I3DS : vers des simulations dynamiques sur l'Internet

Victor Alessandrini

I3DS signifie *IDRIS Internet Interface for Dynamical Simulations*. Les simulations dynamiques sont celles pour lesquelles on dispose d'une interface de suivi des résultats intermédiaires de contrôle, ainsi que de la capacité d'injecter de manière dynamique des nouvelles données dans la simulation. Celle-ci doit, bien entendu, être capable d'adapter sa réponse à l'évolution des jeux de données.

Ce sujet est d'une importance certaine dans un contexte d'évolution vers des simulations de plus en plus complexes, lorsque le traitement de l'information s'effectue en même temps que l'acquisition des données, ou bien lorsque des décisions doivent être prises au fur et à mesure de l'avancement de la simulation. Certains environnements de programmation répartie comme Cactus intègrent des bibliothèques permettant le suivi et le pilotage des codes. Aux USA, la NSF finance un programme national appelé DDDAS (*Dynamic Data Driven Application Software*). Enfin, un certain nombre de projets de R&D dans le domaine des grilles cherche à intégrer cette fonctionnalité dans leurs environnements de programmation.

▶ 1 – Les objectifs de l'IDRIS

Le suivi et le pilotage dynamique des simulations intéressent l'IDRIS pour trois raisons :

- **Les simulations très longues.** La disponibilité des mémoires de plus en plus grosses — presque 1 téraoctet sur le nouveau système IBM — stimule les simulations de plus en plus complexes, où les temps d'exécution augmentent beaucoup plus que de la simple proportionnalité à la taille du problème. Aujourd'hui, des simulations de quelques jours commencent à devenir courantes parmi nos utilisateurs. Sur le nouveau système IBM, nous prévoyons de supporter des simulations de plus de 10 jours. Il va de soi qu'il est exclu de lancer une simulation et de découvrir dix jours plus tard que quelque chose ne s'est pas passé comme prévu. Un outil permettant *le suivi et le pilotage coopératif* d'une simulation par la totalité des chercheurs concernés par son évolution (parfois plusieurs dizaines) est donc nécessaire.

- **La possibilité d'incorporer dans notre environnement de production un certain nombre d'applications d'aide à la décision, à usage externe.** Ces applications demandent d'importantes capacités de calcul, elles manipulent des grosses masses de données, mais les informations critiques à faire parvenir aux décideurs sont relativement limitées. En revanche, la réactivité de l'application aux nouvelles requêtes est critique.

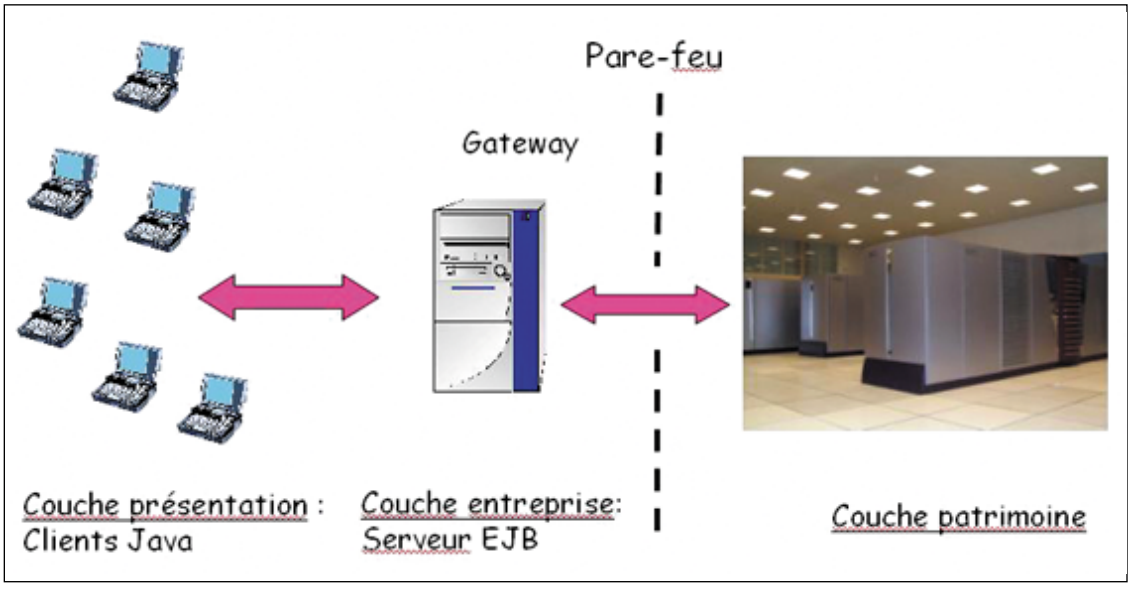
- **L'exécution d'applications en mode ASP** (*Application Service Providers*). Dans ce mode de fonctionnement, un utilisateur se connecte à une application sur l'Internet et non à l'environnement de production de l'IDRIS. Cela permettrait de cacher les supercalculateurs et la haute technicité de notre environnement à des utilisateurs qui n'ont pas vocation à devenir des experts en calcul intensif.

L'interface logicielle I3DS a donc été développée dans cette optique, en adoptant un certain nombre de critères à la base de sa conception :

- **Intégration des applications patrimoines.** Il s'agit de pouvoir projeter sur l'Internet les applications existantes. Toutefois, l'application elle-même est nécessairement scrutée et manipulée. Nous avons donc développé *une méthodologie d'intrusion minimale* qui permet d'aboutir à nos fins. Ceci est probablement l'aspect le plus original de I3DS.

- **Sécurité.** Les applications ouvertes sur l'Internet doivent respecter des normes de sécurité informatique de même qualité et rigueur que celles utilisées dans notre Intranet fortement protégé.

Fig. 1



► 2 – L'architecture de I3DS

L'architecture de I3DS adopte un paradigme devenu classique pour l'intégration des applications d'entreprises. Elle est constituée de trois modules logiciels couplés (*three tier architecture*) qui s'exécutent sur des plates-formes différentes (voir figure 1) :

- La « couche patrimoine », à l'intérieur de l'Intranet de l'IDRIS : il s'agit d'un serveur qui encapsule et exécute l'application cible, sur les machines de calcul.

- La « couche entreprise » : constituée des éléments stratégiques nécessaires à l'articulation de l'Intranet sécurisé avec l'Internet vulnérable. Elle s'exécute sur une machine dédiée (la passerelle ou *Gateway*) située à l'extérieur des pare-feu de l'IDRIS.

- La « couche de présentation » : il s'agit d'un client comportant une interface graphique utilisateur qui s'exécute sur n'importe quelle plate-forme ou PC (Windows, Linux) possédant une machine virtuelle JAVA.

Le client et la passerelle I3DS utilisent JAVA comme langage de programmation, alors que le serveur, qui s'exécute sur une machine de calcul, est écrit en C — et parfois en C + + — mais en cherchant à simplifier au maximum l'interfaçage avec Fortran. CORBA est utilisé comme *middleware* de communication entre les trois modules. D'une part, CORBA est très bien intégré dans JAVA. D'autre part, des projections assez matures de CORBA sur les langages scientifiques de programmation C et C + + sont disponibles aujourd'hui.

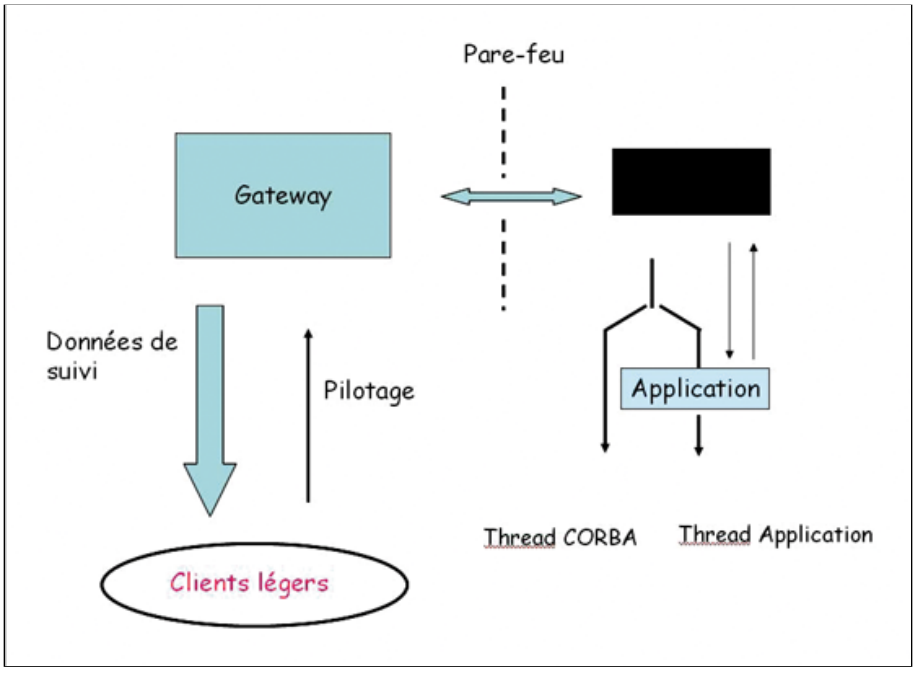
Un schéma plus précis de l'architecture de I3DS est donné dans la figure 2. Nous décrirons par la suite les fonctionnalités et les rôles de chacun de ces trois modules logiciels.

▶ 3 – Le serveur I3DS

Le principe fondamental à la base du choix de l'architecture du serveur I3DS est de cacher totalement l'existence et la complexité de la programmation réseau aux développeurs d'applications scientifiques. L'application ne doit absolument pas être au courant du fait qu'elle est suivie et pilotée. Le serveur I3DS encapsule donc l'application en séparant en dehors de celle-ci tous les éléments liés au couplage avec le service de suivi et de pilotage qu'il fournit.

La figure 2 montre le serveur I3DS pour une application monoprocesseur ou parallèle à mémoire partagée (utilisant, par exemple, OpenMP). Dans ce cas de figure, un seul processus Unix est actif (l'architecture I3DS se généralise sans difficulté à des applications parallèles à mémoire distribuée utilisant MPI). La séparation des fonctionnalités réseau du reste de l'application s'effectue par l'existence, dès le lancement du serveur I3DS, de deux threads, un « thread CORBA » et un « thread application ».

Fig. 2



Le thread CORBA est en permanence à l'écoute des requêtes venant de l'Internet. Il prend en charge la « publication » à l'intention des clients éventuels, des données nécessaires au suivi de la simulation. Il gère aussi les requêtes de contrôle émanant des clients (changer tel ou tel paramètre, supprimer le calcul en cours et passer au jeu suivant de paramètres, renseigner le client sur les valeurs actuelles des paramètres). Ce thread n'est pas vu du tout par le programmeur de l'application.

Chaque fois qu'une demande de modification des paramètres arrive d'un client, une requête est postée par le thread CORBA à une file de requêtes. Le thread application est chargé de retirer, dans l'ordre, ces requêtes de la file d'attente et de relancer l'application avec les

nouveaux paramètres. Un client a, bien entendu, la possibilité de supprimer sa requête s'il estime qu'elle est devenue superflue. Le client a aussi la possibilité de décider qu'un jeu de paramètres doit continuer de manière ininterrompue et jusqu'à nouvel ordre.

Le programmeur de l'application n'a que deux choses à faire pour interfacer son code scientifique avec le serveur I3DS :

- Déclarer, d'une manière particulière, les variables dont les valeurs sont susceptibles d'être modifiées de manière dynamique et utiliser pour leur initialisation une bibliothèque d'entrées-sorties faisant partie de I3DS. Les modifications ultérieures de leurs valeurs sont totalement transparentes.

- Utiliser une opération de *push* définie dans une bibliothèque I3DS pour déposer dans un tampon mémoire prédéfini, et de manière périodique, les valeurs des données de suivi qui doivent être mises à la disposition des clients sur l'Internet.

C'est tout. Toutefois, le programmeur de l'application doit tout de même veiller à la réactivité correcte de son code face à une modification dynamique des données. Si, par exemple, il a décidé que la taille du problème est ouverte à un pilotage extérieur, il doit prévoir la réallocation des tableaux en cas de dépassement de leur taille.

► 4 – La passerelle et le client I3DS

Comme nous l'avons déjà signalé, la passerelle effectue l'intermédiation entre le client et le serveur. Lorsqu'un client est démarré, il commence par établir une connexion avec la passerelle afin de s'identifier. Si la passerelle est satisfaite de l'identité du client, la connexion est maintenue. Dans le cas contraire, elle est refusée.

Le processus d'authentification du client utilise JAAS (*Java Authentification and Authorisation Service*), aujourd'hui intégré dans la version 1.4 de JAVA. L'authentification du client peut se faire à l'aide d'identificateurs divers (mot de passe, clé publique). Une fois le client authentifié, son identité reste connue de la passerelle. Elle sera utilisée ultérieurement pour filtrer ses accès au serveur en fonction des autorisations d'accès qui lui sont accordées.

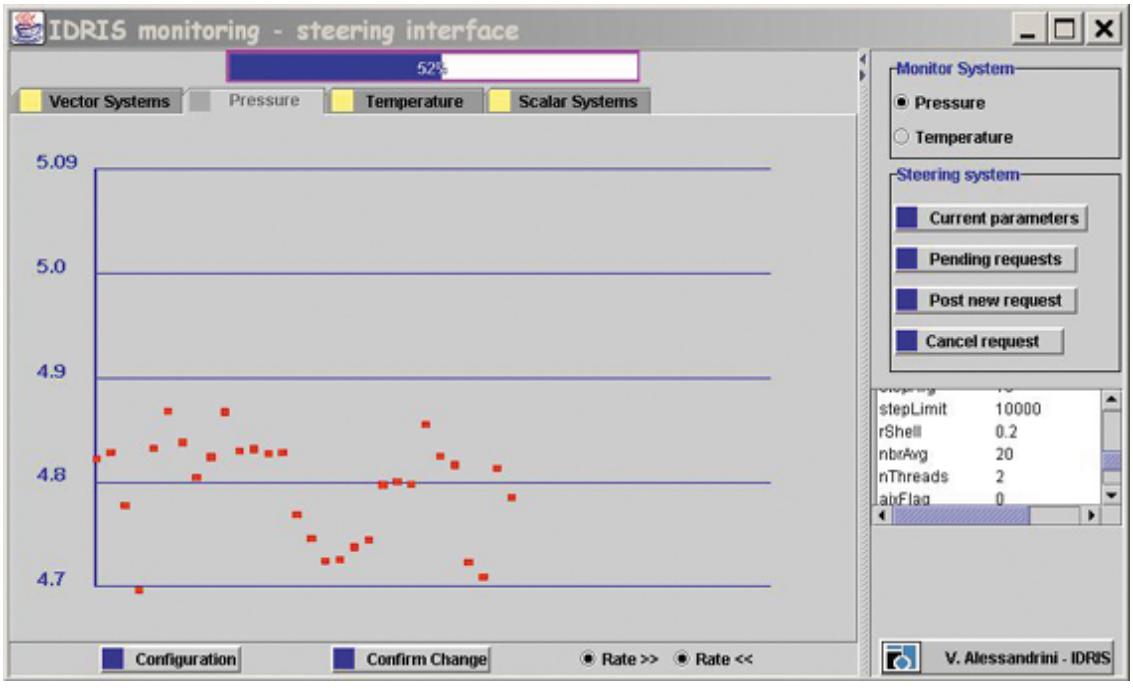
En effet, dans un environnement conçu pour un nombre arbitraire de clients, il est envisageable que, si en principe tous les clients ont le droit de suivre le déroulement de la simulation, seuls certains d'entre eux doivent pouvoir piloter son évolution. JAAS permet d'attribuer à chaque client des droits d'accès différents à chaque service individuel offert par le serveur. L'identité du client accompagne ses requêtes à la passerelle, ce qui permet à celle-ci de vérifier leur légitimité.

La passerelle a donc un rôle d'intermédiaire entre le client et le serveur à des fins de sécurité pour toutes les transactions de pilotage, initiées par le client. La passerelle s'occupe aussi d'envoyer sur l'Internet les données pour le suivi de la simulation, fournies par le serveur.

Pour les données associées au suivi de la simulation, le modèle de communication adopté est un modèle « publication – souscription » et non une communication « point à point » avec chaque client individuel, car il est plus performant pour un grand nombre de clients. La passerelle est en permanence en train d'accéder au serveur pour récupérer les données de suivi et de les publier sur l'Internet sous la forme d'un « événement » à l'intention des clients qui ont décidé « d'écouter » ce type d'événement. La version actuelle de I3DS utilise le *Event Service* de CORBA, mais le même résultat peut être obtenu à l'aide de JMS (*Java Messaging Service*).

Enfin, le client I3DS est basé sur une interface graphique écrite à l'aide de la bibliothèque Swing de JAVA, en cherchant un maximum de convivialité et d'ergonomie. En plus des contrôles de pilotage, elle propose une représentation 2D des données de suivi, adaptée à chaque application. La figure 3 montre l'interface actuelle pour le client, dans le cas du pilotage d'une simulation de dynamique moléculaire où les valeurs instantanées de la pression et de la température du système sont suivies à des intervalles réguliers.

Fig. 3



► 5 – État actuel et perspectives

Un prototype de I3DS pour des codes non-MPI — développé par l'auteur de cet article — existe depuis plusieurs mois. Des démonstrations ont été effectuées à plusieurs endroits, et en particulier à la journée Aristote sur le calcul réparti qui s'est tenue à l'École polytechnique en avril dernier. Les extensions du serveur pour les codes MPI sont en phase finale de développement.

Nous espérons, avant la fin de l'année, pouvoir commencer les tests sur le terrain, à l'aide d'un certain nombre d'utilisateurs précoces. Un article plus détaillé est en phase de rédaction, comportant une description complète des bibliothèques I3DS requises par les développeurs d'applications (écrites toutes en C pour faciliter l'interfaçage avec le Fortran). Leur utilisation devrait être relativement aisée, d'autant plus que le conseil du service de Support aux utilisateurs de l'IDRIS sera disponible.

Nous invitons donc les utilisateurs qui souhaitent bénéficier d'un accès précoce à cette technologie — ce qui nous serait utile pour la rendre plus mature — à nous faire parvenir une manifestation d'intérêt (email à l'auteur).