Intégration d'applications multidisciplinaires en environnement métacomputing

Toan Nguyen — Christine Plumejeaud

INRIA Rhône-Alpes
655, Avenue de l'Europe
Montbonnot, F-38334 Saint Ismier Cedex
Toan.Nguyen@inrialpes.fr

RÉSUMÉ. *Les applications de simultation et d'optimisation mettent en jeu une grande quantité de codes qui induisent des charges importantes sur les systèmes informatiques existant. Les avancées technologiques dans le domaine du matériel et du logiciel permettant d'en améliorer les performances. Le but de cette étude est de montrer que les techniques de spécifications formelles et les plateformes d'intégration réparties fournissent des solutions efficaces au calcul à haute performance dans les environnements de métacomputing. Elles permettent de coupler des codes hétérogènes répartis dans des environnements de résolution de problèmes pour la mise en œuvre d'applications multidisciplines*

ABSTRACT. *Simulation and optimisation involve a large variety of codes that result in high CPU loads on existing computer systems. Advances in both hardware and software, including massively parallel computers, PC-clusters and parallel programming languages somewhat alleviate current performance penalties. It is the claim of this paper that formal specification techniques with distributed integration platforms, provide a sound and efficient support for high performance distributed computing in metacomputing environments. Formal specifications provide rigourous and provable approaches for complex applications definition and configuration, while distributed integration platforms provide standardised deployment and execution environments for coupling heterogeneous codes in problem-solving environments supporting multi-discipline applications.*

MOTS-CLÉS : *Simulation, applications multi-disciplines, spécifications formelles, plateformes d'intégration réparties, environnements de résolution de problèmes, métacomputing.*

KEYWORDS: *Simulation, multidisciplines applications, formal specifications, distributed integration, problem solving environments, metacomputing*

1. Introduction

Les applications de simulation et d'optimisation, comme celles qui sont utilisées pour les maquettes numériques en conception aéronautique, ou pour la simulation numérique de systèmes de propulsion [Lop 00], constituent des défis pour la communauté informatique [KEA 99, NGU 02, SAN 00]. Elles nécessitent des environnements de calcul puissants et sophistiqués, tels des ordinateurs parallèles, des grappes de PC, etc. Elles nécessitent également des logiciels de communication efficaces, par exemple des protocoles à passage de message [KES 99]. Bien que ces problèmes ne soient pas nouveaux, ils ont donné lieu à des solutions spécifiques qui sont devenues des standards de fait, comme MPI [MPI] ou PVM [PVM].

Simultanément, l'informatique a produit des solutions intéressantes aux problèmes de

– la spécification formelle des systèmes répartis et communicant

– l'échange standardisé au sein d'environnements de développement pour les applications réparties.

Cet article explore la conception et l'implantation d'une plateforme d'intégration qui consiste en un environnement informatique dédié à la *spécification*, *configuration*, *déploiement* et *exécution* d'applications d'optimisation multi-disciplines coopératives. De telles applications invoquent une grande variété de composants logiciels issus de diverses disciplines, et connectés entre eux afin de collaborer sur des projets communs.

Par applications *coopératives*, nous entendons ici des applications impliquant des logiciels répartis, appelés ici «tâches», qui interagissent dynamiquement de manière contrôlée et prouvable formellement. Par *spécification*, nous entendons la définition formelle de l'interaction entre les tâches en utilisant un formalisme algébrique, par exemple, l'algèbre de processus SCCS [MIL 83]. Cela inclut les échanges de données entre tâches ainsi que leur synchronisation et le flot de contrôle. Par *configuration*, nous entendons le placement de ces tâches sur des composants logiciels, appelés par la suite «modules» ou «codes». Par *déploiement*, nous entendons l'allocation des différents modules sur du matériel réparti. Par *exécution*, nous entendons le déclenchement des différents modules sur le matériel, afin de produire les résultats des applications.

Ce papier utilise des exemples issus des techniques de spécification formelle pour la définition d'applications complexes impliquant différentes disciplines en conception aéronautique. Nous aborderons le déploiement et l'exécution des applications en nous basant sur des techniques logicielles réparties orientées-objet (par exemple, CORBA) et la programmation par composants logiciels.

Ce travail a donné lieu au développement d'une plateforme d'intégration dénommée CAST (*Collaborative Applications Specification Tool*) permettant l'exécution d'applications de conception dans le domaine aérospatial. La plateforme est basée sur une interface graphique de haut niveau (Figure 2) permettant la spécification formelle d'applications d'optimisation se fondant sur SCCS, l'algèbre de processus communicant issus des travaux de Milner.

Elle est implantée sur un réseau de stations de travail et des grappes de PC connectées par le réseau à très haut débit VTHD [VTH] et communiquant à travers CORBA [Obj98]. Elle a été en partie développée au sein d'un projet européen Esprit HPCN appelé «DECISION» : *Integrated Optimisation Strategies for Increased Engineering Design Complexity*.

Cet article s'articule ainsi. La section 2 est une revue des approches actuelles pour la spécification d'applications réparties. La section 3 consiste en une revue de problèmes d'optimisation multi-disciplines. La section 4 est une revue détaillée de CAST, incluant des exemples tests. La partie 5 est une comparaison d'approches et projets similaires. Nous concluons en partie 6.

2. Spécifier des applications réparties

2.1. *Approches par flot de données*

La modélisation par flot de données a été longtemps utilisée en informatique d'entreprise pour la modélisation de processus et les besoins en ingénierie [DAT]. C'est utilisé avant tout pour spécifier les données d'entrée et de sortie de tâches connectées par un diagramme qui évolue dans le temps.

Derrière cette approche se cache une philosophie de programmation, comme dans IRIS EXPLORER [NAG98], et une absence de fondement formel, à l'exception notable des réseaux de Petri [Pet]. Les paramètres des tâches doivent être connectés entre eux par le biais de canaux définissant leur synchronisation et les échanges de données.

Cette approche date des années 70, lorsque la recherche en informatique répartie n'en était qu'à ses débuts. Ainsi l'approche à flot de données fournit seulement un moyen explicite et manuel pour la spécification et le contrôle d'applications réparties.

2.2. *Approches par flot de contrôle*

Les approches par flot de contrôle ont été beaucoup utilisées pour la modélisation de processus d'entreprise, par exemple l'édition de documents [WOR, DOC, And 98].

Le principal inconvénient de cette approche, comme la plupart des approches à flot de données est une absence de base théorique. Cependant, elle permet de prendre en considération les données et la répartition de l'application sur des réseaux locaux ou distants (LAN, WAN).

2.3. *Approches formelles : algèbres de processus, systèmes de spécification*

Simultanément aux approches par flot de contrôle, un travail théorique important a été mené sur la spécification des logiciels systèmes, notamment sur les processus communicants (SCCS [MIL 83]). Ceci a posé les bases de la spécification formelle de systèmes communicants, plus particulièrement pour les protocoles de communication et la gestion de messages.

La fusion des approches de spécifications formelles avec les approches par flot de contrôle a mené à des voies intéressantes en matière de spécification, configuration, déploiement et exécution d'applications réparties.

Grâce aux standards de communication, à ceux de configuration logicielle et d'exécution comme CORBA, toutes les briques de base permettant de construire efficacement des applications complexes réparties sont disponibles.

3. Spécifier des applications d'optimisation multidisciplines

3.1. *Problèmes d'optimisation*

Modéliser les problèmes de physique avec des méthodes numériques nécessite de bonnes méthodes d'approximation. Même pour des cas simples, ces approches peuvent induire une forte charge CPU : par exemple plusieurs heures de calcul sont nécessaires pour un calcul aérodynamique en 2D sur un profil d'aile, même sur des stations de travail puissantes. Les techniques d'optimisation tendent à réduire le temps de convergence à l'aide de méthodes mathématiques spécifiques. Citons plus particulièrement les approches fondées sur les algorithmes génétiques [MAR 98]. Ils permettent notamment une convergence plus rapide car les solutions candidates peuvent être générées et calculées en parallèle, sur des grappes de PC par exemple.

Un cas précis va nous servir d'illustration dans les prochains paragraphes. Il s'agit de l'optimisation de forme d'une aile dans des conditions aérodynamiques stationaires (Figure 6). L'objectif est de réduire la traînée induite [MAR 98]. Deux algorithmes sont ici en jeu. L'un, qui calcule l'écoulement de l'air autour de l'aile ; l'autre qui optimise la forme de l'aile. Les données d'entrée consistent en la forme de l'aile (RAE2612), son angle d'attaque (2 degrés), la vitesse (Mach 0.84) et le nombre de pas d'optimisation (100). Ce problème nécessite plusieurs heures de calcul sur une station de travail d'entrée de gamme.

Etant données de telles charges de calcul, il est clair que les ingénieurs des départements de conception recherchent les meilleures performances possibles. Deux approches sont actuellement explorées :

– des ordinateurs plus rapides

– des algorithmes plus rapides

La première approche se décompose elle-même en trois :

– des CPUs plus rapides

– des ordinateurs parallèles

– des grappes de PC fortement couplés

– des grilles de calcul.

La seconde approche (algorithmes plus rapides) se base sur la combinaison :

– d'algorithmes génétiques où les solutions candidates sont générées aléatoirement, sélectionnées, croisées, puis fusionnées [MAR 98]

– d'algorithmes inspirés de la théorie des jeux

– des algorithmes parallèles, où des instructions spécifiques pour contrôler le regroupement et le lancement de processus parallèles peuvent être utilisées, comme MPI.

L'importance de la performance dans la convergence des solutions est particulièrement grande car les applications de simulation, impliquant par exemple des maquettes numériques en conception aéronautique sont au cœur des préoccupations de beaucoup de constructeurs d'avions dans un marché très compétitif.

Ainsi les départements de conception tendent à intégrer dans leurs simulations des techniques multi-disciplines, par exemple en couplant l'aérodynamique, l'électromagnétique et la conception de structure. Ceci est détaillé dans la partie suivante (section 3.2).

3.2. *Problèmes d'optmisation couplée*

Les avancées en ingénierie de conception ont rendu possible la coopération de plusieurs disciplines. Cependant, les interactions entre les divers domaines sont encore sujettes à des règles de synchronisation entre les participants et à un concensus quant aux formalismes d'échange de données. Ceci est généralement connu sous le terme de groupware [And 98]. Cela nécessite une technologie avancée en matière de coordination des divers agents impliqués dans les projets (humains ou machines). Cela nécessite également des logiciels sophistiqués pour la négociation et la communication, ainsi que de puissants formalismes de représentation de données et de connaissances. Tout ceci présente des analogies avec les difficultés rencontrées dans le domaine de l'informatique répartie.

Les processus d'optimisation couplée font partie de l'ingénierie multi-disciplines intégrée. Leur importance provient du gain de temps espéré entre la phase de conception et la mise sur le marché, et de la réduction espérée du temps de conception grâce à des cycles d'essais raccourcis. La conception d'ailes d'avion, couplant l'aérodynamique et la structure en est un bon exemple. Bien que l'optimisation en mécanique des structures (CSM) soit bien comprise, ce qui est également le cas en calcul de la dynamique des fluides (CFD) pour l'aérodynamique, leur couplage est délicat à réaliser.

Il est primordial de caractériser précisément où et quand les deux optimisations doivent et peuvent interférer. Ce n'est pas évident car les codes d'optimisation ont souvent été développés de manière indépendante, sans contrainte particulière concernant la coopération. Ceci est également vrai lorsqu'un solveur et un optimiseur sont couplés, car il n'est pas toujours évident de structurer la boucle du programme principal : est-ce le solveur ou l'optimiseur qui doit être le programme maître ?

Une autre question ouverte pour des problèmes d'optimisation couplée, mais aussi pour les environnements généraux de couplage de code, concerne l'interface des données entre les modules. En général, chaque module a ses propres techniques de stockage de données et ses propres formats, par exemple pour les maillages. Il se peut qu'ils ne soient pas compatibles. Ils peuvent aussi se recouvrir partiellement. La compatibilité des données entre différents codes est donc un problème important. Ce point n'est pas abordé dans la suite de ce papier.

4. Plates-formes d'intégration réparties

Avec l'avènement des réseaux d'ordinateurs spécialisés (Intranet) et généraux (Internet), le calcul réparti et les applications de calcul intensif sont des facteurs qui contribuent à améliorer les performances globales de résolution de problèmes complexes comme la simulation multidisciplines. Des WAN aux LAN, NOW et aux clusters de PC, des configurations matérielles et logicielles sont maintenant disponibles qui permettent la répartition des composants des applications. Les applications visées sont ici les applications multidisciplines qui consistent à faire collaborer plusieurs experts à travers un réseau à haut débit et intégrer les résultats des différents sites en un lieu final unique. Le reste de cet article porte sur les problèmes d'optimisation multidisciplines impliquant des calculs répartis sur des environnements hétérogènes.

4.1. *Plates-formes d'intégration*

La recherche en systèmes a souligné les avantages des composants logiciels pour le calcul réparti, par exemple les EJB (Enterprise Java Beans), le CCM (Corba Components Model) [Bal 98]. Ils permettent la définition, le développement et l'exécution d'applications réparties composées de divers modules logiciels hétérogènes qui ne sont pas initialement conçus pour coopérer. Ils peuvent résider sur des plates-formes hétérogènes, à condition de communiquer à travers un médium commun, matériel ou logiciel.

La possibilité de faire coopérer des composants logiciels écrits dans différents langages de programmation tels que Fortran, C, ou C++, a conduit ici à choisir une couche CORBA pour mettre en œuvre la plate-forme d'intégration répartie (Figure 1). Ceci est dû au fait que la plupart des applications d'ingénierie utilisent de nos jours des logiciels existants qui ne peuvent pas facilement être modifiés, par exemple des codes propriétaires ou commerciaux, qui doivent fonctionner sur des sites particuliers, par exemple des serveurs de bases de données, ou qui peuvent être dédiés à des plates-formes spécifiques, telles que des ordinateurs massivement parallèles. L'avènement de la programmation orientée-objet a également souligné la faisabilté d'environnements ouverts pour les applications réparties. Les modules logiciels sont alors considérés comme des composants qui peuvent être connectés dynamiquement à une plate-forme. Cela requiert l'enrobage des modules pour réaliser des interfaces standardisées avec la plate-forme. Cela est réalisé dans CORBA en employant les définitions IDL des interfaces. Parmi les avantages de cette approche figure la possibilité de faire fonctionner des logiciels hétérogènes sur un matériel hétérogène de manière unifiée. C'est un complément à la modularité illustrée par les langages et le développement de logiciels orientés-objet. En outre, cela permet le découplage des interfaces des logiciels de leurs réalisations. De ce fait, des changements dynamiques dans les réalisations des logiciels sont possibles sans qu'il n'y ait d'impact sur les composants qui les utilisent. Cette approche suit la tendance actuelle en calcul réparti définie par l'OMG (Object Management Group) [OMG98]. Pour être mise en œuvre, cette approche requiert la possibilité de définir un formalisme adéquat pour l'échange des données et la synchronisation des composants. Elle utilise l'appel synchrone de méthodes distantes de CORBA, par exemple, ou un traitement de requête plus sophistiqué " multi-threadé ". Cette approche utilise également des répertoires pour stocker et rendre accessibles des objets qui peuvent être partagés à distance grâce au service de nommage et au répertoire des interfaces.

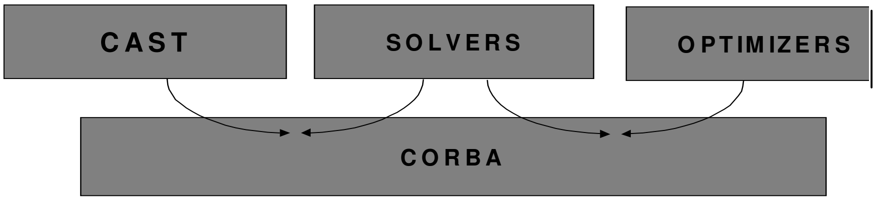

Figure 1. *CAST : intégration de solveurs et d'optimiseurs*

Les particularités de la répartition des données et des modules sont laissées aux programmeurs de logiciels. Ceci contraste avec la programmation parallèle, utilisant par exemple MPI. Dans ce cas, les programmeurs doivent coder à l'intérieur des modules le regroupement et le lancement des processus. Le code d'application est donc mélangé avec des instructions de répartition et de parallélisation. Ce n'est pas le cas avec des environnements où la modularité et la répartition transparente sont possibles, comme CORBA. Dans ce cas, le code des utilisateurs peut être employé tel quel, pratiquement sans modifications. Cela est démontré dans les paragraphes suivants par des cas-tests d'optimisation d'un profil d'aile, où les solveurs et les optimiseurs CFD sont utilisés sans modification via la plate-forme CAST.

4.2. *Une plate-forme d'intégration répartie pour les applications multidisciplines*

4.2.1. *Principes*

CAST est une plate-forme d'intégration conçue pour répondre aux exigences des applications de simulation et d'optimisation réparties dans les projets d'ingénierie complexes. Comme exemples d'applications, on trouve des projets industriels dans l'électronique et l'aérospatial, en particulier l'optimisation multidisciplines : aérodynamique-structure-acoustique et électromagnétique. La population d'utilisateurs visée comprend les directeurs de projets et les ingénieurs qui sont des experts dans leur domaine, c'est-à-dire en électromagnétique, aérodynamique, etc. Ils sont à l'aise en informatique, mais ne sont pas nécessairement compétents dans les spécifications de processus formel, en calcul théorique, en programmation orientée-objet, ou encore en CORBA ou en programmation par composants. Cela oblige la plate-forme d'intégration à fournir une interface utilisateur de haut niveau, n'ayant aucune particularité liée à l'algèbre de processus ou au calcul réparti. Les principes essentiels de la plate-forme sont donc :

– de reposer sur une base théorique solide pour la spécification de processus

– des normes transparentes et largement acceptées pour le calcul réparti

– une utilisation transparente d'une large gamme de logiciels

– un emploi transparent des plates-formes d'ordinateurs réparties hétérogènes

– une interface utilisateur de haut niveau pour faciliter son utilisation par des utilisateurs finaux, qui sont des experts dans des domaines scientifiques non liés à l'informatique.

La plate-forme permet la définition, la configuration, le déploiement et l'exécution d'applications multidisciplines réparties sur NOW, LAN, WAN et clusters de PC. Ces applications sont construites en faisant collaborer des modules qui peuvent fonctionner sur des ordinateurs hétérogènes. Les modules peuvent être réalisés dans des langages de programmation différents et ils échangent des données de manière dynamique. Ils sont synchronisés au moyen de plans définis par les utilisateurs qui comprennent des opérateurs de séquence, de boucles imbriquées et entrelacées, et des opérateurs d'exécution parallèle. Ces opérateurs sont des composants d'une algèbre de processus qui est principalement basée sur l'algèbre de Milner SCCS [MIL 83].

Par exemple, la spécification de l'exemple HBCGA (Hybrid Binary Coded Genetic Algorithm) d'optimisation de forme d'une aile détaillée dans le paragraphe 3.1 engendre la formule SCCS suivante :

```
InitBCGA:InitHybrid:BGGA:(TRUE:(END)+FALSE:(FUN:(TRUE:(HYBRID:
(TRUE:  (=>InitHybrid)+FALSE:(=>FUN)))+FALSE:(=>BGGA))))
```

dans laquelle les différents modules BCGA, FUN, HYBRID et INIT sont connectés par une boucle, représenté par '=>', l'opérateur de séquence étant noté ' :' et l'opéra-

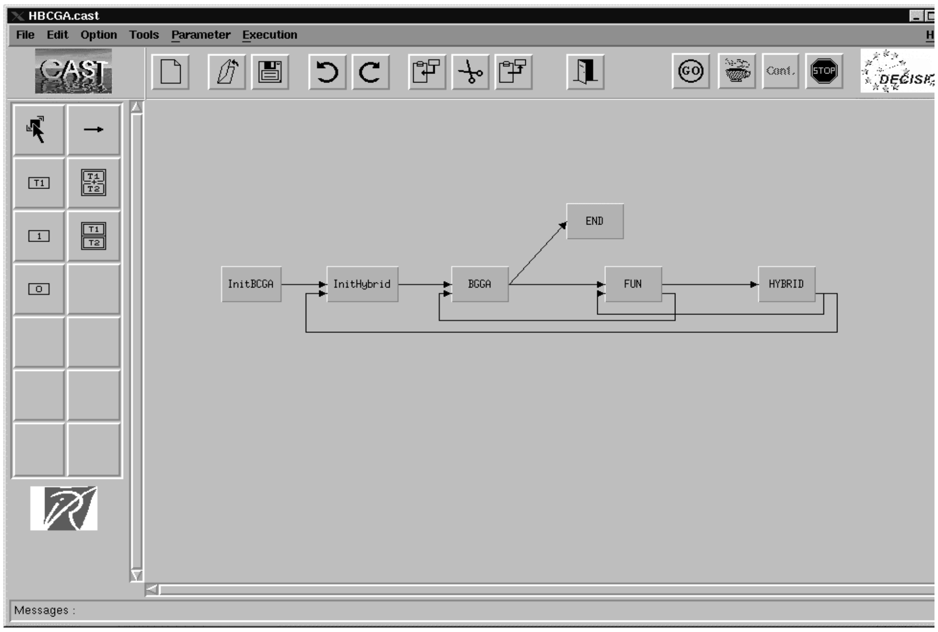

Figure 2. *CAST : l'interface utilisateur*

teur de choix par '+'. D'autres opérateurs SCCS sont disponibles, comme l'opérateur de synchronisation de tâches '&' et l'opérateur de parallélisation '/'. Les icônes sur le menu de gauche sont utilisés pour construire l'application : elles représentent les opérateurs SCCS. Les tâches sont définies par les utilisateurs sur des écrans sélectionnés par sous-menus qui ne sont pas représentés ici. Il faut noter que cet exemple implique deux boucles imbriquées et deux boucles entrelacées. Les icônes de la ligne supérieure sont des outils pour appeler des opérations utilitaires telles que : ouverture d'un fichier, défaire/refaire, enregistrer le fichier, etc. Des applications prédéfinies peuvent être chargées et modifiées en-ligne ou intégrées dans de nouvelles applications, c'est-à-dire que des applications existantes peuvent être sauvegardées et réutilisées plus tard pour en former de nouvelles. La formule SCCS est automatiquement mise en œuvre par CAST 2.0 depuis la spécification graphique de l'application. Elle est utilisée de façon interne pour générer la structure d'exécution répartie.

Les utilisateurs finaux n'ont à aucun moment besoin de connaître SCCS ou toute autre algèbre de processus. La plate-forme d'intégration fournit à l'utilisateur une interface basée sur des icônes de haut niveau. On peut faire appel à des outils de simulation, de vérification, de configuration, de déploiement et d'exécution. Le résultat de l'exemple HBCGA ci-dessus est illustré par la Figure 6. En outre, la plate-forme gère l'intégration de modules répartis sur des réseaux d'ordinateurs hétérogènes. Elle fournit un environnement d'exécution répartie transparent qui est conforme aux spécifications CORBA 2.0 de l'OMG [OMG98].

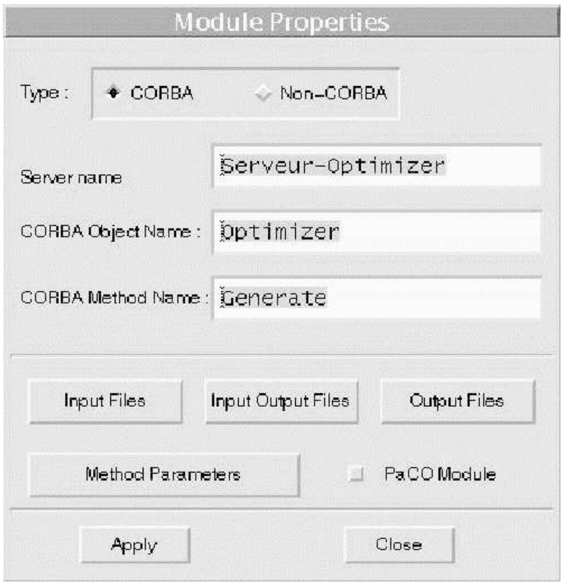

Figure 3. *Nommage d'un code conforme à CORBA*

Les configurations matérielles sont gérées uniformément, c'est-à-dire que les postes de travail, les clusters de PC et les ordinateurs connectés via NOW, LAN, WAN sont gérés de manière transparente via la définition des sites d'exécution des modules. Cela s'effectue en spécifiant les serveurs de noms CORBA pour les différents modules (Figure 3). Actuellement, aucune migration dynamique de modules n'est gérée par CAST. L'utilisateur final n'a pas le choix concernant les sites d'exécution (Figure 4). L'exécution des modules est lancée par une requête dynamique adressée à l'ORB : le serveur de noms inclut le nom des méthodes correspondant aux codes exécutables dans les objets enrobeurs des modules.

4.2.2. *Spécification d'applications complexes*

Le bénéfice tiré de l'utilisation d'une algèbre de processus pour définir les applications est double :

– il oblige les développeurs d'applications à spécifier rigoureusement les interactions des modules et leur planification, sans programmer au préalable les entrées/sorties de modules, retardant ainsi la réalisation à une phase ultérieure dans la conception de l'application

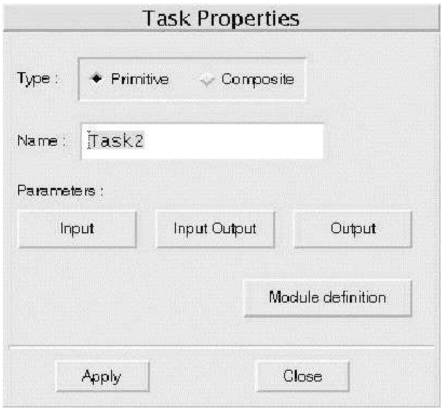

Figure 4. *Nommage d'une tâche applicative de CAST*

– il engendre des spécifications formelles qui peuvent être traitées par vérification automatique et par des algorithmes de réduction, assurant ainsi des propriétés de calcul supplémentaires, comme par exemple, la justesse, l'absence de blocages, ou l'accessibilité, etc. Bien que ce dernier point n'ait pas encore été mis en pratique, on prévoit que l'introduction de la vérification et des algorithmes de réduction permettra d'obtenir un gain significatif dans l'intégration des applications réparties complexes qui impliquent un grand nombre de modules. Cela fournit une base solide et théorique sur laquelle on peut se reposer pour spécifier formellement, prouver et vérifier les processus. Par exemple, la co-simulation de circuits intégrés implique des douzaines de modules qui coopèrent pour simuler les différents aspects de l'électronique : électrique, thermique, temporisation, etc. La gestion et le contrôle des interactions de ces modules peuvent faire appel à des relations complexes. Prouver que ces interactions sont correctement réalisées et que le simulateur le plus simple a effectivement été conçu peut s'avérer être une tâche ardue qui peut tirer profit de la validation de modèles et des outils de vérification hérités des techniques de vérification formelle. Il s'agit d'une pratique courante en conception de circuits intégrés, et qui peut être appliquée directement dans CAST.

Plusieurs techniques de vérification fondées sur des algèbres et des langages de spécification formels sont déjà disponibles et actuellement en cours d'utilisation par les concepteurs de logiciels et de protocoles de communication en temps-réel (LOTOS, CADP, etc). Ils font appel à des algorithmes de vérification de modèles et à des techniques d'observateurs qui peuvent réduire de façon importante les erreurs d'exécution dues à des spécifications incorrectes.

4.2.3. *Mise en œuvre*

Un prototype de CAST a été développé pendant une première période de six mois (Avril-Octobre 1998). Cette plate-forme conforme à CORBA a fait l'objet d'une démonstration au cours de la seconde revue du projet DECISION à Sophia-Antipolis, en octobre 1998. Elle reposait sur l'ORB ILU de Xerox et sur l'environnement de configuration d'applications réparties OLAN [Bal 98]. ILU possède des caractéristiques intéressantes, telles que des interfaces avec une grande diversité de langages comme C, C++, Lisp, Java, et Python. Toutefois, des limites intrinsèques (par exemple : absence de répertoire des interfaces, pas de multi-threading (processus léger)) et de performance ont conduit au remplacement de ces logiciels.

Le remplacement par l'ORB Orbacus [Obj98] de Object-Oriented Concepts Inc., a débuté en novembre 1998. Orbacus permet une intégration aisée avec les langages C++ et Java. Cela a rendu possible une mise en œuvre rapide de la version Corba de CAST sur stations de travail Unix.

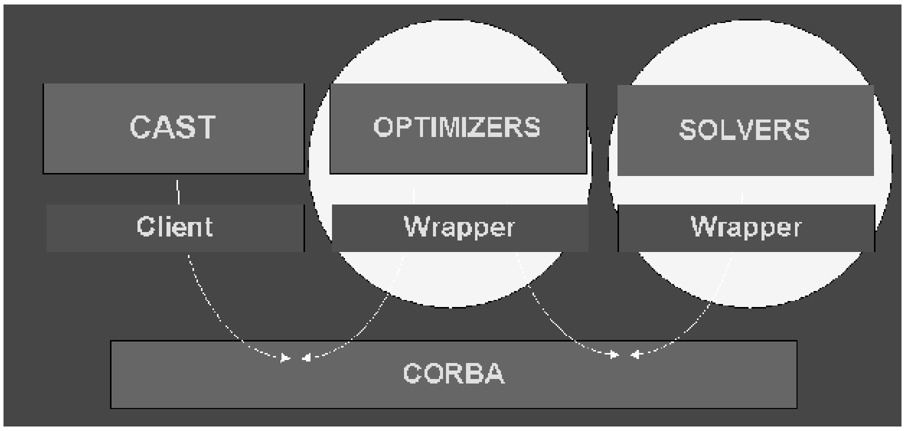

Figure 5. *CAST : une plate-forme d'intégration conforme à CORBA*

Le prototype a fait l'objet d'une présentation en avril 1999. Un facteur déterminant a été le langage C++ choisi pour la mise en œuvre de CAST. L'évolution vers une plate-forme d'intégration compatible avec Corba 2.0, nommée CAST 2.0 en a été largement facilitée (Figure 5). Le langage IDL est utilisé pour permettre à l'utilisateur de définir les interfaces des modules et produire leurs enrobeurs, talons et squelettes. Une fois compilé, cela forme les classes correspondantes C++ qui sont automatiquement intégrées dans le serveur et le client de la plate-forme CAST. Sur le ptototype, pour des raisons de simplicité, de facilité de maintenance et de performance, il n'a été fait aucun usage de l'interface d'appel dynamique ni du répertoire de l'interface dynamique dans Corba. Cela a limité la première version Corba de CAST 2.0 à la seule exécution des modules utilisateurs existants.

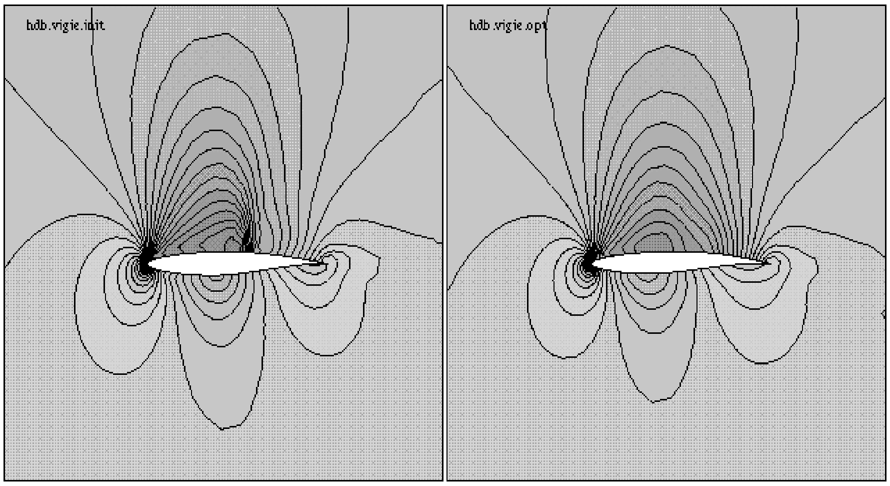

Figure 6. *Optimisation de forme d'une aile : induced drag reduction in high-speed cruise configuration (Isomach curves)*

Depuis, des développements de la plate-forme ont été mis en œuvre pour permettre la connexion dynamique de solveurs et d'optimiseurs. Cette approche emploie maintenant toutes les capacités de création et d'appel dynamique de CORBA. Une phase ultérieure de développement a débuté en janvier 2000. Son but était de permettre l'intégration de codes et l'exécution parallèle sur des clusters de PC connectés à des réseaux à haut débit. Vers la fin de l'année 2000, cette évolution était opérationnelle en ce qui concerne l'exécution des codes parallèles sur des clusters de PC. Interfacer des codes existants qui mettent en œuvre MPI avec CORBA a représenté un autre défi. Différents travaux ont déjà été réalisés concernant l'intégration de codes parallèles dans des environnements de calcul réparti et se sont avérés être une aide inestimable pour le sujet. Notre réalisation utilise la notion " d'objets CORBA parallèles " développés dans PACO par le projet PARIS à l'IRISA [PRI 99]. Un objet parallèle CORBA est un regroupement d'objets similaires fonctionnant en parallèle sur un cluster de PC. Ces objets sont gérés par un objet serveur unique qui est l'interface avec les clients distants. Par conséquent, le parallélisme est transparent pour l'utilisateur final. Cela fournit une intégration en souplesse des codes parallèles écrits en Fortran et utilisant MPI. PACO est lui-même construit sur l'ORB MICO [MIC]. MICO est donc devenu la nouvelle couche ORB utilisée par CAST depuis septembre 2000. L'exemple d'optimisation de forme 2D (Figure 6) figurant au paragraphe 3.1 a été géré sur un cluster de PC en utilisant une version parallèle de l'algorithme génétique [MAR 98]. On l'a ensuite rendu compatible avec PACO. Dans la dernière mise en œuvre, il y a plusieurs serveurs : l'un est dédié à l'algorithme d'optimisation génétique, et les autres sont consacrés aux calculs CFD et au maillage. Les performances vont bien sûr de pair avec l'exécution en parallèle du code sur les processeurs de la grappe. Le tableau figurant ci-dessous donne quelques indications de performance (Tableau 1). Les deux premières lignes du tableau représentent le code d'optimisation exécuté sur une station de travail monoprocesseur Unix (noeud appelé Ships) et les lignes suivantes sur des clusters de PC de 4 à 40 processeurs. Le cluster est situé à l'INRIA Rhône-Alpes à Grenoble. Chaque noeud du cluster est un bi-processeur Pentium III fonctionnant sous Linux à 733 MHz. Les temps CPU sont donnés en secondes.

nœuds	# procs	CPU (sec.)
Ships	4	15960
Ships	8	17820
cluster	4	560
cluster	8	328
cluster	12	252
cluster	16	210
cluster	20	189
cluster	24	178
cluster	28	157
cluster	32	158
cluster	36	150
cluster	40	138

Tableau 1. *Performance pour l'optimisation de forme sur cluster de PC*

L'étape suivante, qui s'est terminée en juillet 2001, a permis l'exécution d'applications de simulation et d'optimisation aérodynamique dans un environnement de metacomputing, comprenant des stations de travail et des clusters de PC répartis sur trois sites différents : INRIA Rennes, INRIA Sophia-Antipolis et INRIA Rhône-Alpes (Figure 9).

Suite à des tests de performance menés sur diverses configurations (Tableaux 1 à 4), le logiciel CAST fonctionne sur le cluster de PC de Grenoble, l'algorithme d'optimisation génétique parallèle fonctionne sur le cluster de PC de Sophia-Antipolis, et plusieurs instances du solveur CFD sont gérés par le cluster de PC de Rennes. CAST est un client CORBA pour le serveur de l'algorithme génétique, lequel se comporte à son tour comme un client des solveurs CFD (Figure 7). Le cluster de PC de Grenoble comprend 225 processeurs bi-Pentium III 733 MHz connectés par un réseau à 100 Mbits/sec FastEthernet, fonctionnant sous Linux (Mandrake 7.0). Le cluster de PC de Rennes est composé de 20 bi-Pentium III, 500 MHz, et de bi-Pentium II, 450 MHz., connecté par un réseau FastEthernet 100 Mbits/sec, fonctionnant sous Linux (Debian 2.2). Quant au cluster de PC de Sophia-Antipolis, il est constitué de 33 processeurs bi-Pentium III (19 fonctionnant sur 933 MHz et 14 sur 500 MHz), sous Linux 2.2.

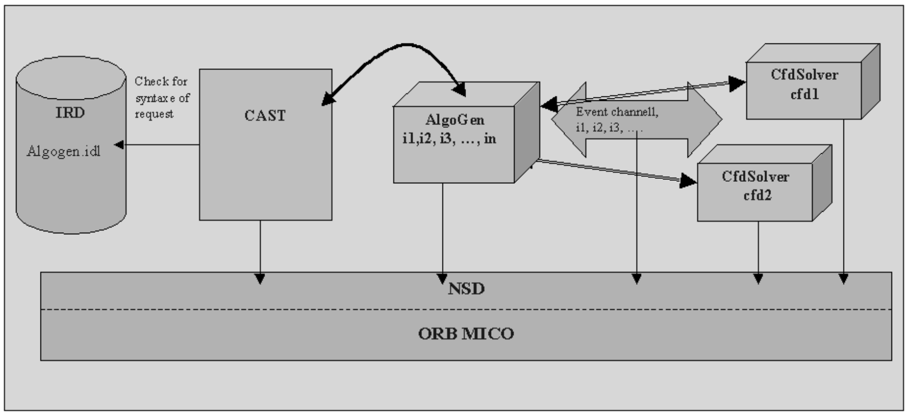

Figure 7. *Clients et serveurs pour l' application d'optimisation de forme*

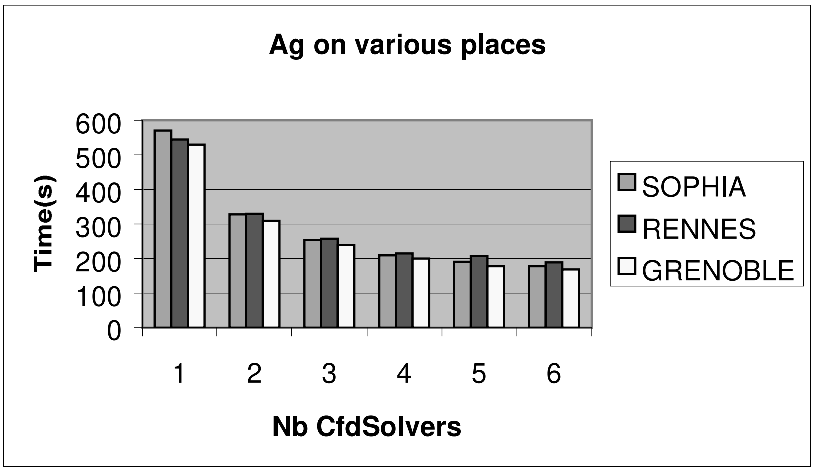

Tableau 2. *Performance à l'exécution de l'exemple d'optimisation*

Plusieurs instances du solveur CFD fonctionnent en parallèle pour traiter les solutions candidates générées par l'algorithme génétique. Chaque solveur CFD se voit attribué plusieurs processeurs pour prendre en compte la décomposition du domaine autour de l'aile, qui est divisée de manière statique en quatre sous-domaines (Figure 8). On a un processeur pour chaque sous-domaine. Le flux d'air autour de l'aile est cal-

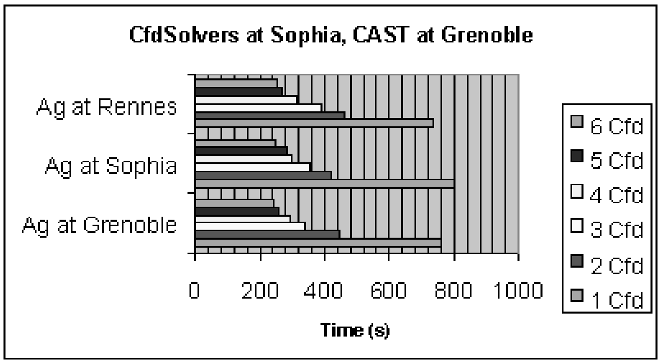

Tableau 3. *Performance pour l'exemple 2D*

culé chaque fois que l'aile est modifiée par l'algorithme d'optimisation, c'est-à-dire pour chaque boucle d'optimisation. Chaque boucle d'optimisation génère plusieurs solutions candidates, qui sont alors évaluées en parallèle, c'est-à-dire que le maillage et le flux sont recalculés pour chaque boucle d'optimisation et pour chaque solution générée par l'algorithme d'optimisation.

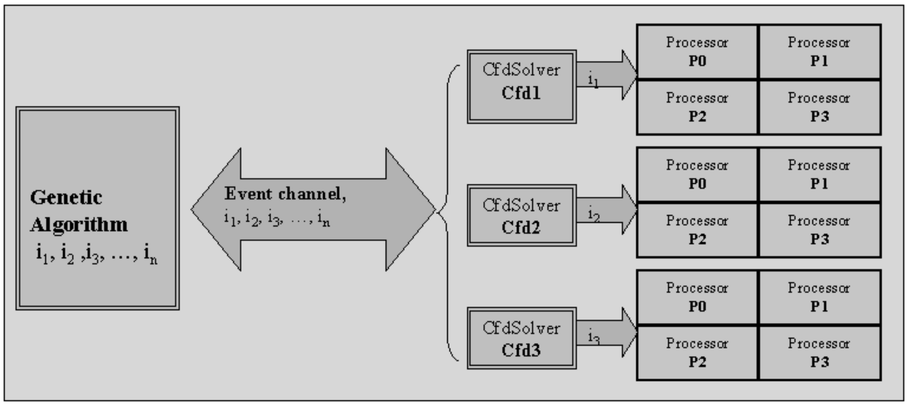

Figure 8. *Interaction entre l'algorithme génétique et les solveurs CFD*

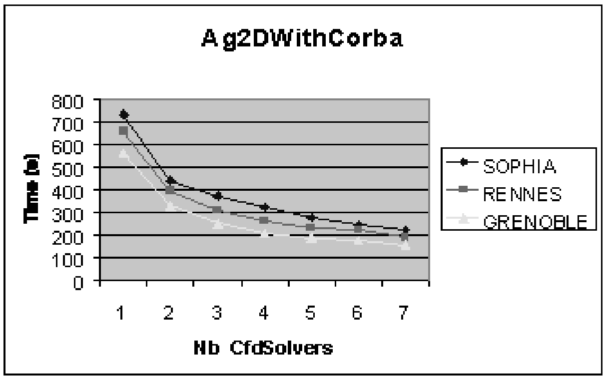

Tableau 4. *Performances pour l'exécution de l'exemple airfoil*

Dans cet exemple, les solveurs CFD sont les tâches qui prennent le plus de temps CPU. Il est donc essentiel de dupliquer et de parallèliser le calcul du maillage et du flux d'air autour du profil aérodynamique. C'est la raison pour laquelle il y a plusieurs instances du solveur exécutés en parallèle pour chaque solution candidate, en fait quatre pour un profil d'aile. De même, plusieurs solutions candidates sont générées par l'algorithme d'optimisation génétique et sont évaluées en parallèle. L'affectation des processeurs est réalisée statiquement : un nombre requis de processeurs est demandé sur chaque cluster de PC avant l'exécution de l'application. Divers tests de performance ont été réalisés pour cette application. Les résultats sont donnés dans le Tableau 2 pour différents sites d'exécution de l'algorithme génétique (Ag), quand le logiciel CAST et les solveurs CFD sont situés à Grenoble, et en fonction du nombre de solveurs CFD. Le Tableau 3 indique les résultats lorsque ces derniers fonctionnent à Sophia-Antipolis. Les clusters de PC sont connectés par le réseau VTHD à haut débit [VTH] (schéma 9).

La plate-forme gère également le couplage de modules non compatibles CORBA avec des modules compatibles CORBA. Cela permet une transition en douceur des logiciels existant et des bibliothèques mathématiques vers une technologie avancée de logiciel réparti. Cette technologie conserve les investissements passés des utilisateurs dans des logiciels spécialisés, permettant leur intégration en douceur avec une technologie plus récente. Elle utilise la notion de " serveur de groupe " qui permet l'enrobage de plusieurs codes non-CORBA simultanément au moyen d'un enrobeur unique. Ceci diffère de l'approche totalement compatible avec CORBA utilisée ci-dessus qui emploie des " enrobeurs de modules " dédiés. Dans ce cas, chaque module CORBA a son propre enrobeur qui l'interface avec la couche CORBA et gère l'appel des clients

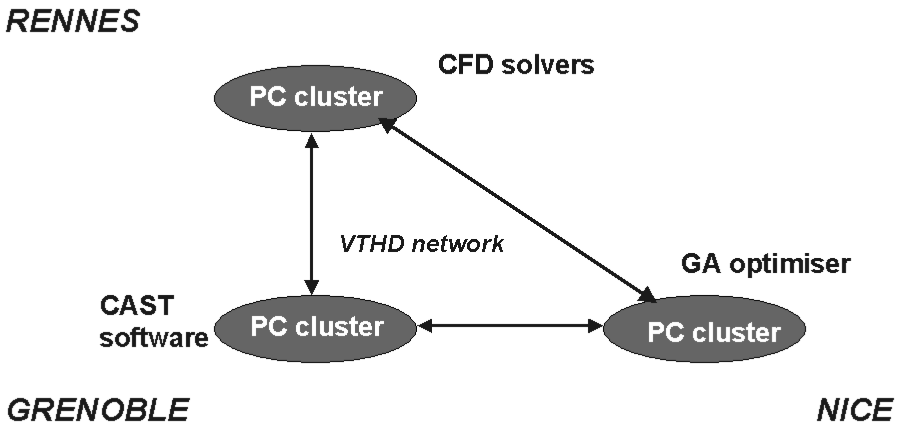

Figure 9. *L' application d' optimisation dans l' environnement de metacomputing*

distants. En fait, les premiers tests pour le nouveau prototype CAST 2.0 étaient gérés en utilisant une version non CORBA de l'optimiseur BCGA et des fonctions analytiques enrobées (c'est-à-dire conformes à CORBA). Le prototype a été réalisé à la fois en C++ et en Java du côté serveur pour évaluer les deux options. Des tests de performance ont montré un handicap prévisible d'un ordre de grandeur en défaveur de Java. Cela n'apparaît cependant pas comme un défaut : d'abord parce que le langage C++ est largement répandu et normalisé ; ensuite, parce que, pour des raisons de portabilité, cette application JAVA du " groupe serveur " a été retenue pour l'intégration de futurs modules non-CORBA. Des tests complémentaires sont effectués avec l'optimisation de la portance d'une aile multi-éléments (Figure 10). L'aile comprend des becs de bord d'attaque, le corps principal et des volets de bord de fuite. Le but est de maximiser la portance dans la configuration d'atterrissage et de minimiser la traînée en configuration de croisière, c'est-à-dire une application à objectifs multiples. Un algorithme génétique parallèle est ici combiné avec la théorie des jeux pour mettre en œuvre l'application. La théorie des jeux est appliquée pour optimiser la position combinée des becs et des volets, qui sont considérés comme les deux joueurs. L'algorithme génétique parallèle est utilisé pour optimiser la position soit des becs soit des volets quand l'autre est fixe. Deux niveaux de parallélisation sont donc utilisés. Combiné avec une approche multi-grille hiérarchique pour modéliser le flux autour de l'aile, un troisième niveau de parallélisme est introduit dans [WAN 01]. Comparés à un cluster de PC privé, des tests non-répartis ont déjà généré des facteurs d'accélération de 15 sur le cluster de PC de l'INRIA Rhône-Alpes à Grenoble. De plus, on s'attend à ce que l'utilisation d'un solveur CFD industriel (c'est-à-dire non transférable) permette d'améliorer encore ces performances.

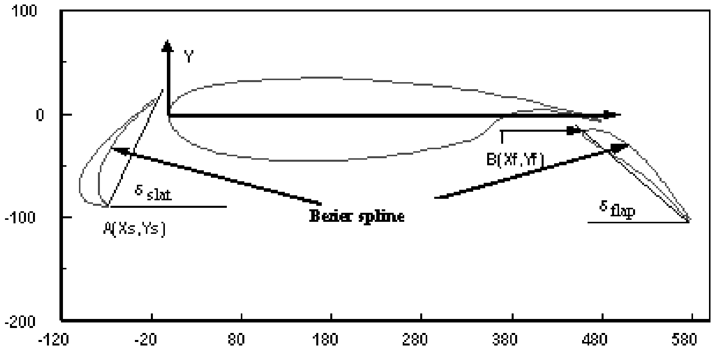

Figure 10. *Géométrie de l'exemple multi-éléments*

5. Comparaison avec d'autres approches et projets

Qu'on les nomme plates-formes d'intégration, environnements de couplage de code ou de résolution de problèmes, des prototypes et des infrastructures ont été développés dans divers projets qui explorent le domaine du calcul à haute performance, en particulier pour les applications de simulation et d'optimisation. Les sciences de l'environnement, les industries aérospatiale, navale et de l'automobile, tout comme les logiciels de visualisation pour les satellites et l'imagerie biomédicale sont les premiers candidats à de tels environnements. La plupart des approches sont consacrées à des aspects spécifiques de certains domaines, par exemple l'optimisation de la structure d'un aéronef. Parmi les thèmes critiques abordés figurent l'interopérabilité et l'intégration de logiciels, ainsi que les interfaces de données, par exemple les questions de compatibilité de maillage.

Ce dernier problème n'est pas abordé dans cet article. Nous nous concentrons, dans le paragraphe suivant, sur quelques projets récents ou en cours qui se réfèrent à l'intégration de logiciels dans des environnements de calcul distribué, et principalement à des projets européens.

FRONTIER (BAe, G.B.) a développé un environnement pour l'optimisation multi-objetifs appliquée à la conception d'ailes. Cet environnement n'utilise pas les caractéristiques dynamiques dans CORBA, mais utilise beaucoup JAVA, définit les séquences de processus, et utilise intensivement des composants logiciels sur étagères (modules COTS). Cet environnement est indépendant de la plate-forme (mais principalement Unix et Windows NT), et les échanges de données sont fondés sur des fichiers de données, les processeurs devant simplement avoir une adresse IP [SPI 99].

Dans un secteur similaire, le projet JULIUS (BAe, Dassault-Aviation) réalise une interface CORBA spécifique pour la visualisation et des outils de CAO [MUR 99]. AUTOBENCH et CODINE (GMD, BMW, Allemagne) utilisent des graphes de processus, ainsi qu'une interface CAD_PDM. Les applications impliquent de l'aérodynamique, de la CFD, de la CSM, de l'acoustique, et le standard d'échange STEP. Ceci a été largement testé dans des applications automobiles et des simulations d'accidents [THO 99]

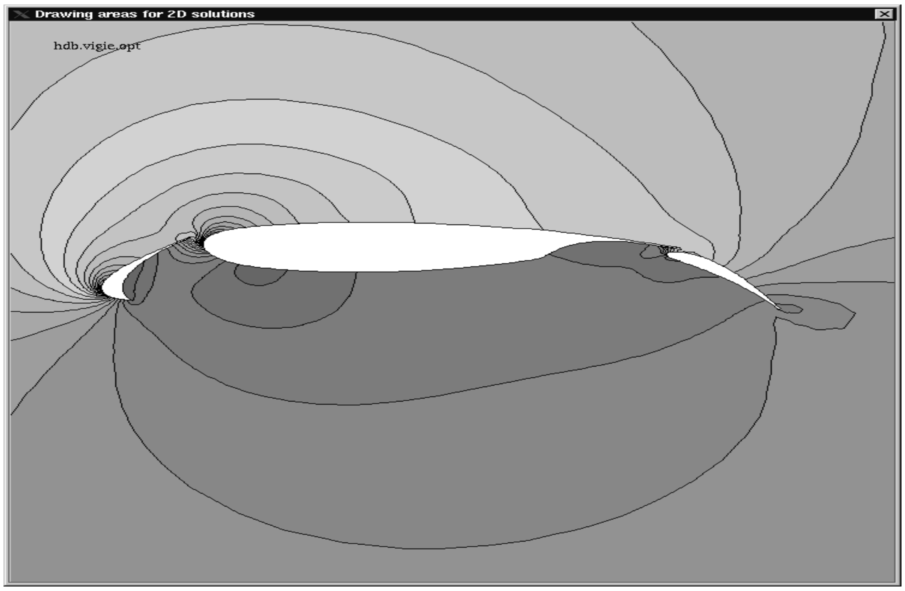

Figure 11. *Optimisation multi-éléments en configuration d'atterrissage (lignes isomach)*

Parmi les autres projets européens qui présentent un intérêt particulier figurent : SOFT_IT (INRIA, Thomson-CSF, Simulog) était un projet européen consacré à l'intégration de logiciels de simulation électromagnétique [COR 99]. PACHA (INRIA, Spacebel, ...) était dédié au développement d'une plate-forme d'intégration basée sur des clusters de PC. JACO3 (INRIA, Aérospatiale) porte sur l'intégration de codes parallèles et est appliqué à des cas-tests électromagnétiques en conception aérospatiale. Il utilise PACO et JAVA [ALL 99]. TENT (DLR, MTU, DASA, Allemagne) était un projet de simulation numérique pour les turbines aéronautiques [STR 99]. Le projet PARIS à l'IRISA, se penche sur les applications de simulation numérique répartie et sur la gestion globale de ressources dans des environnements de metacomputing. Il propose des extensions pour CORBA afin de l'adapter aux composants parallèles tels que des objets répartis (PACO), la gestion globale de ressources (CPU, mémoire, et disques) sur des clusters de PC, de la gestion de programmes parallèles, utilisant une mémoire virtuelle répartie [PRI 99].

Le projet GLOBUS bien connu (Institut des Sciences de l'Information, Université de Californie du Sud, Marina del Rey, USA) propose une infrastructure globale pour le metacomputing. Il fournit des services génériques, comme l'accès transparent à divers ordinateurs distants [KES 99].

Cette brève vue d'ensemble montre qu'aucun des projets mentionnés ne traite particulièrement de la spécification formelle d'applications complexes comme le fait la plate-forme d'intégration CAST . Par conséquent, même si plusieurs prototypes se servent d'une couche CORBA comme d'une structure de communication et de synchronisation, aucun d'entre eux ne comprend à la fois une approche des spécifications d'application de haut niveau et un environnement CORBA de calcul réparti. En outre, aucun d'entre eux ne traite explicitement du couplage transparent de codes compatibles et non compatibles avec CORBA simultanément. Cela semble cependant d'une importance primordiale si on doit utliliser des bibliothèques de logiciels existantes. CAST peut ainsi garantir une transition en douceur des bibliothèques numériques existantes et des librairies mathématiques vers les plates-formes d'intégration avancées. De plus, un certain nombre de projets importants ne font aucun usage des facilités d'appel dynamique de CORBA [SPI 99], comme l'interface d'appels dynamiques (DII) et le répertoire des interfaces dynamiques (DRI). Si cette restriction semble raisonnable dans des applications routinières, elle devient très restrictive dans des applications d'optimisation où le nombre et la variété des modules d'optimisation mis à la disposition des utilisateurs leur donnent la possibilité de passer d'un module à l'autre dynamiquement. Cela constituait un souci dans le projet DECISION [BLA 99]. Cela ne peut être appliqué que dans des plates-formes où les différents modules partagent tous la même interface (ce qui est peu réaliste) lorsque l'utilisation statique de CORBA est employée. Dans le cas contraire, la plate-forme doit fournir un autre niveau de référence, c'est-à-dire des spécifications qui sont indépendantes des définitions de modules. Ceci est réalisé au niveau de la spécification dans CAST, qui fournit une manière facile de définir les applications en termes de tâches logiques et décomposables hiérarchiquement. Par conséquent, le changement de module n'est qu'une affaire de changement de référence. Il est réalisé sans aucune modification des spécifications de haut niveau. Cela permet, à son tour, l'appel à des modules distants de manière transparente, ainsi que le basculement entre des modules qui ne sont pas situés sur le même site. En fin de compte, il ne semble pas que les projets mentionnés ci-dessus aient utilisé au départ une approche orientée objet pour la conception et l'exécution de leur environnement de résolution des problèmes. Cette approche est connue pour garantir le passage à l'échelle, l'extensibilié fonctionnelle, la modularité, la maintenabilité et l'interopérabilité. La plate-forme d'intégration CAST a été conçue avec ces objectifs en vue et est réalisée en C++. Ceci a conduit à un code très compact dont la maintenance est facile. Il a été développé en quatre hommes-années et a été testé sur des exemples d'optimisation d'aile mono et multi-éléments mettant en œuvre des algorithmes génétiques pour l'optimisation et des solveurs CFD avec remaillage dynamique des profils aérodynamiques.

6. Conclusion

Les projets d'ingénierie complexe font appel à des disciplines variées qui doivent interagir de manière efficace pour atteindre les objectifs d'un projet. Ces multiples expertises peuvent avoir des effets en cascade sur les différentes parties d'un projet, même si elles n'opèrent que des changements mineurs sur des aspects très locaux de la conception. Ceci est particulièrement vrai dans les projets aéronautiques où la cohérence de la conception des systèmes et de la cellule est primordiale. Ces interactions complexes nécessitent des spécifications précises et vérifiables. Ceci peut être résolu par des techniques de spécification formelle comme les algèbres de processus. De plus, la grande variété des disciplines impliquées requiert des outils de calcul sophistiqués qui peuvent être connectés à des réseaux d'ordinateurs répartis dans le monde entier. Les technologies du calcul distribué sont donc également nécessaires pour assurer l'intégration des résultats d'un projet. Ceci peut être assuré en faisant appel aux techniques de développement de logiciel orienté-objet et de communication comme CORBA. Cet article présente une plateforme d'intégration appelée CAST. Elle combine les techniques de spécification formelle et de dévelopement logiciel distribué orienté-objet pour spécifier, configurer, déployer et exécuter des applications d'ingénierie complexe de manière unifiée. La plateforme a été développée sur un réseau de stations de travail qui communiquent à travers une couche CORBA. Elle a été testée avec succès sur des applications d'optimisation multi-critères impliquant plusieurs grappes de PC connectées à un réseau rapide à 2 gigabits/sec. Les programmes applicatifs et leurs interactions sont associés de manière transparente à des composants logiciels. Ceux-ci sont réalisés en utilisant des techniques de programmation par objets. Le code utilisateur peut inclure des modules orientés-objet, par exemple des subroutine Fortran, ou des méthdes C++. CAST génère alors automatiquement les interfaces et les échanges de paramètres. CAST offre également une interface utilisateur de haut niveau pour spécifier les applications, ce qui rend totalement transparent les aspects formels de la spécification et les particularités du calcul distribué. CAST permet enfin de coupler du code CORBA avec du code non CORBA. Ceci permet une transition aisée des codes applicatifs et des bibliothèques mathématiques existantes vers les environnements de résolution de problèmes et de métacomputing modernes, ainsi qu'un lien standardisé avec les logiciels tiers qui respectent les recommandations CORBA. La communauté d'utilisateurs visée est constituée des chefs de projet et des ingénieurs des bureaux d'études spécialisés dans un domaine particulier (aérodynamique, électromagnétique, etc).

Bien que le métacomputing ait attiré beaucoup d'attention dernièrement, les applications complexes qui nécessitent de très gros moyens de calcul ont encore besoin d'environnements à haute-performance faciles à utiliser [Bar 00]. Il en va ainsi de la co-simulation de circuits intégrés (projet CADNET), de la simulation numérique des systèmes de propulsion (projet NPSS), de la simulation multi-disciplines d'avions de ligne, comme le Digital Dynamic Aircraft (projet AMAZE). En effet, les systèmes de simulation envisagés aujourd'hui par les constructeurs aéronautiques comprennent simultanément des outils de simulation et d'optimisation en mécanique des fluides, en résistance des matériaux, en électromagnétique et en acoustique. Ceci doit être utilisé depuis les phases initiales de conception jusqu'à la mise au point finale de la dynamique de vol et des systèmes dans des environnements fortement contraints. Cela représente des exigences et des charges sans précédent sur les moyens matériels et logiciels des environnements de résolution de problèmes. Bien qu'il lui manque encore des fonctionnalités (en matière de répartition de charge, de qualité de service, et d'authentification par exemple), la plateforme CAST fournit des services qui en font un bon candidat pour la construction d'un environnement de résolution de problemes à haute performance dans un système de métacomputing dédié aux applications de simulation multidisciplines. La documentation du projet CAST, comprenant un manuel d'utilisation en anglais, est disponible en ligne à l'adresse : `http ://www.inrialpes.fr/sinus/cast`. Des informations supplémentaires sur le projet sont disponibles à l'adresse : `http ://www.inrialpes.fr/sinus`.

Remerciements

Les auteurs remercient Françoise Baude de l'Université de Nice Sophia-Antipolis et Laure Tardivet qui ont assuré spontanément et bénévolement la traduction française de la publication originale. Ils remercient également Alain Dervieux et Jean-Antoine Desideri du projet SINUS à l'INRIA Sophia-Antipolis, ainsi que Jacques Périaux de Dassault-Aviation (Pôle Scientifique), pour leur soutien indéfectible et leurs nombreux conseils

7. Bibliographie

[ALL 99] ALLEON G., « The JACO3 project », *Soft-IT Workshop "Next generation of interoperable simulation environments based on CORBA"*, INRIA-ONERA-Simulog, June 1999.

[And 98] ANDREOLI J.M. ET AL., « The coordination technology area at XRCE Grenoble : research in support of distributed cooperative work », rapport, 1998, Xerox Research Center Europe.

[Bal 98] BALTER R. ET AL., « Architecturing and configuring distributed applications with OLAN », *Proc. IFIP International Conference on Distributed Systems platforms. Middleware'98*, 1998.

[Bar 00] BARNARD S. ET AL., « Large-scale distributed computational fluid dynamics on the Information Power Grid using Globus », *Proc. NASA HPCC/CAS Workshop.*, NASA Ames Research Center, February 2000.

[BLA 99] BLACHON J., NGUYEN T., « DECISION prototype integration platform : CAST CORBA prototype », *Soft-IT Workshop "Next generation of interoperable simulation envi-*

ronments based on CORBA", INRIA-ONERA-Simulog, 1999, Invited Lecture.

[COR 99] CORNU T., « The SOFT_IT project », *Soft-IT Workshop "Next generation of interoperable simulation environments based on CORBA"*, INRIA-ONERA-Simulog, 1999.

[DAT] http ://ptolemy.eecs.berkeley.edu/papers/97/softwareSynth/.

[DCE] http ://www.opengroup.org/dce/dce_topics.htm.

[DOC] http ://www.xerox.com/go/xrx/xrx_research/AX_6.jsp.

[GIR 99] GIRONA S., « Metacomputing tools for stochastic analysis : Promenvir », *European Research Conference Series on Advanced Environments and Tools for High Performance Computing. "Problem-Solving Environments : Infrastructure and Prototypes"*, San Feliu de Guixols, Spain, June 1999.

[ILU98] « XEROX ILU Reference manual », 1998, ftp ://ftp.parc.xerox.com/pub/ilu/2.0a14/manual-html.

[KEA 99] KEANE A., « The creation of a multi-analysis wing design environment », *European Research Conference Series on Advanced Environments and Tools for High Performance Computing. "Problem-Solving Environments : Infrastructure and Prototypes"*, San Feliu de Guixols, Spain, June 1999.

[KES 99] KESSELMAN C., « Globus and metacomputing infrastructure », *European Research Conference Series on Advanced Environments and Tools for High Performance Computing. "Problem-Solving Environments : Infrastructure and Prototypes"*, San Feliu de Guixols, Spain, June 1999.

[Lop 00] LOPEZ I. ET AL., « NPSS on NASA's IPC : using Corba and Globus to coordinate multidisciplinary aeroscience applications », *Proc. NASA HPCC/CAS Workshop.*, NASA Ames Research Center, February 2000.

[MAR 98] MARCO N., LANTERI S., « A two-level parallelization strategy for genetic algorithms applied to shape optimum design », rapport n° 3463, July 1998, INRIA.

[MIC] http ://www.micro.org.

[MIL 83] MILNER R., « Calculi for synchrony and asynchrony », *Theoretical Computer Science*, vol. 25, n° 3, 1983.

[MPI] « MPI : Message Passing Interface », http ://www-unix.mcs.anl.gov/mpi/index.html.

[MUR 99] MURPHY J., « The JULIUS project », *Soft-IT Workshop "Next generation of interoperable simulation environments based on CORBA"*, INRIA-ONERA-Simulog, 1999.

[NAG98] NAG Ltd, « Iris Explorer. Reference Manual », 1998.

[NGU 02] NGUYEN T., PLUMEJEAUD C., « An integration platform for metacomputing applications », Submitted for publication, 2002.

[Obj98] Object-Oriented Concepts, Inc, « ORBacus for C++ and Java. Version 3.1.1 », 1998.

[OMG98] « The Common Object Request Broker : architecture and specification. Revision 2.2 », rapport n° 98-03-11, 1998, Object Management Group.

[Pet] http ://www.daimi.aau.dk/˜petrinet/.

[PRI 99] PRIOL T., « Projet PARIS : Programmation des systèmes parallèles et distribués pour la simulation numérique distribuée », INRIA, May 1999.

[PVM] « PVM : Parallel Virtual Machine », http ://www.epm.ornl.gov/pvm/.

[SAN 00] SANG J., KIN C., LOPEZ I., « Developing Corba-based distributed scientific applications from legacy Fortran programs », *Proc. NASA HPCC/CAS Workshop.*, NASA Ames

Research Center, February 2000.

[SIN] http ://www.inria.fr/Equipes/SINUS-eng.html.

[SPI 99] SPICER D., « FRONTIER : an environment for industrial multi-objective optimisation », *European Research Conference Series on Advanced Environments and Tools for High Performance Computing. "Problem-Solving Environments : Infrastructure and Prototypes"*, San Feliu de Guixols, Spain, June 1999.

[STR 99] STRIETZEL M., « The TENT project », *Soft-IT Workshop "Next generation of interoperable simulation environments based on CORBA"*, INRIA-ONERA-Simulog, 1999.

[THO 99] THOLE C., « Autobench : development of a PSE for the automotive industry », *European Research Conference Series on Advanced Environments and Tools for High Performance Computing. "Problem-Solving Environments : Infrastructure and Prototypes"*, San Feliu de Guixols, Spain, June 1999.

[VTH] « Projet RNRT VTHD », http ://www.telecom.gouv.fr/rnrt/projets/pres_d74_ap99.htm.

[WAN 01] WANG J., « Distributed design optimisation using genetic algorithms and game theory and related applications to high-lift CFD problems in aerodynamics », PhD thesis, Université Pierre et Marie Curie, Octobre 2001.

[WOR] http ://www.do.isst.fhg.de/workflow/index_e.html.