul Haut Débit (HTC : High Throughput Computing).

Le traitement à haut débit (HTC) est opposé au traitement à haute performance (HPC). Là où le premier est évalué sur le long terme, le second implique des résultats à court terme. C'est ainsi que plutôt qu'en MIPS (millions of instructions per second), les performances des applications à haut débit sont mesurées en TIPY (trillions of instructions per year).

Calcul Haut débit (HTC)	**Calcul Haute Performance (HPC)**
• Traitement des calculs à « long terme » • La puissance de milliers de processeurs • Le débit est le principal but • L'ordre dans lequel les différentes tâches s'exécutent n'est pas important • Si l'équipement est indisponible, le travail est migré sur une autre machine	• Traitement des calculs à « court terme » • Utilisation d'un unique processeur couteux • Si un processus échoue, le calcul s'arrête • Tous les processus doivent être redémarrés au même point de synchronisation

La clef à HTC est la gestion et l'exploitation efficaces de toutes les ressources informatiques disponibles. Puisque les besoins de calcul de la plupart des scientifiques peuvent être satisfaits de nos jours par le CPUs et la mémoire de l'ordinateur, le rendement élevé ne joue pas un rôle important dans un environnement de HTC. Le plus grand challenge d'un environnement HTC est comment maximiser la quantité de ressources accessibles aux utilisateurs.

En d’autres termes, le but ici n’est pas de rechercher la performance pure comme dans le cas précédent mais de rentabiliser au maximum des ressources en récupérant les cycles processeurs non utilisés quand celui-ci est oisif (*idle*). On pense bien entendu aux stations de travail qui sont généralement très largement sous utilisées, ne serait-ce qu’en dehors des heures de bureau.

II. Les grilles informatiques

Depuis quelques années, l'informatique répartie et les technologies associées sont dans une constante évolution technologique et économique. Dans ce contexte, le terme « The Grid » ou « grille de calcul », a été introduit pour décrire une infrastructure de calcul répartie utilisée dans des projets de recherche scientifique et industrielle aux Etats-Unis durant les années 1990.

a. Pourquoi les grilles informatiques ?

De plus en plus les applications développées de nos jours sont gourmandes en puissance et en stockage. Ce qui entraîne un problème de dimensionnement des ressources.

Aussi ces applications utilisées par une communauté géographiquement distribuée permet le travail collaboratif. D'où le partage des données et des applicatifs posant ainsi le problème d'accès partagé aux ressources.

Pour les chercheurs, il fallait trouver des possibilités pour accéder aux ressources, interagir avec des collègues, analyser d'énormes volumes de données et de partager des résultats. Ensuite incorporer des ressources existantes qui sont hétérogènes et dynamiques.

L'accès aux services informatiques devrait être comparable en coût et facilité au branchement d'appareils sur le réseau électrique. Cet accès devrait être aisé et transparent quels que soient la puissance nécessaire, la complexité des équipements matériels et logiciels mis en oeuvre, les fournisseurs du service.

Raisons des grilles de calcul

Il est important de savoir quels avantages une grille de calcul est en mesure d'offrir que les infrastructures et les technologies actuelles ne sont pas capables d'assurer. Nous exposons dans la suite quelques unes des raisons pouvant amener à déployer une grille de calcul.

- **Exploiter les ressources sous utilisées** : les études montrent que les ordinateurs personnels et les stations de travail sont inactifs la plupart du temps. Le taux d'utilisation varie entre 30% (milieux académiques et industriels) et 5% (machines grand public). Les grilles de calcul permettront ainsi d'utiliser les cycles processeurs durant lesquels les machines sont inactives afin de faire tourner une application nécessitant une puissance de calcul importante et que les machines qui lui sont dédiées n'arrivent pas à assurer. Les cycles processeur ne sont pas la seule ressource sous utilisée ; souvent les capacités de stockage le sont aussi. Ainsi il est possible qu'une grille agrège toutes ces ressources afin de les partager entre les différents utilisateurs

(on parle alors de « grille de données » ou Data Grid). Une autre conséquence d'une telle utilisation est la possibilité de faire du partage de charge entre les différentes ressources d'une grille.

- **Fournir une importante capacité de calcul parallèle** : le fait de pouvoir fournir une importante capacité de calcul parallèle constitue une caractéristique importante des grilles de calcul. En plus du domaine académique, le milieu industriel bénéficiera énormément d'une telle capacité : bioinformatique, exploration pétrolière, industrie cinématographique etc. en effet les applications sont écrites d'une façon à pouvoir exploiter parallèlement des ressources (clusters, machines multiprocesseur …). Les grilles de calcul peuvent de la même manière fournir des ressources dont l'utilisation pourra se faire en parallèle.

- **Meilleure utilisation de certaines ressources** : en partageant les ressources, une grille pourra fournir l'accès à des ressources spéciales comme des équipements spécifiques (microscope électronique, bras robotique …) ou des logiciels dont le prix de la licence est élevé. Ainsi ces ressources exposées à tous les utilisateurs seront mieux utilisées et partagées et ainsi on évitera d'avoir recours à installer du nouveau matériel ou acheter de nouvelles licences.

- **Fiabilité et disponibilité des services** : du fait que les ressources fédérées par une grille de calcul soient géographiquement dispersées et disponibles en importantes quantités permet d'assurer la continuité du service si certaines ressources deviennent inaccessibles. Les logiciels de contrôle et de gestion de la grille seront en mesure de soumettre la demande de calcul à d'autres ressources.

b. Définition

La grille informatique (ou Grid Computing) a été nommée ainsi par analogie au réseau électrique (ou Electrique Power Grid) qui permet d'obtenir une puissance électrique grâce à une interface simple (la prise de courant) sans se soucier de la provenance de cette énergie et de la manière dont elle a été fabriquée.

La grille reprend ce principe : plusieurs ordinateurs alimentent un réseau pour fournir une puissance globale. Chaque utilisateur dispose alors de ressources considérables, aussi bien en terme de puissance de calcul que de stockage de données, au moyen d'une interface simplifiée.

On peut définir aussi les grilles informatiques comme un système réparti haute performance dont certains des constituants sont des calculateurs parallèles ou des grappes.

Ce concept était à la base essentiellement technique, avec pour objectif principal la mise à disposition de ressources de traitement informatique très importantes et multi organismes avec un accès aussi transparent que possible. Le concept a fait ses preuves autour d'infrastructures prototypes; des grilles de production sont maintenant opérationnelles en entreprises et entre organismes. Le déploiement d'infrastructures de ce type en production a des impacts forts sur le plan des réseaux, de la sécurité, de l'administration au sein de chaque organisme, et sur le plan organisationnel puisqu'un l'un des objectifs visés est celui du partage de ressources entre entreprises ou institutions.
En effet, au delà des besoins en capacité de traitement, l'approche grille permet la co-allocation et le partage dynamique de toutes les ressources de systèmes hétérogènes indépendamment des unités organisationnelles (départements d'entreprises, institutions et entreprises). Des organisations virtuelles (VO) permettent maintenant de définir en tant que de besoin des communautés d'utilisateurs partageant suivant des modalités communes d'utilisation, des sous-ensembles de ces ressources globalisées de traitement, de stockage, des logiciels, des instruments et des services largement distribuées, quelles que soient les organisations propriétaires.

Un nouveau moteur de la puissance informatique

Depuis quelques mois, la grille informatique, fait de plus en plus parler de lui dans l'univers des nouvelles technologies. Il se fait aussi progressivement connaître du grand public à travers des expériences scientifiques et industrielles largement relayées par les médias. Cependant, il reste souvent difficile de comprendre ce qui se cache derrière cette technologie, dont les premières expériences ont débuté il y a une trentaine d'années.
De manière plus formelle, la technologie de grille informatique 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. Ces applications ne seraient pas possibles au sein d'un seul organisme ou d'une seule entreprise. Du point de vue de l'utilisateur, l'origine des ressources utilisées est totalement abstraite et transparente. Ce dernier ne se soucie de rien : ni de la puissance nécessaire pour ses

applications, ni du type de machines utilisées, et encore moins de leur localisation physique. C'est la plate-forme de Grid qui gère tous ces aspects.

En pratique, le Grid Computing essaye d'aller au-delà de l'Internet. Les acteurs impliqués dans ce projet ont un rêve : celui de l'utilisation à la demande. Son objectif est de rendre disponibles toutes les ressources de calcul et de stockage des ordinateurs connectés sur une grille informatique pour des applications de gestion, de communication, de business ou encore des applications typiquement scientifiques.

Il n'y a aucune raison pour que ce qui est possible pour des entreprises ou des centres de recherche ne le soit pas également pour nos collectivités locales. Celles-ci possèdent en effet des parcs informatiques importants et demain elles auront besoin d'une puissance informatique décuplée pour pouvoir utiliser les nouveaux outils de réalité virtuelle et de simulation en 3D. Nos collectivités ont donc tout à fait intérêt à organiser leurs parcs informatiques en grille informatique pour pouvoir bénéficier, sans augmentation de coût, d'une puissance de calcul correspondant à leurs futurs besoins. [2]

Une tentative de classification [3]

- **Classification par objectifs**

Grille d'information: partager la connaissance
Le web
Observatoires virtuels (http://www.france-ov.org/)

Grille de stockage: stocker à grande échelle
Observatoires virtuels
LCG (15 Petabytes/an) (http://lcg.web.cern.ch/LCG/)

Grille de calcul: agréger la puissance de calcul
EGEE (http://www.eu-egee.org/)
DEISA (https://www.deisa.org/)
CiGri (https://ciment.imag.fr/cigri)
SETI@home (http://setiathome.berkeley.edu/)

Grille d'expérimentation: Recherche en informatique distribuée
Grid5000 (http://www.grid5000.fr)
PlanetLab (http://www.planet-lab.org/)
DAS3 (http://www.cs.vu.nl/das3/)
XtremLab (http://xtremlab.lri.fr/)

Grille « best-effort »: Exploiter les temps morts des clusters au sein d'une communauté dont les ressources sont utilisées de manière discontinue dans le temps

- **Classification par infrastructure**

Grilles d'institutions
(EGEE, Grid5000, CiGri,...)
– Grilles de grappes situées dans des centres de calcul
– Ordre de grandeur: centaines de noeuds par grappe
– Noeuds stables et relativement sécurisés

Desktop computing, systèmes Peer-to-peer
(Seti@Home, Boinc, XtremLab,...)
– PC sur Internet
– Ordre de grandeur: millions de noeuds disséminés
– Noeuds très volatiles et peu sécurisés

c. Applications des grilles de calcul

Les principales applications des grilles de calcul se feront dans les domaines suivants :

- **Supercalculateur réparti** (*Distributed Supercomputing*) : une grille de calcul pourra agréger une importante quantité de ressources afin de fournir la puissance de calcul nécessaire pour de nombreuses applications et que même les supercalculateur les plus modernes ne peuvent fournir. Selon la nature et l'étendue de la grille, les ressources agrégées pourront aller de toutes les stations de travail d'une université à tous les supercalculateurs des organismes de recherche d'un pays. Comme exemples d'applications nous pouvons citer la simulation distribuée dans la météorologie, la cosmologie et l'aéronautique …
- **Calcul haut-débit** (*High-Throughput Computing*) : une grille de calcul sera utilisée pour ordonnancer en parallèle une importante quantité de tâches indépendantes les unes des autres. Comme exemples d'applications nous pouvons citer la recherche de clés cryptographiques, les simulations de molécules et l'analyse du génome …
- **Calcul sur demande** (*On-Demand Computing*) : une grille de calcul pourra fournir les ressources pour satisfaire les demandes à court terme d'une application et que les ressources locales ne sont pas en mesure d'assurer (cycles processeur, stockage …). Le défi principal pour les concepteurs de telles grilles est la nature dynamique et aléatoire des demandes faites par les utilisateurs qui peuvent constituer une large population.
- **Calcul Collaboratif** (*Collaborative Computing*) : cette classe d'applications inclut les applications d'interaction entre humains dans des environnements de simulation en temps-réel (conception et interaction avec un nouveau moteur d'avion, conception d'un plan urbain …).

- **Génération, traitement et stockage d'énormes quantités de données (***Data-intensive Computing***)** : dans de telles applications, une grille de calcul pourra absorber et stocker l'importante quantité d'information générées. Comme exemple d'applications nous pouvons mentionner la production d'une carte de l'univers, la prévision météorologique à long terme, les simulations quantiques …

d. Architecture générale d'une grille informatique

Bien que chaque projet ait sa propre architecture, une architecture générale est importante pour expliquer certains concepts fondamentaux des grilles. C'est ainsi que l'architecture d'une grille peut être représentée sous forme de couches, chacune assurant une fonction spécifique.

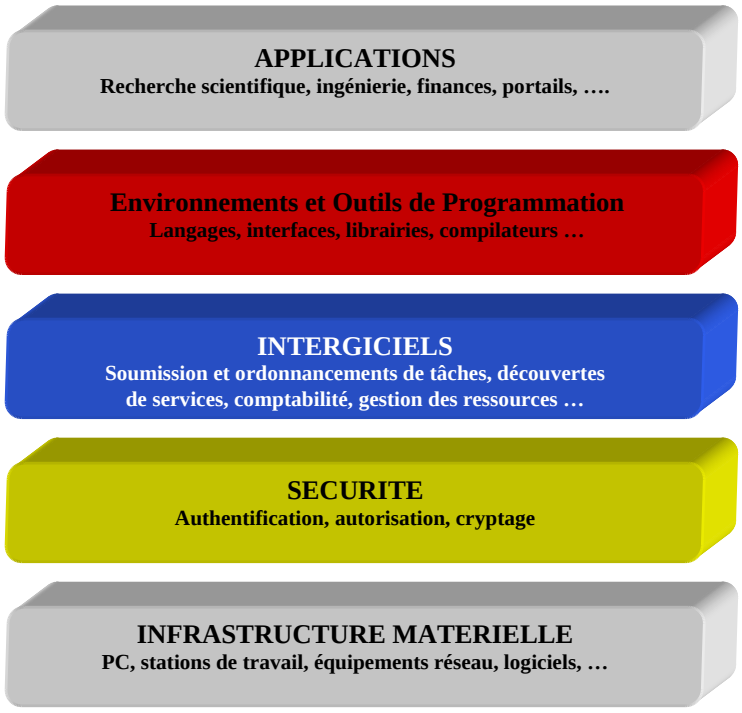

Figure 1 : Architecture en couches des grilles informatiques

Au fond de l'architecture en couches nous avons les ressources proprement dites interconnectées à travers des réseaux locaux et grande distance. Cette **infrastructure matérielle** (fabric) comprend notamment des PC, des stations de travail, des grappes de calcul (clusters), des équipements de stockage, des bases de données, des équipements spéciaux …

Les ressources d'une grille peuvent varier des plus simples (machines de même architecture, même système d'exploitation, réseau d'interconnexion homogène) au plus compliquées

(différentes architectures de machines, plusieurs systèmes d'exploitation, étendue géographique, différentes politiques de sécurité, plusieurs administrateurs …).

La seconde couche fournit les mécanismes de **sécurité** nécessaires à la protection des ressources. Dans les grilles de calcul, la sécurité est un souci beaucoup plus important que dans les environnements d'informatique répartie traditionnels. Le nombre important de ressources et leur étendue géographique constituent un facteur important. Ainsi il est fondamental de bien identifier les ressources à protéger et les degrés de protection nécessaires à appliquer. Parmi les mesures de protection on trouve ceux normalement employés dans les réseaux (pare-feu, détection d'intrusion …) ainsi que ceux spécifiques aux environnements répartis et aux grilles (authentification, autorisation, single sign-on …).

La troisième couche fournit les **intergiciels** *(middleware)* nécessaires à la gestion des ressources, la coordination de l'accès, l'ordonnancement des tâches …
Nous expliquons ci-dessous brièvement les principales fonctions assurées par ces intergiciels:

- **Ordonnancement** *(scheduling)* : un ordonnanceur (scheduler) devra trouver la machine la plus appropriée pour faire tourner les tâches qui lui sont soumises. Les ordonnanceurs peuvent aller du plus simple (allocation de type round-robin) au plus compliqué (ordonnancement à base de priorités …). Les ordonnanceurs réagissent à a charge de la grille. Ils déterminent le taux de charge de chaque ressource afin de bien ordonnancer les prochaines tâches. Ils peuvent être organisés en hiérarchie avec certains interagissant directement avec les ressources, et d'autres métaordonnanceurs) interagissant avec les ordonnanceurs intermédiaires. Ils peuvent aussi superviser le déroulement d'une tâche jusqu'à sa terminaison, la soumettre à nouveau si elle est brusquement interrompue et la terminer prématurément si elle se trouve dans une boucle infinie d'exécution.
- **Réservation** : il est possible dans les grilles de calcul de réserver les ressources à l'avance et ceci afin de garantir une certaine qualité de service et de respecter certaines échéances. Pour cela l'intergiciel devra fournir un système de réservation permettant aux utilisateurs d'exprimer leurs besoins en ressources et d'effectuer la réservation.
- **Services d'information et d'annuaire** (Information and Directory services) : une grille est un environnement où le nombre et la charge des ressources sont dynamiques. Un objectif lors de la conception d'une grille est de rendre les ressources facilement

accessibles à tous les processus. Il est alors nécessaire de fournir des mécanismes permettant d'enregistrer et de découvrir ces ressources et d'identifier leur état.

- **Service de nom** (Naming Service) : Comme dans tout système réparti, une grille devra permettre de référencer ses ressources d'une façon standard et uniforme.

Une grille de calcul devra fonctionner avec un large spectre de technologies matérielles ou logicielles. Tout comme les utilisateurs du WEB ne se soucient pas de savoir si les serveurs WEB tournent sur une machine x86 ou Mac ou bien si le système d'exploitation est Unix ou Windows, les utilisateurs d'une grille ne veulent pas se préoccuper des détails matériels et logiciels de l'infrastructure. Un environnement de grille idéal fournira ainsi un accès transparent aux ressources en cachant toutes les différences « physiques ».

La quatrième couche regroupe tous les outils et les paradigmes pouvant aider les développeurs à concevoir et écrire des applications pouvant tourner sur la grille. On y trouve plus particulièrement des compilateurs, des librairies, des outils de conception d'applications parallèles ainsi que des interfaces de programmation ou API (découverte et réservation de ressources, mécanismes de sécurité, stockage …) que les développeurs d'applications pourront utiliser.

La dernière couche regroupe les **applications** proprement dites qui sont de nature variée : projets scientifiques, médicaux, financiers, ingénierie …

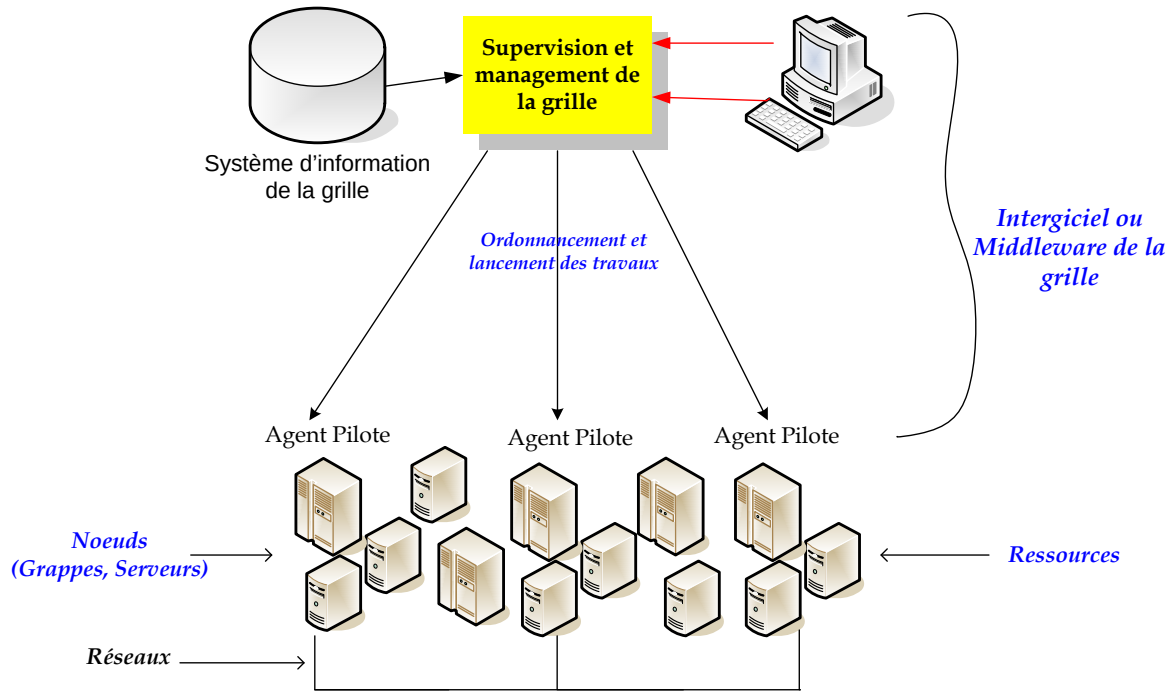

Figure 2 : Architecture simplifiée d'une grille

e. Le paysage des grilles

L'on ne compte plus, aux Etats-Unis tout comme au Japon où l'on trouve par exemple un *Grid technology Research Center* les projets de recherche et développement liés aux grilles, simplement nous noterons que l'effort est mis sur les aspects ingénierie et organisation des architectures matérielles pour aller vers des infrastructures « de production ». Les grilles entrent dans une phase opérationnelle et l'on voit apparaître des organismes fournissant des *Grid and Networking Services*. Le mot d'ordre européen est *Building Grid for Europe a crucial technology for science and industry*, les usages des grilles sont une des priorités affichées du programme *Information Society* de la commission européenne.

Là, le terrain a été défriché par les projets EuroGrid et DataGrid qui ont validé le modèle de grille et qui ont débouché sur les grands projets d'infrastructure Deisa, EGEE et applicatifs BEinGRID.

L'excellence internationalement reconnue de la recherche française (INRIA, Universités, CNRS) a conduit l'INRIA piloter le réseau d'excellence (NOE) CoreGRID lancé en 2004. Enfin, l'on trouve en Europe des projets tels que e-Science, D-Grid, Grid.it, Grid-Ireland, aux dimensions certes très variables mais qui à chaque fois fédèrent les besoins et les initiatives concernant les grilles au niveau national et dont les acteurs jouent tout naturellement un rôle majeur au niveau européen.

En France, les actions de recherche ont été fédérées au travers des ACI Grid, Grid5000, ... Le projet e-Toile soutenu par le Réseau National des Technologies Logicielles et piloté par le Centre Nationale de la Recherche Scientifique (CNRS) a permis de valider, une approche grille haute performance pour les applications de la simulation numérique.

Entreprise / Organisme	Produit	Origine	Technologie
Avaki	Legion (Compute & Data	USA	Mixte
CNRS/LRI	XtremWeb	F	Desktop Grid
CNRS/LAAS+ Alinka SA	Aroma	F	Outils d'administration
CNRS/INRIA/CEA/EDF/CS/SUN	e-Toile	F	Massive Grid
Data Synapse	LiveCluster	USA	Desktop Grid
Entropia	DC Grids	USA	Desktop Grid
GridSystems	Inner Grid	SP	Desktop Grid
GridXpert	Synergie	F -> US	Outils d'administration
Icatis	ComputeMode	F	Extension Cluster>PC
HP	Utility Data Center	USA	Massive Grid (1e gén.)
IBM	Pas de produit propre	USA	
Meshtechnologies	OfficeGrid	DK	Desktop Grid
Pallas–>Intel	Unicore	D–>USA	Massive Grid (1e gén.)
Platform	LSF/Symphony	USA	Mixte
Sun	Sun Grid Engine	USA	Massive Grid (1e gén.)
Université d'Argonne	Globus	USA	Massive Grid
Université de Berkeley	Boinc	USA	Desktop Grid
Université du Wisconsin	Condor	USA	Massive Grid
United Devices	Grid MP	USA	Desktop Grid
W2G technologies	Web2grid	F	Virtualisation (V V M)

Figure 3 : Quelques fournisseurs et produits à considérer

f. Impacts sur les réseaux

Un gros effort a été accompli sur les performances d'interconnexion locale de grappes de serveurs avec des technologies comme Myrinet[1] ou Infiniband[2]. Les grandes grilles doivent traverser des WAN, avec les problèmes de bande passante et surtout de latence liés, ce qui induit des difficultés dans la prévision des délais d'exécution de travaux et surtout dans l'architecture des applications.

La mise en place d'une infrastructure de calcul distribué telle que la grille a des conséquences sur les réseaux. La plus évidente (et celle attendue par un administrateur) peut être une augmentation de l'utilisation des liens, même si cela est loin d'être automatique et forcément significatif. En effet, les applications « grille » n'engendrent pas toutes des transferts de gros volume de données. Le cas échéant, des infrastructures privées sont envisagées afin d'une part de garantir le service pour les expériences et de ne pas perturber l'utilisation normale des réseaux. L'Internet public ne sera alors utilisé que pour l'accès à la grille, la consultation des données ou l'exécution d'applications spécifiques.

Parmi les conséquences inattendues, citons d'abord les conséquences directes sur tous les types de réseau : réseau local, métropolitain puis régional selon la hiérarchie adoptée dans chaque pays, certains pouvant être absents. La première conséquence réside dans le besoin de gestion opérationnelle. Ce n'est pas tant la haute disponibilité des réseaux de bout-en-bout qui est critique, l'architecture distribuée de la grille s'accommodant relativement bien de l'indisponibilité d'un site. C'est plutôt la remontée et la résolution des incidents qui sont nécessaires. La disponibilité des informations dans un système centralisé à destination des utilisateurs de la grille est un besoin qui a un impact fort sur les réseaux s'ils n'ont pas déjà un système opérationnel en place.

g. Utilisation en entreprise

Conçues à l'origine pour permettre l'exécution d'applications nécessitant de grandes puissances de calcul et utilisées principalement par les chercheurs, les grilles informatiques sortent aujourd'hui des laboratoires et commencent à intéresser fortement le monde l'industrie.

---

[1] Myrinet est une technologie récente d'interconnexion de réseau, commuté à faible latence et à haute performance, basée sur l'envoi de messages. Les produits d'interconnexion Myrinet permettent de connecter à très haute vitesse un très grand nombre d'ordinateurs afin de former des clusters très rapides.

[2] Infiniband est un bus d'ordinateur bi-directionnel à faible coût avec un débit de 10 Gbits/s. Il utilise une technologie permettant à plusieurs périphériques d'accéder en même temps au réseau.

En se raccordant à une grille de calcul, les informaticiens transformeront instantanément leur infrastructure de traitement de l'information en un guichet centralisé donnant accès à un nouveau monde – littéralement : le monde entier – de ressources de calcul. Les entreprises pourront immédiatement résoudre les problèmes de leurs clients, donner à leurs collaborateurs répartis partout dans le monde les moyens nécessaires pour leurs transactions commerciales, mais aussi proposer des produits et des services à l'échelle internationale.

Mais le plus intéressant dans ce modèle de traitement, c'est son impact financier. Grâce à la technologie grille de calcul, le budget et les ressources consacrés à l'informatique ne sont plus gaspillés en coûts d'équipement et de gestion. Tout comme les opérateurs des réseaux de distribution d'électricité, d'eau ou de communications téléphoniques, la grille de calcul fournit sa propre infrastructure informatique. Les entreprises paient donc uniquement pour l'utilisation des cycles de calcul, et non pour l'équipement informatique proprement dit.

En entreprise, les grilles se mettent surtout en place dans de très grandes sociétés à haute technologie pour la modélisation, la simulation et la visualisation 3D dans l'automobile, l'aéronautique, le spatial, le recherche pétrolière et le nucléaire. Les autres projets couvrent le calcul intensif et une meilleure gestion du parc informatique avec son obsolescence rapide. Les grilles se généralisent aussi rapidement dans la pharmacie, la biologie, la modélisation moléculaire et la chimie. En finance et bourse, on a pu observer un abaissement drastique des traitements dans l'analyse de risque dans la finance et la bourse. Les grilles ne sont pas réservées aux grandes Entreprises uniquement. Plusieurs fédérations de PME s'y intéressent pour le partage dynamique de leurs équipements, données techniques et codes de calcul, ou en vue d'utilisation en mode informatique à la demande avec des fournisseurs externes, pour des coûts actuellement hors de leur portée. Donc pour les PME, c'est un outil économique de recherche et de développement.

IBM propose une grille informatique pour les PME. Sa nouvelle offre, baptisée « Grid and Grow Express » vise les entreprises de 100 à 1000 utilisateurs. Elle est basée sur sept serveurs lames eServer BladeCenter dont l'entreprise peut adapter la configuration en fonction de ses besoins.

SUN a lancé aussi la grille de calcul « Sun Grid Compute Utility » constitué de 5000 processeurs Opteron et UltraSPARC. Cette grille permet à n'importe quelle entreprise de louer une partie de la grille quelques heures sur Internet afin d'y effectuer un calcul complexe. Elle évite ainsi à l'entreprise d'avoir à investir dans un supercalculateur.

III. LES AGENTS MOBILES

Il existe différents domaines de l'informatique à savoir les systèmes multiagents[3], les agents d'interface[4], les agents mobiles, utilisant le concept d'agent. L'objectif de cette partie est de présenter les travaux touchant à la notion d'agents mobiles en essayant d'en dégager les concepts fondamentaux.

a. Pourquoi la mobilité ?

L'exploitation de la mobilité dans les réseaux ouvre des possibilités très grandes pour améliorer leur fonctionnement général (sécurité, fiabilité, QoS, ...). [5]

Elle permet :

- La réduction de la charge des réseaux : Les systèmes distribués génèrent de nombreux messages pour exécuter leurs tâches. En envoyant les agents où les tâches se font, les messages échangés deviennent locaux et libèrent d'autant la charge des réseaux ; ce qui réduit les délais de communication.

- De s'affranchir des temps de réponse du réseau : Dans les systèmes distribués les protocoles définissent comment les messages et les données sont échangés. Pour modifier un protocole, il faut changer le code sur toutes les machines du système. Les agents encapsulent les protocoles. Changer de protocole revient à interagir avec un nouvel agent ;

- L'autonomie des composants et l'exécution synchrone/asynchrone : Les terminaux mobiles (PDA, ordinateurs portables, téléphones, ...) ne sont pas toujours connectés au réseau. Les systèmes qui nécessitent des connexions permanentes ne sont pas adaptés dans ce cas. Avec des agents, les terminaux mobiles peuvent se connecter pour vérifier s'ils ont des messages. Un agent peut être envoyé sur le terminal mobile au moment de la connexion et travailler sur ce dernier après avoir interrompu la connexion. Le même agent peut attendre la prochaine connexion pour envoyer des informations sur le réseau signifiant par exemple qu'une tâche a été terminée.

---

[3] Les *systèmes multi-agents* provenant des domaines de la robotique et de l'intelligence artificielle qui proposent de structurer une application en différentes fonctions représentées chacune par un agent. Les différents agents coopèrent pour la résolution des problèmes de l'application.

[4] Les *agents d'interface* provenant des domaines de l'interface homme-machine et de l'intelligence artificielle qui essayent de simplifier la vie de l'utilisateur en automatisant des tâches réalisées couramment par l'observation des comportements répétitifs ou en surveillant des ressources comme le courrier électronique.

- L'adaptation dynamique du système aux problèmes : Les agents peuvent se répartir eux-mêmes sur des machines du réseau, afin de mieux tenir compte de l'état du système au moment où ils doivent résoudre leurs tâches.

- La gestion des environnements hétérogènes

- La gestion de la robustesse et de la tolérance aux pannes : Si une machine doit s'arrêter, l'agent qui s'y trouve peut changer de machine pour terminer sa tâche en cours.

b. Définition d'un agent mobile

Voici les premières définitions données aux agents mobiles [7]:

Un agent est une entité autonome, réelle ou abstraite, qui est capable d'agir sur elle-même et sur son environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents, et dont le comportement est la conséquence de ses observations, de ses connaissances et des interactions avec les autres agents.

Il ressort de cette définition des propriétés clés comme l'*autonomie*, l'*action*, la *perception* et la *communication*. D'autres propriétés peuvent être attribuées aux agents. Nous citons en particulier la *réactivité*, la *rationalité*, l'*engagement* et l'*intention*.

Un agent est un système informatique, *situé* dans un environnement, et qui agit d'une façon *autonome* et *flexible* pour atteindre les objectifs pour lesquels il a été conçu.

Les notions "situé", "autonomie" et "flexible" sont définies comme suit:

– *situé*: l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples: systèmes de contrôle de processus, systèmes embarqués, etc ;

– *autonome*: l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;

– *flexible*: l'agent dans ce cas est:

capable de répondre à temps: l'agent doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis,

proactif: l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au "bon" moment;

social: l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

Donc, nous pouvons définir un agent comme un programme informatique qui possède les quatre propriétés suivantes :

Autonomie : Un agent contrôle ses actions et son état interne

Communication : Un agent peut communiquer avec d'autres agents

Réactivité : Un agent perçoit l'environnement dans lequel il se trouve et peut répondre aux changements qui s'y produisent.

Activité : Un agent est capable d'agir sur l'environnement dans lequel il se trouve

c. Particularités des agents mobiles

La migration « forte/faible » : la migration permet le transfert d'un agent d'un site à un autre à travers le réseau. Le problème principal vient du fait que nous ayons à faire à des agents, c'est-à-dire des entités autonomes (pro-actives) qui répondent à des sollicitations externes et internes (leurs propres sollicitations). De ce fait l'agent mobile peut :

- être sollicité de l'extérieur pour faire une migration (sur ordre de l'administrateur de la machine hôte avant un arrêt de la machine) alors qu'il est en train d'effectuer des traitements
- ou la solliciter lui-même alors qu'il est sollicité de l'extérieur.

Deux types de migrations ont été proposés dans les plate-formes existantes : la migration forte permet à un agent de se déplacer quelque soit l'état d'exécution dans lequel il se trouve et de reprendre son exécution après la migration exactement là où elle en était avant ; la migration faible ne fait que de transférer avec l'agent son code et ses données et soit nécessite des moments privilégiés dans le code de l'agent pour pouvoir être lancée (point d'arrêt), soit peut être lancée à tout moment mais avec perte des traitements en cours. La migration forte, bien que beaucoup plus exigeante à implanter que la migration faible, n'en est pas moins indispensable pour toutes les applications pour lesquelles les notions de fiabilité et de tolérance aux pannes sont primordiales [5].

L'accès aux ressources locales : cela pose en fait le problème du piratage des sites visités par les agents mobiles. Ce problème, non spécifique aux agents mobiles, est symétrique pour ces

derniers. En effet un agent mobile est aussi à la merci de se faire pirater par les logiciels qui fonctionnent sur les sites qu'il visite. Cela revient au problème général de la sécurité des machines (lutte contre les virus et autres techniques connues de piratage) avec comme spécificité le fait que les agents mobiles peuvent naturellement se déplacer beaucoup, d'où le problème de leur traçabilité.

La traçabilité : cela peut être **un moyen pour suivre et contrôler les déplacements des agents**. Les agents mobiles peuvent disposer de toute l'autonomie leur permettant de choisir leurs itinéraires (espace) et les moments (temps) pour les explorer. Par exemple il peut être judicieux pour un agent de changer l'ordre des sites qu'il doit visiter à cause d'une indisponibilité passagère d'un lien réseau ; il peut aussi décider de ne pas remettre en question l'ordre de son itinéraire, mais simplement d'attendre un peu que la perturbation du lien réseau s'estompe. On peut aussi avoir des agents dont les missions sont soit exploratoires (robots sur Internet), soit font référence à des buts ou à des besoins, charge aux agents de trouver les bons interlocuteurs pour mener à bien leurs tâches. Bref, même si les agents peuvent être programmés pour donner des informations sur ce qu'il font, il peut être utile de **disposer de mécanismes de traçabilité qui leur sont externes**, pour garder un aperçu de ce qu'ils font.

La communication entre agents : on peut identifier un grand nombre de moyens de communication entre agents, mais on peut les classer suivant qu'il s'agit d'une communication entre deux agents ou entre un agent et un groupe d'agents (groupe au sens large, c'est-à-dire soit un groupe formellement identifié en tant que tel, soit un ensemble d'agents non formellement identifiés comme par exemple des agents présents à un moment donné dans un lieu donné – concommitence spatio-temporelle).
Les moyens de communications sont multiples (messages, sessions, tableau blanc, rendez-vous, de type collaboratifs, par événements distribués ou locaux, ...).

d. Quelques systèmes d'agents mobiles

La notion d'agent est assez récente dans le domaine des réseaux et des systèmes répartis et engendre une activité de recherche très importante. Le système *Telescript* de *General Magic*, qui est à l'origine du premier système d'agent mobiles, a déclenché l'intérêt des grands opérateurs de télécommunication pour cette nouvelle forme de structuration des communications. Beaucoup de projets de recherche sont en cours pour à la fois approfondir

cette nouvelle forme d'architecture des réseaux et étudier l'impact de cette technologie sur l'existant en matière de communication et de services.

Dans cette section, nous tentons de donner une idée de l'état actuel des connaissances en matière d'agents. Cette tâche est assez difficile pour plusieurs raisons:

– Cette technologie est en plein essor

– Il existe beaucoup d'implémentations différentes de la notion d'agent.

– Il y a eu plusieurs tentatives de normalisation pour prévenir la profusion d'implémentations incompatibles entre elles.

– Il y a plusieurs modèles pour décrire le comportement des agents. Chacun fait l'objet de recherches et de documents qui ne permettent pas d'avoir une vue globale et homogène.

– La maîtrise technologique des agents est un enjeu économique et stratégique très important, ce qui explique la difficulté de trouver une normalisation acceptable. En effet les entreprises qui ont déjà implémenté leur système d'agents ont du mal à se mettre d'accord sur un standard.

– La technologie des agents pose des problèmes théoriques et pratiques, notamment en terme de sécurité, très difficiles et dont l'étude en est encore au stade de la recherche.

Nom	**Source**	**Licence**
AgentOWL	http://agentowl.sourceforge.net/	open source
Akira	http://akira-project.org/	open source
LPA Agent Toolkit	http://www.lpa.co.uk/atk.htm	Academic free
MASS	http://mas.cs.umass.edu/research/mass/index.html	open source
SPYSE	http://www.zope.org/Members/drapmeyer/spyse	open source
Tryllian ADK	http://www.tryllian.com/technology/product1.html	open source
WSDL2Agent	http://sas.ilab.sztaki.hu:8080/wsdl2agent/	open source

Systèmes d'agents mobiles en 2005

Les plus populaires	Source
Aglets	IBM - http://www.trl.ibm.com/aglets/
Voyager	http://www.recursionsw.com/products/voyager/voyager.asp
D'Agents	http://agent.cs.dartmouth.edu/general/agenttcl.html
JADE	http://sharon.cselt.it/projects/jade/
MadKit	http://www.madkit.org

Les systèmes d'agents mobiles les plus populaires

IV. CONCLUSION

L'avènement des grilles permettra de gagner plusieurs ordres de grandeur dans la taille des problèmes résolus en calcul scientifique. Les premiers projets de grilles informatiques ont montrés leur faisabilité avec le développement d'infrastructure matérielle et logicielle et des expériences concluantes à l'échelle de l'Internet et d'un intranet. De nos jours, le domaine industriel nécessite d'avoir recours à des environnements de calcul haut débit tels que des grilles pour simuler des comportements réels ou simplement vérifier une cohérence particulière sur des produits virtuels en cours de définition.

Ce sont aussi des systèmes complexes impliquant la mise en oeuvre cohérente de nombreuses couches logicielles. Globalement, bien que les premières grilles de production soient annoncées, il reste de très nombreuses questions et problèmes à résoudre pour que les grilles soient accessibles par des utilisateurs non spécialistes. Au-delà, les propriétés attendues de performance et de tolérance aux pannes ouvrent un vaste espace d'optimisations. Dans ce contexte, il nous parait clair que les notions de mobilité ont un rôle central à jouer dans les grilles informatiques. L'utilisation des agents mobiles peut permettre d'améliorer les performances des communications. En déportant des calculs, ils favorisent la diminution de la communication en bande passante à travers le réseau. Bien que ces agents posent de nouveaux problèmes de sécurité, ils peuvent permettre des interactions distantes robustes sur des réseaux soumis à des défaillances de canaux de communication ou de machines.

PARTIE II : LE SYSTEME CONDOR

I. Le système Condor

Condor est l'aboutissement d'un long projet de recherche initié à l'Université de Wisconsin-Madison et fut opérationnel en tant que système de production au sein de son Département d'Informatique il y a vingt années. Depuis, des centaines d'institutions attachées à l'industrie, à l'Université ou l'administration l'ont adopté avec succès. L'objectif ici est de réaliser des calculs de plusieurs semaines ou mois, en exploitant efficacement les périodes d'inactivité des machines dans un environnement de propriété distribuée.

Condor est un puissant système de contrôle et gestion répartie des ressources d'un réseau de machines hétérogènes (stations de travail et serveurs, clusters, ou des supercalculateurs), spécialisé dans l'exécution concurrente d'un nombre très important de tâches indépendantes entre elles et/ou d'applications reposant sur un calcul intensif. Il est ainsi dédié aux environnements d'exécution dits de « High Throughput Computing ». Condor étudie l'état de charge des stations, déploie des tâches sur les machines non utilisées. Il les déplace lorsqu'elles deviennent indisponibles (selon d'activation du clavier ou de la souris, . . .), surveille les progrès réalisés et informe finalement l'utilisateur de l'accomplissement de tâches.

Simple à l'utilisation, le système met en oeuvre des mécanismes puissants attachés à la sécurité, au « Matchmaking » (i.e. système très souple de mise en correspondance des besoins ou contraintes associés à une tâche donnée et des caractéristiques matérielles ou d'usage liées à chaque ressource), et enfin au « Checkpointing » (i.e. reprise de l'exécution au stade du dernier état sauvegardé).

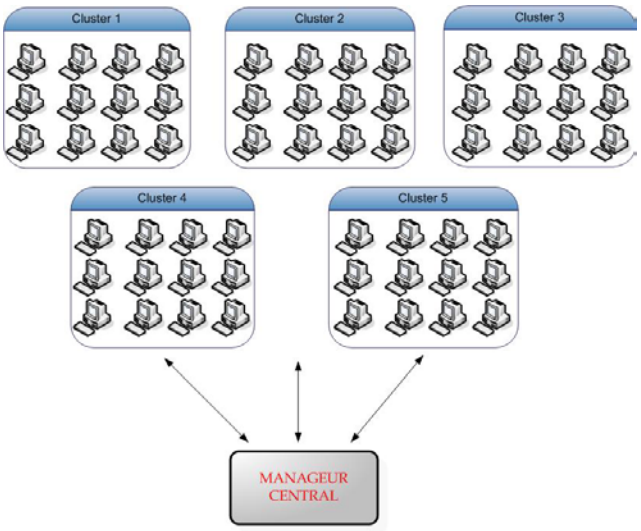

Figure 4 : Exemple de pool Condor

Condor permet la gestion de grilles de calcul englobant des ressources réparties sur plusieurs domaines administratifs. La technologie « Flocking » autorise la coopération de plusieurs infrastructures où ce système est installé. Condor associé au système Globus intègre aussi la plupart des méthodologies et protocoles émergeant du GRID. Ainsi, Condor-G est totalement interopérable avec des ressources distribuées à grande échelle et administrées par le système Globus.

Condor est disponible en libre téléchargement sur le site officiel à l'URL http://www.cs.wisc.edu/condor/downloads avec des distributions pour les plateformes Linux et Windows NT.

II. Fonctionnalités

Les mécanismes de gestion des ressources dans Condor sont étendus. Le système fournit une grande flexibilité à la fois aux soumissionnaires de travaux et aux propriétaires de machines au sein du pool. Cette liste récapitule certaines possibilités du condor.

Soumission distribuée : Au sein d'un pool Condor, il n'y a pas une machine centralisée uniquement pour la soumission des « Jobs » ou travaux soumis; ces derniers peuvent être soumis à partir de plusieurs machines et chacune a sa propre file d'attente. Les utilisateurs peuvent soumettre à partir de leur station de travail.

Priorités d'un job : Les utilisateurs peuvent assigner des priorités à leurs travaux soumis afin de commander leur ordre d'exécution des travaux.

Priorités des utilisateurs : Les administrateurs peuvent assigner des priorités aux utilisateurs en utilisant un mécanisme flexible qui permet une politique de partie équitable, de la commande stricte, de la commande partielle, ou d'une combinaison de ces politiques.

Dépendance des jobs : Quelques ensembles des travaux exigent une commande due aux dépendances entre eux. Par exemple, le job X ne peut être exécuter qu'après Y et Z. L'exécution des dépendances est facilement manipulée dans Condor.

Suspension et reprise d'un job : Basé sur des règles, le Condor peut suspendre un job et le reprendre ultérieurement.

Système d'appel à distance (*Remote Systeme Call*) : En dépit des jobs en cours d'exécution sur les machines à distance, Condor peut souvent préserver

l'environnement local d'exécution par l'intermédiaire des appels de système à distance. Les utilisateurs n'ont pas besoin de rendre des fichiers de données disponibles ou même d'obtenir un compte d'ouverture sur les postes de travail à distance avant que le condor exécute leurs programmes. Le programme s'exécute sous le condor comme s'il fonctionnait sur le poste de travail d'où il a été soumis.

Plusieurs pools de machines peuvent fonctionner ensemble : Le mécanisme de « *Floking* » permet aux travaux d'être exécuté à travers de multiples pools de condor appartenant à différents organismes qui imposent leurs propres politiques de gestion d'accès.

Authentification et autorisation : Les administrateurs détiennent le contrôle des autorisations d'accès, l'authentification est effectuée en utilisant Kerberos[5] et les certificats X.509[6] .

Des plateformes hétérogènes : Un pool Condor peut contenir de multiples plateformes Linux, Unix et Windows NT. Comme exemple, un exécutable qui fonctionne sous Windows 2000 peut être soumis sur une machine tournant sous Linux.
Nous venons de voir quelques fonctionnalités qu'offre le système Condor. Dans la suite de cette partie nous allons voir les mécanismes fondamentaux qui ont fait de Condor un puissant environnement de gestion de ressources.

a. Les classAds

Une ClassAd est une représentation flexible des caractéristiques et contraintes des machines et des jobs (travaux soumis) dans le système Condor. Il est utilisé par les utilisateurs pour spécifier quelles machines devraient exécuter leurs travaux ; les administrateurs les utilisent aussi pour définir les politiques d'ordonnancement.
Toutes les machines du Condor publient leurs attributs tels que la RAM, vitesse du processeur, taille mémoire virtuelle, heure et date, avec d'autres propriétés statiques et dynamiques dans une ClassAd, ainsi que les conditions dans lesquelles elles peuvent exécuter un job Condor et le type de job. Ces attributs reflètent les termes et préférences

[5] **Kerberos** est un protocole d'identification réseau créé au MIT. Kerberos utilise un système de tickets au lieu de mots de passe en texte clair. Ce principe renforce la sécurité du système et empêche que des personnes non autorisées interceptent les mots de passe des utilisateurs.

[6] **X.509** est un standard de cryptographie de l'Union internationale des télécommunications pour les infrastructures à clés publiques (PKI). X.509 établit entre autres les formats standards de certificats électroniques et un algorithme pour la validation de chemin de certification.

individuels que les différents propriétaires des ressources ont « gracieusement » laissé pour participer au pool condor.

En en soumettant un travail au condor, un job classAd est créé. Cette classAd inclut des attributs sur le job, le nom du programme, le soumissionnaire du job, l'heure de la soumission, et d'autres informations nécessaires au processus d'exécution du job. Les conditions et préférences de la machine qui doit l'exécuter sont aussi spécifiées.

Le langage des classads est très simple, puisque leur contenu se limite à une liste d'attributs auxquels sont associés une valeur qui peut être un entier, un réel, une chaîne de caractères ou une expression plus complexe. Les attributs simples sont décrits ainsi :

Machine	**Job**

`Type = "Machine";`
`Disk = 323496;`
`Memory = 64;`
`Untrusted =`
`{"rival","bad"};`
`...`
	`Type = "Job";`
	`Owner = "Raman";`
	`Memory = 31;`
	`...`

Pour les attributs un peu complexes comme les contraintes, leur valeur se présente sous la forme d'une expression structurée à partir d'opérateurs arithmétiques et logiques appliqués sur d'autres attributs :

Machine	**Job**

`Constraint =`
`!member(other.Owner,Untrusted)`
`: DayTime < 8*60*60`
`|| DayTime > 18*60*60;`
	`Constraint =`
	`other.Type == "Machine" &&`
	`Disk >= 10000 &&`
	`other.Memory >= self.Memory;`

La possibilité de définir soi-même ses contraintes aussi bien en tant que client ou fournisseur rend les classads très extensibles et adaptables aux politiques et désirs de chacun. Dans l'exemple proposé ci-dessus, la machine a une liste noire d'utilisateurs et est disponible avant 8h et après 18h, alors que le client demande pour sa tâche un espace disque minimum et une quantité de mémoire supérieure à la sienne. [8]

b. Matchmaking

C'est le mécanisme qui associe un travail non encore exécuté (Client) avec une machine disponible (Fournisseur). Il exige quatre étapes représentées sur le schéma suivant:

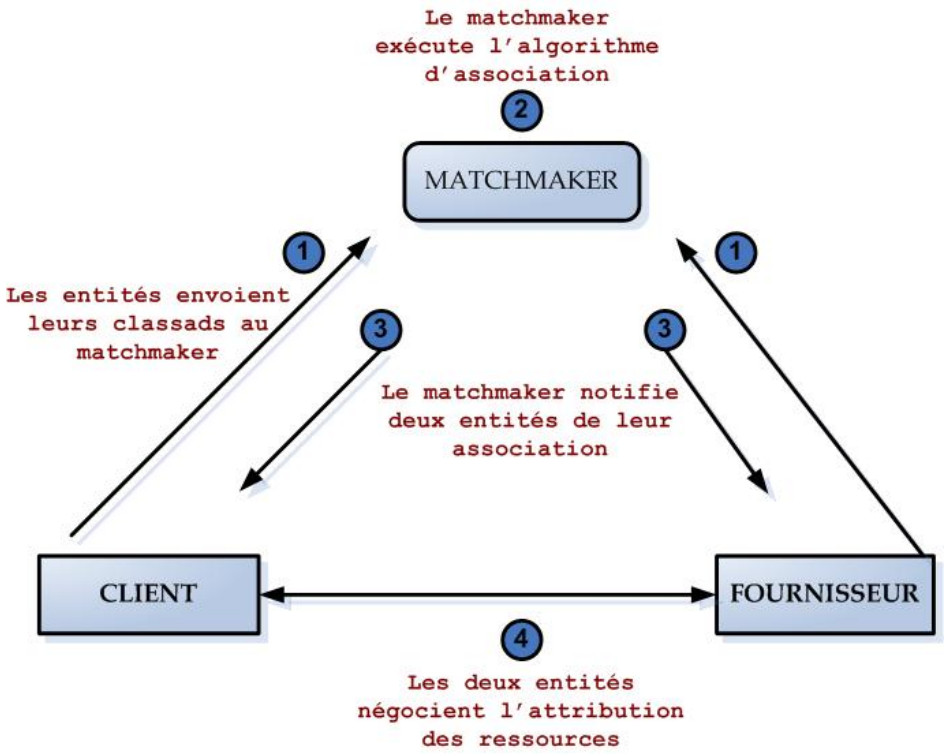

Figure 5 : Matchmaking dans Condor

Le mécanisme de matchmaking se décompose de la manière suivante :

- Les entités qui proposent ou demandent des services décrivent leurs caractéristiques et leurs requêtes dans des annonces classées ou Classads.
- Le matchmaker associe les annonces dont les contraintes correspondent.
- Le matchmaker informe les entités associe et son rôle s'arrête ici.
- Les entités entrent en contact et négocient l'éventuelle attribution de la ressource.

C'est ainsi que le framework Matchmaking repose sur cinq composants :

- la spécification des classads, qui définit le langage utilisé dans les classads,
- le protocole d'annonce, qui définit comment le matchmaker reçoit les annonces,
- l'algorithme de matchmaking, qui définit comment les annonces compatibles sont associées,
- le protocole de matchmaking, qui définit comment les entités sont informées d'une association les concernant,

- le protocole d'attribution, qui définit comment les entités associées négocient l'allocation de la ressource.

Un des points forts de cette approche est de permettre aux fournisseurs de ressources de pouvoir spécifier des contraintes sur les clients qu'ils veulent servir. D'autre part, le matchmaker n'alloue pas directement les ressources, il se contente d'introduire mutuellement une entité client à une entité fournisseur, en leur laissant le soin de s'accorder sur l'attribution de la ressource.

c. Checkpointing et Migration

La capacité de faire migrer un travail en cours d'exécution sur une machine oisive (disponible et non utilisée par son propriétaire) est fondamental à l'ordonnancement des travaux dans un pool Condor. Pour permettre à ces travaux de progresser, Condor utilise un « *checkpoint* » ou point de contrôle de l'état de ces travaux. Le checkpoint (dans condor) d'un programme revient à faire une copie :

- du programme (le code binaire exécuté par la machine),
- de la pile d'exécution (position dans le code du programme),
- de l'état des descripteurs de fichier (pointeurs vers les fichiers ouverts et position dans ces fichiers). Il suffit ensuite de relancer le programme avec ces informations pour le retrouver dans le même état qu'au moment du checkpoint.

Un checkpoint est généré juste avant la migration d'un travail. Il est alors employé comme point de départ pour le travail après migration.

Ainsi, Condor fournit aux programmes la bibliothèque « *checkpointing library* » contenant le signal de point de contrôle appelé « *Condor checkpoint signal* ». Quand un travail reçoit un signal, il crée un fichier contenant les données des processus, aussi bien que des informations sur les dossiers ouverts, les signaux suspendus et l'état actuel de l'unité centrale. Quand le travail est redémarré, l'état contenu dans le fichier de sauvegarde est restauré et le processus reprend le calcul à partir du point où le checkpoint a été généré.

Checkpoint server *(serveur de point de contrôle)*

Par défaut, un checkpoint est écrit sur un fichier du disque local de la machine où la tâche a été soumise. Dans ce cas une quantité suffisante d'espace disque doit être disponible pour stocker les images du checkpoint. Cette image est approximativement égale à la mémoire virtuelle consommée par la tâche pendant son exécution. C'est ainsi

qu’un serveur de checkpoint a été développé pour sauvegarder les checkpoint de tout un pool Condor. Quand un ordinateur est configuré pour utiliser ce serveur, les travaux soumis sur cette machine écrivent et lisent les checkpoint du serveur au lieu de son disque local.

d. DAGMan

Un graphique acyclique dirigé (DAG) est employé pour représenter des programmes dont l'exécution dépend d'un ou plusieurs autres programmes. Dans ce graphe, les noeuds (sommets) sont les programmes, et les arcs identifient les dépendances.

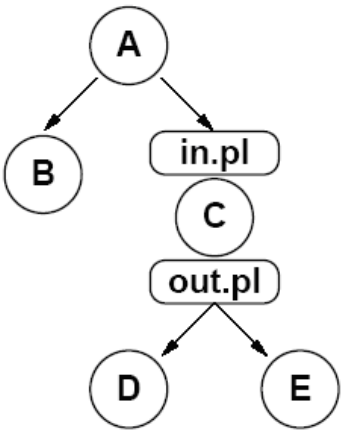

Figure 6 : Graphe acyclique dirigé (DAG)

Le Directed Acyclic Graph Manager (DAGMan) est un méta-ordonnanceur pour les travaux dans Condor. DAGMan soumet les travaux au condor dans l’ordre représenté par le DAG et traite les résultats. Un dossier d'entrée défini avant la soumission décrit le DAG, et un fichier de soumission pour chaque programme du DAG est utilisé par Condor.

Chaque noeud (programme) dans le DAG indique un fichier de soumission. Pendant que DAGMan soumet les travaux au condor, il surveille les fichiers de notation de condor pour imposer l‘ordre exigée par le DAG.

Le Directed Acyclic Graph Manager est responsable de l’ordonnancement et du rétablissement de l’ensemble de travaux soumis au Condor. Le schéma ci dessous montre le langage accepté par DAGMan :

```
JOB A a.condor
JOB B b.condor
JOB C c.condor
JOB D d.condor
JOB E e.condor
PARENT A CHILD B C
PARENT C CHILD D E
SCRIPT PRE C in.pl
SCRIPT POST C out.pl
RETRY C 3
```

Figure 7 : un fichier DAG

- Les premières lignes d'un fichier DAG listent tous les travaux qui seront utilisés dans le graphe. Chaque ligne commence par le mot clé JOB, suivi du nom du job et de son fichier de soumission.
- PARENT-CHILD décrit la relation entre deux ou plusieurs travaux. Dans cette séquence, les travaux B et C ne peuvent s'exécuter sans que A ne soit terminé, les travaux D et E aussi ne peuvent s'exécuter sans que C ne soit terminé. Les travaux qui sont indépendant peuvent fonctionner dans n'importe quelle ordre et simultanément.
- Des traitements peuvent être requis avant ou après la soumission d'un travail au Condor. Pour cela, on utilise la commande SCRIPT suivi de PRE si le traitement s'effectue avant la soumission, ou de POST si le traitement se fait après que le travail soit accompli.

III. Architecture

Un pool Condor comprend :

- une machine unique qui sert de Manageur central
- et un nombre arbitraire d'autres machines qui joignent le pool au fur et à mesure

Conceptuellement, le pool est une collection de ressources (machines) et de ressources demandeurs (jobs). Le rôle du Condor est de joindre les demandes en attente avec les ressources disponibles. Toutes les ressources du Condor envoient les mises à jours

périodiques au manageur central, le dépôt d'information central sur l'état du pool. Le manageur central évalue périodiquement l'état présent du pool et essaie de joindre les demandes avec les ressources appropriées.

a. Les démons du Condor

Pour comprendre l'architecture du Condor, la liste suivante décrit les principaux démons qui tournent dans le système et leur rôle :

condor_master : son rôle est de simplifier l'administration du système. Il assure le bon fonctionnement des autres démons Condor s'exécutant sur les machines du pool. Le master engendre les autres démons. En somme, si l'un des démons d'une machine s'arrêtent anormalement, le condor_master enverra un mail à l'administrateur du système et le démon affecté est automatiquement redémarré. Il tourne sur toutes les machines du pool.

condor_startd : ce démon représente une machine dans un pool Condor. Elle annonce la ClassAd d'une machine. Il permet aux machines d'exécuter les travaux. Le condor_startd est responsable d'imposer la politique sous laquelle des taches à distance seront démarrés, suspendus, repris, évacués, ou tués. Quand il est prêt à exécuter un travail de condor, il engendre le condor_starter, décrit ci-dessous.

condor_starter : ce programme est l'entité qui engendre le travail à distance de condor sur une machine donnée. Il installe l'environnement d'exécution et surveille le travail une fois qu'il s'exécute. Le condor_starter détecte l'accomplissement de la tâche, envoie le statut de cette dernière à la machine de soumission, et s'arrête.

condor_scheed : ce démon représente les jobs dans le pool. Ce démon doit tourne sur toutes les machines habilitées à soumettre des travaux. Les utilisateurs soumettent leur job au condor_scheed, où est stockée la queue des travaux. Les divers outils pour voir et manipuler la queue des travaux se connectent au condor_scheed pour accomplir leur tache.

condor_shadow : ce programme fonctionne sur la machine où un travail a été soumis L'ombre sert des demandes des fichiers au transfert, enregistre la progression du travail, et établit des statistiques quand le travail est terminé.

condor_collector : ce démon est responsable de la collection de toutes les informations sur l'état du pool. Tous les autres démons lui envoient périodiquement les mises à jour

de classAd. Ces classAd contiennent toutes les informations sur l'état des démons, les ressources qu'elles représentent ou les ressources soumis au pool. Le condor_collector peut être vu comme une base de données dynamique des classAd. Il tourne sur la machine désignée comme manageur central.

condor_negociator : il contrôle le matchmaking dans le système condor.

b. Le demon Condor en action

Une machine sert de manageur central. Le démon **condor_master** tourne sur chaque machine du pool. Le central manageur tourne fait tourner les démons **condor_collector** et le **condor_negociator**. N'importe quelle machine dans l'installation qui est capable d'exécuter des travaux doit avoir le **condor_startd**, et chaque machine qui peut maintenir une queue de job et aussi qui permet aux utilisateurs de cette machine de soumettre des jobs doit avoir le **condor_schedd**.

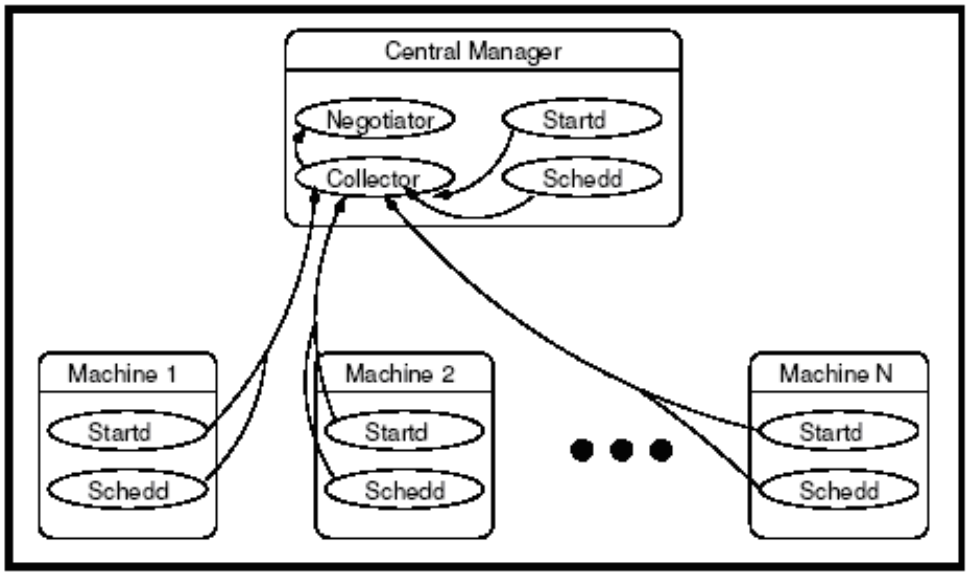

Figure 8 : les démons de Condor

L'interface pour ajouter un job au condor est condor_submit, qui lit le fichier description de la tache à exécuter, crée le classAd, et donne ce classAd au condor_schedd. Ceci déclenche un cycle de négociation.

Pendant un cycle de négociation, le condor_negociator interroge le condor_collector pour connaître la liste de toutes les machines qui sont disposées à exécuter un travail et de tous les utilisateurs avec les travaux de veille. Le condor_negociator communique avec chaque condor_schedd qui a des travaux dans leur file d'attente, et exécute le

matchmaking pour associer les travaux avec des machines disponibles tels que les conditions spécifiées dans les ClassAd du travail et de la machine soient satisfaites.

Quand le condor_schedd démarre une tache, il lance le processus du condor_shadow sur la machine de soumission et le condor_startd lance à son tour processus du condor_starter sur la machine d'exécution correspondante.

Le condor_shadow transfère le classAd du travail et tous les fichiers de données nécessaires au condor_stater qui lance l'application.

Quand le travail se termine ou est interrompu, le condor_starter arrête tous les processus et libère l'espace disque utilisé par l'application.

IV. Les univers de Condor

Condor dispose de plusieurs environnements d'exécution qui sont appelés univers. Ils sont au nombre de six dont les plus utilsés sont les univers Standard, Vanilla et Globus :

- Vanilla
- MPI
- PVM
- Globus
- Scheduler
- Standard

L'univers choisi par l'utilisateur est indiqué dans le fichier de description de soumission. S'il n'est pas indiqué, c'est le *Standard* qui sera utilisé par défaut.

Vanilla

L'univers Vanilla est le plus simple du système Condor. Il est utilisé pour exécuter des jobs séquentiels (non-parallèles). Il ne fournit aucune fonctionnalité pour le checkpointing et ne supporte pas aussi les appels systèmes distants. Le programme sera envoyé, avec les fichiers nécessaires à son exécution, sur une machine disponible. Puis la sortie sera renvoyée sur la machine de soumission. Un travail typique de l'univers *Vanilla* se fonde sur un système de fichier partagé entre la machine de soumission et tous les noeuds afin de permettre aux travaux d'accéder à leurs données.

Standard

L'univers *Standard* est l'univers par défaut de Condor. Cet univers offre les même possibilités que l'univers Vanilla avec en plus le checkpointing et les appels systèmes distants (en particulier les I/O).

MPI

C’est l’environnement de Condor qui permet aux programmes parallèles écrits avec MPI d’être exécuté.

PVM

Il permet d’exécuter des applications en modèle maître-esclave écrits avec l'interface de PVM.

Globus

L'univers de Globus dans le condor est destiné pour fournir l'interface standard de Condor aux utilisateurs qui souhaitent soumettre les travaux aux machines contrôlées par Globus.

Scheduler

Il est employé pour soumettre un travail qui sera exécuté immédiatement par opposition à une machine à distance. Le but est de fournir un service aux programmeurs qui souhaitent contrôler la soumission et le déplacement des travaux dans la file d'attente de condor.

Rapport-gratuit.com
LE NUMERO 1 MONDIAL DU MÉMOIRES

PARTIE III : NOTRE ETUDE

I. Mécanismes de tolérances aux pannes dans Condor

Le Condor actuel ne sait pas gérer toutes les défaillances. Par contre, il offre des mécanismes de base permettant de réaliser des solutions applicatives ou "services système" de gestion des défaillances telles que le mécanisme de point de reprise ou de sauvegarde *(checkpoint)*, DAGMan.

a. Le mécanisme de point de reprise *(checkpoint)*,

Rappelons que le *checkpoint* ou point de reprise est une sauvegarde en fichier image de l'état d'exécution d'un job Condor avec toutes le données nécessaires à un instant T. Ces « checkpointing » sont effectués périodiquement dans le système Condor. Tout dépend de la configuration des ordinateurs du pool.

Par exemple : toutes les deux heures, jamais, etc.

Au moment du lancement du checkpoint périodique, le travail en cours suspend son traitement, exécute le checkpoint et continue immédiatement d'où il a cessé. Il y a également une commande *condor_ckpt* qui permet à l'utilisateur de forcer l'exécution immédiate d'un point de reprise. Dans tous les cas, les travaux de condor continuent leurs exécutions au point de reprise complet le plus récent.

C'est ainsi que ce mécanisme de checkpoint périodique fournit la tolérance aux pannes pour plusieurs raisons :

- Quand un utilisateur soumet une tâche, Condor trouve une machine disponible dans le pool et commence à l'exécuter sur cette machine. Condor a la possibilité de détecter qu'une machine exécutant un job Condor n'est plus disponible (par exemple son propriétaire commence à utiliser le clavier) : il lance alors un checkpoint et fait migrer le travail sur une autre machine.
- Les points de reprise périodique sont aussi indispensables en cas d'arrêt ou de panne totale de la machine d'exécution, dans ce cas le travail est migré sur une machine et continuera son exécution à partir du dernier point de reprise. Donc il n'y aura pas de perte de temps ce calcul.

b. Le gestionnaire de graphe acyclique dirigé *DAGMan*

Le DAGMan, comme nous l'avons vu dans le chapitre précédent permet de résoudre les dépendances de travaux dans un pool Condor. Ces travaux peuvent échouer en raison de la nature du système réparti. Les pannes de réseau et les ressources reprises peuvent

rompre le contact entre Condor et le job en cours d'exécution. De telles pannes ne traduisent pas l'échec du travail lui-même, mais plutôt que le système a échoué. De telles situations sont détectées et relancées par la machine exécutant le programme dans sa responsabilité d'exécuter les travaux sûrement.

Un travail peut aussi produire un résultat erroné si l'utilisateur oublie de fournir un argument d'entrée nécessaire. Dans ce cas-ci, DAGMan se rend compte que le travail est terminé et voit un résultat de programme indiquant une erreur. Il répond en écrivant une délivrance DAG et en quittant avec un code d'erreur. La délivrance DAG est un nouveau DAG énumérant les éléments du DAG initial qui ne sont pas exécutés. Pour remédier à cette situation, l'utilisateur peut examiner la délivrance DAG, fixe toutes les erreurs dans la soumission, et les soumet à nouveau comme un DAG normal.

Quelques erreurs peuvent ne pas être détectées par le système réparti. Par exemple, des fichiers exécutables corrompus devraient être détecté par le système réparti et être relancé au niveau de la machine de soumission. Pour ces raisons, DAGMan permet à l'utilisateur d'indiquer qu'un travail échoué soit relancé, en utilisant la commande RETRY.

II. Etude des défaillances à gérer dans Condor

Dans cette partie nous allons étudier les défaillances observées dans la plateforme Condor, pour ensuite en dégager ceux qui feront l'objet de notre étude. Pour mieux comprendre et cerner notre étude, nous allons faire un bref rappel des concepts de défaillances au sein des systèmes distribués.

a. Définitions

Un système est sûr de fonctionnement s'il permet à ses utilisateurs de placer une confiance justifiée dans les services qu'il délivre.

Critères d'appréciation

- fiabilité (*reliability*) : le système est prêt à fournir le service
- disponibilité (*availability*) : fraction du temps durant laquelle le système fournit le service
- sécurité (*safety*) : un mauvais fonctionnement du système n'a pas de conséquence catastrophique sur l'environnement

– sécurité (*security*) : le système préserve l'intégrité et la confidentialité des informations : le service est effectivement accessible aux utilisateurs autorisés, non aux autres.

Un système est sujet à une défaillance (failure) lorsque son comportement n'est pas conforme à sa spécification.

De la faute à la défaillance

D'une manière générale, c'est la faute qui est la cause de l'erreur. Par exemple une faute logicielle, un court circuit sur une carte, etc. La faute produit une erreur lorsqu'elle est activée. Donc l'erreur est une manifestation de la faute quand la partie fautive du système est utilisée. Ce qui entraîne une défaillance de système.

Figure 9 : Causes des défaillances

Il n'existe pas de méthodes de tolérance aux fautes valables dans l'absolu, seulement des méthodes adaptées à des hypothèses particulières d'occurrence de fautes. Ces hypothèses doivent donc être explicitement formulées, après analyse soigneuse.

b. Les défaillances dans Condor

Précédemment, nous avons vu que la plateforme Condor n'offre pas beaucoup de mécanisme de traitements des défaillances du système à part la technique des points de reprise (*checkpoint*) et le gestionnaire des graphes acycliques dirigés (*DAGMan*).

Condor étant un environnement de calcul haut débit *(High-Throughput Computing)*, son objectif est de gérer et exploiter efficacement toutes les ressources informatiques disponibles. Donc le système doit prendre en charge le plus possible la gestion des défaillances, de façon à minimiser les indisponibilités, et faire travailler les machines le plus longtemps possible sans intervention.

C'est ainsi que nous avons orienté notre étude au niveau de l'étape d'exécution des programmes. En effet, quand l'utilisateur soumet un travail au Condor, il établit ses besoins dans un fichier ClassAd dans lequel il spécifie les caractéristiques de la

ressource, par exemple le type de machine, taille du disque, système d'exploitation, … A partir de là, une ressource est attribuée et le programme commence à s'exécuter.

Nous avons pu constaté que pendant l'exécution, la mémoire utilisée par le travail n'était pas contrôlée, ce qui entraîne des erreurs de saturation mémoire, donc d'indisponibilité de la ressource utilisée.

Précisément, un programme est une série d'instructions dont l'exécution modifie l'*état mémoire* de la machine. Cette mémoire est constituée principalement des valeurs créées ou manipulées par le programme. Or, comme toute ressource informatique, la mémoire disponible pour une application est finie ; un programme tentant d'utiliser plus de ressources que le système ne lui en attribue se retrouvera dans un état incohérent. Au fil de l'utilisation, les performances du système ont tendance à se dégrader.

Certes l'utilisateur a la possibilité de mentionner la taille de la mémoire nécessaire dans sa classAd, mais il s'agit bien sur de la taille totale de la mémoire de la machine, mais pas ce qu'il en reste à un instant T.

Les problèmes de saturation mémoire ne sont pas pris en charge par Condor.

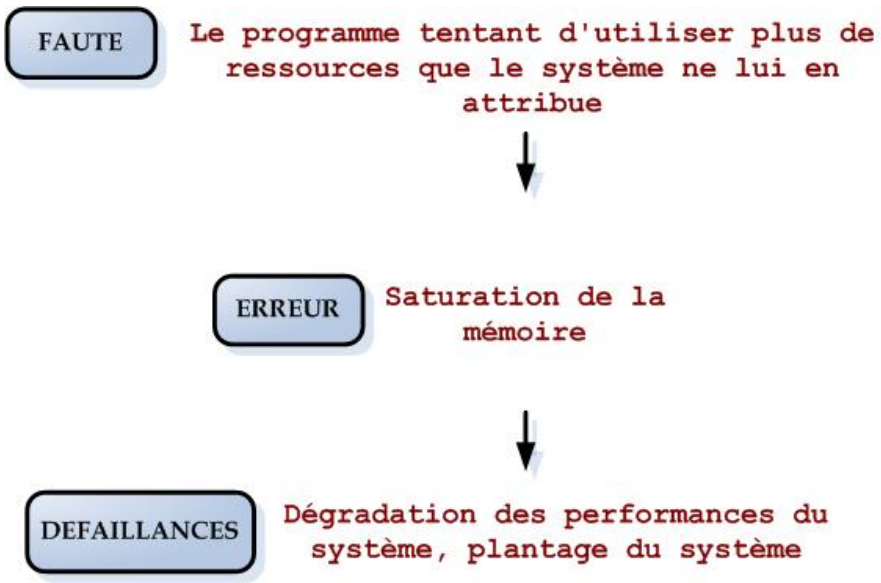

Figure 10 : Saturation mémoire

Nous nous fixons donc comme objectifs de gérer les défaillances causées par la saturation de la mémoire d'une machine en utilisant les agents mobiles.

III. Utilisations des agents mobiles

L'idée est de mettre un agent qui surveille l'évolution de la mémoire, et de notifier la machinerie Condor en cas de dépassement.

a. Les agents mobiles dans les grilles de calcul

Les agents mobiles constituent un outil d'adaptation des applications aux variations de la qualité des services offerts par le système et le réseau ; ils doivent eux-mêmes s'adapter dynamiquement à des conditions d'exécution variables. Les réseaux à grande échelle, tels l'Internet, les grilles de calcul ou de stockage donnent accès à des quantités de données, de services et de ressources répartis.

La programmation par agents mobiles est un paradigme complémentaire pour la programmation des applications réparties. Un agent mobile possède la faculté de se déplacer en emportant son code, ses données, ainsi que son état d'exécution ; il décide lui-même de manière autonome, de ses déplacements. Le rapprochement du client (représenté par un agent) et du serveur permet de diminuer les interactions distantes (en recherche d'information, par exemple) ou de spécialiser un service (adaptation du flux de données d'un serveur multimédia aux capacités du client, par exemple). Le principe de délégation permet à l'agent d'opérer en étant déconnecté du client. La mobilité des agents et leur autonomie améliorent la sûreté (tolérance aux pannes par redéploiement des agents sur des noeuds en service) et permettent le suivi de matériel ou d'utilisateurs mobiles (réseaux actifs, informatique nomade). Enfin, le mode de communication asynchrone permet de s'accommoder de la latence et des perturbations des réseaux à grande échelle ou sans fil. Les agents mobiles sont donc un outil à part entière pour l'adaptation des systèmes.

Avec les grilles de calcul, les agents mobiles trouvent un champ d'application naturel. En effet, les caractéristiques de déplacement, d'adaptation, de coopération vont permettre de tirer partie de ces environnements.

Les agents mobiles vont pouvoir adopter ce comportement naturel de négociation. Dans les grilles, les organisations offrent, généralement, des expertises de haut niveau nécessitant de longue phases de traitement et/ou la mise en oeuvre d'un ensemble de sous-services. Avec un utilisateur nomade, se connectant de manière intermittente à la grille, l'utilisation des agents mobiles permet de déléguer la réalisation de la tâche que l'utilisateur souhaite exécuter.

b. Nos agents mobiles

Conceptuellement, les agents doivent surveiller le niveau de mémoire de chaque machine du pool Condor et dès l'instant que l'utilisation dépasse 90 %, ils signalent un risque de saturation mémoire et donc d'indisponibilité de la machine exécutant le travail soumis.
Dans cette section, nous allons détailler les fonctionnalités et spécificités de notre agent mobile au sein de Condor.

Structure de l'agent mobile

Un agent mobile est défini par un modèle d'agent. Le modèle d'agent définit l'état d'un agent par un ensemble de paramètres d'une part et des primitives d'accès à cet état permettant d'initialiser et modifier les paramètres de l'agent, d'autre part.
Une instance d'agent mobile comporte trois parties :

- des données
- du code
- un contexte d'exécution

Les données sont les valeurs des paramètres définis par le modèle d'agent. Parmi les données d'un agent mobile on trouve son nom, son itinéraire, le programme de la tâche à exécuter, un dossier destiné à recevoir les résultats de l'exécution de ce dernier. Le code met en oeuvre les primitives d'accès aux valeurs des paramètres. Le contexte d'exécution reflète l'état d'exécution courant de l'agent mobile.

Création de notre agent

La création d'un agent consiste à créer une instance d'agent sur une place donnée. Elle s'effectue à partir du modèle d'agent. Les données d'un agent sont initialisées lors de sa création. Une fois créé, l'agent peut être actif c'est à dire que l'exécution du code de la tâche associée à l'agent est déclenchée. La création/activation d'un agent peut prendre plusieurs formes :

- création locale
- création et exécution à la demande
- création et exécution à distance

On parle de création locale lorsque l'agent est créé et activé sur la place située sur le site ayant émis la requête de création. Lorsque la création est locale et que l'activation a lieu sur un autre site à distance (l'agent est transféré après sa création et avant son activation), on parle de

création avec exécution à distance. Dans ce cas, la place sur laquelle l'agent doit être activé est déterminée par l'entité ayant émis la requête de création de l'agent.
Dans le cas de la création avec exécution à la demande, une place initialise le transfert du code de l'agent. L'exécution à la demande peut être utile lorsque une place désire à un moment donné que l'on lui amène un agent mobile depuis une autre place.
Dans le cas de notre étude, la création de notre agent s'effectuera sous la troisième à savoir création et exécution à distance. Techniquement, nous allons placer un générateur d'agent mobile au niveau du manageur central, point de départ de toutes les tâches soumises au sein du pool Condor. Ainsi dés que la commande de soumission d'un travail, *condor_submit*, est lancée un agent mobile sera créé au niveau du serveur central de Condor, pour ensuite migrer vers la machine d'exécution.

Exécution de l'agent

Une fois initialisé, l'agent peut commencer son exécution. L'exécution de l'agent consiste en l'exécution du code de la tâche qui lui est associée. Elle a pour effet de modifier les données de l'agent.
Donc dés l'instant où l'agent mobile migre au niveau de la machine d'exécution, il commence à exécuter en même temps le programme soumis. Il surveille ainsi le niveau de la mémoire de l'ordinateur en effectuant des entrées/sorties.
En effet, les agents mobiles peuvent être amenés à accéder au système de gestion de fichiers ou effectuer des entrées et sorties. Les agents ont donc besoin d'accéder au système hôte. Toutefois, les mécanismes d'accès varient d'un système d'exploitation à l'autre. Notre agent mobile devra accéder aux ressources locales du site à savoir les paramètres et variables représentant l'état de la mémoire d'une machine.
Dés que son utilisation dépasse les 90%, l'agent lance un point de reprise (checkpoint) avec la commande *condor_ckpt* et le travail est migré sur une autre machine. L'agent mobile se déplace aussi en même temps que le travail et commence à exécuter ses tâches comme décrit précédemment.
Précisément, c'est la migration faible qui sera utilisé ici, puisque notre agent se déplace uniquement avec son code et ses données. Ces derniers seront enfin utilisés au niveau de la machine d'exécution pour instancier une nouvelle copie de l'agent. Ce processus continue jusqu'à la fin de l'exécution du travail soumis.

CONCLUSION

Au cours de ce stage, nous avons d'abord évalué l'environnement Condor particulièrement et nous avons constaté que très peu de mécanismes de gestion des défaillances sont développés dans le système Condor. Nous avons ainsi orienté notre étude au niveau de l'étape d'exécution des programmes dans le pool Condor. Nous avons découvert des problèmes de saturation mémoire que la machinerie Condor ne détectait pas.

Ensuite, nous avons fait une étude sur les agents mobiles et surtout leur rôle dans les systèmes de calcul haut débit.

Au terme de cette étude, nous proposons une solution à base d'agents mobiles qui surveillent le niveau de mémoire de chaque machine et dès l'instant que l'utilisation dépasse 90%, ils signalent un risque de saturation mémoire et donc d'indisponibilité de la machine en lançant un point de reprise (ou checkpoint) avec la commande *condor_ckpt*. Ce qui fait migrer le programme sur une autre machine disponible avec une instance de l'agent mobile.

Ce stage a été un excellent moyen pour nous de découvrir l'univers des systèmes répartis et parallélisme, domaines de l'informatique en plein essor, avec pour perspective, des réflexions poussées sur les problèmes de gestion et d'accroissement de la disponibilité des infrastructures répartie, les technologies agents et les protocoles de mise en correspondance entre offre et demandes de ressource au sein d'une grille informatique.

REFERENCES

[1] Foster & Kesselman: **The Grid: Blueprint for a New Computing Infrastructure**, *Morgan Kaufmann*, 2003 (2ème édition en 2004 sous le titre GRID2)

[2] http://www.altivis.fr/-Le-cadre-technologique-.html

[3] Bruno Bzeznik: CiGri: **CiGri: a lightweight Grid for parametric applications**, *Ecole Doctorale - 2007-01*

[4] Marcel Soberman CNRS / STIC, **Les grilles informatiques - état de l'art et déploiement**

[5] **La technologie des agents mobiles**, Cours propose par François BOURDON

[6] Akhil Sahai, "**Conception et réalisation d'un gestionnaire mobile de réseaux fondé sur la technologie d'agent mobile**", Thèse de Doctorat de l'Université de Rennes I, 25 janvier 1999 – Document INRIA TU-0832 : http://www.inria.fr/rrrt/tu-0832.html

[7] *Imed Jarras et Brahim Chaib-draa,* **Aperçu sur les systèmes multiagents**, Montréal Juillet 2002

[8] Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny, "**Condor – A Distributed Job Scheduler**", in Thomas Sterling, editor, Beowulf Cluster Computing with Linux, The MIT Press, 2002. ISBN: 0-262-69274-0

[10] Condor Team, University of Wisconsin–Madison, **Condor Version 6.8.4 Manual**, February 6, 2007, http://www.cs.wisc.edu/condor