Calcul Global, Desktop Grids et XtremWeb

Franck Cappello, INRIA, LRI, Université Paris Sud, fci@lri.fr
Samir Djilali, LRI, Université Paris sud, djilali@lri.fr
Gilles Fedak, UCSD, fedak@lri.fr
Tangui Morlier, INRIA, tmolier@lri.fr
Oleg Lodygensky, LAL, LRI, Université Paris Sud

1 Introduction

Nous présentons dans cette partie les notions fondamentales du calcul global, des Desktop Grid et de la plate-forme XtremWeb. Nous commencons par évoquer l'historique de ces systèmes puis nous présentons XtremWeb. Nous détaillons les interfaces utilisateurs d'XtremWeb avec des exemples concrets. Nous présentons l'application de cette plate-forme au projet d'astroparticules Auger. Enfin, nous évoquerons les thèmes de recherche sur les Desktop Grid et leur rapprochement avec les systèmes de Grid classiques comme Globus.

2 Historique du calcul global

Le Calcul Global est un concept très récent qui peut être considéré comme l'extension à l'échelle d'Internet du principe de vol de cycles : des ordinateurs volontaires se connectent à un serveur pour obtenir des tâches à exécuter. Ces systèmes exploitent un parallélisme trivial pour lequel aucune communication entre tâche n'est nécessaire. Le code des applications, les paramètres et les résultats sont communiqués par le serveur aux machines volontaires. Les premières expériences de ce type de plates-formes ont eu lieu dans les années 90. Il s'agissait alors de "craquer" des clés de cryptage de type RC5 et DES en faisant une recherche exhaustive. D'autres projets ont recherché des grands nombres premiers par la méthode de Mersenne. Plus récemment, des projets ont tenté de "craquer" des clés de cryptage de type RSA. Le potentiel du Calcul Global a été démontré en 1996-1997 avec ces projets qui réunirent 250, puis 3500 et enfin 14000 PCs.

A la fin des années 90, plusieurs projets regroupés sous le terme de "Calcul basé sur Internet" (Web-based Computing) comme JET [11], Bayanihan[12], SuperWeb[1], Javelin [7], Popcorn [10] ou Charlotte [4] ont vu le jour. Ces projets proposaient des environnements de développement et d'exécution pour les applications sur Internet. Le terme Global Computing était alors déjà utilisé pour représenter ces systèmes. La plupart sont nés avec le langage Java et exploitent certaines de ses caractéristiques comme l'existence d'un langage intermédiaire (bytecode) qui permet aux applications d'être portables sans recompilation et d'exécuter les applications dans environnement sécurisé pour les machines participantes. Le projet qui a réellement popularisé le concept de Calcul Global est SETI@home[2]. Le lancement officiel du projet est très récent (Avril 1999) mais le projet et le développement de son code remonte à 1997. SETI@home a démontré plusieurs points : 1) il est possible de fédérer une communauté de plusieurs centaines de milliers d'utilisateurs pour faire du calcul numérique, 2) le système développé soutient une production de calcul considérable (30 Teraflops) ; rappelons que cette valeur n'est pas directement comparable aux performances des machines parallèles classiques, 3) l'organisation de serveur de calcul et du collecteur de résultats permettent de satisfaire plusieurs millions de tâches par jour. Le projet a déjà produit l'équivalent 500000 années de calcul de PC. Actuellement, le code de SETI@home évolue à partir de la plate-forme BOINC (boinc.berkeley.edu) développée par la même équipe à Berkeley.

Ces résultats ont ouvert la voie à des projets académiques et industriels. Il existe actuellement dans le monde plusieurs dizaines de projets de ce type. Les plus connus sur le plan industriel sont United Devices, Platform Active-Cluster, DataSynapse, Grid Systems et COSM. Dans le monde académique, les projets sont souvent dédiés à une application (Xpulsar, Evolution, Distributed.net) ou proposent des environnements complets multi-utilisateurs et multi-applications : XtremWeb (www.xtremweb.net), OurGrid (www.ourgrid.org), Alchemy (www.alchemi.net).

Au delà des plates-formes de calcul, d'autres systèmes distribués à grande échelle ont vu le jour, fondés sur le même principe de la collaboration d'uti-

lisateurs de PC. Les plus populaires sont les systèmes d'échange de fichiers multi-médias comme Napster, Kazaa, Edonkey, etc. Le principe de fonctionnement de ces plates-formes consiste à mettre en relation un utilisateur client avec un utilisateur serveur disposant d'une ressource recherchée par le client. Théoriquement tous les clients peuvent aussi agir comme serveur du système. Les systèmes d'échange de fichier utilisent donc des mécanismes de publication de ressources pour les serveurs, de recherche de ressources pour les clients et de transport de fichiers entre le serveur et le client. Récemment est apparue la notion d'accès et de transferts de segments de fichiers. Plusieurs serveurs collaborent alors à fournir un fichier multi-média pour un client. Les serveurs envoient des segments différents de fichiers. D'autres projets commencent à émerger, destinés au stockage à grande échelle. Il s'agit par exemple des projets OceanStore, InterMemory ou US en France. L'idée n'est plus de calculer ou d'échanger des fichiers, mais d'utiliser les capacités de stockage des PC participants pour construire une plate-forme de stockage aux capacités dépassant celles des systèmes connus. Par exemple, avec la technologie des disques actuelle, il est envisageable de construire des systèmes de stockage à grande échelle de plusieurs PetaOctets. D'autres systèmes reposant sur le même principe de collaboration d'une grande quantité de PC existent dans le domaine des études réseaux. Les projets PlanetLab (www.planet-lab.org), Porivo, CapCal et Gomez Peer coordonnent des PCs connectés sur l'Internet pour évaluer les performances de serveurs connectés sur internet ou de services en les stressant à partir des conditions réelles de l'Internet. Les plates-formes de calcul, d'échange de fichiers à partir de segments, de stockage à grande échelle et de communication réseau reposent sur deux principes fondamentaux : 1) la recherche de ressources et 2) la coordination de PCs. Ces deux principes peuvent être réalisés de manière centralisée ou répartie. Aujourd'hui le terme le plus approprié pour discuter de ces plates-formes est Desktop Grids.

Il existe encore des différences profondes entre ces différents types de Desktop Grids. Les systèmes de Calcul Global ont en commun avec les systèmes Pair à Pair d'échange de fichiers des particularités essentielles qui les différencient des systèmes distribués classiques : échelle, volatilité, connexion Internet. Cependant ils répondent à très peu des caractéristiques évoquées pour les systèmes Pair à Pair. Certes, les noeuds collaborent à un but commun, mais ce n'est que très récemment que sont apparues dans Entropia et Active-Cluster le principe de rôles multiples permettant à chaque serveur d'agir aussi comme un client. Cette caractéristique fut l'une des motivations pour la création du projet XtremWeb en 1999 [6]. Les mécanismes de mise en relation entre un client et un serveur, de transport de données entre les deux et de coordination des calculs utilisent une architecture centralisée rudimentaire. Il est bien sur tentant d'examiner la possibilité de fusionner les deux types de systèmes (Global Computing et P2P) en utilisant des mécanismes distribués. C'est ce que commence à faire le projet P3 (Personnal Power Plant) qui organise une plate-forme de calcul globale au dessus de Jxta (SUN). C'est aussi un des thèmes du workshop annuel Global and P2P Computing.

3 Le projet XtremWeb

Le projet XtremWeb étudie la conception et le développement d'un environnement de Calcul Global et Desktop Grid. C'est une plate-forme généraliste et sécurisée visant les applications du calcul intensif (entier ou flottant). Notons que cela ne signifie pas qu'une plate-forme XtremWeb est comparable directement avec les ordinateurs recensés dans le top 500 par exemple. Cette comparaison n'est pas possible puisque a) les ressources sont par essence volatiles (peuvent se déconnecter à tout instant), b) les performances de communication sont très faibles et c) le nombre de processeurs peut dans certains cas être beaucoup plus grand.

Le projet XtremWeb, comme tous les environnements de Calcul Global, doit satisfaire des contraintes sévères :
- Extensibilité : le système doit être extensible jusqu'à des centaines de milliers de noeuds avec une augmentation de performance correspondante,
- Hétérogénéité : les machines possèdent des configurations matérielles et configurations systèmes très variés,
- Dynamicité : la configuration du système varie constamment ainsi que la latence et le débit des communications,
- Disponibilité : le propriétaire d'une ressource doit pouvoir définir une politique qui limite la contribution de sa ressource, et le système de Calcul Global, doit respecter cette politique,
- Tolérance aux pannes : une défaillance du serveur de calcul ou du collecteur de résultats ne doit pas avoir d'incidence sur les machines de calcul. La perte d'une machine participante ou d'un ensemble de machines participantes est un événement intrinsèque du système,
- "Utilisabilité" : le système doit rester simple à utiliser, à déployer et à maintenir,
- Sécurité : des machines de participantes, de l'application, des données.

Le système XtremWeb, repose sur une architecture 3 tiers : les Workers qui exécutent les tâches de calcul, les Clients qui soumettent les tâches un coordinateur qui

peut être répliqué.

XtremWeb permet à des utilisateurs de lancer des tâches qui vont s'exécuter en concurrence sur les Workers et d'obtenir les résultats correspondants. Pour fonctionner au moins un Client, un Coordinateur et un Worker doivent être identifiés dans le système. Les programmes Workers et Clients sont téléchargés à partir d'un serveur sur les machines participantes. Lors du téléchargement, une liste ordonnée de Coordinateurs est transmise, ce qui permet au Client et au Worker de se connecter à un Coordinateur en essayant d'établir des connexions selon cette liste. Pour soumettre des tâches, un utilisateur commence par ouvrir une session à partir d'un Client. Toutes les tâches soumises dans une session par un Client sont identifiées relativement à cette session. Chaque tâche possède un identifiant unique dans le système, combinaison des identifiants uniques du l'utilisateur, de la session et du numéro de séquence de lancement de la tâche dans la session. Ceci permet à l'utilisateur de consulter le système depuis n'importe quel Client pour obtenir des information sur la tâche et son ou ses résultats.

Chaque tâche soumise par le Client au Coordinateur déclanche l'exécution d'une application préalablement transférée sur les workers avec les paramêtres transmis lors de la soumission de tâche. Ainsi, avant de lancer des tâches, le client peut transférer le code d'une application sur le coordinateur. Le Coordinateur stoke le code des applications dans un cache. Le Coordinateur gère l'exécution des tâches soumises par les Clients. Il les stocke dans une base de données et les converties en < *Works* > qui sont des instances de ces tâches. Lorsque le Worker contacte un Coordinateur pour demander une tâche à exécuter, il transmet au Coordinateur les noms des applications pour lesquelles il dispose déjà localement du code. Si le Coordinateur décide de lancer un Work pour lequel le code n'est pas déjà présent sur le Worker, le Coordinateur commence par transférer ce code sur le Worker. Comme les Workers et les Coordinateurs ne sont pas supposés partager un même système de fichiers, le code de l'application est communiqué au Worker par un message. Le code des applications est stocké sur le Worker dans un cache. Après réception du code, le Worker exécute le Work et envoie le résultat au Coordinateur dès que celui-ci est terminé. Pendant toute la durée de l'exécution, le Worker envoie un signal < *heart beat* > au Coordinateur. Ce signal est utilisé par le Coordinateur pour suspecter les pannes de Workers. Lorsque le Coordinateur ne recoit plus le signal pendant un temps supérieur à une borne, il considère le Worker comme inopérant et lance l'exécution d'une nouvelle instance de la même tâche (un autre Work) sur un autre Worker. Lorsque le coordinateur recoit le résultat d'un Work, il modifie alors l'état de la tâche correspondante comme étant terminée. Le Client peut alors récupérer le résultat de la tâche sur le Coordinateur. Cette étape est laissée à l'initiative du Client à qui revient la charge de collecter les résultats des tâches qu'il a lancées.

L'ensemble des protocoles est concu pour que les Workers et les Clients puissent participer au système même s'ils sont protégés par des pare feux. Toutes les communications entre les Workers (ou les Clients) et les Coordinateurs sont à l'initiative des Workers (ou des Clients) et prennent la forme d'appels de procédures à distance (XML RPC ou RMI). Comme, d'une part, généralement les pares feux autorisent les requêtes sortantes et les réponses aux requêtes sortantes et d'autre part, les Coordinateurs acceptent les requêtes extérieures, les communications entre Workers (ou Clients) et les Coordinateurs sont pratiquement toujours possibles. Cette organisation des communications (PULL) permet un fonctionnement non connecté des Workers et des Clients. D'autres approches fondées sur l'utilisation de PIPE (Jxta, ou Jaber) permettent des communications dans les deux directions (c'est à dire aussi à l'initiative des serveurs). Cependant cette organisation n'a pas été retenue car elle implique un mode de communication connecté.

La coordination de l'ensemble du système est assurée par les Coordinateurs. Les Workers renseignent les Coordinateurs sur leurs caractéristiques statiques et dynamiques : description matérielle du Worker, charge CPU, etc. Les Coordinateurs associent les requêtes des Clients et les demandes de tâches des Workers. Ce mécanisme est proche de celui de < *match-making* > de Condor. Le placement des tâches sur les Workers est effectué en deux étapes : la sélection des tâches et le placement des tâches. La sélection des tâches à partir de la liste des tâches, remplie par les requêtes des Clients, se fait en fonction des applications prioritaires. La priorité est déterminée en attribuant des proportions de tâches à accomplir par application. Les tâches sont placées selon une méthode FIFO. Lorsqu'un Worker demande une tâche, le Serveur sélectionne d'abord la tâche qui peut être exécutée par ce Worker particulier selon l'existence d'un binaire précompilé de l'application compatible avec ce Worker. Cette approche correspond au mode d'ordonnancement eager. De nouvelles politiques peuvent être définies (e.g. LIFO ou d'autres tenant compte des critères spécifiques de l'application et des ressources), et intégrées dans le coordinateur.

La tolérance aux pannes est assurée pour tous les composants du système. L'objectif est de tolérer les pannes des Workers, du Coordinateur et la mobilité des Clients. L'ensemble des composants du système enregistre localement les messages qu'il émet (journalisation optimiste des communications). Le Coordinateur est répliqué selon une stratégie passive (la réplication

est asynchrone aux communications avec les Clients et les Workers). Chaque composant du système utilise un < *suspecteur de défaillances* > associé à des bornes temporelles comme moyen de détection d'anomalies. Le suspecteur de défaillance est réalisé par un signal périodique < *heart beat* > envoyé des Clients et des Workers vers le Coordinateur et entre les Coordinateurs répliqués. Chaque composant du système possède une liste de Coordinateurs ordonnée et détermine son Coordinateur préféré comme étant le premier dans la liste avec lequel il peut communiquer. Il n'y a pas de mécanisme pour maintenir cohérente cette liste, si bien que des composants peuvent choisir des coordinateurs préférés différents. Le système fonctionne pour un Client tant qu'il existe un chemin (qui peut inclure plusieurs Coordinateurs) le reliant à au moins un Worker. La liste des coordinateurs peut être remise à jour sur tous les composants de façon périodique en consultant un site fournisseur de listes.

La sécurité est un élément déterminant dans l'acceptation d'un système de Calcul Global ou de Desktop Grid. De nombreux mécanismes doivent être mis en oeuvre pour protéger : les Workers d'utilisateurs malicieux, les utilisateurs de Worker malicieux, les applications de Coordinateurs malicieux, les données d'espions externes ou internes au système. Le premier mécanisme à mettre en oeuvre concerne l'authentification de tous les composants, des applications et des utilisateurs du système. Pour réaliser une action de bout en bout (l'exécution d'une tâche utilisateur), chacun des éléments du système mis en oeuvre dans cette action doit être en mesure d'authentifier tous les autres. Cette < *chaîne de confiance* > nécessaire sera implémentée prochainement dans XtremWeb par des certificats. Notons qu'elle n'est pas suffisante puisqu'elle ne permet pas d'éviter, par exemple, un utilisateur authentifié de corrompre un Worker en lancant une attaque à partir d'une application authentifiée. De même un ensemble d'utilisateurs peuvent emmener le système à saturation en surchargeant les Coordinateurs de requêtes (dénis de service). Pour traiter le premier type d'attaques, il est nécessaire de confiner l'exécution des applications sur les Workers (sandbox). Dans XtremWeb la sandbox repose sur le mécanisme Ptace existant sur tous les systèmes Unix. XtremWeb n'apporte aucune réponse aux attaques coordonnées. La confidentialité des données est assurée par un chiffrement optionnel des communications. Cela n'évite pas l'espionnage des données sur les Workers, plus complexe à éviter.

4 Interfaces utilisateur

Les utilisateurs potentiels des Desktop Grid de calcul souhaitent exécuter différents types d'applications parallèles. Il s'agit essentiellement d'applications construites comme des sacs de tâches, et qui peuvent avoir une coordination entre tâches de type Workflow ou Dataflow. Une exécution de type Workflow correspond à un ordonnancement impératif des tâches prévu à l'avance. Une exécution de type Dataflow ordonnance les tâches en fonction de la disponibilité de leur résultat et dépend donc des caractéristiques dynamiques de l'exécution.

Une autre demande des utilisateurs est de pouvoir exécuter une application sans avoir à modifier son code. Une tâche dans un système de Desktop Grid de calcul consiste typiquement à exécuter une application séquentielle avec un jeu de paramètres. L'accélération parallèle est obtenue par l'exécution simultanée de plusieurs tâches. Si l'application n'est pas modifiée et pour qu'elle puisse s'exécuter, cela suppose que l'on reconstruise sur la machine d'exécution l'environnement dans lequel elle est prévue pour fonctionner. Cela suppose aussi que les paramètres et les résultats sont fournis et obtenus à partir des mécanismes standards de l'application (entrée standards, fichiers d'entrée, paramètres de ligne de commande, fichiers de résultats, sortie standard, sortie d'erreur).

Fournir l'accès aux fichiers de l'utilisateur à distance permettrait de simplifier la construction de l'environnement de l'application sur la machine d'exécution. C'est typiquement l'un des objectifs des systèmes de fichiers partagés comme NFS. Un tel système est difficilement utilisable dans le contexte des Desktop Grid pour au moins trois raisons : l'extensibilité du système de fichiers, les performances d'accès aux fichiers et la sécurité qui impose de respecter les domaines d'administration. Aussi plutôt que d'accéder à distance aux fichiers utilisateurs, il est préférable de transférer les fichiers nécessaires à l'exécution de l'application sur la machine d'exécution.

Pour ces trois raisons, les interfaces utilisateurs d'XtremWeb sont fondées sur un mécanisme de clonage de l'environnement d'exécution de l'application (les fichiers utilisateurs, à partir d'un répertoire donné).

Pour soumettre une tâche, l'environnement utilisateur local d'exécution de l'application est cloné. Cela signifie concrétement que les répertoires et les fichiers locaux sont archivés dans un fichier d'archive et envoyés au coordinateur conjointement à la requète d'exécution. A la fin du calcul sur la machine d'exécution, une archive est construite avec les fichiers modifiés et les nouveaux fichiers et cette archive est transférée vers le coordinateur comme résultat de l'exécution. Le client peut

collecter cette archive sur le coordinateur, lorsqu'elle est disponible et désarchiver les fichiers qu'elle contient. Du point de vue de l'utilisateur, le résultat de l'exécution correspond donc à celui d'une exécution locale.

Il existe deux types d'interface utilisateur pour XtremWeb : une interface de type système de traitement par lots et une API de programmation Java. Quelle que soit l'interface, dans XtremWeb, les utilisateurs, les Workers, les applications et les tâches sont identifiés par un identifiant unique appelé UID.

4.1 Interface de commandes en ligne

Le tableau 1 présente les commandes principales de la première interface (type système de traitement par lots) :

```
xwhelp

xwformat [ html | csv | xml ]: specifie le format de sortie

xwapps: liste applications installées

xwaddapp <appName> <cpuType> <osType> <appBinaryFile>:
ajoute une nouvelle application

xwrmapp  <appName>: efface une application du serveur

xwworkers: obtenir la liste des workers

xwstatus [jobUID [...]]: obtenir l'état des jobs utilisateurs

xwtasks [jobUID [...]]: obtenir l'état des tasks utilisateurs

xwget | --xwresult [--xwnoextract] [--xwrmzip] [--xwerase]
[--xwoverride] [jobUID [...]]:
obtenir les resultants des jobs utilisateurs

xwremove | --xwdelete | --xwrm | --xwdel  [jobUID [...]]:
remove user jobs

xwsubmit | --xwjob <appName> [yourParameters] [ < inputFile.txt> ]:
créé une nouvelle tâche.
```

TAB. 1 – Commandes principales

Voici quelques exemples d'utilisation de ces commandes :

Détruire toutes les tâches vous appartenant :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwremove
```

Détruire la tâche numéro 100 :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwremove 100
```

Visualiser l'état des vos tâches :

```
$> java -jar XWC.jar --xwconfig aConfigFile xwstatus
```

Visualiser l'état des votre tâche numéro 100 :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwstatus 100
```

Collecter tous les résultants de vos tâches :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwget
```

Collecter tous les résultants de votre tâche numéro 100 :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwget 100
```

Soumettre une nouvelle tâche sans argument :

```
$> java -jar XWC.jar --xwconfig aConfigFile--xwjob appName
```

Soumettre une nouvelle tâche avec argument :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwjob appName youArg0 yourArg1
```

Soumettre une nouvelle tâche avec un fichier d'entrée :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwjob appName < aTextFile
```

Soumettre une nouvelle tâche avec un environnement (un ensemble de fichiers) déjà assembles dans un fichier zip :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwjob
appName -xwenv aZipFile.zip
```

Soumettre une nouvelle tâche avec un environnement comprenant le répertoire courant (et tous les sous répertoires :

```
$> java -jar XWC.jar --xwconfig aConfigFile --xwjob appName -xwenv
```

5 API de programmation en Java

La deuxième interface est une API Java. Les fonctions de cette API sont utilisables selon les droits utilisateurs. Dans XtremWeb, il existe 3 catégories d'utilisateurs : 1) NORMAL_USER : droit de consultation de tâches et d'applications et d'insertion/suppression de tâches, 2) ADVANCED_USER : les même droits précédents plus le droit d'ajout/suppression d'applications, 3) SUPER_USER : les mêmes droits précédents plus le droit d'ajout/suppression d'utilisateurs.

Avant de présenter les fonctions de l'API, voici quelques définitions utiles :

`ClientInterface`

interface qui définit un client par un login/password.

`UserInterface`

un utilisateur XtremWeb est définit par les informations suivantes : login, password, nom, prenom, email, équipe, pays plus son groupe et ses droits (opérations autorisées).

`UserGroupInterface`

description d'un groupe d'utilisateurs. Un groupe a un uid et un label.

`AppInterface`

description d'une application (uid, nom, group d'utilisateurs + d'autres informations non utilisées pour l'instant comme le temps moyen d'exécution et l'occupation mémoire).

`GroupInterface`

interface qui définit un groupe de tâches par un uid, un nom, l'identifiant de la session dans laquelle il a été créé, et l'utilisateur propriétaire.

`MobileResult`

une interface décrivant le résultat d'exécution d'une tâche. Cette interface contient le nom de l'application, la description de la tâche et le fichier zip résultat sous forme de tableau d'octets (si sa taille ne dépasse pas les 250Ko). Les fichiers dépassant les 250Ko seront transférés via une interface TCP avec la fonction boolean getResult (String uid, String fileName) où uid est l'identifiant de la tâches correspondante et fileName est le nom à utiliser pour stocker le résultat sur disque.

`SessionInterface`

interface regroupant les informations décrivant une session. Ces informations sont un uid, un nom, est l'identifiant du client (utilisateur) propriétaire.

Voici les fonctions d'administration et de configuration :

Pour créer une interface de communication avec un serveur XtremWeb :

```
FTCommRMI(ClientInterface client, XWConfigurator config, boolean
caching)
```

avec :
client : login, password de l'utilisateur
config : un fichier de config, qui contient entre autre la liste des serveurs à contacter.
caching : utilisation ou non d'un cache local pour logger les messages échangés avec le serveur.

Les utilisateurs sont gérés par des groupes avec différents droits d'administration (insertion, suppression, ...)

Pour ajouter un utilisateur XtremWeb, si le client en a le droit.

```
boolean addUser(UserInterface user)
```

Pour créer un nouveau groupe d'utilisateurs. Un groupe est définit par un identifiant unique et un label. Un utilisateur a le droit de consultation sur les tâches et applications appartenant aux utilisateurs du même groupe uniquement :

```
boolean addUserGroup(UserGroupInterface group)
```

Pour obtenir un objet contenant la description détaillée(nom, prénom, email, etc.) d'un utilisateur idéntifié par son uid (unique identifier) :

```
UserInterface getUser(String uid)
```

Pour obtenir un objet contenant la description détaillée (label) d'un groupe identifié par son uid :

```
UserGroupInterface getUserGroup(String uid)
```

Pour obtenir un vecteur contenant les uid de groupes d'utilisateurs créés sur le coordinateur XtremWeb :

```
Vector getUserGroups()
```

Pour obtenir un vecteur contenant les uid des utilisateurs créés sur le coordianteur XtremWeb :

```
Vector getUsers()
```

Pour supprimer l'utilisateur identifié par uid. Cet utilisateur n'aura plus le droit de soumettre des tâches ni de faire aucune autre opération sur le coordinateur XtremWeb :

```
boolean removeUser(String uid)
```

Pour supprimer la description d'un groupe d'utilisateurs identifié par uid (non pas les utilisateurs eux mêmes) :

```
boolean removeUserGroup(String uid)
```

Pour obtenir un objet contenant la description de l'application identifiée par un uid :

```
AppInterface getApplication(String uid)
```

Pour obtenir un vecteur d'identifiants (uid) des applications disponibles sur le coordinateur XtremWeb :

```
Vector getApplications()
```

Pour supprimer l'application identifiée par un uid :

```
boolean removeApplication(String appUid)
```

Pour insérer une application décrite par l'interface app dans la liste des applications du coordinateur :

```
boolean addApp (ClientInterface client, AppInterface app)
```

Fonctions de connexion/déconnexion : Pour établir une connexion RMI avec le coordinateur. L'identité du client et l'adresse du coordinateur(s) à contacter doivent être spécifiés préalablement lors de la construction d'une interface de communication de type FTCommRMI :

```
void connect()
```

Déconnexion de l'utilisateur :

```
boolean disconnect(ClientInterface client)
```

Voici les fonctions de manipulation de tâches. Ces fonctions implémentent les actions suivantes : soumettre une ou plusieurs tâches, récupérer l'état d'une ou plusieurs tâches, supprimer une ou plusieurs tâches, récupérer le(s) résultat(s) d'une ou plusieurs tâches.

XtremWeb permet la manipulation de goupes de tâches. Les groupes sont persistants même après la déconnexion de l'utilisateur et ne sont effacés qu'à sa demande explicite.

Pour créer un groupe de tâches décrit par son interface. Le groupe est initialement vide :

```
boolean createGroup(GroupInterface group)
```

Pour supprimer un groupe de tâches. Le groupe ainsi que toutes les tâches qu'il contient seront supprimés. Les tâches en cours d'exécution seront arrêtées avant d'être supprimées :

```
boolean deleteGroup(GroupInterface group)
```

Pour obtenir un vecteur contenant les uid des jobs appartenant à un groupe :

```
Vector getGroupJobs(String groupUID)
```

Pour obtenir un vecteur de MobileResult correspondants aux résultats disponibles de toutes les tâches d'un groupe. Pour s'assurer que les résultats de toutes les tâches sont disponibles, il faut utiliser la fonction groupStatus auparavant :

```
Vector getGroupResults(GroupInterface group)
```

Pour obtenir un vecteur de GroupInterface, correspondant aux groupes de tâches créés par l'utilisateur courant :

```
Vector getGroups()
```

Pour obtenir l'état d'un groupe qui peut avoir l'une des trois valeurs suivantes : WAINTING (toutes les tâches sont en attente d'exécution), RUNNING (au moins une tâche est terminée), COMPLETED (toutes les tâches sont terminées) :

```
int groupStatus(GroupInterface group)
```

Pour soumettre une tâche décrite par un MobileWork en l'insérant dans le groupe de tâches identifié par groupUID (dont la session est identifiée par sessionUID) :

```
String submitJob(MobileWork job, String sessionUID, String groupUID)
```

Voici les fonctions de gestion de sessions. Il faut noter que les sessions sont effacées à la déconnexion. Elles ne sont donc par persistantes.

Pour crée une session décrite par son interface :

```
boolean createSession(SessionInterface session)
```

Pour supprimer une session. Cette opération supprime la description de la session ainsi que toutes les tâches qu'elle contient (celles en cours d'exécution seront arrêtées avant d'être supprimées). Seuls les groupes (avec leurs tâches) seront préservés :

```
boolean deleteSession(SessionInterface session)
```

Pour obtenir un vecteur d'identifiants représentant toutes les tâches appartenant à une session identifiée par son uid :

```
Vector getSessionJobs(String sessionUID)
```

Pour obtenir un vecteur de MobileResult contenant les descriptions des résultats des tâches appartenant à une session identifiée par sont interface :

```
Vector getSessionResults(SessionInterface session)
```

Pour obtenir un vecteur contenant des SessionInterface correspondant aux sessions appartenant à l'utilisateur courant :

```
Vector getSessions()
```

Pour obtenir l'état d'avancement des tâches appartenant à une session décrite par son interface. Le status retourné peut avoir l'une des trois valeurs déjà décrites pour la fonction groupStatus :

```
int sessionStatus(SessionInterface session)
```

Voici les fonctions de gestion des tâches :
Pour soumettre une tâche décrite par sont interface. Cette tâche appartiendra à la session par défaut :

```
String submitJob(MobileWork job)
```

Pour supprimer une tâche décrite par son interface. Si la tâche est en cours, elle sera d'abord arrêtée avant d'être supprimée :

```
boolean deleteJob(MobileWork job)
```

Pour supprimer un ensemble de tâches décrites par un vecteur de uids (chaque uid correspond à une tâche) :

```
boolean deleteJobs(Vector jobs)
```

Pour obtenir un vecteur de uid correspondant aux identifiants de toutes les tâches appartenant à l'utilisateur courant :

```
Vector getAllJobs()
```

Pour obtenir un vecteur de MobileResult regroupant tous les résultats disponibles pour les tâches appartenant à l'utilisateur courant :

```
Vector getAllResults()
```

Pour obtenir un vecteur de MobileResult regroupant tout les résultats disponibles pour les tâches appartenant à l'utilisateur courant et dont l'uid est présent dans le vecteur d'identifiants passé en paramètre :

```
Vector getAllResults(Vector jobIDs)
```

Pour obtenir une description de la tâche dont l'identifiant est passé en paramètre :

```
MobileWork getJob(String uid)
```

Pour obtenir le résultat de la tâche identifiée par uid :

```
MobileResult getJobResult(String uid)
```

Pour obtenir l'état d'avancement de la tâche décrite par l'interface passée en paramètre. Cet état peut prendre une des trois valeurs décrites précédemment pour groupStatus et sessionStatus :

```
int jobStatus(MobileWork job)
```

6 Application d'XtremWeb au projet Auger

Le projet d'Observatoire Pierre Auger [8] constitue la première application scientifique d'XtremWeb. L'observatoire Pierre Auger est un effort international réunissant dix neuf pays, trente trois institutions, trois cents chercheurs avec un budget de $100M pour étudier les rayons cosmiques de très haute énergie, au-delà de 1019 eV. En fait, jusqu'à la détection fortuite de deux évènements au dessus de 1020 eV, la théorie ne prévoyait pas leur existence. L'origine de tels rayons cosmiques est complètement inconnue. C'est un des objectifs du projet que de déterminer si les sources de ces particules sont ponctuelles ou très dispersées dans l'univers. Le projet repose intrinsèquement sur un couplage observation-simulation. Des milliards de gerbes atmosphériques doivent aussi être simulées numériquement, en particulier par le programme AIRES (Air Showers Extended Simulation) [13].

Une exécution séquentielle de Aires dure entre 5 et 10 heures sur un PC moyen de gamme, occupe un espace disque allant jusqu'à 100 Mo, produit un résultat de 10Mo et occupe en mémoire de l'ordre de 10Mo. Cette faible occupation mémoire s'explique par la présence dans le code de AIRES d'un mécanisme de sauvegarde temporaire sur disque. Ce mécanisme sert d'une part à réduire l'occupation mémoire en fonctionnant comme un système de pagination et d'autre part à sauvegarder des contextes intermédiaires de calcul permettant une reprise éventuelle du calcul en cas d'arrêt de l'application. Le temps d'exécution d'AIRES dépend directement d'un paramètre d'affinage de la simulation. Des exécutions précises peuvent demander plus de 10 heures de calcul ou des PC plus performants.

XtremWeb-Auger (xtremweb.lal.in2p3.fr) est entré en production en Janvier 2004. Les gerbes une fois simulées sont stockées dans une base de données localisée à Lyon. Le déploiement d'XtremWeb à grande échelle (plusieurs milliers de PC) dans le cadre du projet Auger est prévu pour la fin 2004.

7 Recherches dans les systèmes de Calcul Global et Desktop Grid

Le projet d'ACI GRID CGP2P (www.lri.fr/ fci/CGP2P.html) vise à étendre les services rendus par les Workers et la capacité d'exécuter des applications parallèles.

Parmi les principaux résultats du projet figure la mise en oeuvre de MPICH-V [5], une bibliothèque de communication pour l'exécution d'applications parallèles MPI sur les systèmes de Calcul Global et les Desktop Grid. Il est intéressant de noter que la réalisation de MPICH-V à nécessité la vérification théorique des protocoles par une des équipes participant au projet CGP2P. MPICH-V est un projet très actif avec de nombreux résultats qui démontrent qu'il est possible de réaliser un environnement MPI tolérant la volatilité des noeuds de calcul, jusqu'à plus d'une faute par minute. Un autre résultat est l'environnement de programmation YML qui permet l'exécution d'applications exprimées sous la forme de graphes Dataflow ou Workflow. Ces graphes sont représentés en XML et un interpréteur ordonnance les noeuds des graphes (les tâches) sur XtremWeb. Une plate-forme a été déployée pour cette étude à Lille. Parmi les autres résultats importants, le projet US d'Amiens [14] développe une infrastructure de stockage à grande échelle qui est destinée entres autres à être intégrée dans XtremWeb. Des études poussées ont démontré que la réplication complète de fichiers est plus intéressante que la réplication de blocks de fichiers lorsque les PC d'un système de Calcul Global sont peu disponibles. D'autres travaux à Grenoble ont caractérisé la disponibilité des PC connectés à l'Internet par les réseaux ADSL. Ces travaux démontrent l'extrême volatilité des PC et leur renouvellement très fréquent. Il existe bien entendu d'autres projets qui s'intéresse à cette problématique. L'un des plus récents est le projet Japonais P3 (Personal Power Plant) développé à l'AIST de Tsukuba. Il s'agit de bâtir une Desktop Grid de calcul par-dessus les mécanismes offerts par l'environnement Jxta de Sun. Un autre projet (JuxMem [3]) de l'IRISA étudie la notion de mémoire partagée par-dessus Jxta.

8 Rapprochement avec les Grilles

Les systèmes de Calcul Global et Desktop Grid se différencient des Grilles de calcul principalement par quatre caractéristiques : 1) le nombre de ressources connecté est plusieurs ordres de grandeur plus grand (typiquement 100 000 ressources) et les ressources sont rarement parallèles (biprocesseur au maximum), 2) les ressources sont extrêmement volatiles, 3) les réseaux qui connectent les ressources sont des LAN, des Intranets et Internet et 4) les utilisateurs sont aussi en nombre très important (typiquement un utilisateur par ressource). Comparativement aux Grilles de calcul, il faut composer avec une infrastructure matérielle déjà présente et évoluant de manière non coordonnée et en fonction d'impératifs sans relation évidente avec le calcul à grande échelle (les PCs sont plutôt concus pour la bureautique et les jeux et les réseaux véhiculent les applications classiques d'Internet : mail, web, flux vidéo,

audio).

Ces caractéristiques engendrent des problèmes nouveaux pour les Grilles : 1) la sécurité doit être fondée sur des mécanismes extensibles à plusieurs centaines de milliers d'utilisateurs et de ressources et 2) la coordination des noeuds, le placement et l'ordonnancement des tâches doivent prendre en compte l'évolution à court terme (la connexion et déconnexion des ressources) et l'évolution à long terme (modification d'infrastructure) du système. Globalement, beaucoup de techniques utilisées pour les Grilles sont inapplicables au Calcul Global et Desktop Grid.

Les différences entre les deux types de systèmes sont profondes car elles ont des racines historiques. L'étude de la fusion de ces systèmes est une question ouverte. La spécification OGSA (Open Grid Services Architecture [9]) proposé par le laboratoire d'Argonne comme servant de base à la version 3 de Globus est une première tentative dans cette direction. Par exemple, le projet CGP2P examine la possibilité d'encapsuler les différentes entités logicielles d'XtremWeb (Client-Serveur-Worker) et MPICH-V (Mémoire de canal, Serveur de checkpoint) comme des Services Web de type Grid pouvant être exécutés sur des ressources banalisées. Actuellement la recherche de service, ni la sécurité, la coordination globale et la coordination d'une application parallèle, entre autre, ne correspondent pas encore aux caractéristiques des Desktop Grids. Cependant en standardisant la représentation des services et en offrant des mécanisme pour gérer les relations Client/Services et Services/Serveurs, OGSA semble une base intéressante pour l'étude d'un rapprochement.

9 Conclusion

Les systèmes de Calcul Global et de Desktop Grid représentent une formidable opportunité pour la mutualisation des ressources de calcul et des données. Le projet XtremWeb a constitué une des premières tentatives dans ce sens en construisant une plate-forme verticale complète dédiée au calcul. Cette plate-forme est déjà utilisée en production dans différents sites. L'exploitation à grande échelle débutera avec l'application au projet de physique Auger. Au-delà de la faisabilité de tels systèmes, des problèmes plus fondamentaux se posent sur le passage à l'échelle des mécanismes connus des systèmes distribués. Nous pensons que des instruments de simulation/émulation deviennent nécessaires pour rendre compte de phénomènes liés à l'échelle. Le projet d'ACI Masse de données < *Data Grid eXplorer* >, en cours, vise à apporter une réponse à ce besoin en construisant le premier grand instrument d'émulation des systèmes à grande échelle.

Remerciements Ce travail est en partie supporté par les ACI GRID CGP2P et DataGraal, l'ACI masse de sonnées Data Grid eXplorer et le RNTL Casper du Ministère de la Recherche et le Projet Pluri-Formation Augenome XtremWeb de l'Université Paris Sud.

Références

[1] A. D. Alexandrov, M. Ibel, K. E. Schauser, and C. J. Scheiman. Superweb : Towards a global web-based parallel computing infrastructure. In *Proceedings of the 11 th IEEE International Parallel Processing Symposium (IPPS)*, April 1997.

[2] D. Anderson, S. Bowyer, J. Cobb, D. Gedye, W. T. Sullivan, and D. Werthimer. A new major seti project based on project serendip data and 100,000 personal computers. In *Astronomical and Biochemical Origins and the Search for Life in the Universe, Proc. of the Fifth Intl. Conf. on Bioastronomy*, 1997.

[3] G. Antoniu, L. Bougé, and M. Jan. Juxmem : weaving together the p2p and dsm paradigms to enable grid data-sharing service. Technical Report RR-5082, INRIA Rennes, 2004.

[4] A. Baratloo, M. Karaul, Z. Kedem, and P. Wyckoff. Charlotte : Metacomputing on the web. In *Proceedings of the 9th Conference on Parallel and Distributed Computing Systems*, 1996.

[5] G. Bosilca, A. Bouteiller, F. Cappello, S. r Djilali, G. Fédak, C. Germain, T. Hérault, P. L. inier, O. Lodygensky, F. Magniette, V. Néri, and A. S. hov. Mpich-v : Toward a scalable fault tolerant mpi for volatile nodes. In *SC02*, Baltimore USA, Novembre 2002.

[6] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. néri, and O. Lodygensky. Computing on large scale distributed systems : Xtreweb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Systems, to appear*, 2004.

[7] B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. E. Schauser, and D. Wu. Javelin : Internet-based parallel computing using Java. *Concurrency : Practice and Experience*, 9(11) :1139–1160, 1997.

[8] J. Cronin and A. Watson. The pierre auger observatory. http ://www.auger.org.

[9] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid : An open grid services architecture for distributed systems integration, globus project, 2002.

[10] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over the internet-the popcorn project. In *Proceedings for the 18th International Conference on Distributed Computing Systems*, 1998.

[11] H. Pedroso, L. M. Silva, and J. G. Silva. Web-based metacomputing with jet. In *Proceedings of the ACM 1997 PPoPP Workshop on Java for Science and Engineering Computation*. ACM, June 1997.

[12] L. F. G. Sarmenta, S. Hirano, and S. A. Ward. Towards bayanihan : builing an extensible framework for volonteer computing using java. In *Proc. ACM Workshop on Java for High-Performance Network Computing*, 1998.

[13] S. J. Sciutto. Aires : Air showers extended simulation. Department of Physics of the Universidad Nacional de La Plata, Argentina, 1995. http ://www.fisica.unlp.edu.ar/auger/aires/.

[14] G. Uatrd and A. Vernois. Data durability in peer-to-peer storage systems. In *Proceedings of the fourth international workshop on Global and Peer to Peer Computing, Chicago*, 2004.