Projet RNTL e-Toile

Rapport scientifique

Réflexions et propositions pour une grille haute performance

15 décembre 2003

Pascale Vicat-Blanc Primet (éditrice)*, Philippe d'Anfray†, Christophe Blanchet‡, Laurent Bobelin§, Olivier Boudeville¶, Fabien Chanussot*, Van-Dat Cung‖, Yves Denneulin**, Abdelaziz Djerrah‖, Marc Drabik§, Yvon Jegou††, Phuong-Nam Huynh‖, Nicolas Lacorne‡, David Lacoste¶, Bertrand Le cun‖, Laurent Lefevre*, Damien Mateo†, René Metery§, Yann Meurdesoif†, Jean-Christophe Mignot*, Aline Pajot¶, Cong Duc Pham*, Rémi Revire**, Geneviève Romier‡‡, Irea Touche†

*INRIA LIP, RESO, Ecole Normale Supérieure de Lyon 46, allée d'Italie, 69007 Lyon, France
†CEA-DSI, Centre de Saclay, 91191 Gif-sur-Yvette Cedex, France
‡CNRS, IBCP, 7, passage du Vercors, 69367 Lyon Cedex 7, France
§CSSI, 200, rue Pierre Duhem, 13799 Aix en Provence-Les Milles, France
¶EDF, 1,avenue du général de Gaulle, 92141 Clamart Cedex, France
‖UVSQ CNRS, PRISM, 45,avenue des Etats Unis, 78035 Versailles, France
**INRIA, APACHE, 655, avenue de l'Europe, 38334 Saint-Ismier Cedex, France
††INRIA, PARIS, IRISA, campus de Beaulieu, 35042 Rennes Cedex, France
‡‡CNRS, UREC, 4, place Jussieu, 75252 Paris Cedex 05, France

Résumé

Un ensemble de ressources de calcul localisées dans plusieurs laboratoires académiques et industriels français ont été interconnectées au sein de la plate-forme expérimentale RNTL e-Toile dans le but d'explorer la technologie émergente des grilles. Cette infrastructure distribuée a été bâtie au-dessus du réseau expérimental très haut débit VTHD. Un ensemble de composants logiciels a été développé pour exploiter au mieux le potentiel d'un tel réseau avancé, puis orchestré au sein d'une architecture ouverte, intégrant les composants spécifiques développés dans le projet autour des éléments clés du système Globus. Plusieurs communautés d'utilisateurs ont suivi et participé aux différents développements afin d'évaluer l'apport de cette technologie vis-à-vis de leurs applications respectives. Ce rapport synthétise les principales activités scientifiques de ce projet de plate-forme expérimentale de grille haute performance et vise à donner une vision d'ensemble des choix, des travaux réalisés et des résultats obtenus dans ce projet.

Table des matières

Table des figures

Chapitre 1

Contexte et choix scientifiques

Lors de l'émergence du concept de grille, les différents acteurs s'accordaient sur ce type de constat : "*[...] la grille étendrait les paradigmes du calcul parallèle sur grappes d'ordinateurs fortement couplés vers des systèmes distribués géographiquement. Dans la pratique, la grille est plutôt utilisée comme une plateforme d'intégration d'applications faiblement couplées - chaque composant pouvant tourner de manière indépendante sur des machines parallèles à faible latence - et pour relier des ressources de calcul, de stockage et de visualisation ainsi que des instruments [...]*" [33]. Cet état de fait résulte sans aucun doute de multiples facteurs dont l'aspect performance pourrait être le principal. On voit néanmoins qu'avec le foisonnement de projets "*grid*" à travers le monde, le concept de grille s'est beaucoup étendu, faisant émerger de nouveaux usages potentiels et fédérant une communauté de plus en plus large. Certains imaginent que les technologies grille permettront à terme de créer un outil de calcul de dimension planétaire ("*world wide computer*") à l'image de ce qu'est *Internet* pour la communication ou le *WEB* pour l'accès à l'information. La grille en tant que nouvel outil informatique soulève de nouveaux verrous non seulement sur le plan du déploiement et de l'ingénierie mais aussi des verrous scientifiques relatifs à la performance, au facteur d'échelle, à la dynamique, à la robustesse, à la sécurité, à la flexibilité. Le paysage de la grille reste encore bien flou et le concept, s'il est sorti de l'enfance, n'est pas encore arrivé à maturité.

C'est dans ce contexte qu'`e-Toile` s'est proposé d'explorer cette nouvelle approche de l'informatique en faisant le pari de l'expérimenter en vraie grandeur avec une infrastructure et des applications réelles [90].

Trois aspects principaux caractérisent les grilles de calcul :

- l'infrastructure composée du nuage réseau et des ressources de calcul et de stockage ;
- le logiciel d'administration et d'exécution appelé *middleware* ou intergiciel ;
- les applications distribuées et exécutées sur la grille ou *grillifiées*.

Dans `e-Toile`, un certain nombre de choix ont été faits à ces trois niveaux afin de répondre aux trois objectifs initiaux du projet :

- construire une plate-forme grille expérimentale à l'échelle nationale et basée sur le réseau expérimental très haut débit VTHD ;
- proposer une solution logicielle en alternative aux solutions existantes ;
- évaluer l'apport de la technologie grille au niveau de ses usages.

L'infrastructure Le choix de construire e-Toile au-dessus d'un réseau très haut débit avait pour but d'étudier l'intérêt d'une technologie d'interconnexion performante et de développer des services de communication en adéquation avec le potentiel offert par le réseau. Ce choix orientait nécessairement e-Toile vers une grille de type "supercalculateur distribué". La technologie émergente des réseaux actifs a d'autre part été déployée en bordure de cette ossature haut débit pour étudier son potentiel vis-à-vis de la grille.

Les développements dans l'intergiciel Le choix de s'appuyer sur un standard de fait et libre tel que Globus [58] a d'une part, donné à la communauté de ce projet, l'expérience d'un déploiement de cet environnement logiciel à une échelle assez large et, d'autre part, a permis d'étudier comment des idées et des approches nouvelles pouvaient s'intégrer dans une architecture ouverte, robuste et facile à déployer, interopérable avec les standards émergents du GGF "*Global Grid Forum*"[57, 9].

Le rôle des applications e-Toile a souhaité dès le début intégrer les communautés scientifiques et industrielles afin qu'elles se familiarisent avec cette nouvelle approche de l'informatique pour en mesurer pleinement le potentiel et les difficultés. Il est aussi fondamental que les utilisateurs finaux soient impliqués dans la définition des objectifs et des attentes vis-à-vis de "l'outil". Il fut nécessaire et très bénéfique que quelques applications candidates participent à l'évaluation des composants de l'intergiciel et réfléchissent à l'élaboration des *benchmarks*.

La figure 1.1 présente l'architecture de la grille e-Toile et situe dans quatre niveaux d'abstraction différents, les composants qui ont été développés.

Au niveau de l'infrastructure sont positionnés les services réseau de haut niveau directement interfacés avec les services de VTHD : QoSINUS pour la gestion dynamique de la qualité de service, DyRAM pour le *multicast* fiable actif et Tamanoir pour l'environnement de nœud actif. Au second niveau, les composants du noyau e-Toile interfacés avec les composants de Globus apportent les fonctionnalités de gestion, de contrôle et d'optimisation de la plate-forme : allocation de ressources (Allocator), chargement (Loader), surveillance des ressources (Spam), stockage des informations (SIC) et interface utilisateur (Guide). Au niveau des bibliothèques de communication haute performance mises à disposition des applications, G-Madeleine offre une interface multi-protocoles étendue à la grille et permet un support efficace de MPI. MOME permet le partage de zone mémoire à l'échelle de la grille tandis que NFSp/Gxfer propose une solution de

stockage distribué et d'accès rapide aux données. Au dernier niveau, les applications hétérogènes d'optimisation combinatoire (Bob++), de dynamique moléculaire (CHARMm), de Génomique (GriPPS), de simulation numérique (Dymoka) représentent les logiciels principaux qui ont été portés sur les deux versions de la plate-forme : version Globus, puis version avec l'intergiciel e-Toile.

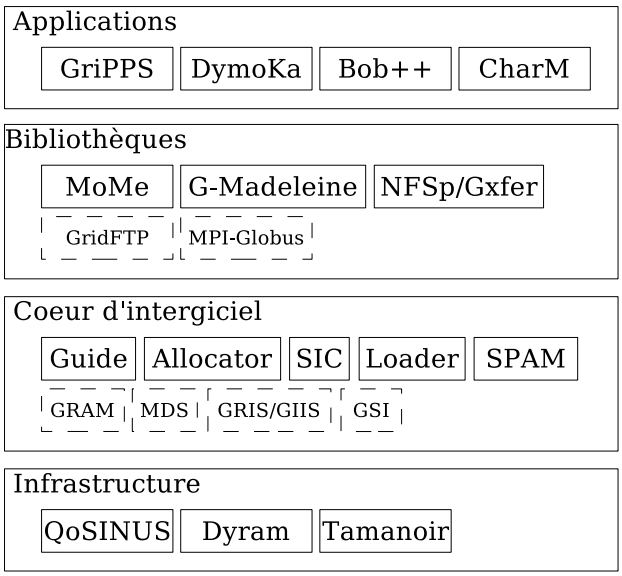

FIG. 1.1: Composants développés et exemples d'applications "grillifiées" dans e-Toile. Les composants Globus en pointillés ont été remplacés par des composants e-Toile, en particulier le MDS et les GIIS/GRIS

Ce rapport s'attache à montrer comment les trois aspects fondamentaux de la grille ont été étudiés de manière interdépendante et se sont articulés dans une architecture intégrée. Son objectif est de proposer des réponses aux trois principales questions que nous nous sommes posées :

- que peut apporter un réseau très haut débit et programmable à la grille ?
- comment intégrer des solutions nouvelles dans le cadre contraint du standard de fait Globus ?
- comment supporter, intégrer de multiples applications hétérogènes et évaluer leurs gains respectifs ?

La suite du document est structuré de la manière suivante : le chapitre 2 développe la dimension réseaux en étudiant plus particulièrement les services offerts à la grille par VTHD en terme de performance et de qualité de service ainsi que les apports de la technologie des réseaux actifs déployée en bordure de la grille. Les deux chapitres suivants examinent les aspects liés à l'intergiciel. Les bibliothèques de communication de haut niveau et les outils de transfert de données créés ou

adaptés à l'environnement e-Toile dans le but d'exploiter pleinement la ressource réseau sont présentés au chapitre 3. Le cœur de l'intergiciel e-Toile, en charge de l'allocation et de la surveillance des ressources est décrit au chapitre 4. Le chapitre 5 développe les travaux réalisés dans le domaine des applications et les nombreuses questions qui ont été ouvertes à ce niveau. Finalement les conclusions synthétisant les principaux apports de ce projet et les perspectives des nombreux travaux réalisés sont regroupées au chapitre 6.

Chapitre 2

Que peut apporter un réseau très haut débit dans une grille ?

2.1 Construction d'une plate-forme de grille en périphérie du réseau très haut débit VTHD

Une des spécificités des grilles est de s'appuyer sur une interconnexion de réseaux permettant de couvrir des distances importantes et une forte hétérogénéité inter-site. Le choix du type de réseau longue distance (WAN) sous-jacent a une incidence directe sur le type de grille visée pour couvrir des besoins et des objectifs spécifiques. Cette dimension est très souvent négligée et mal étudiée. Le type de réseau choisi a un impact scientifique, technologique mais aussi politique et financier significatif dans la conception et l'utilisation d'une grille. En effet, le réseau longue distance introduit des problématiques d'hétérogénéité, mais aussi de performance et de sécurité à de multiples niveaux. On peut de manière simplifiée, distinguer trois types de "nuages réseau" pour l'interconnexion des ressources réparties :

- *Internet*, réseau commun et très accessible de base et qui permet de construire immédiatement une grille ;
- un réseau privé virtuel (VPN) dont la vocation est de limiter et de protéger l'accès aux ressources réparties ;
- un réseau très haut débit pour obtenir des transferts performants, garants de la performance globale de l'environnement de grille.

Internet et plus particulièrement la technologie TCP/IP répond aux problèmes d'hétérogénéité et d'extensibilité, les réseaux privés virtuels à celui de la sécurité et les réseaux très haut débit à celui de la performance.

Ces trois types de réseaux sont actuellement utilisés pour l'interconnexion des ressources réparties dans le cadre des plates-formes de grille expérimentales internationales telles que EU DataGRID[55] qui s'appuie sur l'interconnexion des réseaux nationaux de la Recherche européen autour de GEANT [8], TeraGrid [24] qui est bâtie sur un réseau très haut débit (40 Gb/s) ou EU DataTAG[5] qui interconnecte les grilles européennes et américaines par un réseau expérimental à

10 Gb/s.

Des réseaux plus flexibles, proposant des services plus sophistiqués que les services IP actuels et réunissant à la fois les propriétés d'extensibilité, de sécurité et de performance sont étudiés par la communauté réseau internationale [59]. Ces réseaux, par les services avancés qu'ils proposeraient, permettraient de créer, à la demande, des environnements de calcul adaptés aux besoins de diverses communautés d'utilisateurs (organisations virtuelles). Ces services réseaux ne sont cependant pas encore réellement disponibles et déployés aujourd'hui dans un contexte multi-domaine mais devraient voir le jour dans les dix prochaines années.

Les ressources de calcul (et/ou de stockage) agrégées dans une grille peuvent être classifiées selon les mêmes critères de simplicité, de performance et de sécurité. Une grille peut regrouper :

- des ressources de *commodité* tels que les ordinateurs de bureau non utilisés pendant la nuit et le week-end ;
- des ressources privées (à accès très limité) ;
- des ressources très haute performance (typiquement des machines parallèles ou des grappes de PC).

Evidemment, l'idéal est de rechercher l'homogénéité selon ces critères entre les ressources réseau et les ressources d'extrémités : choisir un réseau privé et des ressources protégées pour assurer la sécurité ou un réseau haute performance et des ressources haute performance dans l'autre cas. Il existe des instances de ces types de grille mais aussi des instances hybrides :

- des environnements de calcul à très large échelle sans sécurité ni performance mais regroupant un très grand nombre de calculateurs. Un exemple est planetLab [22] ;
- des grilles privées virtuelles comme celles que se construisent en interne les industriels (CISCO, SUN, CEA ou EDF) ;
- des grilles très haute performance comme celle visée par TeraGrid.

e-Toile se place donc dans cette dernière catégorie des plates-formes privées haute performance. Ainsi, le réseau physique longue distance, constitué d'un seul domaine, est déconnecté de l'*Internet* et propose des débits très importants. Ce réseau est la plate-forme support de l'initiative française pour l'*Internet Nouvelle Génération* menée sous l'égide du Réseau National pour la Recherche en Télécommunications (RNRT). VTHD est géré et exploité par France-Télécom. Les ressources y sont relativement protégées. La plate-forme expérimentale e-Toile rassemble des ressources mises à disposition par les partenaires et qui sont essentiellement des grappes de PCs. Ainsi, contrairement à ce que l'on trouve dans TeraGrid qui rassemble des *clusters* à très haute performance interconnectés par des réseaux rapides, ou à ce que l'on aura dans la plate-forme expérimentale nationale Grid5000, les ressources de calcul d'e-Toile ne sont pas particulièrement performantes et nombreuses. Ce point important a donc orienté nos études plutôt sur les aspects communications puisque nous disposions d'un moyen d'intercon-

nexion assez exceptionnel. En effet, tous les sites `e-Toile` sont raccordés à `VTHD` à travers des interfaces `Gigabit Ethernet` (`IP/Gigabit Ethernet`). Les liens de raccordement ont été réalisés à partir de fibres monomode connectées entre le point d'entrée du site `e-Toile` et le routeur d'accès au dorsal `VTHD` situé sur un site France Télécom. La figure 2.1 situe les différents sites `e-Toile` au sein de `VTHD`

FIG. 2.1: Topologie d'`e-Toile` et `VTHD`

2.1.1 Déploiement d'outils de supervision

L'exploitation de la grille nécessite sa surveillance, pour cela, il est nécessaire de visualiser l'état de la grille, tant du point de vue des ressources que des performances réseau. Dans ce but, plusieurs types d'outils complémentaires ont été déployés sur la plate-forme physique. Ces outils sont indépendants de l'intergiciel utilisé sur les plates-formes logiques de la grille. Comme de nombreux outils développés dans le cadre d'autres projets (dont nous sommes parfois nous-même partenaires) sont disponibles, nous avons choisi de les réutiliser et de les adapter à la plate-forme `e-Toile`. La consultation des différents résultats et tableaux de bord qu'ils produisent se fait généralement à travers une interface *WEB* disponible aussi bien depuis l'*Internet* à travers une passerelle sécurisée que depuis le réseau

VTHD. L'accès aux informations, réservé aux membres du projet, est contrôlé au moyen des certificats personnels délivrés par l'autorité de certification du projet.

Surveillance et visualisation des ressources de la grille : Map Center Map Center [39, 38] est un outil de surveillance et de visualisation de la grille développé par les équipe de l'UREC et de l'INRIA RESO dans le cadre du projet Européen DataGrid. Cet outil est non intrusif et transparent vis-à-vis des ressources supervisées. Il n'introduit aucun effet de bord ou interférence avec l'environnement matériel et logiciel de la plate-forme ou l'administration des système locaux. Map Center offre aux utilisateurs de la grille (administrateurs, programmeurs, utilisateurs finaux), différents tableaux de bord en temps réel. L'utilisateur peut ainsi vérifier la disponibilité d'une ressource, d'un service ou l'accessibilité d'un site à travers le réseau. L'outil, diffusé en "*Open Source*" dans la suite DataGrid a été adapté à la plate-forme e-Toile. Ainsi, e-Toile est référencé sur le site *WEB* de Map Center [15] qui regroupe les grilles internationales : DataGrid, DataTag, PPDG... La figure 2.2 donne un exemple de visualisation temps-réel de l'état des ressource e-Toile avec l'outil Map Center.

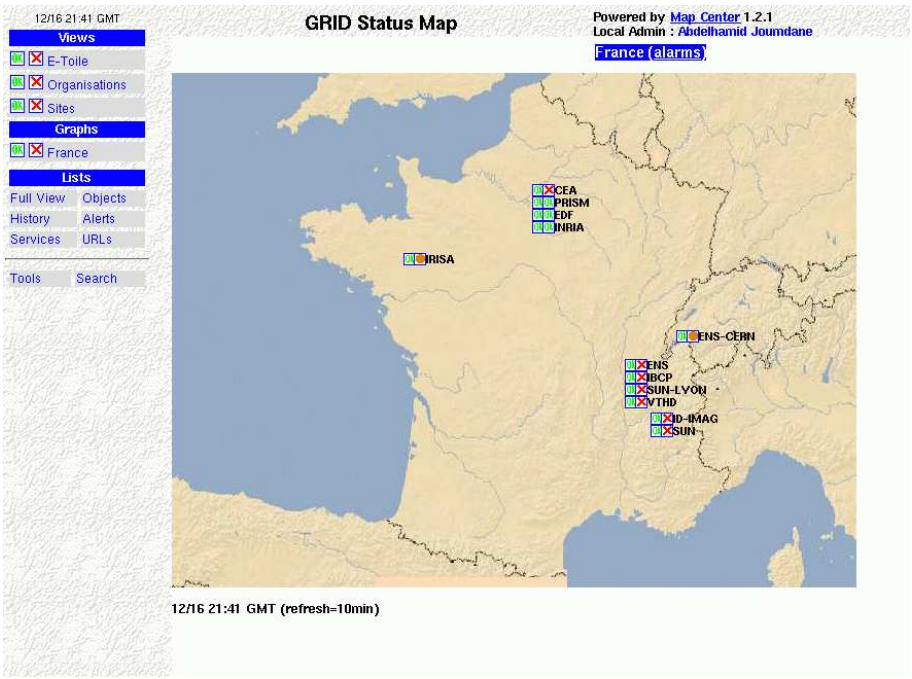

FIG. 2.2: Vue de Map Center

Plusieurs outils de mesure des performances de bout en bout développés dans le cadre du projet Européen DataGrid ont été déployés sur e-Toile. Ils permettent la collecte de données et l'élaboration de statistiques sur les performances

des liens. Ces performances, historisées dans une base de donnée spécifique, peuvent être consultées à partir d'une interface *WEB* destinée aux utilisateurs finaux. Cette base de données est aussi directement accessible par les composants de l'intergiciel, ce qui permet à l'allocateur de disposer d'informations sur les liens réseaux et les performances associées afin d'optimiser l'algorithme d'allocation de ressources et de placement de tâche.

Visualisation de la charge des ressources de calcul Dans la cadre d'`e-Toile`, nous avons choisi de déployer un outil largement diffusé, `GANGLIA`, développé à l'Université de Berkeley [7]. Cet outil est complémentaire des précédents et permet de visualiser à travers une interface web la charge des ressources de calcul de la grille, depuis un niveau global rassemblant toutes les ressources jusqu'à chaque nœud de chaque site, cela sur une période choisie par l'utilisateur. La figure 2.3 ci-dessous, donne un exemple de graphe produit par `GANGLIA`, qui permet, au moment de l'exécution de travaux sur la grille de vérifier le bon équilibrage de charge et le bon fonctionnement des nœuds.

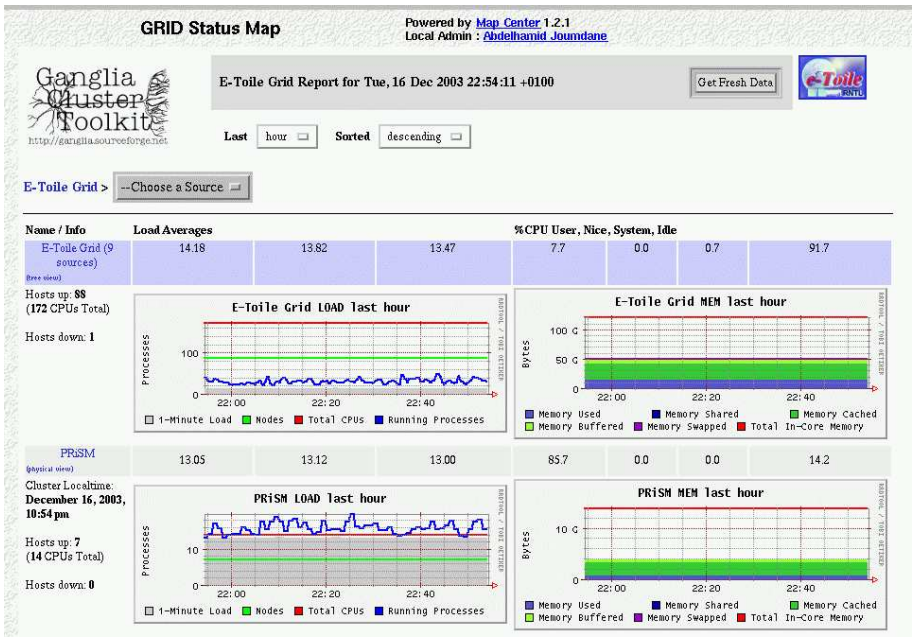

FIG. 2.3: Vue de `GANGLIA`

2.1.2 Caractéristiques et services offerts par `VTHD`

Le réseau `VTHD` (Vraiment Très Haut Débit) est un réseau expérimental, mis en place par le Réseau National pour la Recherche en Télécommunications et France

Télécom R&D. Il n'offre donc pas de garantie de disponibilité de service, mais permet, de part sa vocation de recherche et ses performances, d'étudier grandeur nature les caractéristiques d'un réseau haut débit et de ses services avancés et d'en évaluer l'intérêt dans le domaine des grilles.

VTHD offre un service de transport en IPV4 avec qualité de service différenciée, un service IPV6, un service de Réseau Privé Virtuel avec MPLS ainsi que des services Multicast IPV4. Dans e-Toile nous avons plus particulièrement étudié les services de qualité de service DiffServ et de multicast. Les services VPN et IPV6 apparaissent aussi comme des services très importants pour la grille ; ils resteront à explorer dans le futur.

Les services DiffServ dans VTHD Le protocole IP propose un unique service *Best Effort* pour l'acheminement des paquets au travers d'une interconnexion de réseaux hétérogènes ; il ne permet pas de définir et d'offrir des garanties de services ni d'associer des traitements spécifiques en fonction des flux transportés, ce qui est un besoin important des applications multimédia mais aussi du calcul réparti. Au cours des quinze dernières années, la communauté scientifique et industrielle a cherché à faire évoluer le paradigme IP à service unique vers une architecture de services plus riches et aux performances mieux contrôlées. Après l'approche IntServ [42] basée sur la réservation stricte de ressources mais qui pose des problèmes de passage à l'échelle dans l'*Internet*, l'approche DiffServ [34] qui exploite le champ "type de service" (TOS) [75] du modèle IP classique a été proposée. Différents modèles de service de cette approche DiffServ ont été standardisés. Le modèle *Expedited Forwarding* (EF)[53, 64] permet de construire un service dit *Premium* à délai et taux de pertes déterministes et offre ces garanties statistiques à des agrégats de flux. Le service *AF* [63] est destiné à assurer un débit de paquets. Cette approche de différentiation requiert théoriquement des mécanismes d'approvisionnement et de comptabilité déployés dans les équipements en bordure de domaine pour assurer ces garanties.

Le modèle de qualité de service implémenté sur VTHD est à peu près conforme à l'architecture DiffServ (*Differentiated Services*), standard de l'IETF (*Internet Engineering Task Force*). La configuration des services différenciés de VTHD est décrite dans [87]. Quatre classes de service sont définies de façon à offrir des garanties spécifiques à chacun des types de trafic transportés par le réseau :

- EF : cette classe correspond à un service de bas délais. Elle est indiquée pour les applications nécessitant des délais de propagation relativement faibles. La garantie de délai est assurée par un surdimensionnement de la bande passante attribuée à cette classe de service dans le cœur de réseau et par un contrôle d'accès aux entrées ;
- AF-TCP : il s'agit d'une classe garantissant en théorie un débit minimal. Elle est destinée aux applications telles que les transferts de fichiers, *telnet*, les applications *WEB* privilégiées. Un certain nombre de travaux [83, 70] ont

montré que le comportement de TCP au-dessus de la classe AF ne permettait pas d'offrir cette garantie. Un marquage adaptatif doit donc être réalisé pour assurer cette propriété. C'est une des adaptations que réalise le service QoSINUS que nous proposons plus loin ;
- AF-UDP : cette classe est destinée aux applications nécessitant des transferts UDP avec un débit assuré ;
- *Best Effort* : qui correspond à la classe par défaut, où aucune qualité de service n'est garantie. C'est le service par défaut de l'*Internet* actuel.

En théorie, dans l'architecture DiffServ, le marquage des paquets se fait à l'entrée d'un domaine DiffServ. Dans VTHD, le marquage des paquets peut être réalisé par l'application émettrice des paquets, par un routeur du réseau, ou par un élément placé en coupure à l'entrée du réseau.

Afin de garantir la qualité de service proposée, la bande passante est allouée statiquement de la manière suivante entre les différentes classes : à l'entrée du réseau, EF dispose de 10% de la bande passante, AF-TCP de 30%, AF-UDP de 30%, et *Best Effort* de 30%. En cœur de réseau, 30% sont réservés pour EF, 30% pour AF-TCP, 30% pour AF-UDP et *Best Effort*, les 10% restant étant dédiés aux flux de contrôle du réseau. Les files d'attentes d'AF-TCP, AF-UDP et *Best Effort* sont de plus gérées par un mécanisme de gestion active de file d'attente WRED (*Weighted Random Early Discard*). L'allocation de bande passante est réalisée par un ordonnancement de paquets de type WRR (*weighted round robin*). Il faut noter que des configurations différentes, et notamment pour la classe EF ont été étudiées et déployées dans d'autres réseaux DiffServ [81, 47].

Le service IP Multicast dans VTHD Le Multicast IPV4 est actuellement déployé au sein du réseau VTHD à la fois sur le modèle ASM (*Any Source Multicast*) et SSM (*Source Specific Multicast*). Dans le modèle ASM, les routeurs implémentent le protocole de routage Multicast PIM *Sparse-Mode*. Un point de rendez-vous (RP multicast est configuré sur le routeur NC de Rennes. Toute demande d'enregistrement d'une source ou d'abonnement d'un récepteur doit passer par le RP. Cela peut poser des problèmes si le RP doit gérer un grand nombre d'états dans sa table de routage Multicast. De plus, il y a toujours création d'un arbre partagé dont la racine est le RP avant le basculement sur l'arbre de plus court chemin. Le Multicast entre différents AS est également activé à l'aide de *peerings* MSDP (*Multicast Source Discovery Protocol*) / MBGP entre les RP de différents AS Multicast.

Dans le modèle SSM, les routeurs implémentent aussi la version SSM de PIM. Cette configuration ne nécessite plus de point de rendez-vous. On s'abonne directement à un canal (Source, Groupe) à condition de supporter IGMPv3 à l'accès. Le Multicast inter-AS ne requiert pas de protocole supplémentaire si ce n'est MBGP afin de différencier les AS Multicast des AS non multicast. L'avantage principal de ce modèle est que l'on dispose d'un nombre de canaux (S,G) illimité. Cela résout les problèmes d'adressage que l'on pourrait rencontrer

en PIM *Sparse-Mode*. De plus, la complexité est réduite puisque l'arbre de distribution pour un canal (S,G) est toujours routé à partir de la source S.

2.1.3 Etude des performances de DiffServ sur VTHD

Des tests consistant à étudier le comportement de bout en bout des classes DiffServ de VTHD ont été menés dans e-Toile. Les performances de *Best Effort*, EF et AF-TCP ont été mesurées dans cinq différentes conditions du réseau, artificiellement chargé à 25%, 50%, 75% et 100% avec un trafic UDP à 267 Mb/s, 535 Mb/s, 840 Mb/s et 1000 Mb/s respectivement. Les mesures de performance obtenues sont présentées dans les figures 2.4,2.5 et 2.6.

Ces expérimentations montrent que le trafic EF est correctement protégé du trafic concurrent des autres classes. Les performances de *Best Effort* sont pour leur part très affectées lorsque le lien est congestionné. Le débit TCP est alors divisé par 10. Ce résultat correspond au comportement attendu de TCP qui réagit courtoisement à toute congestion en diminuant son débit d'émission. La figure 2.5 montre un cas extrême où le réseau est congestionné par des flux UDP. Il est évident que dans le cas de flux TCP, l'équité du partage de bande passante s'applique et les flux TCP sont moins pénalisés. Il n'y a cependant jamais de protection de trafic contrairement à ce que l'on peut observer avec la classe EF. Enfin, les performances de la classe AF-TCP sont correctement protégées par rapport à *Best Effort*, mais impactées par EF, comme il fallait s'y attendre.

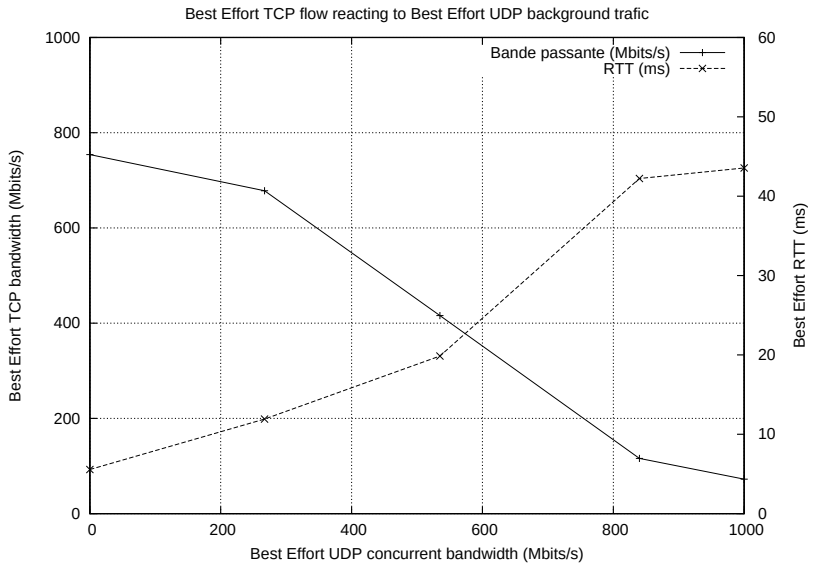

FIG. 2.4: Performances de la classe *Best Effort* dans différentes conditions de charge

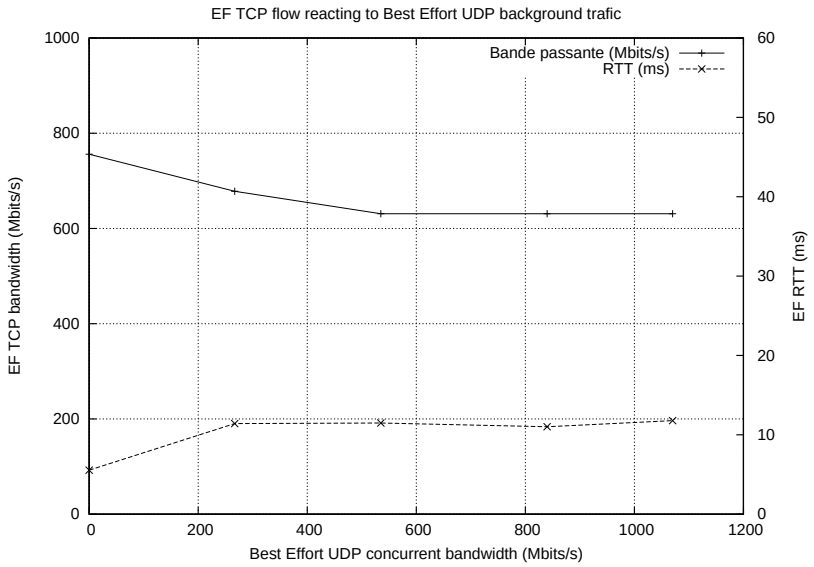

FIG. 2.5: Performances de la classe EF dans différentes conditions de charge

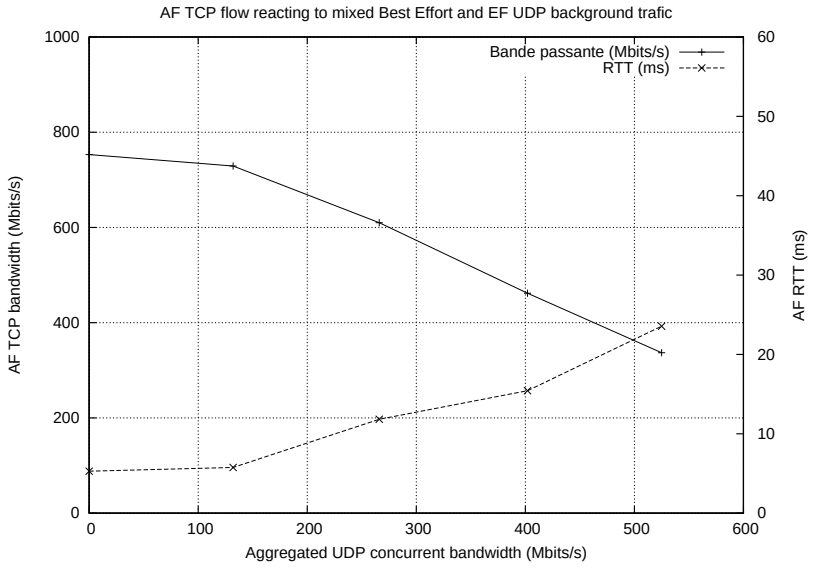

FIG. 2.6: Performances de la classe AF dans différentes conditions de charge

D'autres mesures ont été également effectuées, correspondant à des cas d'utilisation typiques d'une application de visualisation d'images médicales distribuées.

Le temps de transfert de fichiers de différentes tailles à été mesuré dans les conditions de trafic décrites précédemment, en utilisant les différentes classes disponibles dans Diffserv [80].

Des test additionnels sont planifiés. Il s'agit en particulier de comparer de manière plus exhaustive toutes les classes disponibles dans VTHD les unes par rapport aux autres et d'étudier leur comportement à un niveau moins macroscopique. Il parait en particulier intéressant de mesurer l'impact d'un changement dynamique de classe des paquets d'un flux TCP sur le débit obtenu. Le but est de caractériser ce comportement afin de concevoir les algorithmes de gestion dynamique de la qualité de service tels que ceux développés dans le service QoSINUS décrit plus loin. Par ailleurs, des expérimentations de la qualité de service avec les nouveaux protocoles de transport à l'étude tels que HighSpeed TCP ou SCTP seraient souhaitables.

2.2 Apport de la technologie réseaux programmables

2.2.1 Performances de l'approche Tamanoir sur VTHD

Les réseaux actifs constituent aujourd'hui un domaine de recherche fortement étudié, notamment aux USA, en Europe et au Japon. Ils ont pour but d'exploiter le réseau de manière plus flexible et plus intelligente. Un réseau actif, contrairement à un réseau traditionnel, n'est pas un simple support passif de transmission de paquets. Il peut être vu comme un ensemble de nœuds (routeurs) *actifs* qui réalisent des opérations personnalisées sur les flux de données qui le traversent, et qui autorisent les utilisateurs, les opérateurs, ou les fournisseurs de services à injecter leurs propres programmes dans ses nœuds, permettant ainsi de modifier, stocker (cacher) ou rediriger le flux de données à travers le réseau. Les routeurs actifs consomment plus de cycles CPU et/ou d'espace mémoire qu'un équipement réseau classique. L'équipe RESO de l'INRIA a conçu une architecture de routeurs actifs, Tamanoir, capable de fournir suffisament de ressources de calcul pour supporter des débits réseaux de plus en plus importants.

Les travaux que nous avons menés dans le cadre du projet RNTL e-Toile concernent l'étude de l'apport de la technologie des réseaux actifs dans le cadre des grilles [69, 61, 60, 40]. Différents services actifs, pour la gestion dynamique de la qualité de service et le transfert en multicast de fichiers, ont été conçus et développés. Par ailleurs, un outil permettant de gérer des nœuds Tamanoir et des services actifs déployés dans une grille de calcul longue distance a été mis au point. Nous avons également adapté Map Center [38] et élaboré des modules complémentaires à son interface *WEB*. Cette adaptation nous permet aujourd'hui d'interroger l'état de l'infrastructure active, d'un nœud Tamanoir et d'une instance de service exécutée sur un TAN au travers d'un navigateur. Une étape fondamentale de nos travaux a été de proposer une validation expérimentale de notre architecture. Pour cela nous avons mené des campagnes de mesures sur une technologie réseau haut débit (1 Gb/s) longue distance autour de l'épine dorsale VTHD.

2.2.2 Gestion dynamique de la qualité de service de bout en bout : l'approche QoSinus

En ce qui concerne la gestion de la qualité de service, l'aspect réseau actif est utilisé dans le plan contrôle, c'est à dire que les données ne sont pas modifées par les routeurs. En revanche, la qualité de service d'une session de transfert est programmée par l'utilisateur, appliquée par le nœud actif le plus proche de l'émetteur et transmise aux nœuds suivants. Les requêtes de QoS exprimées par l'application émettrice sont envoyées sous la forme d'un paquet actif à destination du récepteur. Les différents routeurs actifs traversés sur le chemin peuvent alors programmer la QoS pour le flux applicatif à venir. L'avantage de l'approche active dans ce contexte est la facilité de déploiement du service de gestion de la qualité de service. Les routeurs actifs étant situés en bordure de cœur de réseau, il peuvent intercepter les requêtes de QoS, sans que l'application ait besoin de connaître une autre machine que celle destinataire de ses flux. Cela est particulièrement intéressant dans le cas où les paquets sont routés à travers plusieurs domaines DiffServ. C'est la raison pour laquelle, le système QoSINUS, conçu et développé dans le cadre d'e-Toile a été proposé comme solution de gestion dynamique de la qualité de service dans un contexte multidomaines et en interaction avec le système de réservation de ressources GARA [56, 91] dans le cadre du projet Européen GRANDE en cours de négociation dans le programme FP6.

Programmation de la QoS De nos jours, beaucoup de routeurs utilisés au cœur des réseaux hautes performances assurent des services de QoS de type DiffServ. C'est le cas de VTHD. De leur côté, les applications ont besoin qu'un certain niveau de qualité de service soit assuré à leur flux de données. Mais il n'est pas nécessairement évident pour les applications de traduire leurs besoins en terme de classe DiffServ affectée aux paquets composant le flux. Les performances de bout en bout de ces classes ne sont pas connues des applications et risquent pour certaines de varier dans le temps. De plus, même si la sémantique de chacune de ces classes est sensée être standardisée, leur implémentation peut varier d'un réseau à l'autre. Il est donc plus intéressant pour les applications de pouvoir spécifier leurs besoins de qualité de service en terme de délai maximum, débit minimum et taux de perte maximum, c'est-à-dire dans une sémantique générale de bout en bout.

Chacune de ces caractéristiques peut être définie par un paramètre quantitatif (100 Mb/s ou 10 ms) ou qualitatif, comme "*high*, *medium* ou *low*". Une application désirant transmettre un flux supportant un taux de perte important (inférieur à 5 %), peut avoir un besoin de débit fixe et une contrainte importante au niveau du délai de transmission. Un tel besoin de qualité de service peut s'exprimer de la manière suivante : $E2E_{packetdelay}$= 150 ms , $E2E_{rate}$ = 64 Kb/s , avec $E2E_{packetdelay}$, le délai de bout en bout de transfert des paquets IP et $E2E_{rate}$ le débit effectif. Dans ce cas on ne précise pas le taux de perte qui sera optimisé au mieux. Une application interactive pourra spécifier une contrainte de débit pour

assurer le transfert des interactions en temps correct : $E2E_{rate}$ = 15 Kb/s ou $E2E_{transferdelay}$ = 500 ms, avec $E2E_{transferdelay}$ le délai de transfert d'une interaction d'un bout à l'autre du réseau.

QoSINUS permet de faire dynamiquement le lien entre une telle spécification de performance de niveau flux et le marquage des paquets qui permet leur traitement DiffServ dans les routeurs du cœur de réseau. Pour cela, un service actif est déployé aux points d'accès du domaine DiffServ. Ce service permet à une application de programmer une qualité de service sans avoir à savoir comment elle sera assurée par le réseau. Par ailleurs, ce service mesure et surveille en continu les classes de services réseau pour allouer localement au mieux et dynamiquement les requêtes des flux. Dans le cadre d'e-Toile, les clients de QoSINUS sont typiquement les applications de la grille requérant des garanties de délais comme par exemple les flux MPI ou l'allocateur pour la diffusion des barrières de synchronisation. L'intérêt de la technologie active est de permettre le déploiement d'un protocole client-passerelle personnalisé sans requérir l'implémentation d'un standard tel que rfc2205 dans les équipements terminaux et les routeurs d'accès. Des protocoles tels que XQoS développé au LAAS[1] ou RSVP [43] sur souche active peuvent aussi être déployés dynamiquement pour être testés et comparés. Le protocole choisi est simple et correspond aux besoins actuellement identifiés. Les mécanismes de déploiement dynamique de Tamanoir permettent d'envisager de l'enrichir par des mises à jour très aisées si de nouveaux besoins apparaissent.

Du point de vue d'un client, l'utilisation de QoSINUS se déroule en deux phases, comme le montre la figure 2.7.

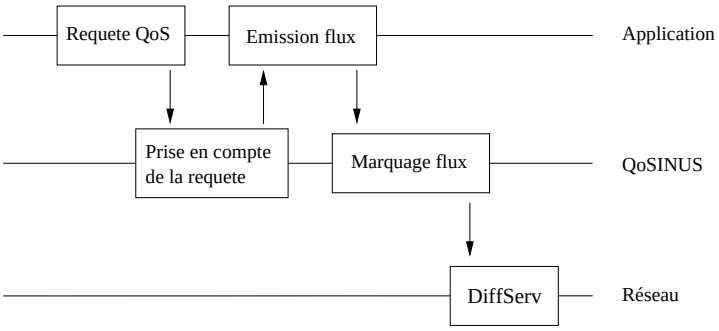

FIG. 2.7: Scénario d'utilisation de QoSINUS

Lors de la première phase, le client spécifie la qualité de service requise pour son flux de données. Le système QoSINUS analyse la requête et lui attribue un code de session. La deuxième phase correspond à l'invocation de la qualité de

[1] CNRS-LAAS Laboratoire d'Analyse et d'Architecture des Systèmes à Toulouse. http ://www.laas.fr

service. Le client transmet ses flux de données lors d'une session identifiée par le code attribué précédemment. Une fois ses données transmises, le client ferme la session.

L'API décrite ci-dessous correspond aux deux phases d'utilisation du système : négociation et invocation.

- QoSSet : cette fonction permet à une application de spécifier ces besoins de QoS en termes de délai, débit et taux de perte. Ces valeurs sont qualitatives ou quantitatives ;
- QoSInvoke : cette fonction est appelée pour indiquer à QoSINUS que le flux applicatif pour lequel une requête de QoS a été spécifiée est sur le point d'être émis ;
- QoSRelease : lorsque l'application a terminé d'émettre son flux de données, elle appelle cette fonction pour permettre au système de libèrer les ressources de QoS allouées pour le flux ;
- QoSRequest : cette fonction permet de spécifier les besoins de QoS de manière plus détaillée que QoSSet. La requête est exprimée dans un document XML qui peut contenir les informations suivantes :
  - l'identifiant du flux (source, destination, protocole utilisé, DSCP) ; la conformité, qui décrit le débit que l'application ne doit pas dépasser ;
  - le traitement des paquets non conformes (suppression, remarquage, *shaping*) ;
  - les garanties requises, en terme de délai, débit et taux de perte ;
  - le planning d'émission du flux, définissant quand et pour combien de temps le flux va être émis.

Structure du logiciel QoSinus Le logiciel QoSinus se compose de deux parties : une API cliente utilisée par les applications pour spécifier les qualités de service à assurer pour leurs flux et un service actif exécuté au niveau des routeurs actifs Tamanoir pour la gestion dynamique des classes de services et l'application de la qualité de service de niveau réseau. Le service actif QoSINUS s'exécute dans l'environnement actif Tamanoir. Ses fonctionnalités sont les suivantes :

- réception et analyse des requête d'application ;
- contrôle d'admission des flux d'application en entrée de cœur de réseau ;
- monitoring des classes DiffServ dans le cœur de réseau ;
- marquage dynamique des paquets dans la classe la plus adaptée.

Plusieurs algorithmes de conversion (*mapping*) entre la requête application et le marquage DiffServ des paquets ont été développés dans le cadre d'e-Toile. En particulier, une conversion statique a été tout d'abord conçu pour valider l'architecture globale du composant. Dans ce cas, la bande passante disponible au point d'entrée du réseau est répartie statiquement entre les classes de service. La traduction entre requête QoS et classe DSCP est définie par des règles statiques simples comme "*une requête de bas délai implique un marquage dans la classe Premium*". A la réception d'une requête, une classe est déterminée. S'il reste assez de bande

passante pour cette classe, la requête est acceptée et la bande passante disponible mise à jour ; sinon, la requête est rejetée.

Le *mapping* dynamique adopte une autre approche. Ici, les performances des classes Diffserv sont mesurées en continu. Le système choisit alors périodiquement la meilleure classe pour chaque flux en fonction de cette information. Il peut ainsi s'adapter aux variations de performances des classes Diffserv du réseau. Un autre algorithme destiné aux flux TCP a été proposé. Dans ce cas, l'application spécifie une quantité de données à transmettre et un planning d'émission (i.e. une date de début et une date de fin d'émission). QoSINUS filtre alors les messages d'acquittement de TCP pour évaluer la quantité de données déjà transmise par un flux. Il peut ainsi comparer les données transmises au planning requis par l'application et ordonner les flux en privilégiant les flux les plus en retard. En marquant les paquets des flux les plus en retard dans la classe de service la plus performante, il permet à ces flux de "*rattraper leur retard*".

L'idée consiste ici à, par exemple, diminuer temporairement la priorité d'un transfert massif de donnés en cours de transmission, pour permettre à un flux plus court, moins massif et plus interactif de recevoir une qualité de service correcte. Le transfert massif ayant par la suite le temps de "*refaire son retard*". QoSINUS cherche à optimiser à la fois des performances des flux individuels mais aussi l'utilisation globale des classes de service offertes au point d'accès. Une première expérience a été conduite au sujet de cette approche. Elle a permis de valider l'utilisation des ACK TCP pour évaluer l'état d'avancement d'un transfert TCP. Cet algorithme intègre un ordonnancement de type EDF "*Earliest deadline first*" entre plusieurs flux. Cet algorithme est en cours d'évaluation.

L'architecture du service QoSINUS est décrite de manière plus complète dans [78]. Dans [80] les objectifs et les résultats scientifiques de ce service avancé sont détaillés. Le service QoSinus est opérationnel et déployé dans la plate-forme. Nous avons montré qu'il correspondait à un besoin de maîtrise des délais de transfert d'un bout à l'autre d'un domaine. Il a été démontré lors des journées RNTL 2003 et à la conférence IEEE Cluster 2003 [79]. Une plus large utilisation dans le cadre d'e-Toile avec les composants de l'intergiciel et les applications est requise pour valider l'ensemble des fonctionnalités et des algorithmes proposés.

2.2.3 Transfert multicast fiable actif

Un service de communication point à multipoint offre un moyen efficace de diffuser des unités de données à un groupe de récepteurs, en ce sens qu'une seule copie de chacune de ces unités est envoyée par la source (s'il n'y pas de pertes). Le multicast IP (RFC 1112) fournit au niveau réseau un support efficace pour la diffusion non fiabilisée des paquets pour un grand nombre d'applications. Le problème de la fiabilité en point à point est bien maîtrisé et des solutions satisfaisantes (du moins pour TCP) ont été déployées. En revanche, l'assurance de la fiabilité dans le contexte du multicast est un problème plus ardu.

Les services actifs pour le multicast fiable qui ont été proposés et évalués dans la communauté de recherche jusqu'à présent sont le cache des paquets, l'agrégation des messages de contrôle et le multicast partiel des paquets retransmis. Nous avons proposé et évalué 4 nouveaux services actifs plus légers qui sont :

- l'élection dynamique d'un retransmetteur : un routeur actif d'assistance peut élire un récepteur pour retransmettre un paquet perdu par un de ces voisins ;
- la détection rapide des pertes dans les routeurs : un routeur actif d'assistance peut générer un NACK vers la source s'il détecte une perte de séquence dans le flux des paquets ;
- l'aggrégation des RTTs : les routeurs actifs d'assistance participent à l'aggrégation des RTTs par segment afin de générer une valeur de RTT plus précise permettant une régulation du débit (contrôle de congestion) plus fluide par la source ;
- le partionnement des récepteurs en sous-groupes pour améliorer la gestion des groupes hétérogènes.

La proposition DyRAM Nous avons intégré dans un protocole appelé DyRAM (*Dynamic Replier Active reliable Multicast*), les nouveaux services que nous avons proposés, ainsi que les services d'agrégation globale et de *subcast* [74].

Un prototype de DyRAM a été initialement développé [73] et affiche des résultats très encourageants car ils montrent que des services actifs légers peuvent être implémentés très efficacement et que leurs coûts d'exécution est faible dans les routeurs. Par exemple, le temps de traversée d'un paquet de données dans un routeur actif est de l'ordre de $20\mu s$, celui de traitement des NACK et des paquets de retransmission est compris entre $120\mu s$ et $135\mu s$. Ces temps mesurés sur une plateforme à base de Pentium Pro 200MHz sont perfectibles avec des processeurs plus rapides. L'élection dynamique d'un retransmetteur prend un temps variable qui dépend d'une part du nombre de récepteurs sous le routeur actif, et d'autre part du nombre d'itérations nécessaires pour trouver le bon retransmetteur. Ce temps est cependant très faible, de l'ordre d'une centaine de μs pour 25 récepteurs sous le même routeur d'assistance. Nous implémentons un support similaire à ftp pour utiliser DyRAM à des fins de distribution de code et de données pour le calcul scientifique sur une grille de calcul. Ces travaux ont donné lieu à des séminaires chez SUN Labs et des démonstrations à IPDPS 2003 (stand ACI GRID). Des démonstrations dans le cadre du projet RNTL e-Toile ont également été effectuées et ont prouvé la faisabilité du concept de grille active précedemment énoncé.

Les principaux problèmes ont été rencontrés au déploiement du multicast sur le réseau VTHD. Nous sommes en contact permanent avec les opérateurs de VTHD et avec les différents partenaires pour activer et configurer le support de IP multicast dans les routeurs de VTHD et ceux des sites e-Toile.

Chapitre 3

Bibliothèques et outils de communication performants

3.1 Système et transferts de fichiers distribués : NFSp et GXFER

Le stockage, la manipulation et le transfert de données sont un des problèmes actuels sur les architectures parallèles et un de ceux au cœur de la problématique des grilles. Par exemple, un des projets de grille les plus ambitieux, Datagrid, est tout entier tourné vers cet objectif : pouvoir manipuler de grandes quantités de données tout en gardant une bonne efficacité, proche des capacités du matériel et des capacités de l'interconnexion. Nous présentons dans cette partie la solution de stockage et de transfert qui a été étudiée au sein d'e-Toile : le couple NFSp/Gxfer.

NFSp Une des architectures de grille les plus courantes, du fait de son excellent rapport performance/coût, est formée de grappes de processeurs interconnectées. Il devient alors intéressant d'utiliser cette architecture non seulement pour satisfaire des besoins de calcul mais aussi de stockage, c'est-à-dire exploiter l'espace disque disponible sur les disques des nœuds de la grappe. Pour ce faire en conservant une bonne qualité de service et simplifier l'administration il est important d'être peu intrusif vis-à-vis de la configuration des nœuds clients. Pour cela nous proposons une extension distribuée du système de fichiers distribués NFS que nous avons baptisée NFSp.

Du point de vue client, il s'utilise comme un serveur NFS traditionnel, avec un montage distant d'un système de fichiers exporté par le serveur, mais, du côté serveur, le stockage des données s'effectue sur un ensemble de nœuds au lieu d'un seul. Chaque client peut aussi être utilisé pour le stockage. Le point de montage ne gère que les méta-données (nom, propriétaire, droits d'accès, taille, liste des *inodes* de stockage, etc) des fichiers alors que le contenu de ces derniers est stocké

sur des nœuds séparés. Chaque requête de lecture ou d'écriture est reçue par le méta-serveur, le point de montage chargé de la gestion des méta-données, qui la transmet au nœud de stockage concerné. Ce dernier répond ensuite directement au client. Deux implantations existent, une fonctionnant en mode noyau et une en mode utilisateur, les performances en lecture sont bonnes, par exemple 150 Mos/s avec 16 serveurs interconnectés par un réseau 100 Mb/s. En écriture la limite dépend de la connectivité du nœud qui héberge le méta-serveur qui représente un goulot d'étranglement de l'architecture.

Gxfer Le transfert efficace de fichiers sur grille, entre des nœuds d'une grille, est un problème important : en effet, si le temps de transfert des données vers le(s) site(s) sur lequel elles seront analysées ou traitées est important par rapport au temps de traitement effectif, la solution grille perd une bonne partie de son intérêt par rapport à l'utilisation d'une architecture de type grappe de processeurs. Ce transfert doit tenter de tirer partie de la totalité des capacités réseau de la grille, et donc, typiquement, d'exploiter des liens de l'ordre du Gb/sec et jusqu'à 10 Gb/sec. Pour obtenir de tels débits de communication une solution consiste à disposer de serveurs de stockage dédiés, typiquement des SANs (*Stockage Area Network*), à même de pouvoir utiliser de tels débits. L'inconvénient de ce type de matériel est son coût qui peut être prohibitif pour certains participants de la plate-forme de grille.

Une autre solution consiste à faire un ensemble de communications point-à-point pour transférer un ensemble de données, typiquement un fichier. C'est cette approche qui est utilisé dans le cas de gridftp mais elle oblige à répliquer le stockage des données à transférer sur l'ensemble des nœuds devant participer au transfert ou à en répartir "à la main" des tranches qui serviront au transfert parallèle. C'est en étudiant ce type de solutions que nous est venue l'idée d'utiliser la distribution du stockage, qui est au cœur de l'infrastructure NFSp, pour réaliser des transferts parallèles. Cependant, conscients qu'il est difficile d'imposer l'utilisation de notre solution de stockage à l'ensemble des partenaires d'une grille, nous avons décidé de concevoir une solution de transfert générique pouvant exploiter des serveurs de stockage quelconques, type SAN, systèmes de fichiers locaux, NFS ou NFSp.

Le composant chargé de réaliser les transferts s'appelle Gxfer, pour *Grid transfert*, son rôle est de copier des fichiers entre deux sites distants. Si un des sites dispose d'une solution de stockage distribué alors le transfert pourra se faire en utilisant plusieurs flux, si ce n'est pas le cas il fonctionnera comme un "simple" transfert point à point. Dans le cas d'un transfert parallèle Gxfer se chargera de gérer la synchronisation entre les différentes machines communiquant et de regrouper les informations éparses afin de reconstruire le fichier chez le destinataire de la communication. L'utilisation de Gxfer se fait de la même manière que pour une opération de copie locale, simplement en invoquant la commande gxcp plutôt que cp. La syntaxe est identique, un fichier distant se désignant sous la forme

site_distant :nom_du_fichier.

Les performances de Gxfer sont très intéressantes, nous avons transféré un fichier de 1 Go en moins de 10 secondes (9.6 sec) entre les sites de Grenoble et de Lyon reliés par un lien à 1 Gb/s, les deux serveurs de stockage concernés étaient des PC sous Linux, les moins performants étant ceux de Grenoble des Pentium 733Mhz avec 256 Mo de mémoire et reliés par un réseau 100 Mb/s. Les deux sites utilisaient NFSp comme serveur de stockage. Les essais supplémentaires que nous avons pu mener ont montré un bon passage à l'échelle.

En conclusion nous avons pu valider grâce au projet e-Toile que NFSp est un composant de stockage robuste et avec lequel il est possible de mettre en place une solution de transfert efficace sur grille basée sur des communications parallèles entre les nœuds de stockage. Les performances sont au rendez-vous ainsi que la robustesse.

3.2 Bibliothèque de communication G-Madeleine et Interface MPI performante

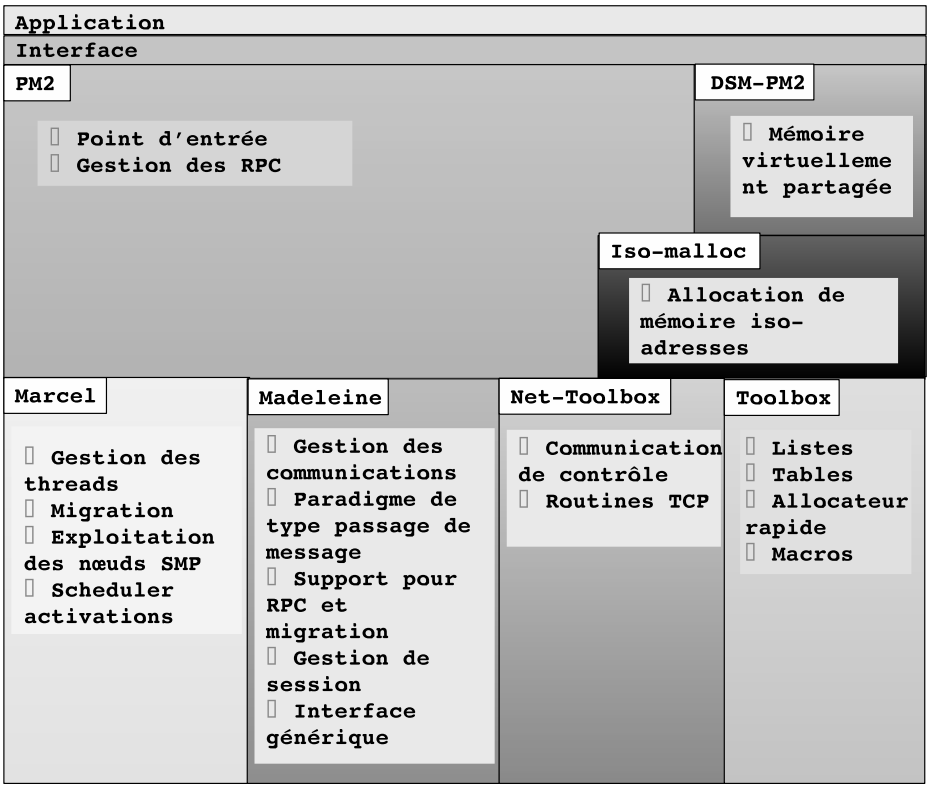

FIG. 3.1: Architecture de Madeleine

Dans le cadre du projet e-Toile, Madeleine la bibliothèque de communication développée par l'équipe de Raymond Namyst[1] à été portée sur Globus. Les développeurs disposent de l'interface native Madeleine ainsi que de MPI au-dessus de Madeleine. Ce portage fournit aux applications une API de communication dont le modèle est complètement connecté sans avoir besoin d'établir les communications entre les nœuds et sans connaitre le réseau de communication sous-jacent (*cluster*, TCP, grille). Madeleine fournit de plus un maximum de flexibilité pour les différents modes de *bufferisation* des données. Plus particulièrement dans le cadre d'e-Toile, la transparence de Madeleine a été généralisée à un deploiement multi-site et l'authentification et la sécurité d'e-Toile ont été introduites. La figure 3.1 décrit l'architecture de ce composant.

3.3 Mémoires virtuelles distribuées sur la grille : Mome

Une simulation numérique complexe peut entraîner l'exécution de plusieurs programmes, chacun de ces programmes étant en charge d'une partie des traitements. L'échange de données entre les programmes d'une simulation se fait en général par des fichiers. La présence de ces fichiers pose un certain nombre de problèmes lorsqu'il s'agit de déployer les programmes de simulation sur une grille informatique : les données doivent être présentes sur le site sélectionné par le gestionnaire de la grille pour l'exécution. Idéalement, il faudrait pouvoir déclencher automatiquement le transfert direct des fichiers du site où ils ont été produits vers le ou les sites de consommation pour éviter le passage par un serveur centralisé.

Il est parfois difficile de décider d'avance des transferts de données à effectuer, par exemple, lorsque le choix des tâches à exécuter dépend du résultat de l'exécution de tâches précédentes.

Une deuxième difficulté provient de l'utilisation de grappes de calculateurs pour l'exécution de codes parallèles de simulation : chaque nœud de la grappe est chargé d'une partie des calculs et, en général, n'accède qu'à une partie des données en entrée et produit une partie des résultats. L'utilisation d'un système de gestion de fichiers classique sur les grappes constitue un goulot d'étranglement. Il faut rassembler les contributions de chaque nœud pour former le fichier résultat puis diffuser la totalité de ce fichier sur les nœuds clients.

L'utilisation d'une technologie de type "mémoire partagée répartie" pour la mise en œuvre d'un répertoire de données a plusieurs objectifs :

- transférer directement les données des nœuds producteurs des grappes vers les nœuds consommateurs ;
- éviter le passage par une gestion centralisée des données ;
- offrir aux programmes un espace de nommage uniforme des fichier.

Mome est une mémoire partagée répartie logicielle développée dans le projet Paris de L'IRISA. Les fonctionnalités de ce logiciel ont été étendues au cours du

[1] maintenant à l'université de Bordeaux

projet e-Toile afin de constituer un répertoire de données distribué sur la grille. Sous cette forme, une instance de répertoire peut-être initialisée par un utilisateur sur l'ensemble des nœuds de la grille. Cette initialisation entraîne le lancement d'un processus de type *daemon* sur chacun des nœuds des grappes. Lorsqu'un code de simulation parallèle est déployé sur une grappe, chacun des processus locaux se connecte au *daemon* local. Après connexion, chaque processus peut ouvrir, lire, écrire, détruire ou créer des segments dans la mémoire répartie Mome. L'accès aux données d'un segment se fait par projection de ce segment sur l'espace d'adressage du processus. Les modifications par un processus quelconque sur un élément d'un segment sont automatiquement propagées aux autres processus suivant le modèle de cohérence en cours.

Le répertoire partagé réparti Mome est persistant : les processus *daemon* assurent la conservation de l'ensemble des données des segments même lorsqu'aucune application n'est connectée. La mise en œuvre actuelle supporte également la défaillance des applications : les processus *daemon* conservent toujours la dernière copie de chaque page de la mémoire virtuelle. Pour son fonctionnement interne la mémoire partagée répartie Mome ne fait jamais référence aux adresses de projection des pages dans les mémoires des processus connectés, ce qui permet le partage de données par des processus hétérogènes. Il est de ce fait possible de déployer des applications "couplées par la mémoire" sur la grille.

Ce premier prototype de répertoire de données distribué déployé au cours du projet e-Toile a mis au clair certaines limitations de la mise en œuvre de la mémoire partagée Mome : cette mise en œuvre considère que tous les nœuds de calcul sont au même niveau et ne tient pas compte des caractéristiques des réseaux d'interconnexion présents. Une nouvelle mise en œuvre plus adaptée à la hiérarchie formée par l'interconnexion des grappes a été étudiée et devrait être disponible dans l'avenir.

Chapitre 4

Comment intégrer des solutions nouvelles dans le cadre contraint de Globus ?

4.1 Réflexions sur une architecture ouverte et extensible

Dans le but de mener à bien une réflexion sur l'architecture de base d'un intergiciel de grille de calcul haute-performance, un comité d'architecture a été constitué, dès le démarrage du projet. Pour ce faire le comité a créé des groupes de travail :

- allocation/*monitoring* ;
- communication/mémoire partagée ;
- *core middleware*/système d'information.

qui se sont réunis pour débattre sur l'état de l'art dans leurs différents domaines, pour analyser les plus values apportées au projet par leurs "briques logicielles" et pour choisir un intergiciel de base. C'est ainsi qu'il a été décidé d'utiliser, comme intergiciel de base, la boîte à outils Globus. La version 2.2 a été déployée comme intergiciel de base pour la version 0 de l'intergiciel e-Toile. La version 0 de l'intergiciel e-Toile devait permettre de tester l'infrastructure de la grille (grappes et réseau VTHD) et permettre aux applications de se familiariser avec les concepts implémentés dans une grille de calcul et de données.

4.1.1 Architecture de l'intergiciel e-Toile

De manière à consolider les réflexions des groupes de travail et à figer l'architecture de l'intergiciel e-Toile, un séminaire de 3 jours, découpé en sessions plénières et en ateliers, a été mis en place. La production de ce séminaire a permis de dégager l'architecture innovante de l'intergiciel et de spécifier les grandes fonctionnalités des "briques logicielles" identifiées comme indispensable à une grille de calcul haute-performance. La conception de cette architecture a été faite, en ayant toujours à l'esprit, le remplacement des briques Globus par des composants inno-

vants, robustes et extensibles. C'est ainsi qu'a été construit le cœur de l'intergiciel, autour d'un Système d'Information et de Contrôle :

- une interface utilisateur conviviale dotée d'un langage de description de travaux extensible ;
- un allocateur de ressources sans état ;
- un système de surveillance et de mesure intégré ; les briques Communication, Mémoire Partagée, Parallélisme, apportant quant à elles, en complément des fonctionnalités assurées par le noyau, les aspects haute-performance de la grille expérimentale `e-Toile`.

Les deux seuls outils de `Globus` utilisés dans la version actuelle d'`e-Toile` sont le `GSI` (module d'authentification des utilisateurs et des ressources par certificats) et le `GRAM` (*Globus Resource Allocation Manager*).

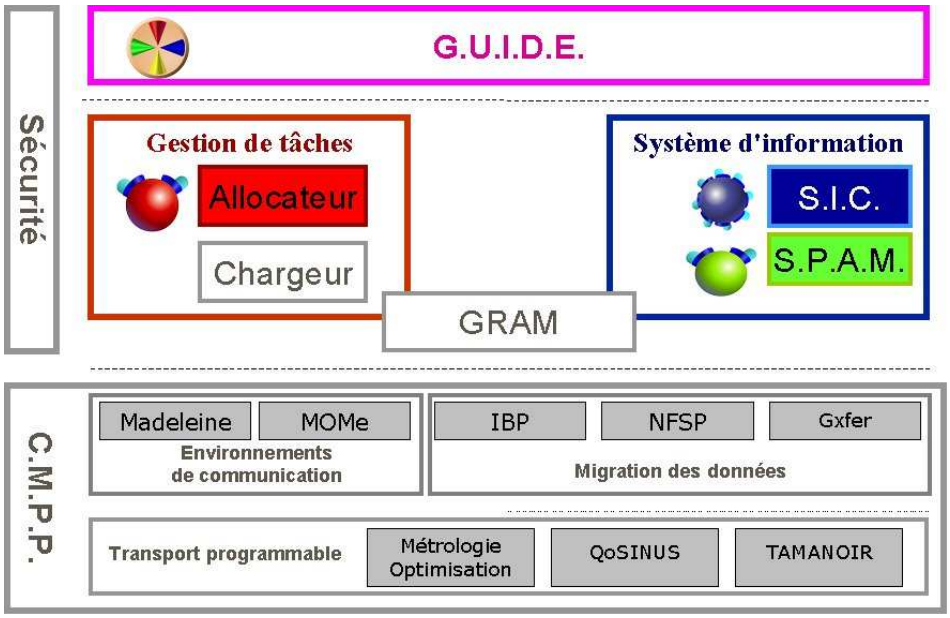

FIG. 4.1: Architecture de l'intergiciel `e-Toile`

4.2 Un allocateur sans état

Le problème de l'allocation d'une application sur une grille de calcul est complexe. Il s'agit de répondre au mieux aux attentes des utilisateurs quant au choix des machines sans qu'il soit nécessaire de donner nominativement les machines. L'utilisateur devrait pouvoir spécifier ses vœux par des règles présentant des abstractions comme le choix sur l' architecture, le type de système d'exploitation, la capacité mémoire. Certaines de ces règles ne peuvent être violées, c'est ce que nous appelons des contraintes fortes du problème de placement. D'autres, en revanche,

sont plutôt des vœux formulés par l'utilisateur, elles ont la même forme que les contraintes fortes mais l'allocateur peut se permettre de les violer, si l'état de la grille ne permet pas d'y répondre

Dans le cadre d'une grille de calcul, une partie du problème d'ordre beaucoup plus technique et/ou administratif, est de trouver une solution logicielle à la coopération des différents gestionnaires de travaux se trouvant sur chacune des machines parallèles composant la grille, sans pour autant imposer un gestionnaire spécifique sur toutes les machines. Les machines intégrées dans une grille appartiennent à des entités administratives souvent différentes. La machine doit donc disposer d'une certaine autonomie vis-à-vis de la grille. Elle doit pouvoir être utilisée par l'entité propriétaire, ignorant son appartenance à la grille [86, 52]. Contrairement à une solution où le système a l'autorité complète sur tous les systèmes qu'il utilise, dans le cadre d'une grille, il faut trouver une solution permettant de faire coopérer des outils qui ne sont pas prévus pour le faire au départ.

Le paragraphe suivant 4.2.1 discute des différentes formes que peut prendre une application. En effet en fonction du modèle d'exécution de l'application, le problème ne sera pas identique, et donc sa résolution sera différente. Au paragraphe 4.2.2, nous discutons plus précisément du problème de l'allocation mais d'un point de vue modélisation et optimisation. Dans le paragraphe 4.2.3 nous présentons les solutions techniques que nous avons voulu mettre en place pour répondre à ce problème.

4.2.1 Les modèles d'applications

Lorsqu'une application vise à exécuter plusieurs processus, il est obligatoire de dissocier les modèles dans lesquelles les processus d'une application sont lancés de manière synchrone de ceux où les processus sont lancés de manière asynchrone. Comme spécifié par le langage de description de travaux (`LDT`), et en accord avec les partenaires applicatifs d'`e-Toile`, nous avons spécifié trois formes principales d'applications.

Le modèle `SPMD` C'est le modèle habituel de programmation des machines parallèles classiques. P processus issus du même binaire sont lancés sur p processeurs. Les P processus de l'application doivent donc démarrer au même moment. C'est une contrainte difficile à prendre en compte vis-à-vis de l'optimalité de l'allocation. Encore une fois, les différentes applications peuvent avoir des priorités, des temps de calcul donnés ou estimés inconnus. Beaucoup de solutions ont été apportées, lorsque l'ensemble des processeurs est homogène, comme par exemple sur une grappe, puis sur une plate-forme hétérogène. Parmi les solutions citées plus haut, seuls `Codine`, `CONNECT :Queue`, `DJM`, `DQS`, `LSF`, `NC Toolset`, `PBS` (voir [65]) gèrent hétérogénéité des machines.

Mais il faut bien comprendre que cette contrainte de synchronicité dans le lancement de processus alliée au problème d'utilisation exclusive d'un processeur par un processus est un problème majeur dans l'allocation de ressources. Par des exemples simples, il est facile de montrer que ces deux contraintes peuvent conduire à une sous utilisation de la machine.

Le modèle de processus en cascade Parmi les applications d'e-Toile certaines consistent à lancer plusieurs applications sur des données différentes ou avec des paramètres différents. Pour éviter à l'utilisateur d'effectuer plusieurs requêtes de même nature, nous avons préféré donner la possibilité de n'effectuer qu'une requête pour n lancements asynchrones. Le coté "facile" de ce modèle est que les différents travaux composant l'application peuvent être ordonnancés indifféremment. Le côté "difficile" est la nécessité de prendre en compte le fait que tous ces travaux "indépendants" font partie d'une même application, et que l'ordonnanceur essaie d'optimiser le temps d'attente entre la date de soumission et la date de lancement du dernier des travaux.

Coopération de travaux SPMD Plus généralement, de nombreuses applications sont construites à partir d'applications plus petites communiquant les unes avec les autres. Ces petites applications peuvent être de simples processus séquentiels, mais elles peuvent aussi être des applications SPMD. Ce modèle permet donc de lancer de manière synchrone plusieurs applications qui ont des binaires différents.

Beaucoup d'autres modèles d'application auraient pu être considérés, mais nous nous sommes volontairement limités à ces derniers, car ils représentaient l'ensemble couvrant toutes les applications proposées par les partenaires dans le cadre d'e-Toile.

4.2.2 Problèmes d'allocation dans les grilles

Positionnement du problème Le problème consistant à allouer des tâches à réaliser à des unités de traitement s'appelle dans sa grande généralité ordonnancement ou *scheduling*. La forme classique d'un problème d'ordonnancement consiste à décider des dates de début d'exécution sur plusieurs machines d'un ensemble de tâches bien connues à l'avance (temps d'exécution, date de disponibilité, date à laquelle la tâche doit être terminée, etc...). Cette forme est dite *offline*, car le processus de décision de l'ordonnancement est fait *a priori*. Les tâches sont assignées statiquement sur les processeurs.

A l'inverse, lorsque la décision de placement ou d'ordonnancement est faite à la volée, le problème est dit *online* [84, 71]. C'est par exemple le cas lorsque les tâches ne sont pas connues à l'avance mais découvertes au fur et à mesure de leur arrivée. Beaucoup d'autres raisons peuvent aussi conduire à obtenir des problèmes *online* et il est bien évident que, dans le contexte des grilles de calcul, nous sommes en présence de problèmes de type *online*.

En plus des tests expérimentaux [28], beaucoup d'études ont porté sur des algorithmes *online*, avec garantie de performances. Ainsi, il est défini pour les algorithmes *online* le $\rho - competitive$ [84, 71]. Un algorithme online est $\rho - competitive$ si pour toutes ses entrées la solution est au plus ρ fois moins bonne que la solution optimale. Nous n'entrerons pas ici dans les détails des démonstrations, mais pour certains problèmes, il existe des algorithmes $\rho - competitive$, pour d'autres il n'en a pas été trouvé ou prouvé !

Maintenant, si le problème d'ordonnancement est difficile dans le contexte de machines parallèles homogènes, il l'est encore plus dans le cadre des machines parallèles hétérogènes. Sa difficulté est encore multipliée, lorsque les applications comme celles que nous avons définies doivent pouvoir s'exécuter de manière synchrone sur plusieurs machines.

Le problème d'optimisation En premier lieu, afin de répondre aux contraintes fortes d'architecture/système données par la requête `LDT`, un sous-ensemble de machines de la grille est sélectionné par requête au `SIC` stockant les informations statiques des machines. Une deuxième phase est le choix des machines réelles sur lesquelles sera exécutée l'application à partir du sous-ensemble de machines obtenues. Cette phase désigne donc plus une optimisation du placement et de l'ordonnancement. Bien souvent cette deuxième phase est oubliée, un simple `first-fit` est utilisé. En effet, le problème d'optimisation peut être pris sous 2 angles :

Le Placement : l'utilisateur de la grille effectuant une requête a toujours une intuition assez précise des machines qu'il veut utiliser. Si son application consomme beaucoup de temps `CPU`, mais communique assez peu il voudra des CPU puissants, mais pas forcément un réseaux à haut débit et de faible latence. Au contraire, certaines applications requièrent des communications intensives, la vitesse du `CPU` pouvant être un facteur moins bloquant. L'optimisation du `placement` désigne ici la satisfaction de l'utilisateur à obtenir les machines.

L'ordonnancement : l'utilisation de la grille doit toutefois être aussi optimisée dans le sens où toutes les requêtes ne doivent pas forcément utiliser les mêmes machines. Dans le cadre des grilles, où l'ordonnancement tourne en régime permanent, les critères possibles sont l'attente d'une application pour être exécutée et le taux d'utilisation des machines de la grille. Si beaucoup de machines ne font rien alors que d'autres sont très chargées et que l'utilisateur avait laissé libre le choix sur ces machines, l'ordonnacement aurait pu être meilleur. Le but est remplir les "trous".

Le problème d'allocation et/ou d'ordonnacement est très différent selon qu'il a été décidé d'exécuter l'application sur une seule et unique machine, ou plusieurs machines.

Placement Mono-Machine Lorsque plusieurs machines de la grille ont la capacité d'accepter la totalité d'une application synchrone ou une simple partie com-

plète d'un application asynchrone, le problème est relativement simple : l'allocateur revient ici à être un simple *Ressource Broker*. En effet son rôle est de choisir parmi les machines possibles la machine la moins chargée. Il joue le rôle d'aiguillage d'une requête vers un unique gestionnaire de travaux.

Placement Multi-Machine C'est ce cas qui pose le plus de problème. Car en effet, si d'un point de vue technique, ce problème n'est pas simple à résoudre, il l'est encore moins d'un point de vue optimisation. Un des choix les plus difficiles est de définir un découpage possible d'une application en n sous-applications. Autrement dit, il faut définir une partition de l'application. Le but est que chacune des partitions soit associée à sa machine cible, afin que l'application totale s'exécute au mieux. Beaucoup de paramêtres sont à prendre en compte pour définir cette partition. Une liste non exhaustive peut être :

- la charge des machines cibles : le lancement de toutes ces partitions devant être synchrones, il faut que les machines soient disponibles dans un futur proche ;
- les vitesses des processeurs : si les processeurs des machines ont des vitesses très différentes et que l'application a beaucoup de points de synchronisation, la vitesse globale de l'application sera calibrée sur le processeur le moins rapide ;
- les débits réseau : si une application communique beaucoup il est préférable de limiter le nombre de partitions sur différentes machines plutôt que de la scinder en beaucoup de petits morceaux, ralentissant d'autant l'exécution de l'ensemble. En revanche si l'application communique peu ce critère sera moins important.

Les critères énoncés ci-dessus ne sont que "quelques uns parmi beaucoup". Ce problème est vraiment très difficile. Il l'est encore plus avec les écueils techniques auxquels nous sommes confrontés pour apporter une solution, même non performante, comme nous le discutons dans le paragraphe suivant.

4.2.3 Coopération des gestionnaires

Il existe beaucoup de gestionnaires de files d'attentes (*batch system*). Les premiers ont été proposés dans le contexte des systèmes distribués tels que `Amoeba` ou `Chorus`. D'autres se placent comme des logiciels qui sont des sur-couches aux systèmes d'exploitation. Tous permettent de lancer de manière transparente des processus, sans que l'utilisateur ait connaissance de la machine physique sur lequel le processus s'exécute. Les implémentations sont nombreuses : `Codine`, `Condor`, `DJM`, `DQS`, `LoadBalancer`, `LoadLeveler`, `LSF`, `CONNET :Queue`, `NQE`, `PBS`, `NC Toolset`, `Task Broker` (voir [65]). Beaucoup adressent le problème des processus indépendants les uns des autres. Certains ne gèrent que des machines homogènes. Lorsque les processus ont des priorités, des temps de calculs estimés ou fixés, et que l'ensemble complet des tâches que nous avons à placer est inconnu, on parle plus volontiers de problème d'équilibrage de charge. Parfois, des

contraintes sur le placement sont ajoutées, par exemple un processus ne peut s'exécuter que sur un ensemble restreint de processeurs. Dans ce cas, le problème est plus souvent qualifié de placement de tâches.

Comme dit précédemment un des problèmes lorsque l'on veut exécuter de manière synchrone des applications sur plusieurs machines, est de réussir à faire coopérer les différents gestionnaires de travaux présents sur ces machines. Le problème revient donc à créer une sur-couche à ces ordonnanceurs permettant d'en proposer une vue uniforme. Dans Globus ce composant est le GRAM [52], il présente une interface unique pour attaquer tout gestionnaire de travaux pouvant déjà exister sur une machine. C'est sur ce point que le projet Globus a fourni le plus d'effort.

Le service Globus s'occupant de la coopération s'appellait DUROC, il consistait à faire coopérer plusieurs systèmes GRAM lancés sur plusieurs machines. L'implémentaion [52] où les caractéristiques d'une machine sont stockées dans le MDS (*Monitoring and Discovery Service*) n'était pas satisfaisante. Le principe utilisé était d'encapsuler les applications réelles à lancer de manière synchrone dans des applications spécifiques Globus. Ces applications "à la Globus" ne lancent effectivement les applications réelles que lorsqu'elles ont toutes passé un point de synchronisation, garantissant ainsi qu'elles ont toutes été lancées par le gestionnaire de travaux local. Le gros défaut de cette proposition est d'occuper des machines inutilement, par des processus dont le rôle n'est qu'attendre que les autres parties de l'application soient lancées par les autres gestionnaires.

Dans la version 3 de Globus, il avait été prévu une nouvelle architecture appelée GARA [56, 91] (*Globus Ressource Management Architecture*) dont l'idée principale etait d'utiliser la fonctionnalité de réservation, pour assurer le lancement synchrone d'applications grille. Ce nouveau composant redéfinit DUROC comme deux entités, l'une s'occupant de la réservation et l'autre de l'allocation (ainsi GRAM a été renommé en LRAM (*Local Resource allocation manager*). Même si GARA a été spécifié pour la version 3.0 de Globus, il n'est toujours pas disponible.

Nous avons voulu implémenter l'idée de réservation. Le prototype actuel du composant est spécifique pour l'ordonnanceur Maui [16] car il est le seul ordonnanceur du domaine public à disposer de possibilité de réservation dynamique. Son autre intérêt est aussi de pouvoir être intégré à des gestionnaires de files d'attentes, comme OpenPBS et SGE. En effet, Maui est juste un ordonnanceur, pouvant remplacer ceux fournis avec ces gestionnaires. Parmi toutes les machines de la grille, certaines disposent de Maui et donc de la capacité d'y faire une réservation, d'autres ont un autre gestionnaire de travaux, et donc ne disposent pas de cette fonctionnalité.

Le but de l'algorithme lorsqu'on recherche un ensemble de machine pouvant accueillir une application en mode multi-machine, est d'obtenir à partir de Maui

et du `SIC`, les intervalles de temps où des processeurs des machines sont potentiellement libres, puis lorsque la décision a été prise, les réservations sont effectuées auprès des différentes instances de `Maui` concernées.

4.2.4 Algorithme du prototype allocateur

Le mode de fonctionnement du prototype de l'allocateur avec `Maui`, est présenté dans la figure 4.2.

Dans le prototype, le but était de valider la chaîne technique dans l'intergiciel. Maintenant, tout n'est pas réglé, cela est essentiellement dû à la sémantique non exclusive des réservations sous `Maui`.

```
1) Requête au SIC pour obtenir la liste L des machines
   satisfaisant les contraintes fortes de la requête utilisateur,
   cette liste comprend des machines disposant ou non de Maui.

2) Construction d'un sous-ensemble L' de L où chacune des
   machines de L' peut contenir l'application synchrone:

3) SI L' n'est pas vide ALORS

     a) Choix de la machine la moins chargée dans L'.

   SINON

     a) Construction du sous-ensemble L_m  de L composé
        des machines disposant de Maui.

     b) Requête aux différents Maui de L_m, pour obtenir
        la liste des réservations faites sur ces  machines.

     c) Décision de placement incluant le partitionnement
        de l'application, vis-à-vis des réservations déjà
        faites sur ces machines.
```

FIG. 4.2: Fonctionnement de l'allocateur avec `Maui`

Dans le prototype, le but était de valider la chaîne technique dans l'intergiciel. Maintenant, tout n'est pas réglé, cela est essentiellement dû à la sémantique non exclusive des réservations sous `Maui`.

4.2.5 Conclusion sur l'allocateur

Définir un allocateur, pour un intergiciel est un problème technique difficile, mais aussi un problème algorithmique très complexe. Ce composant a été spécifié

et conçu dans le cadre du projet. Sa validation n'est malheureusement pour l'instant pas terminée. Il reste en particulier des problèmes techniques liés à la communication des informations pertinentes à `Maui`, caché derrière le `GRAM`, puis le gestionnaire de travaux.

Concernant le nombre restreint de modèles d'applications visés, l'allocateur étant sans état, il n'est pas très difficile d'ajouter d'autres allocateurs ayant pour but d'autres sémantiques d'allocation. Tous se synchronisent par les données présentes dans le `SIC`.

4.3 Un système de surveillance et de mesure intégré

L'originalité du noyau de l'intergiciel `e-Toile` réside dans le `SIC`, son Système d'Information et de Contrôle innovant. Le `SIC` implémente un système de gestion de base de données. Deux zones d'information sont gérées par le `SIC` :

- la zone réservée au noyau de l'intergiciel ;
- la zone utilisateurs (composants `CMPP`, applications).

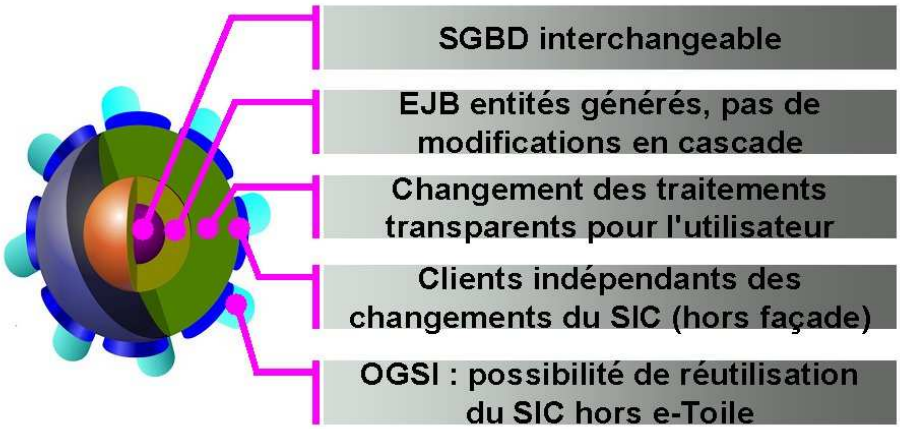

FIG. 4.3: Le système d'information et de *monitoring* (`SIC`) d'`e-Toile`

Les grandes fonctionnalités implémentées dans cette couche logicielle sont les suivantes :

- fournir une vision d'ensemble de la grille `e-Toile` aux utilisateurs ;
- disposer d'un spectre de données plus large pour permettre à l'allocateur de fonctionner sans état ;
- fournir les moyens de détecter d'éventuelles pannes ;
- fournir un niveau d'abstraction permettant aux utilisateurs d'être indépendants du mode stockage des données.

La raison majeure pour n'avoir pas conservé le MDS (*Monitoring and Discovery Service*) de Globus est son instabilité. Le manque de souplesse ainsi que le temps de réactivité et de rafraîchissement des informations (LDAP) ont motivé le développement d'un système d'information et de contrôle novateur. Le SGBD retenu est MySQL, outil du domaine public largement diffusé et standardisé. La mise en œuvre d'un SGBD distribué de type Oracle 10g serait tout-à-fait envisageable ; le SIC ayant été conçu de manière à rendre le SGBD "interchangeable". L'originalité de cette brique logicielle réside également dans ses différents types d'interface :

- administration du SIC ;
- administration des sites ;
- SPAM (service de publication et de *monitoring*) localisateur ;
- publication d'évènements ;
- *monitoring* ;
- allocateur ;
- SPAM ;
- chargeur.

Des fonctionnalités supplémentaires, telles que :

- le processus de vérification de la cohérence des informations ;
- l'interface OGSI et la compatibilité OGSA ;
- la détection des pannes sur site ;
- l'archivage des données.

pourraient faire l'objet de développements futurs.

4.3.1 Composants de surveillance et de contrôle

Les composants GRAM de Globus permettent, de s'interfacer avec la plupart des gestionnaires de travaux ou *batch managers* utilisés dans le domaine scientifique : LSF, OpenPBS, SGE, CONDOR, NQE, ... C'est la raison pour laquelle le comité d'architecture a opté pour l'utilisation de ce composant de Globus pour la récupération d'informations, ce que ne font pas les outils de *monitoring* classiques tels que GANGLIA (dont les capteurs sont dépendants du système *batch* installé).

Les différents services de publication d'information SPAM (service de publication et de *onitoring*) implémentés dans e-Toile sont bâtis au-dessus des composants GRAM de Globus. Ils permettent une portabilité accrue ainsi qu'une indépendance vis-à-vis du gestionnaire de travaux installé en local. Dans le cas de PBS, les SPAMs détectent la présence des commandes PBS pour publier des informations plus fines que celles mises à disposition par les GRAMs. Les services SPAMs sont l'équivalent des capteurs ou encore des *dendrites* de GANGLIA adaptés à Globus et implémentant une interface cliente EJB (1.1) ainsi qu'une interface *Grid Service* OGSI. Trois types de service SPAM sont implémentés dans la version actuelle d'e-Toile. Sur la tête de grappe, SPAM H publie les informations concernant la tête du grappe (la station d'accueil) et SPAM J publie les

informations concernant les travaux, sur chaque nœud de la grappe, SPAM N publie les informations locales à ce nœud.

Les SPAMs publient les informations à destination du SIC :

– lors de l'installation (informations statiques) ;
– au redémarrage (*boot*) de chaque machine (informations statiques) ;
– cycliquement (*via* /CRONTAB) (informations dynamiques).

Toutes ces informations sont remontées dans le SIC à une fréquence paramétrable dans le fichier de configuration des SPAM. De part sa conception, ce système de SPAMs permet de publier dans le SIC, de façon automatique, les informations relatives aux ressources de la grille. Ces informations sont visualisables à travers l'interface *WEB* ainsi que par le biais de l'interface en ligne de commande. Elles sont donc exploitables pour diverses utilisations, voire des extensions futures. Des fonctionnalités supplémentaires, telle que le support de systèmes d'exploitation autres que UNIX/LINUX ou bien le support d'autres gestionnaires de travaux pourraient faire l'objet de développements futurs.

4.4 Langage de description de travaux extensible, interface utilisateur conviviale

Le GUIDE (Grille Utilisateur Interface d'e-Toile) représente non seulement l'interface utilisateur mais aussi l'ensemble des composants d'interfaçage avec la grille ; il s'agit de la brique logicielle représentant le point d'entrée dans la grille e-Toile. Un effort tout particulier a été porté sur le LDT (Langage de Description des Travaux) et ses attributs ainsi que sur l'interface graphique. En effet, ce point d'entrée dans la grille nous a paru primordial car la plupart des APIs dans ce domaine sont souvent rébarbatives et nécessitent une connaissance approfondie du système, "tournant" derrière l'interface pour mener à bien l'exécution d'un travail, ce qui n'était pas souhaitable dans le cadre d'e-Toile.

4.4.1 Langage de description de travaux

Le LDT est l'équivalent de l'UI (*user interface*) que l'on trouve dans le projet DataGrid ou encore de la machine cliente Globus. Il fournit un langage qui appartient à la même catégorie de langage que le RSL de Globus ou le JDL de DataGrid. Cependant, il possède une capacité de description plus fine et plus souple que ses concurrents.

De par sa conception même, le LDT fournit à l'utilisateur un haut niveau d'abstraction. La complexité de la grille est parfaitement masquée à la vue de l'utilisateur. Celui-ci n'a pas besoin d'expertise pour soumettre des travaux sur la grille contrairement aux autres intergiciels de grille. De plus, ce langage est parfaitement extensible, et ce, à "moindre frais" grâce à une génération automatique des objets Java et à l'utilisation des champs *Requirements* et *Rank*. En revanche, l'utilisateur doit parfaitement connaître la structure de l'application qu'il souhaite "lancer" de

manière à pouvoir décrire finement ses travaux, dans le fichier d'entrée de soumission de travaux, au format `XML`.

```
<Application>
    <Job jobname="sampleJob" project="sampleProject">
            <InputFiles protocol="gridftp">
                    <File path="/usr/local/incoming/input"/>
            </InputFiles>
            <OutputFiles protocol="gridftp" transfert="onexit">
                    <File path="/usr/local/outgoing/output"/>
            </OutputFiles>
            <Requirements>
                     (queue_name = "sample queue")
            </Requirements>
            <Rank>3</Rank>
            <Executable binary="/usr/local/bin/myProgram" >
                    <Arguments/>
                    <Environment/>
            </Executable>
    </Job>
</Application>
```

FIG. 4.4: Schéma d'une requête en `XML` au `LDT`

Un autre point fort indéniable est l'utilisation du langage `Java`. Langage portable par excellence *a contrario* des *scripts* de type `UNIX`.

4.4.2 Interface utilisateur graphique

Le deuxième volet du `GUIDE` est l'interface utilisateur *WEB* graphique. Il s'agit d'un outil permettant de visualiser l'état des travaux et des ressources de la grille `e-Toile`.

Son interface conviviale et intuitive en fait un outil de *monitoring* simple et efficace. Des fonctionnalités supplémentaires, telle que le rafraîchissement automatique des pages*WEB* ou bien la soumission graphique des travaux pourraient faire l'objet de développements futurs.

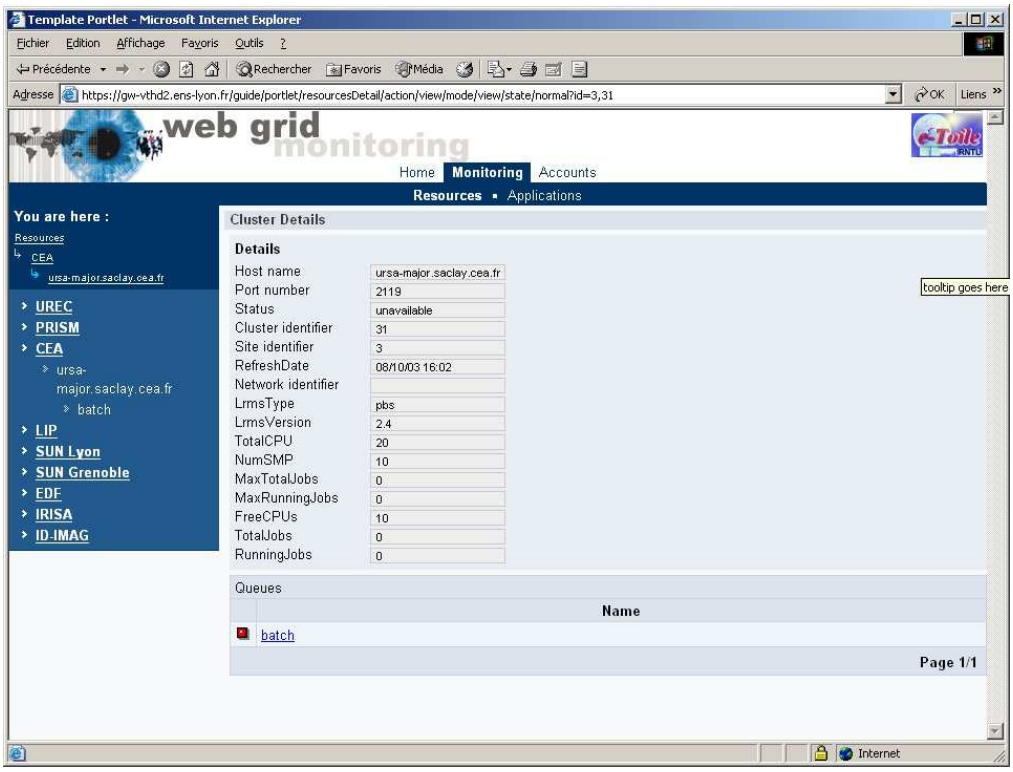

FIG. 4.5: Interface utilisateur du `GUIDE`

Chapitre 5

Comment supporter de multiples applications hétérogènes ?

5.1 Propositions de classification

Les développements des grilles de calcul et de données concernent de plus en plus d'applications dans des domaines de plus en plus variés.

L'on trouve en premier lieu tout ce qui vient du monde de la simulation numérique et du calcul distribué avec les modèles de programmation par échanges de messages ou dans le cas d'applications faisant intervenir plusieurs codes une approche client/serveur voire un modèle objet-distribué (cela s'est traduit par exemple dans les projets européens `Metodis` [17], `Damien` [4]).

Le concept de Grille de calcul et de données a émergé, dans ce domaine, lorsque l'on a essayé de dépasser les limitations de ces applications. Bien sûr, les bornes les plus évidentes étaient liées à la possibilité de fédérer et d'utiliser les ressources "*hardware*" (puissance de calcul, mémoire, bande passante, ...), c'est bien là le but des projets d'intergiciels -ou "*middleware*"- comme `Globus` [10], `Legion` [13], `Unicore` [25], ...et cela a constitué la problématique centrale des projets structurants tels que `EuroGrid` [6] ou `DataGrid` [55]. Mais rapidement sont apparus des problèmes liés à la conception même des applications en termes d'interopérabilité, de modularité, de déploiement, d'interactivité, de partage de données distribuées et plus généralement de passage à l'échelle. Ces problèmes sont liés à des nouveaux usages des outils informatiques rendus possibles par les grilles.

Vu des applications, l'intergiciel est le composant clef qui procure à l'utilisateur une vision uniforme des ressources distribuées et hétérogènes. Cette fois le terme "ressources" est pris au sens le plus large : matérielles, logicielles, applicatives, mais aussi bases de données, instruments scientifiques, etc.

La "grillification" des application est donc le problème soulevé par l'évolution des besoins, la multiplication des ressources disponibles et des intervenants ainsi que l'émergence de nouveaux modes de travail. Nous distinguerons trois axes, "res-

sources" , "applications", usages".

Axe "ressources" Ce premier type de classification fait référence aux limitations "*hardware*" et se révèle insuffisant. Nous distinguerons les applications :

- ***"cpu intensive"***, les applications qui effectuent de grandes quantités de calculs. Typiquement, on évalue (e.g. en `Gflops`) la performance d'une application qui effectue des calculs flottants ;
- ***"data intensive"***, les applications qui manipulent de grandes quantités de données en mémoire ou sur des sites de stockage. Là aussi on dispose d'indicateurs précis (e.g. `Go`) ;
- ***"network intensive"***, les applications [distribuées] qui nécessitent beaucoup de transferts d'information et sont sensibles aux caractéristiques du réseau : débit, latence et qualité de servive -`QoS`- en général. Tout cela se mesure de façon précise : taille des données (e.g. `Go`), bande passante utilisée sur le réseau (e.g. `Go/s`).

Axe "applications" Deuxième typologie, les aspects liés au caractère distribué des applications sur la grille. Ici les classifications sont beaucoup plus grossières :

- **la distribution**, en terme de nombre de tâches et de sites. On distinguera les applications faiblement distribuées (jusqu'à une dizaine de composants), distribuées (quelques centaines de composants) ce sont celles qui nous intéressent ici et enfin les systèmes distribués à grande échelle (jusqu'à plusieurs dizaines de milliers de composants) ;
- **la granularité**, qui mesure la taille de l'élément de distribution et qui est liée à la taille des tâches rapportée au volume des communications entre ces tâches. Le grain de l'application sera "fin", "moyen", "gros" ou encore "variable". Typiquement le grain "moyen" est adapté à la distribution sur une grappe ; le grain "gros" à la distribution sur plusieurs sites.
- **la régularité**, nous distinguerons les applications régulières (parallélisation automatique, applications SPMD, ..), semi-régulières (e.g. couplage d'applications régulières) et enfin irrégulières ou hétérogènes (spécialisation des composants) ;
- **la dynamicité**, l'application est statique si les tâches et leurs dépendances mutuelles peuvent être décrites avant l'exécution, c'est le plus souvent le cas pour des applications basées sur les échanges de messages. Les aspects dynamiques incluent, le redéploiement (e.g. équilibrage de la charge), la création de nouvelles t âches mais aussi la tolérance aux défaillances.

Axe "usages" Viennent enfin les usages [de l'application] et l'interaction avec l'intergiciel. Classification beaucoup plus qualitative :

- **les données et les résultats** de l'application : nature (interactif, ...), taille, transfert depuis ou vers des sites de stockage, traitement (filtrage, analyse dynamique, ...) ;

- **la sécurité** : quels besoins en terme d'authentification et de droits des usagers de la grille ; de protection, confinement et cryptage des données. Comment harmoniser les politiques locales de sécurité sur chaque site et l'organisation mise en place au niveau de la grille ;
- **la "synchronicité"** des composants de l'application notamment pour les phases cruciales de lancement et de terminaison. Ainsi pour une étude paramétrique consistant à effectuer de nombreuses exécutions d'une même application, les tâches sont asynchrones en revanche, les différents composants d'une application parallèle sont fortement couplés et nécessitent un lancement synchronisé ;
- **les contraintes temporelles** notamment l'interactivité (synchronisation des composants de l'application et des utilisateurs) mais l'on peut aussi ranger dans cette catégorie les modèles à temps contraint (e.g. temps réel).

En conclusion, si l'on veut affiner la caractérisation des applications déployées sur les grilles, on voit que l'on peut exhiber une taxonomie très riche. En revanche on ne dispose guère de métriques sauf pour la classification traditionnelle ("CPU, Mémoire, Réseau") ; cela pose d'ailleurs le problème de la définition d'une suite de "*benchmarks*" pour les grilles, voir à ce sujet les travaux spécifiques menés dans le cadre du GGF (*Global Grid Forum*) [11] ou encore le projet GRASP [48].

5.2 Méthodologie de développement et d'évaluation des gains

La grille e-Toile est une grille haute performance. La partie applicative du projet s'intéresse en priorité à des grandes applications de la simulation numérique qui n'ont pas été conçues *a priori* pour les grilles de calcul mais d'autres aspects ont pu être abordés dans e-Toile grâce à la participation de l'IBCP.

Le premier problème rencontré est donc celui de l'adaptation des applications patrimoniales à la grille avec deux axes de questionnement, le premier concernant les modèles, le second la portabilité.

Les **modèles** de programmation et d'exécution utilisés sont-ils bien adaptés au contexte de grille et notamment y a-t'il une compatibilité avec les approches déjà utilisées en informatique distribuée : client/serveur "à la Corba" [3] avec des mises en œuvre optimisées comme PadicoTM [54], objets distribués [32] par exemple "à la Java" en suivant les réflexions menées dans le forum JavaGrande [12], et plus récemment dans [66].

L'application est-elle **portable** ?

- quel est son mode de diffusion ("source", "binaire") ; comment son utilisation est-elle contrôlée ? (libre, licence utilisateur ou site , "jetons", ...) ;
- résiste-t-elle à l'**hétérogénéité** des matériels, des systèmes d'exploitation, des environnements d'exécution.

Il est alors possible d'évaluer l'impact du déploiement sur la grille de l'application : simple portage, encapsulation (par exemple dans des composants), restructuration (notamment des structures de données dans le cas de codes procéduraux) et enfin réecriture complète.

Nous retiendrons une interrogation globale sur les modèles de programmation et d'exécution de la grille et le problème crucial de la résistance à l'hétérogénéité.

La contrepartie de l'hétérogénéité c'est l'interopérabilité qui est normalement assurée par l'intergiciel. Néanmoins, les projets qui ont une dimension grille mettent en avant des communautés d'utilisateurs, le partage de connaissances ou d'information au sein de ces communautés nécessite un travail de normalisation (format de données, définition de meta-données, ...) qui peut aussi conduire à des modifications profondes des logiciels. En effet des "formats" différents de données ne couvrent pas forcément les même champs pertinents d'information.

Concevoir ou déployer une application sur la grille implique de se poser la question des fonctionnalités de l'intergiciel. Dans la cadre d'`e-Toile`, après une étude comparative des outils disponibles, nous avons choisi `Globus` comme intergiciel de base. Ce dernier semblait en effet offrir les fonctionnalités minimales propre au modèle de grille : gestion des utilisateurs, des sites, des applications, des données. Ainsi dans le cadre d'`e-Toile` l'interrogation a-t-elle surtout portée sur les aspects performances et la nécessité de remplacer certains composants de `Globus` par des briques plus performantes ou aux fonctionnalités étendues.

Dans ce cadre se pose le problème de l'accessibilité de ces fonctionnalités à l'utilisateur au niveau applicatif, c'est typiquement le cas pour les modules "réseau actif" : qualité de service sur le réseau, traitement "au vol" des données.

La mesure des gains entre `Globus` -plate-forme version 1 d'`e-Toile`- et l'intergiciel `e-Toile`- version 2 du projet - pose une nouvelle fois le problème des métriques à mettre en place pour mesurer les "performances" de l'application sur la grille. Le question ne se pose plus en terme aussi simples que, par exemple, la mesure des `Gflops` sur un super-calculateur. Il faut en premier lieu caractériser les attentes des applications vis-à-vis de la grille :

- fédérer ponctuellement des ressources pour pouvoir lancer un "calcul plus grand" ;
- produire "plus de résultats" (études paramétriques ou statistiques) ;
- se positionner par rapport à des contraintes extérieures (e.g. le temps de restitution, l'interactivité...) ;
- rendre possible les communications et les interactions entre des composants logiciels initialement isolés.

Pour des applications de la simulation numérique qui s'exécutent sur des supercalculateurs ou des grappes de `PCs` ("*clusters*"), nous pourrons effectuer des comparaisons de performances brutes mais celle-ci n'auront de sens que si elles sont

"modulées" par quelques indicateurs "économiques" (typiquement dans le cas où les ressources utilisées n'auraient pas pu être disponibles localement, etc). Dans les autres cas il faudra mettre en rapport le taux d'utilisation "utile" des ressources avec la notion de service rendu aux utilisateurs. On retrouve là les difficultées à définir des *benchmarks* pour la grille et aussi la nécessité de disposer d'un modèle économique qui permette de mettre en rappport ressources utilisés et résultats obtenus.

5.3 Exemples d'applications déployées

Vu des applications, le projet `e-Toile` comprend trois grands volets :

1. Mise en place d'une plate-forme matérielle **performante** et installation d'un intergiciel de base (version 1).
2. Développement et intégration de composants logiciels spécifiques, cruciaux dans l'optique applications scientifiques hautes performances et d'outils plus "transversaux" facilitant l'accès et l'usage de la plate-forme (version 2).
3. Définition et déploiement d'applications pour tester la validité (version 1) et les performances (version 2) de la plate-forme.

Notons que cette plate-forme est, au niveau Français, exceptionnelle. Il s'agit bien de valider un modèle de grille haute performance avec des applications numériques distribuées à fort besoin de calcul, de mémoire et effectuant de nombreux et volumineux transferts de données. Dans ce cadre, certaines fonctionnalités de l'intergiciel prennent une importance primordiale :

- réserver voire synchroniser des ressources de calcul. Ce mode de fonctionnement "intrusif" vis à vis des ressources locales n'est pas `a priori` celui de la grille ;
- utiliser des réseaux à haut débit avec une qualité de service garantie pour assurer, par exemple, une bonne synchronisation entre les différents composants de l'application distribuée qui s'exécutent simultanément sur plusieurs sites ;
- utiliser des outils pour optimiser le lancement des composants de l'applications, par exemple transférer le binaire de l'application sur les nœuds de calcul ;
- utiliser des outils pour optimiser les mouvements de données sur la grille : données en entrée des applications, messages échangés entre composants, émission des résultats vers des sites de stockage.

Cette approche est celle des trois partenaires applicatifs initiaux (`CEA`, `EDF`, `PRiSM`). L'`IBCP`, qui a rejoint le projet en tant qu'utilisateur de la plate forme pour des applications comportant également la notion de "services *WEB*" auprès de la communauté scientifique, a une approche un peu différente qui sera détaillée plus bas.

5.3.1 CEA Atlas

Contexte scientifique et enjeux l'application provient d'une expérience en cours au CERN dont le CEA-DSM-DAPNIA est partenaire. Les grands collisionneurs sont les derniers-nés des accélérateurs de particules avec lesquels les physiciens explorent la structure ultime de la matière. Ils ont permis de grandes avancées dans la compréhension des interactions fondamentales et des particules élémentaires. Les nouveaux projets mettent en jeu des appareils de plusieurs dizaines de kilomètres qui sont construits en collaboration mondiale. Le dernier né de ces collisionneurs est le LHC (*Large Hadrons Collider*) [14] en fin de construction au CERN à Genève. A cet appareil sont associés des détecteurs qui permettront de reconstituer les évènements (collisions ou désintégrations) et de les confronter aux théories. Le LHC est composé de quatre détecteurs : Alice, CMS, le LHCb et Atlas.

Les détecteurs comportent de multiples composants utilisés pour distinguer les différents types de particules et en mesurer l'énergie et les moments. Les ordinateurs connectés aux détecteurs collectent et interprêtent les données produites et présentent des extrapolations de résultats aux physiciens.

Le but du détecteur Atlas est de récolter les informations sur les collisions proton-proton. L'expérience, unique (au moins) par sa taille, implique près de 2000 physiciens issus de 150 universités et laboratoires de plus de 34 pays [1].

Pour Atlas, l'information doit être suffisamment précise et complète pour identifier certains évènements qui ont une chance d'être observés sur 10^{12} collisions. Cette information doit être analysée et stockée très rapidement. En effet, chaque événement représente 1 à 1.5 Mo de données. Sur le billion de collisions généré par seconde dans le LHC, seulement dix à cent collisions sont potentiellement intéressantes, toutes les autres sont rejetées. Néanmoins, on estime que l'expérience produira plusieurs PetaOctets de données par an.

Le code Atlas, qui nous intéresse ici, met en œuvre une méthode de Monte-Carlo de génération physique d'événements et de simulation du détecteur afin de permettre à la communauté de se procurer des jeux de données de l'ordre de 107 événements (ce qui correspond à une dizaine de TeraOctets de données). On espère ainsi mieux comprendre le fonctionnement de l'installation mais aussi anticiper sur les problèmes posés par les grandes masses de données produites.

Ainsi le code Atlas est une application monoprocesseur que l'on va exécuter "un grand nombre de fois", chaque exécution produit un fichier de résultat dont la taille est de l'ordre du Go. Enfin l'analyse des résultats nécessite l'accès à l'ensemble des données produites. Notons que cette application est utilisée dans le "*testbed 0*" du projet DataGrid. D'autres approches peuvent être utilisées pour ce type de problèmes, voir par exemple [72], basées sur des plates-formes de calcul global comme XtremWeb [26] qui gérent de façon plus systématique la distribution des tâches. Mais ici le point clef est aussi la gestion des grands volumes de données produits.

Travaux dans le cadre d'e-Toile Les objectifs liés à Atlas dans le projet e-Toile sont :

- tester les stratégies de déploiement (allocateur, chargeur...) ;
- tester les transferts à haut débit ;
- utiliser la spécifité "réseau actif" (pour l'analyse des résultats).

Une version Atlas en kit (*rpms* sous Linux) a été mise à la disposition de la communauté. La distribution comprend les sources mais nécessite un environnement de compilation et d'exécution très contraignant (Linux, Fortran, C, C++, bibliothèques spécifiques du CERN, bibliothèque GEANT, ...) Pour cela il n'est pas possible d'envisager l' exécution sur un site distant d'un binaire construit "en local". L'application doit d'abord être installée et validée sur un site, elle peut ensuite être exécutée en mode "ASP". Le code Atlas s'exécute donc sur un nœud de grappe, les ressources consommées pour un Pentium 4 avec 512 Mo de mémoire vive sont récapitulés dans le tableau 5.1. Pour faire des tests sur un lancement type de l'application, les développeurs de ce code conseillent de mettre au minimum 250 collisions.

Nombre d'événements	Temps	Taille du fichier de sortie
10	0h22	100 Mo
20	0h43	167 Mo
30	1h02	234 Mo
40	1h22	300 Mo
50	1h45	364 Mo
100	3h17	676 Mo
150	5h01	1 Go
250	8h07	1.7 Go

FIG. 5.1: Les besoins de l'application Atlas

Quelques mesures ont été effectuées pour mesurer les besoins en terme de transfert de données. Des fichiers faisant respectivement 8, 16, 32, 64, 512, 1024, 2048 et 4096 Mo ont été transférés entre deux sites e-Toile (Saclay et Lyon) par FTP et *globus-url-copy*, en jouant sur les options de *globus-url-copy*. Ces résultats sont résumés dans le tableau 5.2.

Par rapport à FTP, *globus-url-copy* n'est pas par défaut optimisé pour les transferts de données volumineuses. Il l'est en revanche pour les petits fichiers. Il faut

utiliser l'option *-p n* sur *globus-url-copy*, avec *n* égal à 2 ou 3, pour obtenir une amélioration (de l'ordre de 1.5). Ce paramètre permet l'envoi du fichier en plusieurs flux. Les performances restent du même ordre si on augmente *n* au delà de 4. En réalité, pour améliorer la performance avec *n* grand, il faudrait augmenter la taille des fichiers (bien au delà de la dizaine de `GigaOctets`). Nous retiendrons une mauvaise performance et un paramétrage *a priori* complexe pour l'utilisateur.

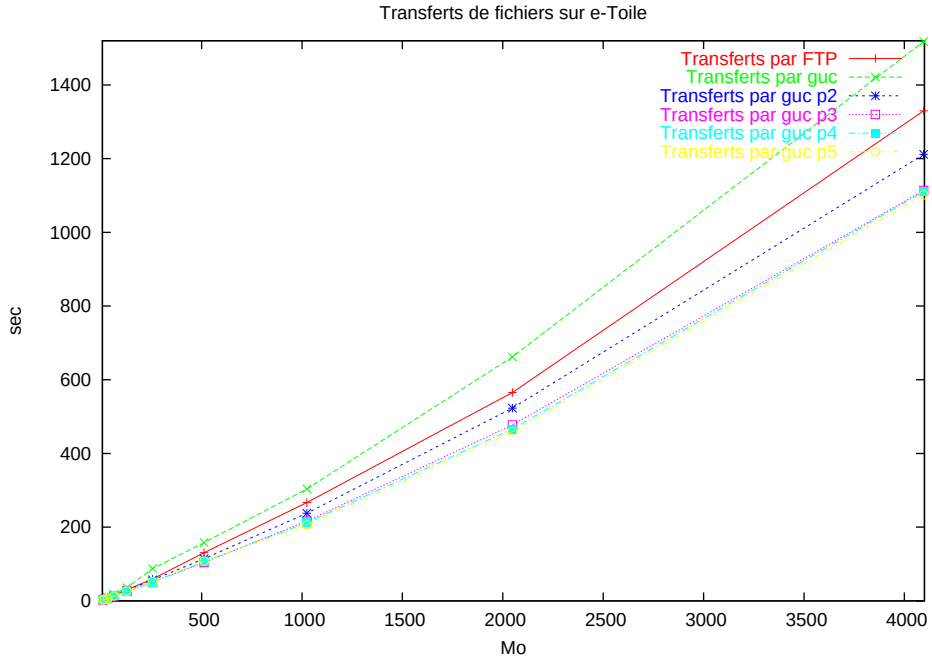

FIG. 5.2: Transferts de fichiers dans `e-Toile` 1, comparaison de plusieurs protocoles

Les travaux sur le version 1 ont donc consisté à déployer l'application sur les sites `e-Toile` et à l'utiliser comme un service distant. Il a fallu attendre le premier déploiement du version2 pour pouvoir exprimer, à travers le `LDT`, le lancement "en rafale" (asynchronisme) sur **plusieurs sites** à travers un seul *script*. L'exemple de la figure 5.3 montre le squelette d'un tel *script*, l'application asynchrone est constituée de trois tâches (*jobs*) qui lancent respectivement 6, 10 puis 10 exécutions de l'applications sur trois sites différents d'`e-Toile`.

La stratégie consiste ensuite à définir des sites de stockage où seront transférés les résultats de ces exécutions. L'utilisation des briques logicielles `NFSp/Gxfer` semble bien sûr indispensable mais n'a pu encore être mise en œuvre. Enfin une dernière série de tests impliquerait les fonctionnalités "réseau actif". En effet si l'analyse des résultats implique la lecture des fichiers créés, il n'est pas possible

d'en rapatrier un grand nombre sur un poste de travail. Néanmoins un traitement de bas niveau type "filtrage" pourrait être mis en place et seule une petite quantité d'information pertinente parviendrait à l'utilisateur pour être traitée par le programme de reconstruction/analyse.

L'intérêt n'est pas seulement de rendre possible ce type d'analyse, mais de la paramétrer "au plus haut niveau" selon l'information que l'utilisateur final recherche dans les fichiers résultats de la simulation ou dans ceux produits par l'expérience elle-même.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Application>
<Application universe="standard" synchronization="asynch">
  <Requirements>(1=1)</Requirements>

  <Job jobname="Atlas1" project="DEMO_RNTL" nb-process="6">
   <Requirements>
      cluster_hostname="cocteau.prism.uvsq.fr")
   </Requirements>
   <Executable binary="./demoAtlas" stdout="/home/anfray/AT1.out"/>
  </Job>

  <Job jobname="Atlas2" project="DEMO_RNTL" nb-process="10">
   <Requirements>
     (cluster_hostname="ursa-major.saclay.cea.fr")
   </Requirements>
   <Executable binary="./demoAtlas" stdout="/home/anfray/AT2.out"/>
  </Job>

  <Job jobname="Atlas3" project="DEMO_RNTL" nb-process="10">
   <Requirements>
     (cluster_hostname="serveur1.etoile.edf-labs.net")
   </Requirements>
   <Executable binary="./demoAtlas" stdout="/home/anfray/AT3.out"/>
  </Job>

</Application>
```

FIG. 5.3: *script* `XML` de lancement des tâches de l'application `Atlas` sur la grille `e-Toile`

5.3.2 CEA CHARMm

Contexte scientifique et enjeux Cette partie est représentative des besoins en terme de calcul du `CEA-DSV` dans le domaine de la dynamique moléculaire pour l'étude des mouvements et de l'évolution de la configuration spatiale des systèmes moléculaires.

Toute molécule est une association de divers atomes. La cohésion de cet ensemble résulte des interactions, répulsives et attractives, entre les atomes. Une molécule sera donc modélisée par un système purement mécanique de forces, sans tenir compte de la nature microscopique et électrostatique des interactions entre les électrons et les noyaux. La modélisation moléculaire représente les atomes par des sphères et les liaisons entre les atomes par des ressorts. De plus, les atomes sont modélisés comme des charges ponctuelles (leur polarisabilité électronique est négligée). Pour réduire le temps de calcul, on fait une approximation qui consiste à calculer l'interaction entre un atome et le reste du système par sommes de paires, de sorte que la paire d'atomes ne voit pas les autres atomes du système. De plus, l'interaction entre deux atomes distants de plus d'une distance définie à l'avance (dite *cutoff*) est ignorée. En partant d'un temps t donné, on connait les positions $r_i(t)$ des atomes. A partir de ces données, on peut calculer la fonction d'énergie potentielle du système $E(t)$ qui permet de calculer la force $\vec{F_i}(t)$ exercée sur chaque atome i de masse m_i se trouvant à la position $\vec{r_i}(t)$ à l'instant t.

$$\vec{F_i}(t) = -\frac{dE(\vec{r_1}, \ldots, \vec{r_n}, t)}{d\vec{r_i}(t)}$$

A l'aide de la deuxième loi de Newton, on peut en déduire l'accélération $\vec{a_i}(t)$ de chaque atome i à l'instant t.

$$\vec{F_i}(t) = m_i \vec{a_i}(t)$$

Puis en intégrant cette équation, on peut obtenir la vitesse $\vec{v_i}(t)$ de chaque atome i à l'instant t.

$$\vec{a_i}(t) = \frac{d\vec{v_i}(t)}{dt}$$

Avec un développement limité, on peut trouver la position de chaque atome à l'instant $t + dt$.

$$\vec{r_i}(t + dt) = \vec{r_i}(t) + \vec{v_i}(t)dt + \frac{1}{2}\vec{a_i}(t)$$

On recommence ainsi le processus jusqu'à ce que l'on ait simulé assez d'instants.

Le logiciel de dynamique moléculaire utilisé est le code CHARMm, *Chemistry at HARvard Macromolecular Mechanics* [2, 45]. Il a été conçu pour permettre de réaliser des minimisations d'énergie et des dynamiques moléculaires de protéines, d'acides nucléiques et de lipides, que ce soit dans le vide, en solution ou dans un environnement cristallin. Un des objectifs de ce code est donc de simuler le mouvement d'une protéine au sein d'un système de molécules. Le principe général consiste à appliquer l'équation de Newton : $\Sigma \vec{f} = m\vec{a}$ à un système biologique. En fait, on crée la molécule ou le système de molécules à l'aide d'outils de modélisation puis on les étudie avec CHARMm. Pour cela, on applique une certaine force au système, il s'agit de la fonction d'énergie potentielle. Cette force a été définie au préalable et validée avec des molécules dont on a pu tester expérimentalement

le comportement. Les différents paramètres de la fonction peuvent être obtenus par des expériences de spectroscopie ou par la mécanique quantique. Ce logiciel a de nombreuses fonctionalités. On peut par exemple utiliser plusieurs protocoles de dynamique, notamment la dynamique classique, la dynamique de Langevin, la mécanique quantique.

Les applications de ce logiciel au CEA-DSV sont multiples, citons pour ce qui est visé dans le cadre du portage sur la grille e-Toile :

- l'étude d'un système membranaire : on se place dans une boîte cubique contenant des lipides qui constituent la paroi de la membrane et une protéine, le tout avec de l'eau. La protéine se trouve sur une traverse de la membrane et intervient dans le processus de communication entre l'intérieur et l'extérieur de la membrane. On cherche à voir l'interaction entre les lipides et la protéine et déterminer la trajectoire de cette dernière ;
- l'étude de la protéine du prion : la maladie due à cette protéine apparaît lorsqu'elle change de conformation, c'est à dire de configuration géométrique. Elle peut par exemple se replier sur elle même. On modélise la protéine dans une boîte cubique remplie d'eau et on étudie les conditions et conséquences des changements de conformation en présence ou non d'une membrane ;
- l'étude de l'anexine : cette protéine intervient dans la mort d'une cellule. On l'étudie aussi dans une boîte remplie d'eau en présence ou non d'une membrane.

Ces trois applications utilisent les protocoles de la dynamique classique.

Pour donner une idée du dimensionnement, le premier système, par exemple, est composé d'environ 60000 atomes. La résolution consiste à intégrer l'équation de Newton sur des pas de temps de l'ordre de la femtoseconde ($10^{-15}s$). On peut alors simuler des phénomènes de l'ordre de la nanoseconde ($10^{-9}s$), ce qui nécessite à peu près un mois de calcul et constitue l'état de l'art alors que les phénomènes biologiques que l'on souhaite comprendre ont une durée caractéristique de l'ordre de la milliseconde.

Travaux dans le cadre d'e-Toile Les objectifs liés à CHARMm dans le projet e-Toile sont les suivants :

- déploiement optimisé (exécutable construit en local) ;
- allocation de resources utilisant à la fois les modes synchrone et asynchrone ("rafales" de travaux parallèles multi-grappe) ;
- parallélisme : "meta-MPI" (i.e. nœuds locaux et distants).

L'application est utilisée de façon intensive -étude paramétrique-. CHARMm est un code parallèle qui nécessite aussi beaucoup de mémoire vive. Le code passe mal à l'échelle du fait du grand volume des communications entre les tâches. Ainsi concrètement même s'il n'est guère possible de dépasser quelques dizaines de nœuds, nous pourrons envisager l'utilisation de plusieurs grappes pour une seule exécution.

la distribution du code comprend les sources et ne nécessite pas un environnement trop spécialisé. Il est possible de construire le binaire localement et de l'envoyer pour exécution sur un site distant "compatible" (système d'exploitation, bibliothèques dynamiques, etc..) avec le site local. Cette façon de travailler s'est avéré être la plus simple à mettre en œuvre. La version parallèle utilise la bibliothèque `MPI`, il reste à définir le lancement de l'application à travers `Globus` et les ordonnanceurs locaux (type `OpenPBS`).

Dans le cadre du version 1, pour lancer des exécutions sur plusieurs grappes, il faut passer par `MPICH-G2` [18] qui est une implémentation du standard `MPI` "au-dessus" de `Globus`. Tout cela a posé d'énormes difficultés de paramétrage -notamment pour "traverser" les ordonnanceurs locaux- et de toutes façons, l'utilisation de `MPICH-G2` ne résoud pas le problème de la synchronisation des travaux sur des sites distincts. L'aptitude de l'application à passer à l'échelle a été évaluée sur la grappe du `CEA`. Cela donne, par exemple, pour 500 étapes de simulation la courbe de la figure 5.4 qui pourra servir à valider l'approche multi grappe.

N	tps total	tps cpu
1	3h 16' 12"	3h 16' 12"
2	2h 04' 48"	1h 55' 12"
4	1h 27' 36"	1h 19' 48"
8	1h 07' 48"	1h 01' 12"
16	52' 42"	48' 09"

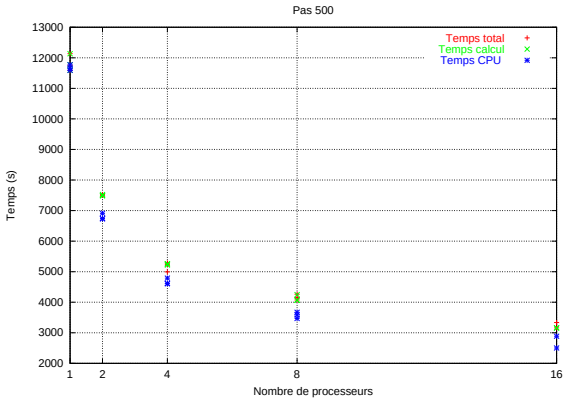

FIG. 5.4: Les besoins de l'application `CHARMm`

Les travaux futurs impliquent l'utilisation du portage de Madelaine au-dessus de `Globus` qui généralise la notion de multi-grappe hétérogène [30, 29] ainsi que des briques de transfert à haut débit (`NFSp/GXFER`). Ils nécessitent aussi la possibilité d'utiliser un allocateur qui autorise le mode synchrone. L'approche multi-grappe pose bien sûr le problème de la tolérance aux pannes. Dans ce domaine, les solutions sont le plus souvent recherchées au niveau applicatif, mais elles remettent en cause la description statique du parallélisme d'une application comme `CHARMm`. Les approches menées dans le domaines des plates formes de calcul global à grande échelle comme `MPICH-V` [31] permettent d'envisager des solutions au niveau intergiciel.

5.3.3 CEA Orchidée

Contexte scientifique et enjeux Orchidée [68] est un code de climatologie développé conjointement au LMD, Laboratoire de Météorologie Dynamique du CNRS, et au CEA-DSM-LSCE, Laboratoire des Sciences du Climat et de l'Environnement. Ce code implémente un modèle général de végétation et en simule les échanges avec l'atmosphère. Il permet donc de déterminer l'influence de la végétation dans le système climatique. En général, Orchidée est utilisé "en mode couplé", c'est à dire en interaction avec le code de circulation atmosphérique LMDZ et le code d'océan OPA. Néanmoins, Orchidée implémente un modèle de végétation trop général pour représenter correctement l'influence de l'homme sur la végétation, en particulier en ce qui concerne les cultures qui représentent une part importante de le surface au sol en Europe.

Très brièvement, le domaine de calcul est découpé en mailles et sur chaque maille cohabitent un ou plusieurs modèles de végétation. Pour une approche "grossière" le modèle de végétation n'est qu'une fonction qui effectue un bilan très simplifié des échanges. Pour une approche plus "fine" il sera parfois nécessaire de sous-traiter le calcul à un autre code qui modélise de façon précise l'apport d'un certain type de végétation ou d'activités humaines pour cette parcelle.

Ces modèles locaux peuvent être eux aussi très coûteux en temps calcul ou mémoire et bien sûr des informations doivent circuler entre les applications. Il n'est pas exclus d'utiliser, à [long] terme, jusqu'à une dizaine de codes différents et très hétérogènes de par leur provenance, leur nature (parallèles, ...) et d'une façon générale de par les langages et techniques de programmation utilisés. Enfin d'autres contraintes doivent être introduites dans le modèle actuel, par exemple le "routage de l'eau", qui impliqueront de nombreuses communications entre tous les modules qui s'exécutent simultanément.

Du point de vue modélisation informatique, nous nous intéressons à un problème où le graphe logique de l'application n'est pas connu *a priori*. Ainsi, il n'est pas possible de prévoir le nombre de codes qui seront impliqués dans le cours de la simulation ni bien sûr le patron des communications entre les différents composants. Nous avons aussi à traîter plusieurs niveaux de parallélisme, typiquement "intra" et "inter" applications.

Actuellement, l'étude porte sur trois applications, Orchidée qui est le code "maître" et deux modèles locaux :

- le code STICS [44], développé par l'INRA, Institut National de la Recherche Agricole, qui modèlise la croissance de culture des champs de blé ou de maïs pour les surfaces cultivées ;
- le code PASIM [21] qui implémente un modèle pour les prairies incluant les interactions avec les herbivores. Ce code est en cours de reécriture.

Le couplage actuel s'effectue par échange de fichiers à la fin de chaque simulation. Cette méthode est très contraignante et fait perdre beaucoup d'efficacité au couplage. L'idéal serait d'obtenir un couplage fort, à chaque itération d'un pas de

temps et sans échange de fichiers. L'objectif est d'utiliser une technologie modulaire permettant à terme de coupler un grand nombre de modèles spécifiques implémentés par des applications parallèles.

L'objectif est donc de tester une approche "conception orientée grille" pour une plate-forme logicielle modélisant la végétation dans le système climatique. Il est clair que la validation de ce type d'approche doit se faire sur une plate forme matérielle qui permette de simuler des conditions extrêmes en terme de distribution et de dynamicité tout en offrant une large panoplie d'outils susceptible de la rendre "performante" sans intervention lourde de l'utilisateur.

Travaux dans le cadre d'`e-Toile` Dans un premier temps, pour encapsuler les codes sans restructuration profonde et utiliser un modèle simple à mettre en œuvre, nous avons envisagé l'utilisation de `Corba` et si possible un modèle de composant type `OpenCCM` développé dans le cadre du consortium `ObjectWeb` [20]. Une autre approche possible était celle proposée dans `ProActive` [23] utilisant des objets distribués et `Java` toujours dans le cadre d'`ObjectWeb`.

L'idée est ensuite de faire fonctionner l'application distribuée `Corba`, sur la plate-forme `e-Toile` en utilisant la couche de compatibilité `COG` [76].

L'analyse de la grillification de cette application a beaucoup nourri les reflexions présentées plus haut. Notamment l'impact sur les codes utilisés s'est révélé assez profond. Ainsi le code `Pasim` s'est-il révélé impossible à utiliser en l'état et sa reécriture complète a du être envisagée. Elle est en cours de réalisation dans le cadre d'un autre projet du `LSCE`

Des modifications en profondeur du code `STICS` ont été nécessaires pour rendre le code apte au couplage : le code a été construit pour ne traiter qu'un seul point géographique alors qu'il est nécessaire de travailler sur des centaines de points simultanément. Ce travail a également permis de produire une version paralléle du code `STICS`. La seconde phase du travail a été l'encapsulation `Corba`. Le code client est `Orchidée`, il va, à intervalles réguliers, interroger les différents modèles de végétation traités par `STICS` (actuellement le blé et le maïs). Le code `STICS` joue ici le rôle du serveur, et on exécute un serveur par type de végétations.

L'ensemble de l'application `Corba` est actuellement en cours de portage sur `e-Toile`, le code `PASIM` doit être intégré au système dès que sa reécriture - incluant les spécifications pour l'encapsulation `Corba`- sera terminée.

5.3.4 EDF, application `MODERATO`

`MODERATO` est un code de modélisation pour le contrôle non destructif par gammagraphie. L'arrêté d'exploitation du 22/10/1999, relatif à la surveillance du circuit primaire principal et des circuits secondaires principaux des réacteurs nucléaires à eau sous pression stipule que les `END` (examens non destructifs) employés en exploitation doivent faire l'objet d'une qualification ; cette qualification consiste à démontrer que l'application de la procédure d'examen correspond bien aux exigences de l'exploitant, une application étant définie comme une méthode

d'END mise en œuvre selon une certaine technique sur un composant particulier ; il y a environ 200 applications de ce type à qualifier à EDF (exemple : contrôle d'une soudure bien répertoriée). L'argumentaire théorique étant insuffisant pour constituer un dossier de qualification, il faudrait pouvoir procéder par maquettage, ce qui est coûteux et lent : on a donc recours à la modélisation nettement moins coûteuse et beaucoup plus rapide. De plus, cette modélisation va également permettre l'optimisation des configurations de contrôle, c'est-à-dire par exemple analyser la dégradation de l'information en fonction de la position du défaut, analyser la sensibilité au désalignement du tir de photons en fonction de la forme du défaut, optimiser la position de la source pour limiter les fausses alarmes, déterminer le nombre de tirs minimum nécessaire afin de limiter les recouvrements (limiter le coût et le temps d'expérimentation).

D'un point de vue modélisation, la chaîne de contrôle par gammagraphie est constituée de trois composants majeurs : la source (Ir ou Co), la pièce (géométrie complexe, grande épaisseur) qui subit le bombardement de photons et le film radio, avec filtre et écrans, qui encaisse les électrons engendrés par l'impact des photons, ce qui se traduit par un noircissement du film.

Le code MODERATO [82] calcule le transport photonique à travers la pièce à contrôler par une approche Monte-Carlo (MC). Un deuxième module de calcul MC traite le transport photonique et électronique à travers une cassette de film. Les temps de calcul par la méthode MC étant prohibitifs, le code fait appel à deux techniques d'accélération :

Le transport photonique à travers la pièce peut être décomposé en un rayonnement direct (photons n'ayant subi aucune collision dans la pièce et arrivant donc en ligne droite sur le film) et un rayonnement diffusé (photons ayant subi une ou plusieurs collisions avant d'atteindre le film). Seul le rayonnement diffusé est calculé avec la méthode MC, le rayonnement direct pouvant être obtenu facilement par une méthode de rayon rapide. Le calcul MC du diffusé est effectué avec un nombre réduit de photons, le résultat est ensuite extrapolé et combiné avec le résultat du rayonnement direct obtenu par une méthode rayon (mode 0). Le transport photonique et électronique dans la cassette de film est simulé au préalable afin d'obtenir une caractérisation de la réponse du film en fonction de l'énergie et de l'angle d'incidence des photons arrivant sur la cassette. Cette réponse est ensuite décrite de façon approchée par deux polynômes et une fonction de saturation (mode 1).

Portage sur e-Toile L'application MODERATO telle qu'elle a été écrite est mono processeur et tourne sous Windows ; pour traiter 1*109 photons il faut environ 17 heures sur 1 seul processeur Xeon 2.3 Ghz.

Une partie de l'application (celle qui ne concerne pas l'interaction avec l'utilisateur qui doit demeurer sous Windows) a été portée sous Unix/Linux (HP, Linux/Solaris).

Pour une configuration typique, le code MC passe entre 90% et 95% du temps dans le calcul du transport photonique. L'encaissement des photons dans la cas-

sette prend entre 5 et 10% du temps de calcul. Cette observation suggère qu'une première approche pour le portage sur la grille e-Toile doit être de parallèliser les calculs des trajets des photons, puisque ces trajets sont par ailleurs complètement décorrélés ; l'encaissement des résultats sera traité séparément.

La linéarité de l'accélération est prouvée pour une simulation en mode 1 (film déjà caractérisé), ce qui laisse envisager une bonne utilisation future en grille. En revanche, en mode 0, l'encaissement sur le film ralentit l'application, et le *speed up* stagne au-delà de 4 processeurs ; néanmoins il est linéaire jusqu'à ce point critique. L'encaissement doit donc être parallélisé, et une décomposition du film s'impose. Ces développements vont être réalisés.

Néanmoins d'ores et déjà, la grille e-Toile a permis de faire tourner MODERATO en mode 1, et d'en mesurer l'intérêt pour les utilisateurs. A titre d'exemple, huit études ont été réalisées sur 2*109 photons chacune et lancées sur 90 processeurs de la grille e-Toile (Site CEA Saclay, ENS Lyon, et EDF Clamart) : les résultats ont été obtenus en moins de 24 h (alors qu'en monoprocesseur, il aurait fallu 2*8*17=272 heures, soit plus de 11 jours). Indépendamment du gain considérable en temps de restitution des résultats, le fait de faire exécuter MODERATO sur une grille avec un grand nombre de photons, a permis de résoudre des bogues de l'application non encore décelées, mais aussi de valider certaines hypothèses physiques qui avaient été faites. Le fait de repousser considérablement la limite du temps calcul utilisable, permet d'envisager à la fois plus de finesse dans la modélisation, mais aussi de lancer plus d'études permettant d'optimiser encore plus les opérations physiques de contrôle non destructif *in situ*, ce qui représente un grand intérêt économique et de gain de temps.

Ces premiers résultats sont très encourageants, et l'effort sur MODERATO va être poursuivi ; il devra prendre en compte les exigences de l'utilisateur final, notamment celle de pouvoir disposer d'une interface graphique en mode interactif permettant d'effectuer des vérifications en cours d'étude, indispensable sur des études longues (plusieurs jours) et pour vérifier le positionnement du film. Il faudra également fournir une interface de soumission des études la plus légère possible pour l'utilisateur.

5.3.5 EDF, application Dymoka

Application de simulation en dynamique moléculaire, Dymoka[85] est un code générique de calcul de grandeurs physiques à l'échelle atomique permettant la modélisation de l'effet des défauts atomiques d'un métal sur sa déformation. Différents types de simulation de matériaux à partir d'un potentiel d'interaction atomique tabulé et dépendant du matériau simulé, peuvent être effectués. Un exemple d'application du code est l'étude des défauts d'irradiation neutronique dans le fer et ses alliages (alliage fer-cuivre). Les potentiels empiriques utilisés sont de type EAM (*Embedded Atom Method*) ; ils sont de courte portée et comprennent une partie

n-corps. Les méthodes employées pour cette simulation sont la dynamique moléculaire et la méthode de Monte Carlo avec l'algorithme de Metropolis.

Dymoka est issu du code CDCMD de l'université du Connecticut, qui a été adapté aux besoins d'EDF. Il existe une version séquentielle et une version parallèlisée. La parallèlisation de Dymoka qui a été faite, est de type échange de messages avec MPI afin d'assurer la portabilité sur le plus grand nombre de machines parallèles et vectorielles.

Portage sur e-Toile Dans la version parallèle, une grosse simulation impliquant un million d'atomes nécessite quelques centaines d'heures CPU (15 heures *elapse* sur 32 processeurs Compaq). L'intérêt de porter la version parallèle sur une grille est de pouvoir traiter des simulations impliquant un ordre de grandeur de 10-100 millions d'atomes (temps d'exécution proportionnel au nombre d'atomes). Notons que la version séquentielle pourrait aussi bénéficier de la grille : on a alors une étude paramétrique avec plusieurs lancements simultanés. Un autre intérêt de la grille pour Dymoka est de pouvoir bénéficier d'une grosse capacité de stockage. En effet, ces simulations produisent de très grandes quantités de données : une simulation d'irradiation neutronique avec 1 à 2 millions d'atomes génére environ 300 Moctets de données sous forme de fichiers binaires.

L'application Dymoka, dans sa version parallèle a été portée et testée sur la grille e-Toile en utilisant MPICH-G2. Elle est maintenant prête à tourner sur la dernière version du middleware e-Toile en utilisant MPICH-Madeleine. Des tests de transfert à haut débit (GXFER et NFSp), et de stockage temporaire dans la grille (IBP) sont également envisagés.

5.3.6 EDF, application ROCK

L'application ROCK [41] est un outil de calcul pour la neutronique. C'est un exemple typique d'application non portable "raisonnablement" sur une grille, c'est-à- dire dont l'adaptation pour exécution sur la grille e-Toile impliquait des remaniements tels que nous avons décidé de ne pas continuer le portage. L'intérêt de la démarche était de prendre une classe d'application distribuée assez ambitieuse et exigeante, dédiée à l'exécution sur grappe (*cluster*), et d'essayer de la transformer de manière à tirer parti autant que possible des ressources plus nombreuses fédérées par la grille.

A l'usage, le portage a rencontré des difficultés d'ordre technique qui, malgré la qualité de l'application, ont considérablement freiné sa progression au point de rendre son achèvement hors de portée. A l'origine, l'exercice consistait à toucher aussi peu que possible à l'application, pour sélectivement débrancher ceux de ses services que la grille devait désormais, à sa manière, prendre en charge, tout en rajoutant ceux qu'elle rendait nécessaires.

Les principaux problèmes rencontrés ont concerné la couche de communication réseau, nativement fondée sur l'utilisation directe de *sockets* UNIX exploitées

par un courtier de message (intergiciel) développé en interne et particulier à ROCK, et qui communiquait indifféremment *via* le système de fichiers et le réseau. La substitution des couches basses du courtier natif par celui du lot 1 de la grille e-toile, Globus_io, le découplage des accès aux fichiers et aux *sockets*, la réécriture de la logique de connexion et de transport en s'interdisant de passer en multi-tâche ont contribué à augmenter de beaucoup les délais, mettant même à jour des cas d'intégration où une réécriture des *pipes* UNIX (primitives systême de très bas niveau) à base de *handle* Globus semblait nécessaire, bien que la raison s'y opposât.

D'autres difficultés étaient à lever, notamment la négociation d'une classe de service spécifique qui permette de se substituer à l'ordonnanceur de la grille (les graphes paramétrés de ROCK n'étant gérables par aucun ordonnanceur de grille connu), la création d'un système d'information applicatif rendu nécessaire à l'application dans un contexte de grille, le déploiement dynamique de l'application, des données en entrée et le rapatriement des résultats sur les calculateurs retenus, la gestion de la sécurité étaient toutes des difficultés qui se rajoutaient à celle qui a mobilisé l'essentiel de nos efforts (l'intergiciel), sans pour autant être insurmontables prises isolément. Par ailleurs, du point de vue du périmètre de prise en charge, beaucoup d'entre elles se situaient à la frontière entre grille et applications.

Bien que le portage ait dû être arrêté faute de temps, il s'est révèlé riche d'enseignements à plusieurs niveaux, au point que le début d'une démarche raisonnée de portage d'applications sur grille en ait émergé. La préparation de l'application, préalablement à son portage, devrait aller systématiquement de pair avec la réalisation d'une maquette validant l'intégration avec les composants de la grille, de telle sorte que ces deux étapes nous amènent à mi-chemin du portage. Ce dernier n'est jamais facile, *a fortiori* dans les cas défavorables, dans lesquels pourtant nous ne nous étions pas placés, où l'application elle-même souffre de carences d'architecture ou dont le concepteur originel n'est pas consultable. Ensuite, toutes les applications ne se prêtent pas aisément au portage sur grille, même quand elles pourraient en tirer un bénéfice certain. Enfin, le dimensionnement objectif des efforts de réécriture et de remplacement des services natifs, bien qu'impossible à quantifier avec précision, est néanmoins le seul guide qui permette, une fois les verrous techniques levés, de déterminer l'ampleur des développements nécessaires, ampleur qui *in fine* constitue un élément essentiel de la faisabilité du portage.

5.3.7 IBCP le contexte particulier de la bioinformatique

L'IBCP a pour domaine de recherche en bioinformatique l'analyse de séquence de protéine [35, 37, 49]. Parmi les algorithmes les plus couramment utilisés dans ce domaine par la communauté internationale [77], il est possible de distinguer une classification de ceux-ci suivant leurs besoins en puissance de calcul, leur type de calcul (étude paramétrique, codes parallèles par échanges de messages) et leurs besoins en espace de stockage. Ainsi, on distinguera les applications qui :

– ont des temps de calcul inférieurs à la dizaine de minutes, et les autres ;

– ont un mode d'exécution qui se prête à une parallélisation par les données ("paramétriques") et ceux qui nécessite des échanges ("parallèles") ;
– utilisent en entrée des données dont la taille est de quelques Ko, inférieure au Mo et les autres.

Les problèmes bioinformatiques auquels peut répondre plus particulièrement le concept de grille informatiques sont les cas où les analyses bioinformatiques sont génératrices de "gros" processus qui demandent plusieurs dizaines d'heures pour s'exécuter, ou qui recapitulent de grands nombres de requêtes par de multiples utilisateurs commme dans le cas d'un portail [49, 36]. Dans le cadre de ces "grosses" exécutions, si l'algorithme se prête à une modélisation par échanges de messages, une solution peut être trouvée avec les bibliothèques adéquates. Mais le plus souvent, ces applications ont un temps d'exécution inférieur à la dizaine de minutes, leur complexité réside dans un très grand nombre de répétitions nécessitant une confrontation des données d'entrée à un jeu de données de référence de grande taille qui doit être considéré dans son ensemble (banque de séquences par exemple).

Du fait du statut actuel des applications bioinformatiques les plus couramment utilisées (diffusion en "binaire", code source inaccessible, manque de ressources humaines pour reprise et adaptation du code, validation du code modifié avant acceptation par la communauté), il est préférable de considérer l'application à "grillifier" comme une boîte noire [36]. Le travail d'adaptation porte alors sur les données. Des adaptations du code source, lorsqu'il est disponible seront également envisagées afin de proposer à la communauté de nouveaux modes de développement en rapport avec les architecture de grille. Ces deux méthodologies doivent conduire à des ensembles de recommandations à destination de la communauté de développeur dans le domaine de la bioinformatique.

L'évaluation des gains doit porter en premier lieu sur des comparaisons des performances brutes dans les deux contextes : local et distribué. Mais la grille doit avant tout apporter à la bioinformatique, la possibilité d'effectuer des analyses à grande échelle nécessaires à l'étude des données brutes issues des grands programmes biologiques (génomes complets, détermination de structure...).

Les gains les plus probants en bioinformatique sur la grille viendront certainement des progrès en terme de :
– gestion et maintenance (mise à jour) des données biologiques non figées, i.e. données "modifiables" sur la grille ;
– mise à disposition de ces données modifiables auprès des algorithmes, c'est-a-dire leur accès par les applications à leur demande et avec les meilleures performances ;
– sécurité de ces données, sensibles dans la plupart des cas.

Clairement, des contraintes fortes de sécurité, d'integrité et de gestion des accès aux données pénalisent leur distribution dans le cas d'application sensibles. A cela viennent s'ajouter la contrainte des mises à jour parfois quotidiennes de ces données et de leur diffusion dans leur ensemble en direction des applications.

Les applications de l'IBCP Le projet GriPPS [88] a rejoint e-Toile en mars 2002, ses réalisations ont pour objectif d'étudier les contraintes spécifiques des applications biologiques dans un contexte de grille de ressources distribuées sur un réseau haut débit. Les études portent sur une application développée à l'IBCP, dont le code source est maîtrisé. Les résultats obtenus permettront de tirer des conclusions non seulement sur l'application étudiée, mais également sur des problématiques récurrentes en bioinformatique liées aux algorithmes bioinformatiques et aux bases de données biologiques [36]. De la dimension grille viendront certainement des progrès en terme de

- codes non maintenus ou non-accessibles ;
- données sous forme de fichiers-texte "à plat" de grandes tailles ;
- multiples formats bioinformatiques incompatibles entre eux ;
- mise à jour et synchronisation des données, sécurité et integrité, gestion fine des accès...

GriPPS s'intéresse à l'adaptation d'une application de recherche de motifs protéiques dans un ensemble de séquences de protéines de référence. Les données en entrée peuvent atteindre plusieurs centaines de Mo, voire le Go, et les temps d'exécution plusieurs heures. La distribution d'une exécution est réalisée par la scission des données en sous-ensembles de taille adaptée. Ceux-ci sont alors répartis sur les nœuds disponibles avec l'application, et les résultats demandent ensuite à être synchronisés.

les résultats sur e-Toile Les résultats obtenus par GriPPS portent moins sur des critères de performances pures que sur des critères de faisabilité et d'adéquation avec les contraintes specifiques liées aux applications du domaine scientifique qu'est la bioinformatique.

Ainsi, nous avons démontré la réalisation de la grillification de l'application PattInProt [19] dans le contexte "pur Globus v2.2" de la version initiale, puis sur les composants spécifiques d'e-Toile [79].

L'adaptation à la grille a mis en évidence un fort travail au niveau de l'application pour proposer les fonctions de haut niveau telles que l'allocation des ressources disponibles, l'ordonnancement, la gestion et le transfert des données. Ces réalisation ont été faites sous la forme de "scripts" implémentant de manière "basique" les fonctionnalités de niveau applicatif d'un intergiciel. Notons que ce type de réalisation est très peu compatible avec la communauté d'utilisateurs visée : biologistes et bioinformaticiens. Ces études ont mis en évidence la part non négligeable imputable à la distribution des données, dans les performances globales.

L'application PattInProt a ensuite été portée sur les composants développés par les partenaires d'e-Toile. Ces composants de haut-niveau ont simplifié le portage en prenant en charge les fonctionnalités telles que l'allocation, le chargement ou le transfert des données. L'adaptation de l'application peut, dans ce cas, être assimilée à la construction du "script" adéquat dans le langage de description des travaux d'e-Toile (LDT), utilisant les fonctionnalités disponibles sur la

grille. Cela permettra une adoption plus réaliste par les chercheurs de la communauté bioinformatique. Des composants en particulier comme `NFSp/Gxfer` ou `Qosinus` ont montré leur adéquation particulière avec nos applications bioinformatiques pour le traitement des données biologiques mises en œuvre, sous réserve de l'approfondissement de leur intégration dans l'intergiciel.

5.3.8 PR*i*SM Parcours arborescents en Optimisation Combinatoire

L'équipe `OPALE` du PR*i*SM s'est investie dans ce projet, outre l'étude et la réalisation de l'allocateur de ressources décrites par ailleurs dans ce document, au déploiement de la bibliothèque `BOB++` sur la grille `e-Toile` afin d'en effectuer l'évaluation. La bibliothèque `BOB++` est une plate-forme d'aide au développement d'applications de type *Branch-and-X* (*X* désigne soit *Bound*, *Cut*, *Price* ou encore *Infer*). Ces applications sont pour la plupart issues de l'optimisation combinatoire (ordonnancement de tâches, allocations de ressources, etc...) ou de l'intelligence artificielle (jeux solitaires ou à deux joueurs). Pour les résoudre exactement, les méthodes communément utilisées sont celles fondées sur les parcours d'arbres explorant l'espace des solutions. `BOB++` permet d'une part de faciliter le développement de telles applications et d'autre part d'obtenir à partir d'un seul et même code applicatif plusieurs exécutables pour les machines allant d'une seule machine comme un simple PC aux machines de type grappe (*cluster*) ou grille.

Il est à noter que le travail de parallélisation de la bibliothèque `BOB++` comme l'étude des modèles de programmation théoriques et le choix des plate-formes d'exécution *multithread* (`Athapascan`), ainsi que le développement des applications en optimisation combinatoire comme le problème d'affectation quadratique (QAP, affectation de n usines sur n sites afin de minimiser le coût de transport de marchandises entre les usines) [67] et le problème du voyageur de commerce, sont réalisés dans le projet `ACI-GRID DOC-G` (Défis en Optimisation Combinatoire sur Grille de calcul).

Les travaux de parallélisation de la décennie [50, 62] ont montré que les parcours arborescents de type *Branch-and-X* ont des efficacités remarquables sur des machines parallèles allant du `SMP` (mémoire partagée) aux grappes de `PC` (mémoire distribuée). Un travail récent d'une équipe américaine [27] a même rapporté la résolution de l'instance de données de taille 30 de `Nugent` (`Nug30`, 30 usines pour 30 sites), alors considérée comme très difficile du problème de l'affectation quadratique. Cette résolution a été menée sur une grille de calcul composée de 2510 machines, nécessité 7 jours de temps de calcul et développé une arborescence de plus de 12 milliards de sommets. D'ailleurs, ce résultat n'a toujours pas été égalé ou reproduit depuis.

Les parallélisations proposées, comme celles dans `BOB++`, sont principalement fondées la technique de décomposition des arborescences parcourues lors des recherches dans l'espace des solutions. Cette décomposition consiste à affecter une

branche de l'arborescence (une sous-arborescence) à un processeur indépendamment des autres branches. Si cette technique a l'inconvénient d'utiliser plus de mémoire, puisqu'elle nécessite en général la recopie systématique de tout le contexte des sommets de l'arborescence, elle présente aussi l'avantage d'autoriser une **granularité variable de calcul** en ajustant la taille des sous-arborescences fournies aux différents processeurs. De plus, les caractères irréguliers et imprévisibles des arborescences à cause de la génération au fur et à mesure des sommets à évaluer et à explorer font que ces problèmes sont classés comme **irréguliers**. Seul un **ordonnancement/équilibrage de charge dynamique** au cours du calcul permet une exploitation efficace des machines parallèles.

Clairement, les applications développées avec la bibliothèque BOB++ sont de type **CPU intensive** et **network intensive**. Les résultats expérimentaux sur la grille d'`e-Toile` ci-après confirment bien ces deux caractéristiques.

Le choix des tests de bibliothèque `BOB++` sur la grille d'`e-Toile`, s'est donc naturellement porté sur le problème du `QAP` et sur la plate-forme d'exécution `Athapascan`, issus du projet `ACI-GRID DOC-G`. Le choix du problème `QAP` s'explique par la disponibilité d'une bibliothèque connue d'instances de données, la `QAPLIB` et elle est accessible par le WEB [46]. Ainsi, nous pouvons étalonner nos résultats par rapport à ceux des autres équipes de recherche. Quant au choix de la plate-forme d'exécution `Athapascan`, il est dicté par le fait que l' **ordonnancement dynamique par vol de travail** (*work stealing*) est intégré dans le modèle de programmation d'Athapascan. Par ailleurs, Athapascan peut tout à fait être portée sur les bibliothèques de communication comme `MOME` (mémoire partagée virtuelle) ou `Madeleine` (multi-protocoles de communication) du projet `e-Toile`.

Le principe du déploiement des applications de la bibliothèque `BOB++` sur la grille d'`e-Toile` consiste à générer un binaire de l'application que l'on envoie sur l'ensemble des serveurs de la grille avec les fichiers de données nécessaires. Notre application de test ne nécessite pas l'usage de bibliothèques commerciales, ni de couplage de codes. Par ailleurs, les fichiers de données et de résultats de petites tailles. Pour une instance de problèmes, le fichier de données est de l'ordre *a priori* de quelques `Ko`. Il en est de même pour les fichiers de résultats qui sont essentiellement constitués de statistiques permettant une analyse du comportement du programme grillifié. L'application est peu exigente en terme de transfert de fichiers.

En revanche, les applications en optimisation combinatoire comme le `QAP`, sont extrêmement consommatrices de temps `CPU` et les méthodes de type *Branch-and-X* ont un comportement irrégulier et dynamique dans le temps. Par conséquent, le calcul exploite au maximum tous les processeurs disponibles de la grille `e-Toile` alors que l'ordonnancement dynamique/équilibrage de charges quant à lui prendra toute la bande passante offerte par le réseau haut débit `VTHD` pour le transfert des sommets de l'arborescence entre les processeurs. Nous verrons sur les graphiques ci-après, issus des captures d'images de `GANGLIA`, la confirmation de

ces deux caractéristiques de notre application grillifiée.

```
<Application universe="mpi" synchronization="synch"
                            service-class="1">

 <Job jobname="Qap-qap.ior.8131" nb-process="6"
      estimated-time="10000">
  <InputFiles protocol="gridftp">
     <File path="/home/lecun/e-TQAP/DATA/nug18s.dat"/>
     <File path="/home/lecun/e-TQAP/.a1rc"/>
     <File path="/home/lecun/e-TQAP/qap"/>
     <File path=
      "/home/lecun/e-TQAP/realsh.cocteau.prism.uvsq.fr.sh"/>
     <File path="/home/lecun/e-TQAP/qap.ior.8131"/>
  </InputFiles>
  <OutputFiles>
     <File path=
      "/home/lecun/e-TQAP/qapresult.cocteau.prism.uvsq.fr.txt"/>
  </OutputFiles>
  <Requirements>
     (cluster_hostname="cocteau.prism.uvsq.fr")
  </Requirements>
  <Executable binary="realsh.cocteau.prism.uvsq.fr.sh" >
  </Executable>
 </Job>

 <Job jobname="Qap-qap.ior.8131" nb-process="10"
      estimated-time="10000" >
  <InputFiles protocol="gridftp">
    <File path="/home/lecun/e-TQAP/DATA/nug18s.dat"/>
    <File path="/home/lecun/e-TQAP/.a1rc"/>
    <File path="/home/lecun/e-TQAP/qap"/>
    <File path=
     "/home/lecun/e-TQAP/realsh.ursa-major.saclay.cea.fr.sh"/>
    <File path="/home/lecun/e-TQAP/qap.ior.8131"/>
  </InputFiles>
  <OutputFiles>
    <File path=
     "/home/lecun/e-TQAP/qapresult.ursa-major.saclay.cea.fr.txt"/>
  </OutputFiles>
  <Requirements>
     (cluster_hostname="ursa-major.saclay.cea.fr")
  </Requirements>
  <Executable binary="realsh.ursa-major.saclay.cea.fr.sh" >
  </Executable>
 </Job>

</Application>
```

FIG. 5.5: *script* XML de lancement de l'application QAP sur plusieurs sites de la grille e-Toile

Sur le lancement de notre application à partir de l'intergiciel e-Toile (Guide-allocateur-chargeur), cela n'a posé aucune difficulté en passant de Globus aux fichiers XML de l'intergiciel e-Toile. L'exemple du fichier XML de la figure 5.5 a permis le lancement de notre application sur deux grappes (PRiSM Versailles et CEA Saclay) de la grille d'e-Toile.

Toutefois, la spécification (voir les documents fournis par le projet concernant ce sujet) prévue dans l'intergiciel e-Toile est plus riche que celle de Globus, autorisant notamment la possibilité de spécifier différentes classes d'applications suivant leurs comportements (synchrone ou pas, ayant ou non une estimation de la durée totale de l'application). Cette spécification permet à l'intergiciel d'optimiser l'allocation des ressources pour l'exécution des applications. Typiquement, notre application bien qu'elle ait un comportement irrégulier et dynamique à l'exécution, nécessite néanmoins un lancement synchrone du processus principal avec les autres, d'où la nécessité de préciser `synchronization="synch"` après le marqueur `<Application universe...>`.

Sur le plan des tests conduits, nous avons principalement mené deux types :

Tests de fonctionnalité et de faisabilité sur l'intergiciel e-Toile La simplicité dans le lancement de notre application (un seul binaire, un seul fichier de données de petite taille, des fichiers de résultats de petite taille à rapatrier) a permis de mener rapidement ces tests et d'obtenir des résultats conluants sur les instances de petite taille du problème QAP.

Les seules *difficultés* rencontrées étaient en général dues au mauvais usage du langage de soumission de jobs en XML au début des tests.

Tests de performance de la grille e-Toile Cette série de tests, menés sur des instances de données de taille moyenne à grande du problème QAP, a pour but de valider ou non les performances brutes de la grille physique. La question ici est d'essayer de savoir si l'on était capable ou non d'exploiter, avec une application grillifiée et ayant des bonnes performances sur les machines parallèles et des grappes, une grille de calcul aussi bien qu'une machine parallèle ou une grappe. La réponse n'est pas *a priori* évidente tant les matériels (processeurs, mémoires, cartes de réseaux, etc) et les systèmes mis en jeu dans une grille de calcul sont **hétérogènes**.

Nous avons donc comparé les résultats obtenus d'une part sur une seule grappe (machines homogènes), et d'autre part sur la grille e-Toile. Cela pose par ailleurs des problèmes difficiles et ouverts sur les mesures de performance à adopter. Nous y avons apporté notre contribution qui est illustrée par les résultats expérimentaux suivants.

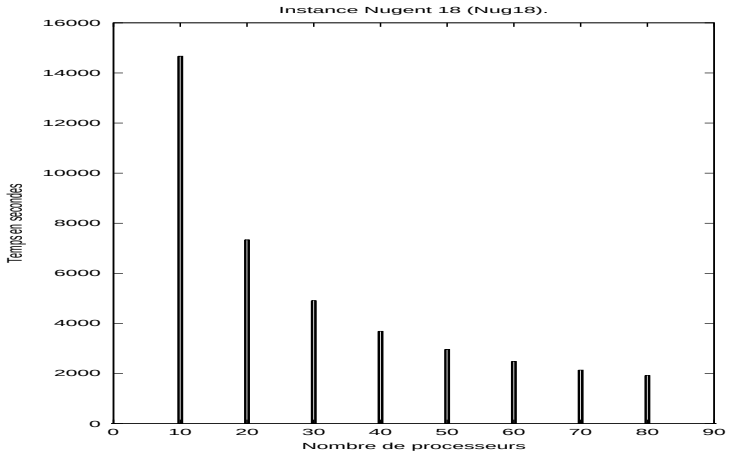

FIG. 5.6: Résultats du `Nug18` sur l'icluster d'`ID-IMAG`, de 10 à 80 processeurs.

La figure 5.6 montre les résultats que nous avons obtenus sur la grappe icluster de `ID-IMAG` à Grenoble pour l'instance `Nug18` (18 usines et 18 sites). Pour ces tests, nous avons eu l'accès exclusif à toute la grappe. Les temps sont ceux des exécutions de 10 processeurs en 10 processeurs. De 10 processeurs à 80 processeurs, nous avons une **accélération de 7,6** avec une **efficacité à 95%**. Notre application est donc bien efficace sur une grappe homogène de machines.

Or, sur la grille `e-Toile`, pour l'instance de taille supérieure `Nug20` (20 usines et 20 sites), nous avons obtenu un temps $T_{20} = 1648$ secondes sur la grappe du `CEA` composée de 20 `AMD MP 1800+` et un temps $T_{84} = 499$ secondes sur **quatre grappes** (`CEA` Saclay 10 `bi-AMD MP 1800+`, `SUN ENS`-lyon 15 x `bi-PIII` 1.4`Ghz`, `PRiSM` Versailles 7 x `bi-Xeon` 2.4`Ghz`, `EDF` Clamart 10 x `bi-Xeon` 2.2 `Ghz`) totalisant 84 processeurs. Ce qui montre une **accélération brute à** $T_{20}/T_{84} = 3,1$.

Nous voyons ici que nous sommes confrontés à un problème sur les mesures de performance classiques. Notre idée d'une mesure de performance, grossière certes mais réflétant néanmoins la réalité, est de ramener la puissance des 84 processeurs à un seul processeur virtuel (car il n'existe pas) de puissance équivalente. Par conséquent, en considérant le critère des fréquences des différentes familles de processeurs mises en jeu dans la grille : 20 x 1.8 `Ghz` (`AMD`) + 30 x 1.4 `Ghz` (`PIII`) + 14 x 2.4 `Ghz` (`Xeon`) + 20 x 2.2 `Ghz` (`Xeon`), nous obtenons un processeur équivalent à 155.6 `Ghz`. Soit 86.44 processeurs à 1.8 `Ghz` et donc comparé à 20 processeurs (`AMD`) à 1.8 `Ghz`, nous aurions eu une accélération maximale théorique à 4.32 pour 3.1 observée. Soit une efficacité de 71.76% [51, 89]. Comparée au 95% observée précédemment sur une grappe homogène, cette perte d'efficacité est due principalement à deux facteurs :

- le surcoût de l'utilisation d'une grille (communication sur un réseau métropolitain et allocation des ressources sur des systèmes hétérogènes) ;
- le surcoût de recherche de l'application.

Toutefois, cette efficacité est une borne inférieure, car l'accéleration théorique surestime les processeurs `AMD` et `PIII` par rapport au `Xeon`. L'efficacité réelle est certainement supérieure à 71.76%.

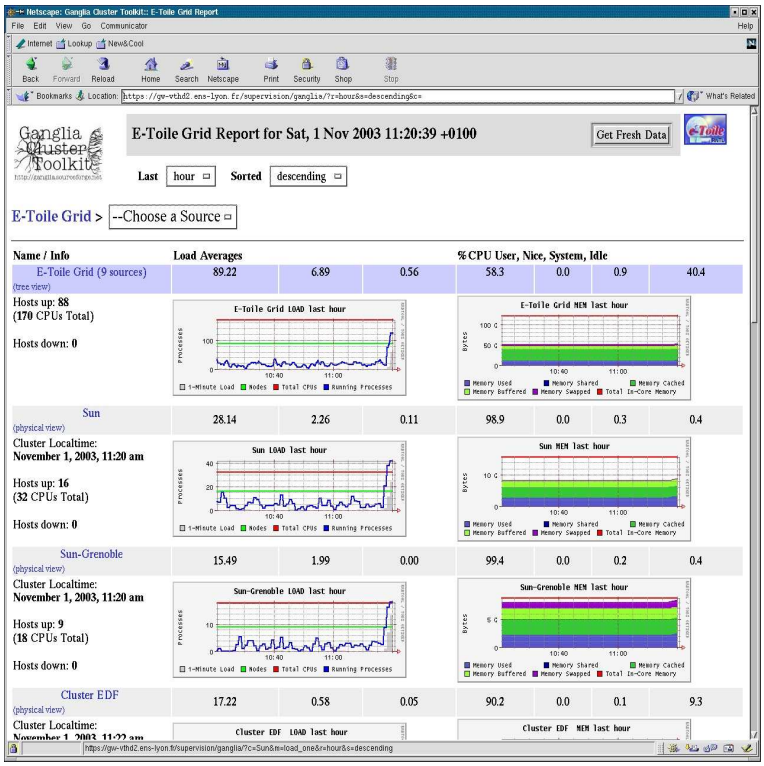

FIG. 5.7: Montée en charge.

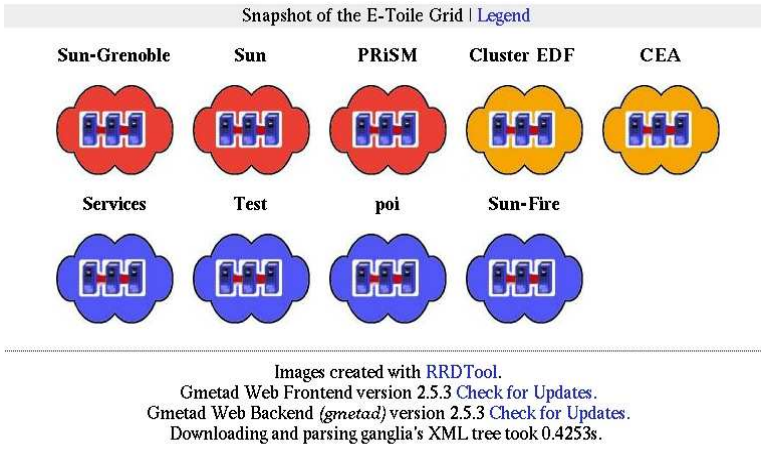

FIG. 5.8: Régime permanent de calcul sur 5 grappes (chargées à presque à 100%).

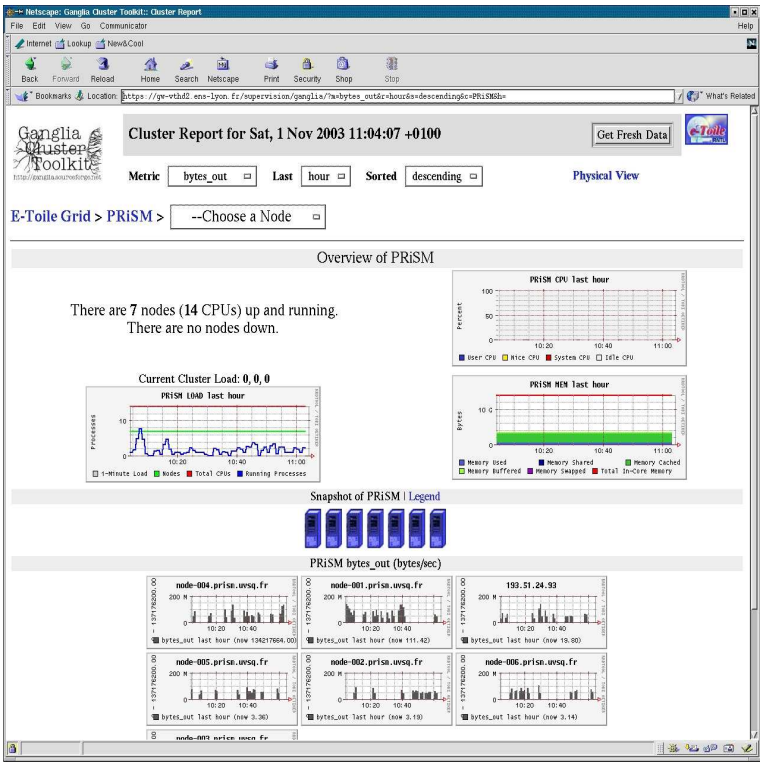

FIG. 5.9: Pics de communication sur la grappe du PR*i*SM.

Des tests ponctuels avec plus de 120 processeurs de la grille `e-Toile` ont également été réalisés avec des efficacités de même qualité. Les figures 5.7 et 5.8 montrent l'état de charge, respectivement la montée en charge et le régime permanent des grappes de la grille `e-Toile` (`CEA`, `EDF`, `Sun`, `ENS`-Lyon et Grenoble, PR*i*SM) en pleine résolution de l'instance du `Nug20`. Les grappes en rouge signifient que la charge `CPU` est de plus de 100%, alors que celles en orange sont entre 75% à 100% de charge `CPU`.

La figure 5.9 montre les pics de communications sortantes (*bytes_out*) lors des ordonnancements dynamiques par vol de travail entre les nœuds de calcul de la grappe. Ces graphiques sont des captures d'images de l'outils de supervision `GANGLIA` [7] installé pour le grille `e-Toile`.

Il est à noter au passage, que les accélérations ne sont plus calculables par rapport au meilleur programme séquentiel, puisque pour les instances de très grande taille, même avec le meilleur programme séquentiel, cela peut prendre des jours ou des mois de calcul, voire tout simplement inaccessible. Nous avons choisi ici de comparer le temps obtenu sur la grille par rapport au temps obtenu sur l'une des grappes homogènes de la grille.

D'autres briques développées dans le cadre de ce projet `e-Toile` permettraient d'avoir un gain dans les performances. Nous pouvons citer `NSFp` et `Gxfer`

pour le transfert de fichiers. Si toutefois notre application n'a que des petits fichiers à transférer, NSFp aurait permis d'avoir une image unique des fichiers sur la grille, évitant ainsi les transferts systématiques de fichiers avant l'exécution vers les machines cibles et le rapatriement des fichiers pour l'analyse à la fin de l'exécution.

Sur le plan des communications, la mémoire partagée virtuelle Mome et la couche de communication multi-protocole Madeleine auraient permis respectivement une meilleure gestion des variables globales dans l'application et une meilleure utilisation des réseaux. Malheureusement, à cause du développement en parallèle de ces composants en même temps que les applications, l'intégration n'a pu se faire dans le temps imparti du projet.

Sur l'aspect de la sécurité, d'une part nos applications ne sont pas aussi sensibles que celles du CEA et d'EDF, et d'autre part, relativement faciles à protéger, car seuls les calculs et les données en mémoire des machines sont à prendre en compte dans une éventuelle politique de sécurité. Les données et les résultats sur fichiers sont de taille relativement faibles.

En revanche, la tolérance aux pannes est un point crucial pour nos applications qui peuvent nécessiter plusieurs semaines, voire des mois de calculs. Or, la probabilité qu'un sous-ensemble de machines tombe en panne ou en maintenance est proche de un lorsque la durée de calcul est longue. Il importe donc que les calculs ne soient pas relancés à partir du début lorsqu'une machine tombe en panne ou est isolée pour une maintenance. Des techniques de checkpointing ou de reprise à chaud pendant la résolution sont très intéressantes. Or, nos applications à base de parcours arborescents peuvent facilement adopter ces techniques. Il suffit de sauvegarder les racines (sommets) des sous-arborescences en cours d'exploration. Ainsi, lorsqu'une machine est arrêtée, il suffit de relancer la recherche à partir de la racine de l'arborescence en quesiton sur une autre machine ou la même machine redémarrée.

Si ces deux points ont été considérés dans le projet e-Toile, le temps imparti au projet n'a pas pu permettre leur mise en œuvre complète dans l'intergiciel e-Toile.

Les résultats obtenus sur une application d'optimisation combinatoire, le problème de l'affectation quadratique, ont permis de montrer la faisabilité d'une exploitation efficace d'une grille telle que la plate-forme e-Toile. C'est en l'occurence, le cas des applications à base de parcours arborescent avec un ordonnancement dynamique des calculs fondé sur le vol de travail.

Il reste cependant encore une marge quant à une efficacité supérieure, nous n'avons pas pu, malgré la puissance offerte par plus de 120 processeurs de la grille d'e-Toile, résoudre les instances de données au-delà de 24 usines et de 24 sites.

Sur le plan applicatif, un prolongement de ce projet de 6 mois nous permettrait d'améliorer la bibliothèque BOB++ pour les grilles de calcul afin d'atteindre des efficacités au delà de 72%, et par la même occasion de résoudre les instances de données de taille supérieure à 24 afin d'approcher, voire de dépasser les meilleurs résultats de la littérature pour le problème du QAP.

5.4 Synthèse des résultats obtenus sur les versions successives de la grille e-Toile

En conclusion, du point de vue des applications, les portages effectués sur la version initiale, ont démontré la validité du concept de grille haute performance :

- du concept de grille tout d'abord, puisque il a été possible expérimentalement de fédérer et de partager les ressources logicielles et matérielles mises à disposition par les partenaires, tout cela en s'appuyant sur un ensemble de fonctionnalités minimales telles celles délivrées par Globus ;
- du concept de grille haute performance ensuite, qui est plus spécifique à e-Toile. Il s'agit de mettre en évidence les composants logiciels stratégiques qui permettent de déployer les applications retenues.

Même si peu de travaux ont été menés, à cette date, sur la version e-Toile, les insuffisances relevés sur la version initiale, montrent bien les besoins résumés dans le tableau 5.10 : les applications ne peuvent être déployées efficacement sans les fonctionnalités (améliorées ou étendues) apportés par les modules spécifiques intégrés dans l'intergiciel e-Toile :

	Atlas (CEA)	Charmm (CEA)	Orchidée (CEA)	Rock (EDF)	Ecoss (EDF)	Dymoca (EDF)	Gripps (IBCP)	BOB++ (PRiSM)
Tamanoir	XX				XX			
Multicast	XX			XX	XX	XX	XX	
Qosinus	XX	XX	XX	XX	XX	XX	XX	XX
IBP	XX	XX		XX	XX	XX	XX	
Gxfer	XX	XX		XX	XX	XX	XX	XX
NFSP	XX	XX		XX	XX	XX	XX	XX
Mome			XX					XX
Madeleine		XX	XX	XX	XX	XX		XX

FIG. 5.10: Les besoins des applications e-Toile en terme de composants de l'intergiciel

Chapitre 6

Conclusions et perspectives

Au terme de ce projet, on constate qu'e-Toile a eu l'ambition de s'attaquer à de nombreux et difficiles verrous et en particulier :

- à la problèmatique de communication très haute performance ;
- à l'exploration de la technologie des réseaux actifs ;
- à la définition d'une architecture d'intergiciel ouverte et extensible permettant à la fois d'intégrer des composants Globus et des solutions et approches nouvelles ;
- à la prise en main de l'outils et de la culture grille par des communautés d'utilisateurs variées.

L'ensemble de ces questions a été étudié et un certain nombre de réponses proposées. Il reste de nombreux points à creuser et surtout, à bien finaliser les études expérimentales. Aujourd'hui l'ensemble des logiciels ont été écrits, validés fonctionnellement, intégrés et déployés. La plate-forme est opérationnelle. Les études de performance peuvent donc pleinement être conduites.

Au plan des services réseaux avancés, e-Toile s'est focalisé sur les aspects performance et qualité de service. Nous avons montré que les services offerts par VTHD étaient intéressants pour les applications de grille, mais le gain réellement obtenu doit encore être exploré de manière plus systématique, en particulier en lien avec les travaux réalisés dans le domaine de l'ordonnancement et de la réservation de ressource. Si l'aspect différenciation de service a été bien étudié, l'aspect transport haute performance demande de nouveaux travaux. D'autres services proposés dans VTHD demandent aussi à être étudiés. Les services VPN de niveau 1 ou 2 ainsi que le service IPV6 apparaissent de plus en plus importants pour la grille car pouvant répondre aux besoins d'extensibilité et de protection. Grâce à l'environnement Tamanoir nous proposons aujourd'hui un équipement actif apte à supporter un lien Gigabit et donc déployable autour d'une épine dorsale au gigabits. Dans le cadre d'e-Toile deux services s'appuyant sur la technologie Tamanoir ont été crées et expérimentés : un service extensible de gestion dynamique de la qualité de service VTHD et un service de transfert de fichiers en multicast. Bien que nous nous soyons focalisés sur la performance, nous ne nous sommes cependant pas limités

au support de services légers (uniquement destinés au plan contrôle par exemple). Nous avons aussi montré que l'environnement était apte à supporter des services de haut niveau, capables aussi d'agir sur le plan données des paquets lorsque celà est nécessaire (compression à la volée par exemple). Les différents services qui ont été déployés sont opérationnels et devront être validés plus en profondeur par les applications et les composants de l'intergiciel. La technologie des réseaux programmable offre donc des possibilités intéressantes mais qui demandent encore à être approfondies en relation avec les niveaux supérieurs.

En conclusion de la partie "intergiciel", nous dirons que nous avons fait la preuve de la faisabilité du développement et de l'intégration de solutions nouvelles dans le cadre contraint de `Globus` en proposant une architecture innovante, extensible et réutilisable. Un effort important a porté sur l'étude de l'allocateur et d'un service de surveillance et d'information flexible et performant. Nous avons montré que définir un allocateur, pour un intergiciel, est d'une part un problème technique difficile, mais aussi un problème algorithmique très difficile. Un composant de type allocateur sans état a été spécifié et conçu dans le cadre du projet. Sa validation n'est malheureusement pour l'instant pas terminée. Il reste en particulier des problèmes techniques à résoudre pour permettre de communiquer les informations pertinentes à `Maui`, caché derrière le `GRAM`, puis le gestionnaire de travaux. Concernant le nombre restreint de modèles d'applications visés, l'allocateur étant sans état, il n'est pas très difficile d'ajouter d'autres allocateurs ayant pour but d'autres sémantiques d'allocation. Tous se synchronisent par les données présentes dans le `SIC`. Les aspects sécurité, notamment dans le domaine innovant de la gestion de certificats d'attributs, sont en cours de développements et pourraient être implémentés dans le cadre de développements futurs. La mise à grande, voire très grande échelle (donc hors `VTHD`) reste un point à aborder pour faire la preuve de la *scalability* et des performances de cette architecture.

Au niveau des applications, la plupart des expériences effectuées sur la plate-forme `e-Toile` en version `Globus` se situent dans le prolongement des utilisations des supercalculateurs : études paramétriques (grand nombre d'exécutions d'un code séquentiel), déploiement d'applications parallèles de type `SPMD` ou encore `multi-SPMD`, voire la combinaison des deux. Ainsi les couplages d'applications sont ils réalisées le plus souvent avec des outils qui étendent les fonctionnalités du modèle de programmation par passage de messages aux échanges entre applications ("`meta-MPI`"). C'est l'état de l'art dans un certain nombre de domaines d'application (météo, climatologie, électromagnétisme...) où le contrôle n'est assuré que *via* l'échange des données.

Les application portées sur la plate-forme en version `Globus` sont néanmoins représentative d'un large éventail d'usage de la grille pour les applications scientifiques (régulières, irrégulières, synchrones, asynchrones...). S'y ajoute l'expérience plus orientée portail de l'`IBCP`.

Dans le cadre de la version avec l'intergiciel `e-Toile`, où l'on dispose d'outils plus spécifiques, les réflexions ont plus porté sur une "conception orientée grille" des applications. Elles ont mis en évidence l'importance des outils transversaux :

- interface homme-machine (`Guide`) ;
- système d'information (`SIC`) et notamment collecte d'informations, traces ("*Monitoring*") ;
- allocation/chargement permettant le déploiement optimisé et gérant les aspects synchrones ;
- sécurité.

Le projet `e-Toile` a montré que l'interaction entre les communautés de "développeurs" et les communautés d' "usagers réels", si elle a pu s'avérer lourde au début du projet, a apporté, à partir du milieu du projet, un véritable richesse. L'engagement des équipes d"'utilisateurs" permet de véritablement valoriser et de prendre du recul sur les travaux menés en amont.

Une prolongation du projet permettrait de tester les composants logiciels spécifiques de l'intergiciel `e-Toile`, d'en mesurer complètement les performances et de conclure le volet applications sur une plate-forme version `e-Toile` stabilisée.

Notons qu'en ce qui concerne les perspectives, il reste évidemment encore beaucoup de verrous à explorer, notamment en ce qui concerne l'organisation de la sécurité et le traitement de la tolérance aux défaillances. Au niveau réseau des solutions plus flexibles, proposant des services plus sophistiqués que les services IP actuels et réunissant à la fois les propriétés d'extensibilité, de sécurité et de performance sont étudiés par la communauté réseau internationale. Ces réseaux, par les services avancés qu'ils proposeraient, permettraient de créer, à la demande, des environnements de calcul adaptés au besoin de diverses communautés d'utilisateurs (organisations virtuelles). Ces services réseaux ne sont cependant pas encore réellement disponibles et déployés aujourd'hui dans un contexte multidomaine mais devraient voir le jour dans les prochaines années. Des études sur l'utilisation efficace de ces services dans le contexte des grilles devront être envisagées dans le cadre de futur projets nationaux, si l'on souhaite capitaliser l'avance qu'`e-Toile` a permis d'obtenir dans le domaine des grilles haute performance.

Par ailleurs, l'expérience unique et très positive de l'intégration de composants logiciels variés et issus des laboratoires de recherche au sein d'une architecture logicielle ouverte et le déploiement d'applications de grille aux contraintes multiples sur une plate-forme d'échelle nationale, montre la faisabilité et tout l'intérêt de ce type de projets intégrés. Il faut noter qu'une véritable communauté très hétéroclite s'est créée et a permis d'explorer un domaine fortement complexe et multidisciplinaire de manière très fructueuse tant sur le plan de l'accroissement de la connaissance que sur le plan des résultats produits. Ces résultats sont en cours de publication et dissémination et seront présentés et démontrés dans les prochaines conférences internationales du domaine ainsi qu'au *Global Grid Forum*.

Bibliographie

[1] Atlas : A toroidal lhc apparatus. http://atlas.web.cern.ch/Atlas/.

[2] Charmm (chemistry at harvard molecular mechanics). http://yuri.harvard.edu/.

[3] Corba : Common object request broker architecture. http://www.corba.org/.

[4] Damien : Distributed applications and middleware for industrial use of european networks. http://www.hlrs.de/organization/pds/projects/damien/.

[5] Datatag : Research & technological development for a transatlantic grid. http://www.datatag.org.

[6] Eurogrid : Application testbed for european grid computing. .

[7] Ganglia : distributed monitoring and execution system. http ://ganglia.sourceforge.net/.

[8] Geant network : a multi-gigabit pan-european data communications network. http://www.dante.net/.

[9] Global grid forum : promote and support the development, deployment, and implementation of grid technologies and applications. http://www.ggf.org/.

[10] The globus alliance is developing fundamental technologies needed to build computational grids. http://www.globus.org.

[11] Grid benchmarking research group of the global grid forum. http://www.nas.nasa.gov/GGF/Benchmarks/.

[12] Java grande forum (jgf) : to develop community consensus and recommendations for either changes to java or establishment of standards (frameworks) for grande libraries and services. http://www.javagrande.org/.

[13] Legion : Worldwide virtual computer. http://legion.virginia.edu/.

[14] Lhc : The large hadron collider. http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/.

[15] Map center : An open grid status visualization tool. http://mapcenter.in2p3.fr/.

[16] Maui : an advanced job scheduler for use on clusters and supercomputers. http://supercluster.org/maui.

[17] Metodis : Metacomputing tools for distributed systems. http://www.hlrs.de/organization/pds/projects/metodis/.

[18] Mpich-g2 : a grid-enabled implementation of the mpi v1.1 standard. http://www3.niu.edu/mpi/.

[19] Network protein sequence @nalysis, pattinprot : a tool to scan a protein database of one or several sequences for one or several patterns. http://npsa-pbil.ibcp.fr/.

[20] The objectweb consortium, openccm. http://www.objectweb.org/openccm.

[21] Pasim : The pasture simulation model. www.nccr-climate.unibe.ch/download/wp3/p32/p32_pasim.html.

[22] Planetlab : An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/.

[23] Proactive : The java library for parallel, distributed, concurrent computing, with security and mobility. http://www-sop.inria.fr/sloop/javall/.

[24] Teragrid : build and deploy the world's largest, most comprehensive, distributed infrastructure for open scientific research. http://www.teragrid.org.

[25] Unicore : Uniform interface to computing resources. http://www.unicore.org/.

[26] Xtremweb (xw) : a software platform designed to serve as a substrate for global computing (large scale distributed system) experiments. http://www.xtremweb.org/.

[27] K. M. Anstreicher, N. W. Brixius, J.-P. Goux, and J. Linderoth. Solving large quadratic assignment problems on computational grids. *Mathematical Programming*, 91(3) :563–588, 2002.

[28] Olaf Arndt, Bernd Freisleben, Thilo Kielmann, and Frank Thilo. A comparative study of online scheduling algorithms for networks of workstations. *Cluster Computing*, 3(2) :95–112, 2000.

[29] A. Aumage and G. Mercier. Mpich/madiii : a cluster of clusters enabled mpi implementation. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2003)*, Tokyo, Japan, may 2003.

[30] O. Aumage, L. Bougé, A. Denis, L. Eyraud, J.F. M ?haut, G. Mercier, R. Namyst, and L. Prylli. High performance computing on heterogenous clusters with the madeleine ii communication library. *Cluster Computing*, 5 :43–54, 2002.

[31] G. Balsica, A. Bouteiller, F. Capello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodigensky, F. Magniette, V. Neri, and A. Selhikov. Mpich-v : Toward a scalable fault tolerant mpi for volatile nodes. In *ACM/IEEE International Conference on Super Computing SC 2002*, Baltimore, nov 2002.

[32] F. Baude, D. Caromel, and D. Sagnol. Distributed objects for parallel numerical applications mathematical modelling and numerical analysis modelization. *M2AN special issue on Programming tools for Numerical Analysis*, 36(5) :837–861, 2002.

[33] F Berman, G Fox, and T Hey. *The Grid : Past, Present and Future*. Wiley, 2003.

[34] Steven Blake, David Black, Mark Carlson, Elwyn Davies, Zheng Wang, and Walter Weiss. An architecture for differentiated services. rfc 2475, ietf, dec 1998.

[35] C. Blanchet. Logiciel mpsa et ressources bioinformatiques client-serveur web dédiées à l'analyse de séquences de protéine.

[36] C. Blanchet, C. Combet, and G. Deléage. L 'analyse de séquences de protéines avec gps@ sur une grille de ressources informatiques. In *in proceedings JOBIM 2002*, jun 2003.

[37] C. Blanchet, C. Combet, C. Geourjon, and G. Deléage. Mpsa : Integrated system for multiple protein sequence analysis with client/server capabilities. *Bioinformatics*, (16) :286–287, 2000.

[38] F Bonnassieux, R Harakaly, and P Primet. Mapcenter : un modèle ouvert pour la découverte, la supervision et la visualisation des environnements distribués à large échelle. In *Conference JRES*, november 2003.

[39] Primet P. Bonnassieux F., Harakaly R. Automatic services discovery, monitoring and visualization of grid environments : the mapcenter approach. In *Across Grid workshop*, February 2003.

[40] F. Bouahfs, B. Gaidioz, J.P. Gelas, L.Lefèvre, M. Maimour, C. Pham, P. Primet, and B. Tourancheau. Designing and experimenting an active grid architecture. *Future Generation Computer Systems (FGCS), Special issue on : Advanced Grid Technologies*, 2003.

[41] O. Boudeville. Retour d'expérience sur le portage de l'application rock sur grille de calcul.

[42] Robert Braden, David Clark, and Scott Shenker. Integrated services in the internet architecture : an overview. rfc 1633, ietf, jun 1994.

[43] Robert Braden, Lixia Zhang, Lixia Zhang, Shai Herzog, and Sugih Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. rfc 2205, ietf, sep 1997.

[44] N. Brisson and al. Stics : a generic model for the simulation of crop and their water and nitrogen balances. i. theory and parametrization applied to wheat and corn. *Agronomie*, 1998.

[45] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. Charmm : A program for macromolecular energy, minimization, and dynamics calculations. *J. Comp. Chem.*, 4 :187–217, 1983.

[46] R. E. Burkard, E. Çela, S. E. Karisch, and F. Rendl. Qaplib - a quadratic assignment problem library. Sur le WEB `www.opt.math.tu-graz.ac.at/qaplib`. Publié la première fois dans European Journal of Operational Research, vol. 55, pp. 115-119, 1991.

[47] M. Campanella, T. Ferrari, S. Leinen, R. Sabatino, and V. Rejis. Specification and implementation plan for a premium IP service. Delivrable 9.1, GEANT project (IST-2000-26417), 2001.

[48] G. Chun, H. Dail, H. Casanova, and A. Snavely. Benchmark probes for grid assessment. Sur le WEB `grail.sdsc.edu/projects/grasp/`. Dans le cadre du projet GRASP "Grid Assessment Probes".

[49] C. Combet, C. Blanchet, C. Geourjon, and G. Deléage. Nps@ : Network protein sequence analysis. *Tibs*, (25) :147–150, 2000.

[50] V.-D. Cung. *Contribution à l'Algorithmique Non Numérique Parallèle : Exploration d'Espaces de Recherche*. Thèse de doctorat d'université, Université Pierre et Marie Curie – Paris VI, April 1994.

[51] Van-Dat Cung, Abdelaziz Djerrah, and Rémi Revire. Résolution du qap sur grille de machines avec athapascan. In *Journées Bordelaises de formation GRID2 : Applications, Algorithmique et Ordonnancement pour la grille*, September 2003. `http ://www.irisa.fr/grid2/annonceBordeaux.html`.

[52] Karl Czajkowski, Ian Foster, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A ressource management architecture for metacomputing systems. In *In proceeding of the IPPS/SPDP'98 Workshop on job scheduling strategies for parallel processing*, pages 62–82, 1998.

[53] Bruce Davie, Anna Charny, Jon C. R. Bennett, Kent Benson, Jean-Yves Le Boudec, William Courtney, Shahram Davari, Victor Firoiu, and Dimitrios Stiliadis. An expedited forwarding phb (per-hop behavior). rfc 3246, ietf, mar 2002.

[54] A. Denis, C. Perez, and T. Priol. Padicotm : An open integration framework for communication middleware and runtimes. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid2002)*, pages 144–151, Berlin, Germany, may 2002.

[55] next generation computing infrastructure EU DataGrid. http ://www.eu-datagrid.org. 2002.

[56] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. Distributed resource management architecture that supports advance reservations and co-allocation. In *Intl Workshop on Quality of Service*, 1999.

[57] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid : An open grid services architecture for distributed systems integration. In *Globus Project, 2002, www.globus.org/research/papers/ogsa.pdf.*, 2002.

[58] Ian Foster and Carl Kesselman. Globus : A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2) :115–128, Summer 1997.

[59] Pascale Primet Franck Bonnassieux, Mathieu Goutelle. Network services - final report. Rapport de recherche, European DataGrid project, December 2003.

[60] Alex Galis, Jean-Patrick Gelas, Laurent Lefèvre, and Kun Yang. Active network approach to grid management. In *ICCS*, pages 1103–1113, Melbourne, Australia, June 2003. ICCS. LNCS 2658, ISBN 3-540-40195-4.

[61] Jean-Patrick Gelas and Laurent Lef ?vre. Towards the design of an active grid. In Lecture Notes in Computer Science, editor, *Computational Science - ICCS 2002*, volume 2230, pages 578–587, Amsterdam, The Netherlands, apr 2002. ISBN 3-540-43593-X.

[62] B. Gendron and T. G. Crainic. Parallel branch-and-bound algorithms : Survey and synthesis. *Operations Research*, 42(6) :1042–1066, 1994.

[63] Juha Heinanen, Fred Baker, Walter Weiss, and John Wroclawski. Assured forwarding PHB group. rfc 2597, ietf, jun 1999.

[64] Van Jacobson, Kathleen Nichols, and Kedar Poduri. An expedited forwarding phb. rfc 2598, ietf, jun 1999.

[65] Joseph A. Kaplan and Micheal L. Nelson. A comparison of queueing, cluster and distributed computing systems. RR 1090025, NASA Langley Research Center, June 1994.

[66] T. Kielmann, P. Hatcher, L. Bougé, and H. Bal. Enabling java for high performance computing : Exploiting distributed shared memory and remote method invocation. *Communications of the ACM*, 44(10) :110–117, oct 2001.

[67] T. C. Koopmans and M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25 :53–76, 1957.

[68] G. Krinner, N. Viovy, P.Friedlingstein, N. de Noblet, J. Ogée, P. Ciais, S. Sitch, J. Polcher, and I. C. Prentice. Orchidee a dynamical global vegetation model for studies of the coupled atmosphere-biosphere system. *submitted to Global Change Biology*, 2003.

[69] L. Lefèvre, C. Pham, P. Primet, B. Tourancheau, B. Gaidioz, J.P. Gelas, and M. Maimour. Active networking support for the grid. In Noaki Wakamiya Ian W. Marshall, Scott Nettles, editor, *IFIP-TC6 Third International Working*

Conference on Active Networks, IWAN 2001, volume 2207 of *Lecture Notes in Computer Science*, pages 16–33, oct 2001. ISBN : 3-540-42678-7.

[70] Wei Lin, Rong Zheng, and Jennifer C. Hou. How to make assured service more assured. In *ICNP*, pages 182+, 1999.

[71] Lipton and Tomkins. Online interval scheduling. In *SODA : ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1994.

[72] O. Lodigensky, G. Fedak, F. Capello, V. Neri, and A. Cordier. Augermome & xtremxeb : Monte carlos computation on a global computing platform. In *CHEP Conference on High Energy Physics*, La Jolla, California USA, march 2003.

[73] M. Maimour, J. Mazuy, and C. Pham. The cost of active services in active reliable multicast. In *Proceedings of the 4th IEEE Annual International Workshop on Active Middleware Services (AMS 2002)*, pages 67–72, Edinburg, UK, July 2002.

[74] M. Maimour and C. Pham. Dynamic replier active reliable multicast (dyram). *Journal of Cluster Computing*, 2004. A paraître.

[75] Kathleen Nichols, Steven Blake, Fred Baker, and David Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. rfc 2474, ietf, dec 1998.

[76] M. Parashar, G. von Laszewski, S. Verma, J. Gawor, K. Keahey, and N. Rehn. *A CORBA Commodity Grid Kit*. John Wiley and Sons, 2002.

[77] G. Perriere, C. Combet, S. Penel, C. Blanchet, J. Thioulouse, C. Geourjon, J. Grassot, C. Charavay, M. Gouy, L. Duret, and G. Deléage. Integrated databanks access and sequence/structure analysis services at the pbil. *Nucleic Acids Res.*, 31(13) :3393–3399, 2003.

[78] P. Primet and F. Chanussot. Spécification du service actif qosinus. Technical Report RT-0287, INRIA, 2003.

[79] Pascale Vicat-Blanc Primet, Fabien Chanussot, Christophe Blanchet, Nicolas Lacorne, and Philippe d'Anfray. E-toile : High performance grid middleware. In *IEEE International Cluster Conference. Grid Demo session*, December 2003.

[80] Pascale Vicat-Blanc Primet, Johan Montagnat, Fabien Chanussot, and Mathieu Goutelle. Network quality of service in grid environments : the qosinus approach. In *submitted to IEEE International IPDPS Conference.*, 2004.

[81] R. Sabatino. The SEQUIN project and GEANT network - qos for european research and education networks. In *International Workshop on Quality of future Internet Services QofIS*, pages 24–26, Coimbra, Portugal, September 2001.

[82] A. Schumm, O. Bremnes, and H. Ourly. Le code de modélisation radiographique moderato. reprend une communication faite à la conférence “NDT in progress”, Prague, Octobre 2003.

[83] N. Seddigh, B. Nandy, and P. Pieda. Bandwidth assurance issues for tcp flows in a differentiated services network.

[84] J. Sgall. Online scheduling – a survey, 1997.

[85] Y. Souffez and C. Domain. Dynamique moléculaire : Bilan du csn au csv.

[86] Globus Team. Dynamically-updated request online coallocator duroc v0.8. Globus http ://www.globus.org/, October 1998.

[87] Géraldine Texier and Laurent Toutain. Utilisation de la différentiation de services dans les applications. Technical report, VTHD Technical Document, 2002.

[88] A. Vernois, P. Vicat-Blanc, F. Desprez, F. Hernandez, and C. Blanchet. Gripps : Grid protein pattern scanning. In *in proceedings of the International HealthGrid 2003 : 1st Conference Workshop of HealthGrid cluster*. Elsevier, January 2003.

[89] Pascale Vicat-Blanc Primet. Présentation et expérimentation de l’intergiciel de grille e-toile sur la plate-forme nationale de test. In *Réseau National des Technologies Logicielles : Journées RNTL 2003*, October 2003. `http ://www.telecom.gouv.fr/rntl/programme03.htm`.

[90] Pascale Vicat-Blanc Primet and Philippe d’Anfray. Les grilles haute-performance et le projet etoile. *Matapli*, (71), May 2003.

[91] V. Taylo W. Smith, I. Foster. Scheduling with advanced reservation. In *In proceeding of the IPDPS Conference*, May 2000.