NICAISSE Sébastien Centre de Données astronomiques de Strasbourg

Analyse et mise en œuvre d'un cluster dans le domaine astronomique

Stage de troisième année 2002-2003

Remerciements

Je remercie Mr Schaaff pour m'avoir permis d'effectuer ce stage au sein du centre de données de Strasbourg (CDS) et pour m'avoir apporté de précieux conseils qui m'ont aidé à résoudre les problèmes que j'ai rencontrés.

Je remercie Mrs Derriere sébastien et Ochsenbein françois pour m'avoir conseillé dans la mise en place de ce projet et m'avoir exposé les bases en astronomie.

Je remercie également Mmes Faivre et Acousta, les deux infirmières qui se sont occupées de moi durant ce stage. Je m'excuse pour tous les désagréments que je leur ai causés.

Je n'oublierai pas non plus, Mr Faivre qui est venu me rechercher devant la mairie de Linghosheim, le soir du 1re Avril.

J'embrasse ma nièce Solène qui m'a fait connaître, lors de ce stage, les joies de la Varicelle. Mais, bon, j'espère qu'elle attendra quelques années pour m'initier à la scarlatine.

SOMMAIRE

Introduction

Les performances des ordinateurs de bureau standard, à base de processeur ont atteint un tel niveau qu'il est aujourd'hui possible de construire à partir de ces machines, des supercalculateurs aussi rapides que les calculateurs dédiés type CRAY, IBM, SP ou SGI Origin 2000. D'autre part, des problèmes technologiques ou simplement les coûts de certains composants (commutateurs pour l'accès à la mémoire par exemple) limitent les possibilités de passage à l'échelle des machines multiprocesseurs. Dans ce contexte, les clusters (ou grappes) de PC apparaissent comme une très bonne alternative aux autres moyens de calcul.

Le projet CoCat (co-processeur catalogue) n'est pas aussi ambitieux que de mettre en oeuvre un supercalculateur. Dans notre cas, nous nous contenterons d'élaborer un cluster composé de six machines interconnectées par un réseau de 100 Mb. Ce cluster accélérera la gestion des requêtes, effectuées par les astronomes du monde entier, sur des catalogues astronomiques.

Ce cluster sera une pièce maîtresse dans VizieR qui est l'un des trois principaux projets du CDS. Afin de choisir la solution la plus adaptée à notre problème, un état de l'art sur les clusters a été effectué. Cette étude nous montre les différents styles de clusters et les technologies de gestion des processus parallélisés. Puis, nous présentons une analyse à propos de l'organisation des données au sein du cluster. Ensuite, nous étudions la solution CLIC, les outils que cette distribution propose ainsi que les problèmes rencontrés. Enfin, nous prospectons l'avenir de ce cluster dans une grille de calcul.

1. Le Centre de Données astronomiques de Strasbourg (C.D.S).

1.1. Présentation du CDS :

Le Centre de Données astronomiques de Strasbourg (CDS), créé en 1972, est un centre de données dédié à la collection et à la distribution dans le monde entier de données astronomiques. Il est situé à l'Observatoire de Strasbourg. Le CDS est un laboratoire de l'Institut National des Sciences de l'Univers (INSU), rattaché au CNRS. Le personnel du CDS comprend 10 chercheurs, 8 ingénieurs, 5 techniciens, et plusieurs collaborateurs à contrat temporaire et invités.

Le CDS héberge la base de données SIMBAD, base de référence mondiale pour l'identification d'objets astronomiques. Son but est de:

- rassembler toutes les informations utiles, concernant les objets astronomiques, disponibles sous forme informatisée : références bibliographiques des différents objets, données d'observations produites par les observatoires du monde entier, au sol ou dans l'espace;
- mettre en valeur ces données par des évaluations et des comparaisons critiques ;
- distribuer les résultats dans la communauté astronomique ;
- conduire des recherches utilisant ces données.

Le CDS a signé des accords d'échanges internationaux avec les organismes suivants:

- NASA,
- National Astronomical Observatory (Tokyo, Japon),
- l'Académie des Sciences de Russie,
- le réseau PPARC Starlink au Royaume-Uni,
- l'Observatoire de Beijing (Chine),
- l'Université de Porto Allegre au Brésil,
- l'Université de La Plata en Argentine,
- InterUniversity Center for Astronomy and Astrophysics (Inde).

Le CDS joue, ou a joué, un rôle dans d'importantes missions astronomiques spatiales: contribuant aux catalogues d'étoiles guides (EXOSAT, IRAS, Hipparcos, HST, ISO, SAX), aidant à identifier les sources observées (Hipparcos, Tycho, ROSAT, SIGMA) ou organisant l'accès aux archives (IUE), etc. Le CDS contribue au XMM Survey Science

Center, sous la responsabilité de l'équipe "Hautes-Energies" de l'Observatoire de Strasbourg.

Le CDS est membre de la Fédération des Services d'Analyse de Données Astrophysiques et Géophysiques (FAGS).

1.2. Les services du CDS :

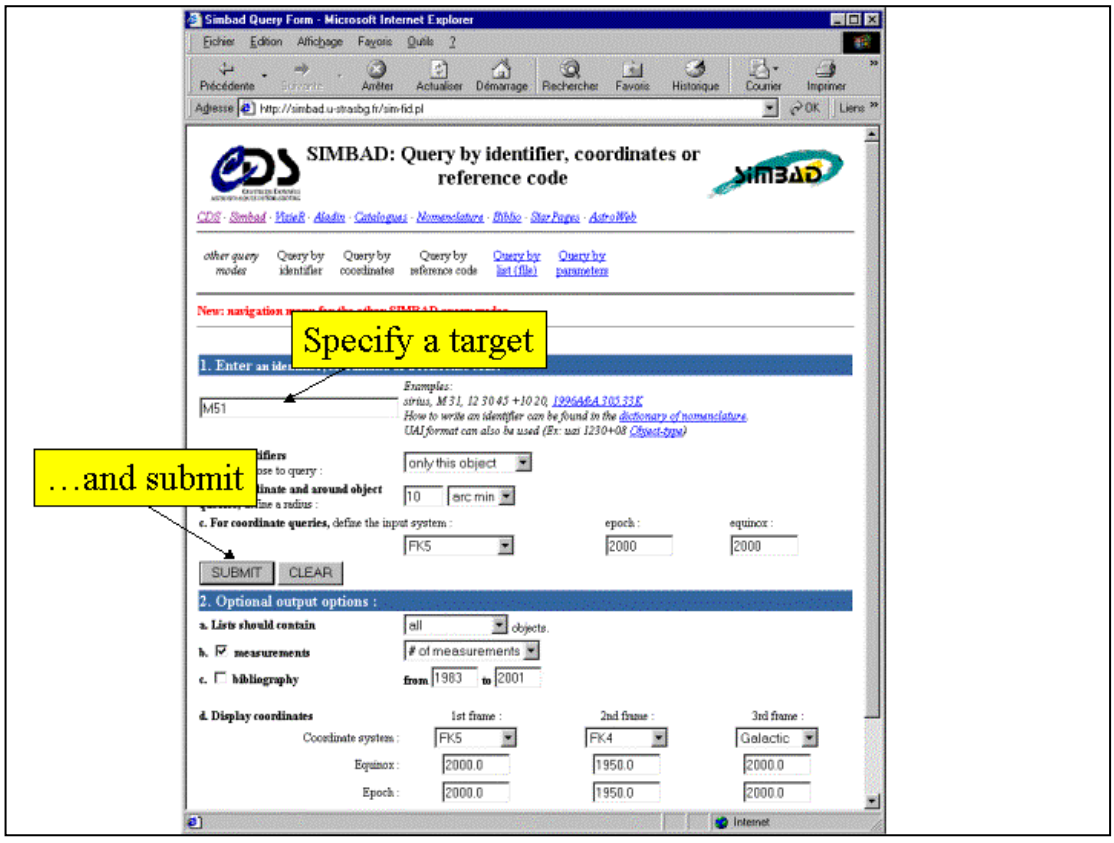
1.2.1. Simbad :

Simbad est une base de données de référence pour les identifications et la bibliographie d'objets astronomiques.

Simbad contient plus 7,5 millions d'identificateurs pour plus de 2,8 millions d'objets différents. Pour chaque objet, figurent dans la base quelques mesures (position, magnitude dans différents domaines de longueurs d'ondes), ainsi que les références bibliographiques où l'objet est cité (plus de 110 000 articles sont concernés).

Cet ensemble de données résulte d'un long travail d'identification croisée entre de nombreux catalogues, listes d'objets et articles de journaux, entrepris au début des années 1980, et constamment développé et mis à jour depuis.

Page web permettant d'effectuer une requête sur Simbad :



L'utilisateur peut choisir le format du fichier où sont entreposés les résultats de la requête. En effet, Simbad peut générer des fichiers html, xml ou xls (fichiers Excel).

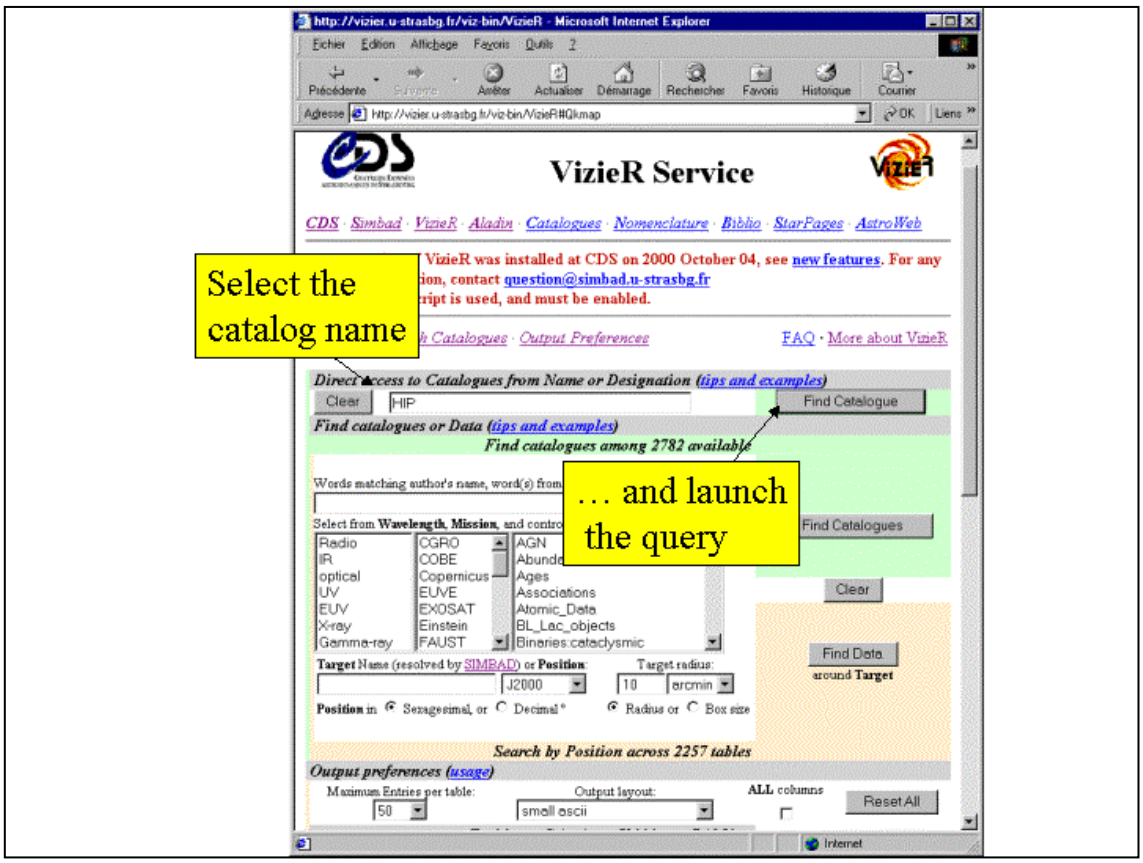
1.2.2. VizieR :

VizieR est une base de données rassemblant plusieurs milliers de catalogues astronomiques sous un format homogène.

Une description standardisée du contenu des catalogues permet leur inclusion dans un système de gestion de base de données (SGBD) relationnel. Un ensemble de liens, entre les tables de VizieR, et avec des services externes (bibliographiques, archives externes, serveurs d'images), permettent de naviguer entre les données des catalogues et d'autres données associées.

Il faut noter que les très grands catalogues (plus de 10^7 enregistrements) ne sont pas gérés par un SGBD relationnel. Pour des raisons de performances, des outils spécifiques ont été réalisés pour ces catalogues dont certains dépassent le milliard d'objets.

Page web permettant d'effectuer une requête sur VizieR :

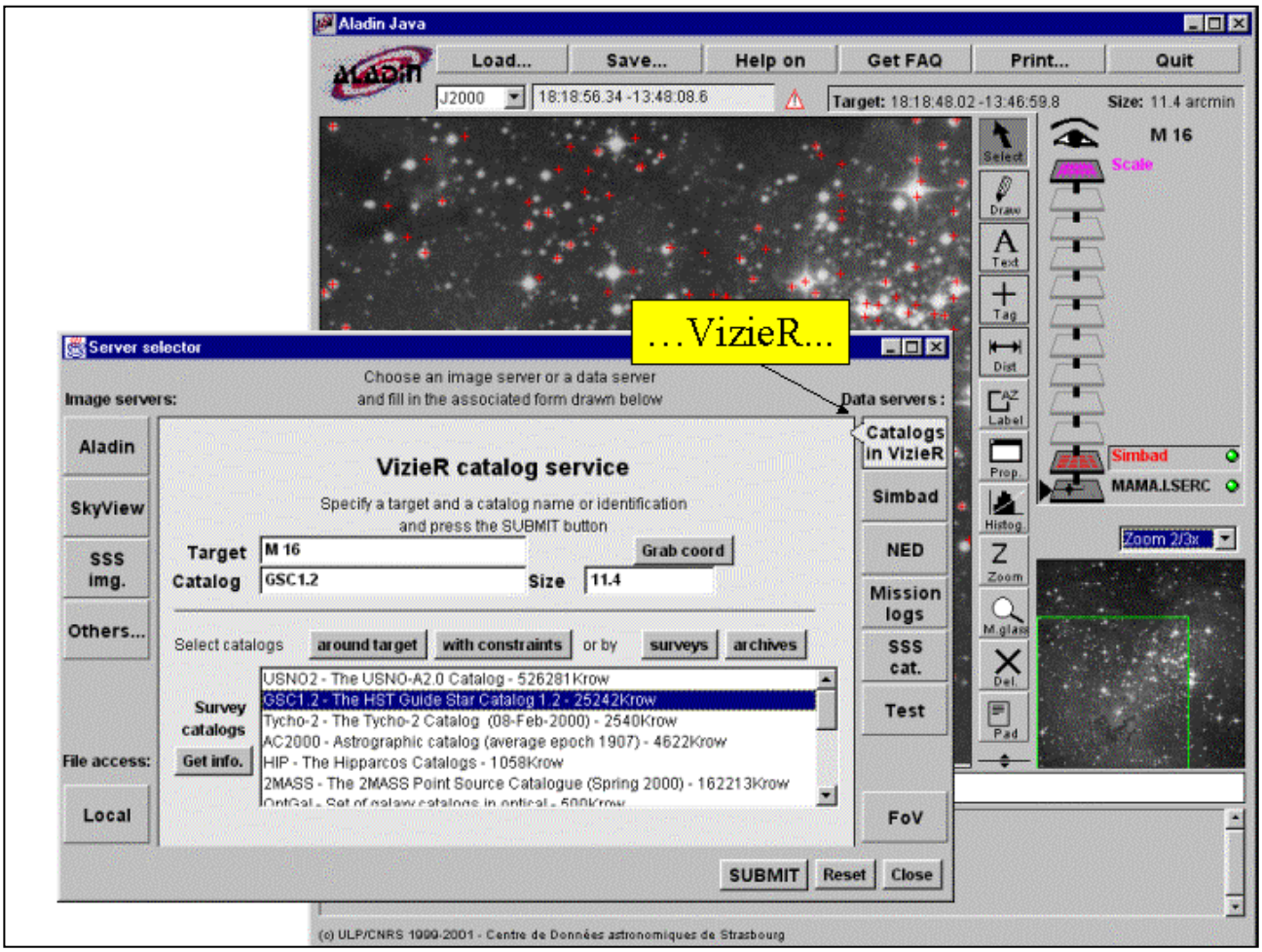


1.2.3. Aladin :

Aladin est un atlas interactif du ciel permettant d'accéder simultanément à des images numérisées du ciel, ainsi qu'à des catalogues et bases de données astronomiques.

Cet outil permet de superposer, sur des images du ciel optique, les objets présents dans Simbad, les sources de catalogues contenus dans VizieR, mais aussi d'autres données locales ou situées sur des serveurs distants (archives, HST, …).

Exemple d'utilisation d'Aladin :



1.2.4. Les autres services :

Parmi les autres services offerts par le CDS, on peut citer le dictionnaire de nomenclature, les services bibliographiques et les services de pages jaunes (AstroWeb, Sar's family, AstroGlu).

Le CDS est également impliqué dans le développement d'outils et de méthodes d'intérêt général pour l'échange de données et la standardisation.

Parmi les projets actuellement en développement, figurent un système de gestion des très grands catalogues, reposant sur une base de données orientée objet, la mise au point de stratégies d'indexation multicritères des catalogues, et l'étude d'un système pour automatiser l'identification croisée des catalogues.

2. Présentation du projet CoCat et de son contexte.

2.1. Contexte du projet :

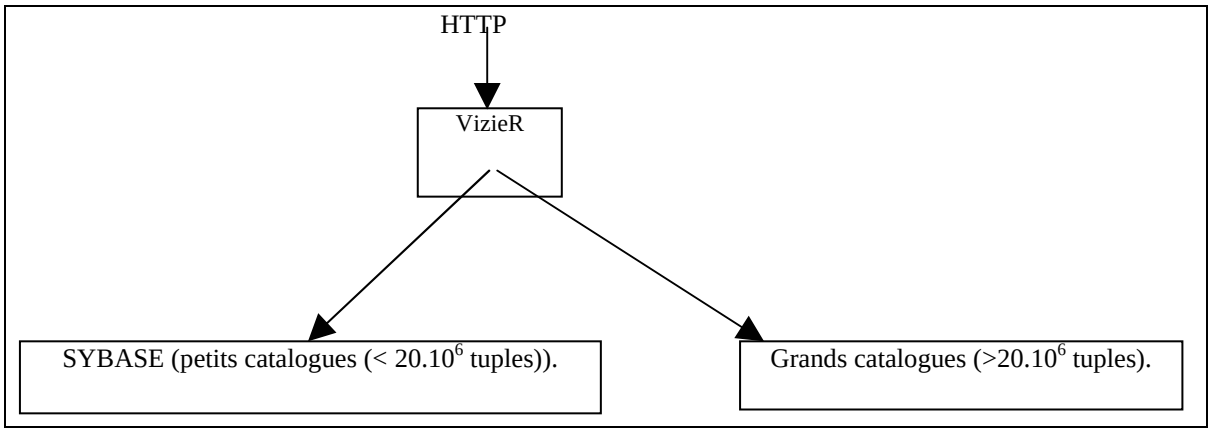
2.1.1. Les catalogues astronomiques :

Un catalogue astronomique est un listing de descriptions d'objets célestes. Le catalogue est constitué de deux parties :

- une partie "méta-données" qui indique l'auteur du catalogue, la période où les mesures ont été effectuées, la description de la structure des données, etc ... ;
- une partie "données" qui liste les différents objets avec leurs différentes mesures physiques.

Ces catalogues se présentent sous la forme de fichiers ascii compressés. Leur taille varie entre quelques méga-octets à quelques giga-octets. Une application spécifique est associée à chaque catalogue pour la décompression et le traitement des informations.

2.1.2. Architecture existante :



VizieR est une architecture logicielle qui permet à un utilisateur d'effectuer des requêtes sur des catalogues traitant de l'astronomie par le biais d'une page web. Les catalogues sont triés suivant leur taille. En effet, les grands catalogues (dépassant 20 millions de tuples) ne sont pas dans la base de données Sybase. La base de données Sybase n'est pas assez rapide lorsqu'elle gère des tables de plusieurs dizaines de giga octets. VizieR lance le logiciel spécifique au catalogue choisi par l'utilisateur, extrait les informations des catalogues sélectionnés par l'utilisateur. Puis, ces informations sont utilisées afin d'effectuer de multiples traitements. Les résultats de ces traitements sont ensuite envoyés à VizieR via le réseau interne. Enfin, VizieR affiche les résultats sous la forme d'une page html ou d'un fichier (xml, excel, etc…). C'est dans le cadre de ces

grands catalogues qu'une grappe de PC sera mise en oeuvre afin de réduire les temps de réponses.

2.2. Objectif du projet CoCat :

Le projet CoCat (co-processeur de catalogues) a pour objectif de mettre en place un cluster. Ce cluster permettra d'accélérer le fonctionnement de VizieR. En effet, pour le moment, VizieR s'exécute sur une machine Sun quadri-processeurs de 400 Mhz. L'équipe de ce projet est composée de Mr Schaaff, ingénieur de Recherche, de Mrs Derrière et Ochsenbein, astronomes et de moi-même.

2.2.1. Architecture hardware du cluster :

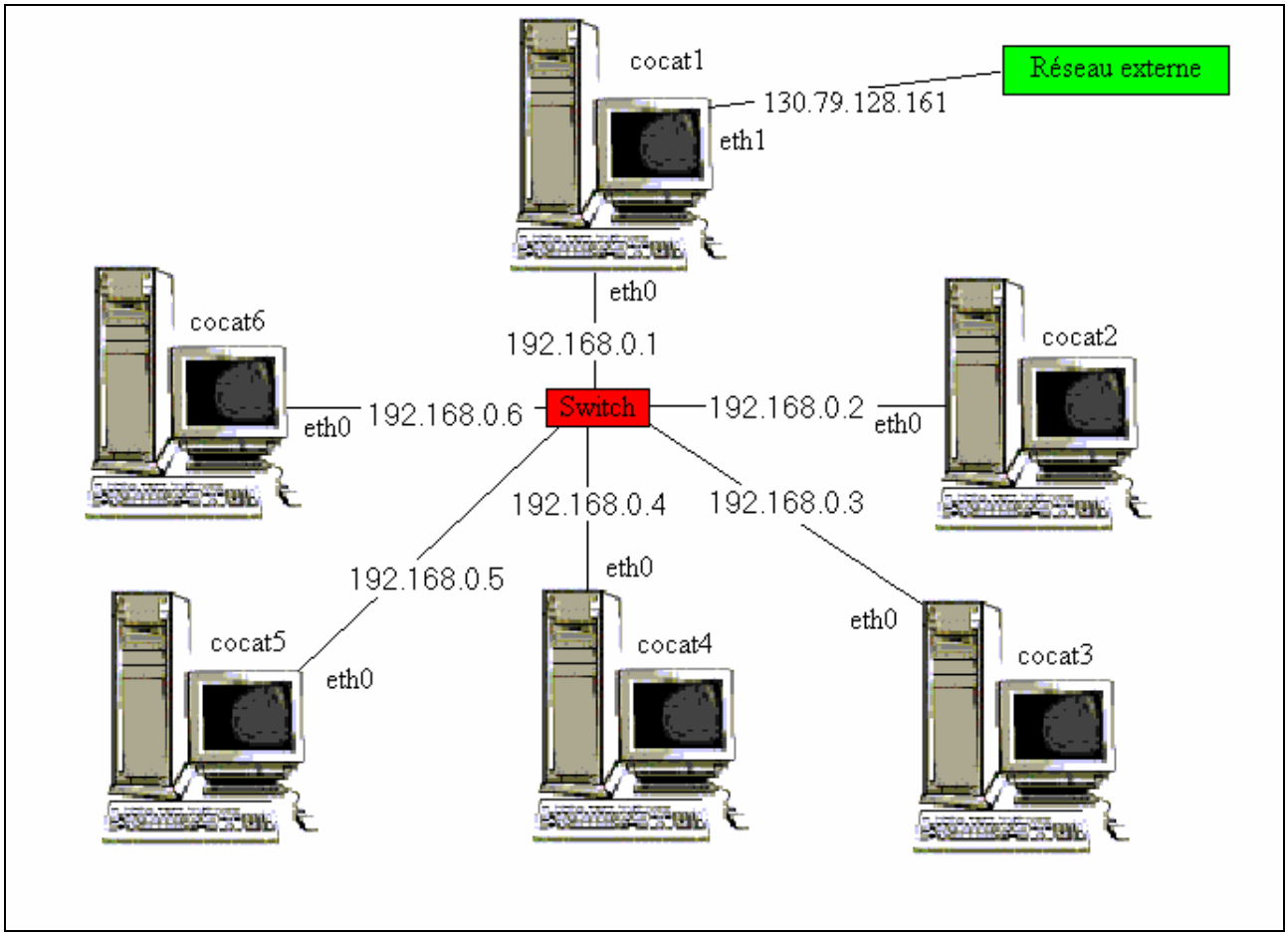
Pour mettre en oeuvre le cluster, nous disposons de six machines reliées par un réseau de 100Mb, contenant chacune, un microprocesseur AMD 2,4 Ghz, deux disques durs IDE de 200 Go, une mémoire vive de 1 Go et deux cartes réseaux (Broadcom et 3Com), ainsi qu'un routeur.

Cette architecture constitue un sous-réseau dont l'une des machines s'interface, à l'aide de sa carte réseau Broadcom (eth1), avec le réseau du CDS. Par contre la carte 3Com (eth0) permet à toutes les machines du cluster de communiquer entre elles.

Nom et adresse IP des différentes machines du cluster :

NOM	***ADRESSE IP DU SOUS-RESEAU***	***ADRESSE IP EXTERNE AU SOUS-RESEAU***
cocat1	192.168.0.1	130.79.128.161
cocat2	192.168.0.2	
cocat3	192.168.0.3	
cocat4	192.168.0.4	
cocat5	192.168.0.5	
cocat6	192.168.0.6	

Schéma récapitulatif :



2.2.2. Organisation du projet :

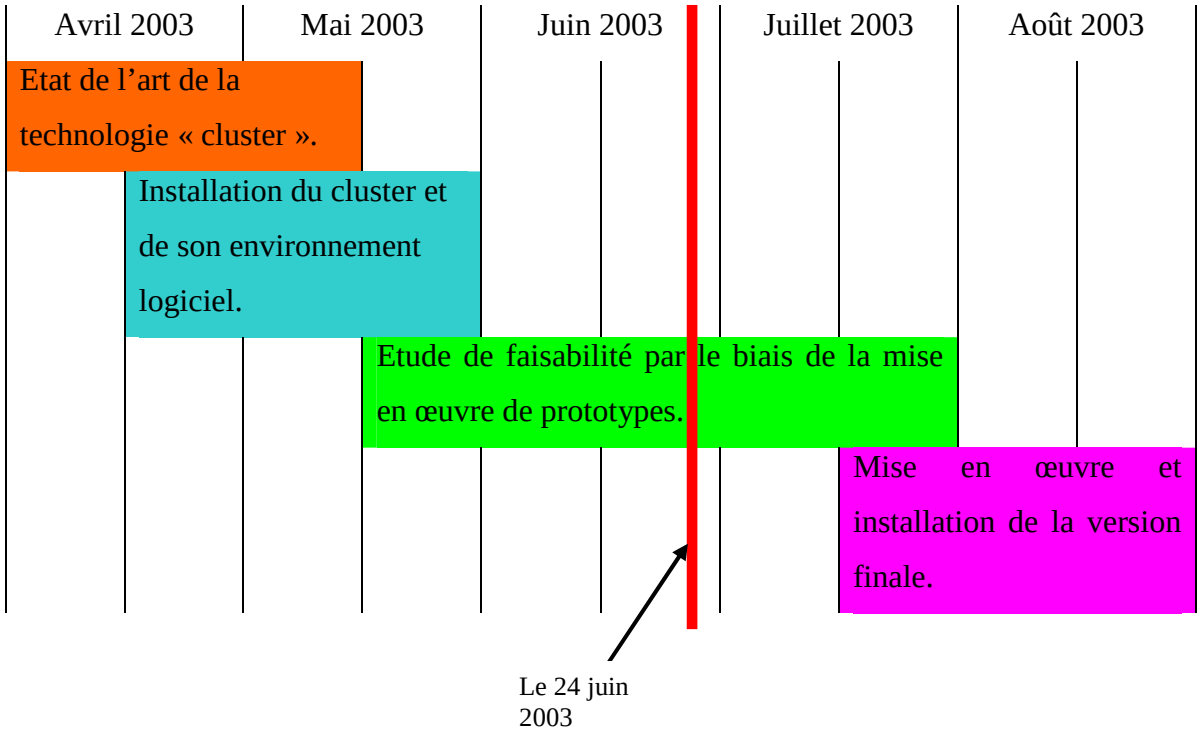
La mise en œuvre du projet s'effectue à l'aide de la méthode « extrem programming ». Cette méthode consiste à rester en relation avec les différents clients dans toutes les étapes de la construction du projet. Ainsi, les utilisateurs s'aperçoivent plus rapidement que le projet n'évolue pas dans le sens souhaité et peuvent donc le réajuster avec l'équipe de développement. Ce type de projet est donc en constante évolution et réclame de fréquentes réunions avec les différents acteurs. Lors de ce projet, nous avons choisi que la fréquence des réunions soit hebdomadaire afin de répondre au mieux aux attentes des utilisateurs.

De plus, l'utilisation de cette méthode ne permet pas l'utilisation d'outils de modélisation comme UML car la structure du projet change constamment à cause des différentes doléances formulées par les utilisateurs.

Le projet s'organise en quatre principales étapes :

- Etat de l'art de la technologie « cluster ».
- Installation du cluster et de son environnement logiciel.
- Etude de faisabilité par le biais de la mise en œuvre de prototypes.
- Mise en œuvre et installation de la version finale.

Prévision de l'évolution du projet CoCat sous la forme d'un gant :



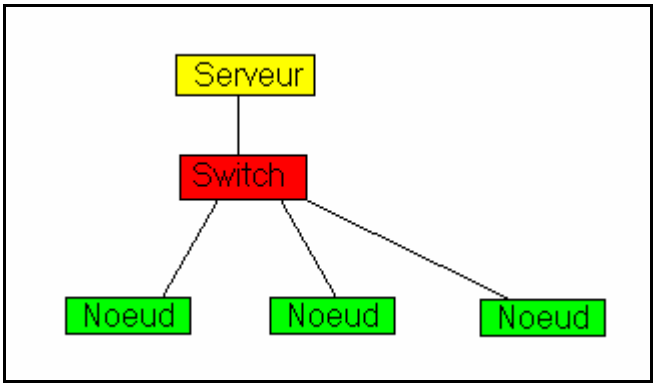
Pour simplifier l'installation du cluster et la mise en œuvre des premiers prototypes, il a été décidé que le cluster soit composé de trois machines jusqu'à la soutenance du 24 juin.

3. Etat de l’art de la technologie « cluster ».

2.1. Qu'est-ce qu'un cluster* ?

Un cluster (ou grappe en français) est un ensemble de machines reliées par l'intermédiaire d'un hub ou d'un switch, géré par un système d'exploitation spécifique. Cette architecture a pour but de modéliser une pseudo-machine. Elle est composée de deux parties, des nœuds et un serveur. Les nœuds modélisent les ressources (taille de la mémoire vive, taille du disque dur, etc…) de la machine virtuelle. Elles sont égales à la somme des ressources des différents nœuds. Le serveur a pour fonction de gérer l'ensemble du cluster.

Schéma de l'architecture « cluster » :



Ce type d'architecture permet d'obtenir une machine aussi puissante mais plus flexible qu'une architecture multi processeurs. En effet, un cluster, composé de cinq nœuds, est équivalent à un quadri-processeur. Une architecture multi processeurs n'est pas évolutive comme un système de type cluster. En effet, Cette architecture est flexible car il est possible d'ajouter un processeur en insérant un nouveau nœud. De plus, un cluster se révèle plus économique à puissance égale.

Par contre, les risques de pannes se révèlent supérieurs dans le cadre d'un cluster, le nombre de machines étant plus important. Ainsi, un cluster a une probabilité moindre d'être au maximum de ses possibilités. De plus, la gestion d'un cluster est de plus en plus complexe avec l'accroissement du nombre de machines.

Le mode dégradé :

Il existe deux modes dégradés, l'un est de type mineur et l'autre de type majeur. Le mode dégradé majeur oblige le cluster à cesser de fonctionner. En effet, l'instabilité du système peut être provoquée si le serveur ou le switch devient inactif. Par contre, le mode

* Pour voir les différentes formes de cluster. Voir l'annexe I : Les différents types de clusters.

dégradé mineur diminue les performances du cluster sans remettre totalement en question son fonctionnement. Par exemple, l'inactivité d'un nœud n'interfère pas dans le bon fonctionnement du cluster, si celui-ci est géré par un système d'exploitation spécifique à ce genre de système.

3.2. Environnement permettant la gestion des processus qui effectuent un traitement en commun :

Afin d'augmenter la vitesse de fonctionnement d'un processus, on le parallélise, on le divise en plusieurs processus que l'on disperse dans les différents nœuds du cluster. Pour cela, il faut que ces différents processus travaillent en commun. Ils doivent donc s'échanger des informations. Cette communication est gérée par un environnement logiciel. Lors de cette étude, deux principaux environnements en sont ressortis :

- LAM/MPI
- PVM.

3.2.1. LAM/MPI* :

LAM (Local Area Multicomputer) est un environnement de programmation et de développement respectant le standard MPI 1.1. Cet environnement permet à des systèmes hétérogènes de travailler ensemble via un réseau. Le logiciel LAM aide à la gestion des processus fonctionnant sur un cluster. Il permet d'optimiser la vitesse des processus, aide au debuggage des applications parallèles et facilite leurs développements.

3.2.1.1. Qu'est-ce que le standard de communication MPI** :

MPI est une bibliothèque d'échanges de messages pour machines parallèles hétérogènes, c'est un standard créé pour le développement d'applications parallèles portables. Une application MPI est un ensemble de processus exécutant chacun son propre code (modèle SPMD) et communiquant via des appels à des sous-programmes de la bibliothèque MPI. Le modèle d'exécution d'une application MPI est le SPMD (*Single Program Multiple Data*), soit l'exécution du même programme pour tous les processus. Les paramètres et les données sont privés à chaque processus, que l'on soit sur une

* : Pour plus d'informations à propos de l'environnement LAM/MPI, voir l'annexe II : LAM/MPI

architecture à mémoire distribuée ou partagée. Le partitionnement, l'échange des données, le contrôle de la synchronisation des processus MPI sont de la responsabilité de l'utilisateur. On peut donc facilement simuler un modèle maître-esclave.

L'interface MPI permet donc l'exploitation des machines multi processeurs par passage de messages. Celle-ci doit être pratique, portable, efficace et flexible. La première version finalisée date de mars 1994, le projet a principalement été mené par le centre de recherche d'IBM T.J. Watson, le NX/2 Centre, le PARAMACS.

Les principaux objectifs de ce consortium étaient de concevoir une bibliothèque permettant les points suivants :

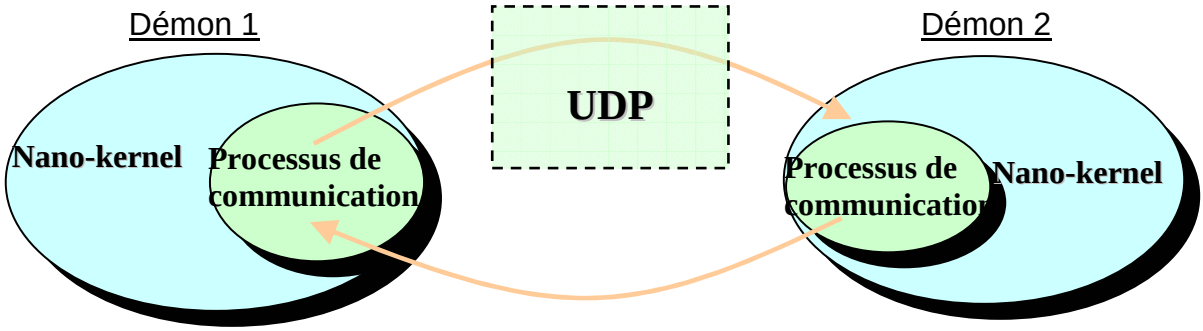
- communications bas et haut niveau
- portabilité quasi totale
- bonne modularité (pour le développement de bibliothèques)
- supportant l'hétérogénéité
- supportant divers groupes et topologies de processus

Remarque : la librairie MPI supporte divers langages comme le fortran 90, le C, le C++ ou encore Java.

3.2.1.2. Fonctionnement des démons de LAM :

LAM se présente comme un simple démon présent sur chacune des machines. Chaque démon est structuré comme un micro-noyau fournissant des services de passage de messages et de synchronisation. Certains processus internes au démon forment un sous-système de communication qui permet le passage de messages vers les démons des autres machines. La communication entre les différents processus se fait via UDP c'est à dire en mode non connecté et par paquets.

[**] : Pour plus d'informations des différents modes de communication proposés par la librairie MPI, voir l'annexe IV : La librairie MPI



La communication se fait point à point : Un message va d'un processus vers un autre sans nécessairement passer par le serveur. Les programmes écrits pour LAM sont compatibles avec les autres implémentations de MPI. LAM détecte les erreurs de communication dues à un «crash» d'une machine ou à une rupture de communication. LAM bloque toutes les communications vers la machine posant problème, les processus restants sont informés de la panne de manière asynchrone.

On peut donc gérer ce type d'erreurs dans les applications en supprimant les communications vers les processus qui ne réagissent plus, puis en les créant de nouveau.

3.2.2. PVM* :

PVM (Parallel Virtual Machine) est une machine virtuelle qui gère des processus s'exécutant au sein d'un cluster homogène ou hétérogène. PVM comporte les fonctionnalités suivantes :

- gestion des nœuds,
- gestion des processus,
- gestion des messages entre processus,
- gestion des groupes de processus,
- détection des processus instables.

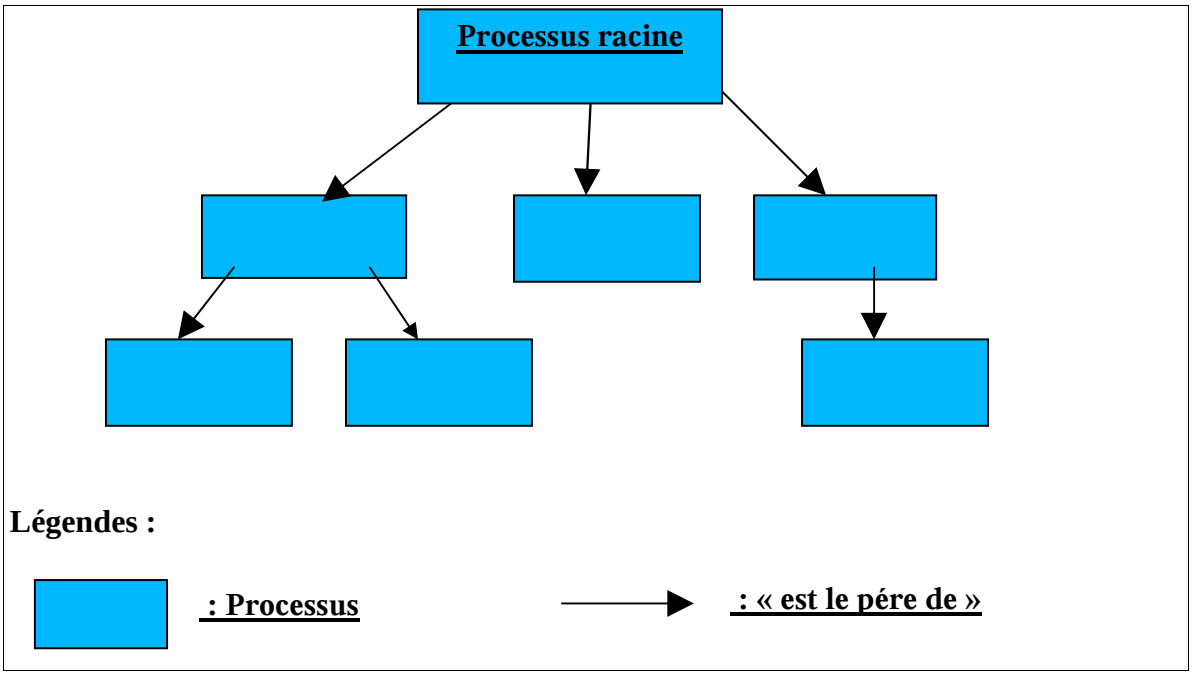
3.2.2.1. Organisation logique des processus au sein de la machine virtuelle PVM :

Dans une machine virtuelle, les processus s'organisent sous la forme d'un arbre. A la racine de cet arbre, il y a un processus qu'on appelle «processus racine». Cette organisation est due à la spécification de PVM. En effet, l'utilisateur peut créer qu'un seul processus et seuls ces mêmes processus peuvent en créer d'autres. Ainsi, les processus ont

* : Pour plus d'informations à propos de l'environnement PVM, voir l'annexe III : PVM

tous un père sauf le processus «racine». Lorsqu'un fils est créé, il hérite de l'environnement de son père. De plus, la machine virtuelle donne un identifiant unique à chaque processus. Cet identifiant permet de localiser le processus au sein du cluster ainsi que le groupe auquel il appartient.

Schéma récapitulatif :



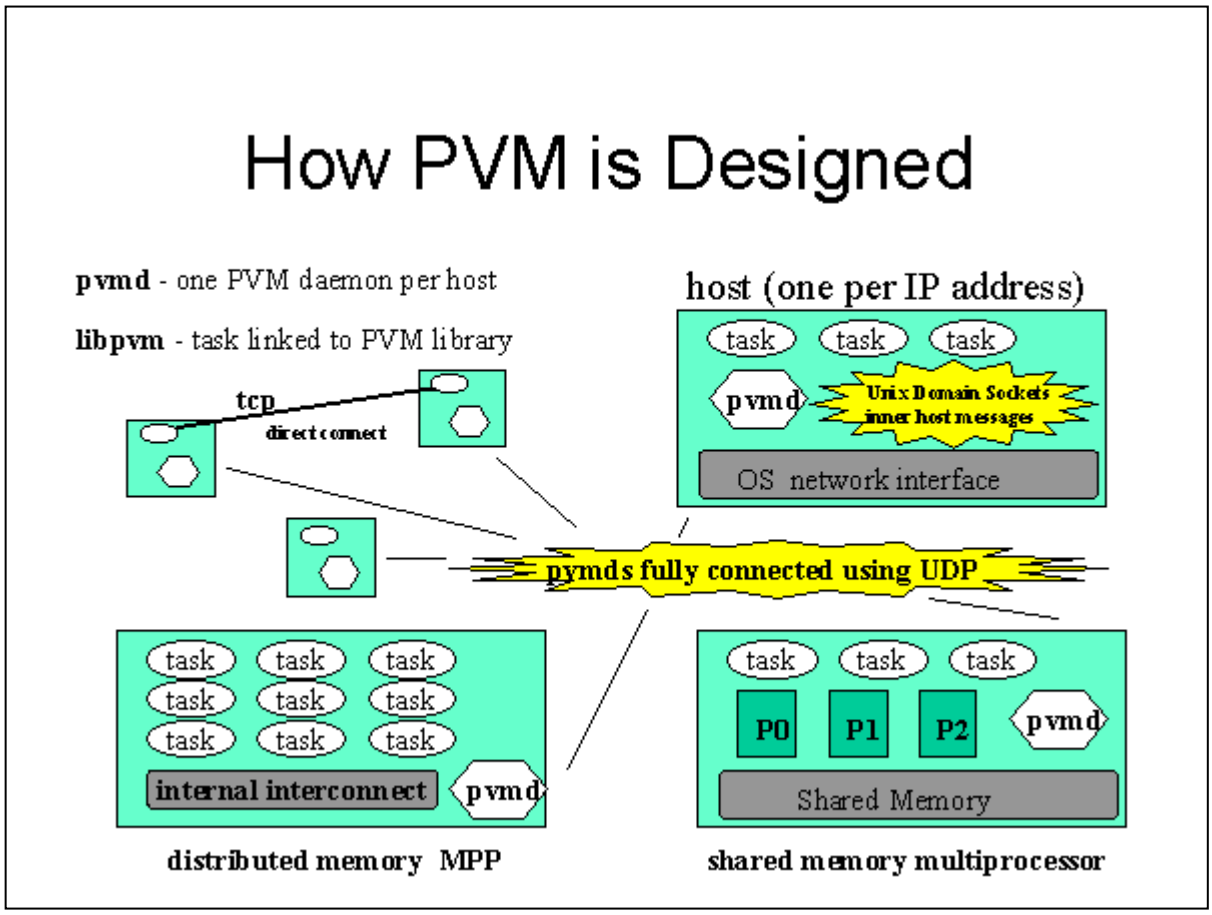
Cette organisation permet à plusieurs machines virtuelles de fonctionner ensemble. Ainsi, une machine virtuelle ne peut pas avoir accès à un processus dont elle n'est pas responsable.

3.2.2.2. Fonctionnement de la machine virtuelle PVM :

3.2.2.2.1. Organisation des machines virtuelles PVM :

PVM est un démon appelé «pvmd». Ce démon gère les messages qui transitent entre processus et contrôlent la présence des processus dont il a la charge. Lorsqu'il est lancé sur l'un des nœuds du cluster par un utilisateur, il devient le «pvmd maître». Il lance les démons, sur les autres nœuds, qui deviennent les «pvmd esclave». Seul le maître peut créer des esclaves et peut détecter leur instabilité. Les démons communiquent entre eux à l'aide des protocoles UDP/IP. Mais, pour amorcer la communication avec un nœud, le «démon maître» utilise les protocoles TCP/IP.

Fonctionnement des démons pvmd au sein d'un cluster :



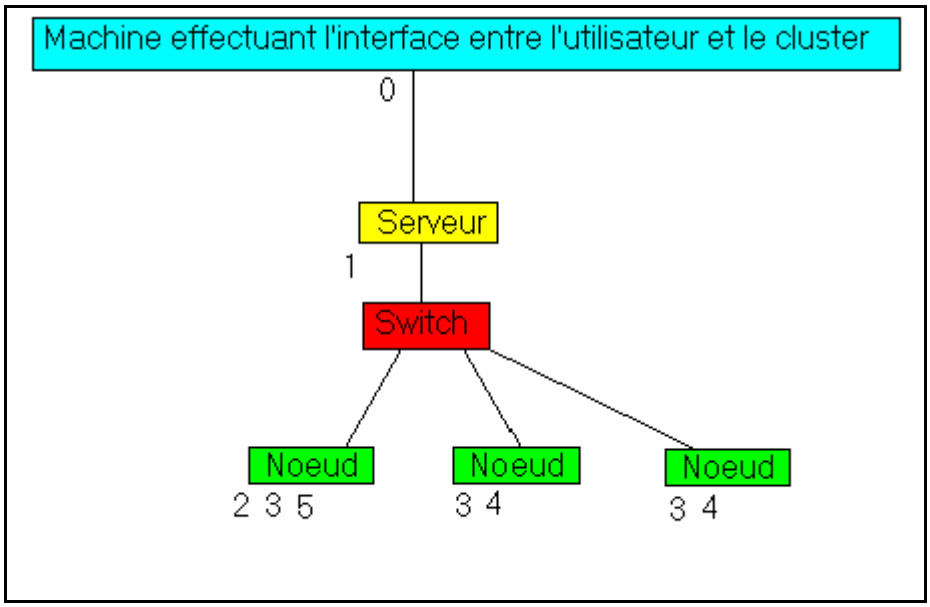
3.2.2.2.2. Le «démon maître» :

Le «démon maître» est composé d'un gestionnaire de messages et d'un gestionnaire de ressources (ressources manager RM). Le manager des ressources est un processus PVM qui a pour fonction de gérer l'ensemble des processus ainsi que de synchroniser les différents nœuds du cluster.

3.3. Algorithme général de la communication inter-processus :

Les différentes organisations de données imposent que les processus communiquent suivant le schéma maître-esclave.

Algorithme général utilisé :



Explication :

Etape 0 : la machine effectuant l'interface entre l'utilisateur et le cluster (le serveur VizieR), envoie au serveur les différents traitements que l'utilisateur souhaite effectuer.

Etape 1 : Le serveur recherche un nœud qui fonctionne. Lorsqu'il le trouve, il lui donne le rôle de maître d'une manière totalement dynamique. Ainsi, en mode dégradé mineur, le système continue de fonctionner. Puis, il lui envoie les traitements à effectuer

Etape 2 : Le nœud maître cherche les nœuds qui fonctionnent, subdivise les traitements en problèmes plus petits et les envoie aux différents nœuds stables.

Etape 3 : Tous les nœuds effectuent les différents traitements.

Etape 4 : Lorsque les nœuds ont terminé, ils envoient leurs résultats au nœud maître.

Etape 5 : le nœud maître combine les résultats qu'il a précédemment rapatriés. Puis, il envoie le résultat de cette combinaison au serveur qui le transmettra au serveur VizieR (la machine effectuant l'interface entre l'utilisateur et le cluster).

3.4. Etude de l'organisation des données au sein du cluster.

Le but de cette étude est de connaître les différentes configurations possibles afin d'optimiser les performances du cluster. Cette organisation doit respecter quelques contraintes :

- Repousser le moment où le changement de l'architecture du cluster deviendra inévitable (ajout d'un nœud, changement des disques durs ou des mémoires vives).

- Le cluster doit répondre à toutes les requêtes en mode dégradé mineur.
- Les processus doivent s'exécuter le plus rapidement possible.

3.4.1. L'organisation dite « redondante » des données:

Chaque nœud du cluster contient un exemplaire de tous les catalogues disponibles. Cette architecture diminue fortement le nombre de connexions entre nœuds car les informations sont localisées.

Mais, cette organisation recèle certains désavantages. En effet, la taille des catalogues double tous les ans. A ce rythme, les 400 Go que contiennent les disques durs de chaque nœud seront rapidement saturés.

3.4.2. L'organisation dite « distribuée » des données :

Dans ce type d'organisation, les différents nœuds se partagent les catalogues. Elle impose que les catalogues de mêmes caractéristiques d'un point vue astronomique soient regroupés entre eux. Cet agencement permet d'utiliser la totalité de l'espace des douze disques durs disponibles au sein du cluster.

Cette organisation provoque un ralentissement des traitements qui s'accroît avec la dispersion des catalogues dans les différents nœuds. En effet, cette dispersion implique de nombreux échanges de messages à travers le réseau. De plus, en mode dégradé mineur, la perte d'un nœud engendre la perte de catalogues.

3.5. Présentation et analyse de fonctions astronomiques types :

La fonction astronomique « croisement de catalogues » est l'une des fonctions accédant à une grande quantité de données et d'une grande puissance de calculs. Sa présentation permet au lecteur de se rendre compte du type de fonctions que le cluster est amené à gérer. Ensuite, nous analysons une fonction d'extraction d'étoiles au sein d'un catalogue. Cette fonction a été choisie pour sa mise en œuvre dans toutes les applications gérant un catalogue.

3.5.1. Présentation de la fonction «croisement de catalogues» :

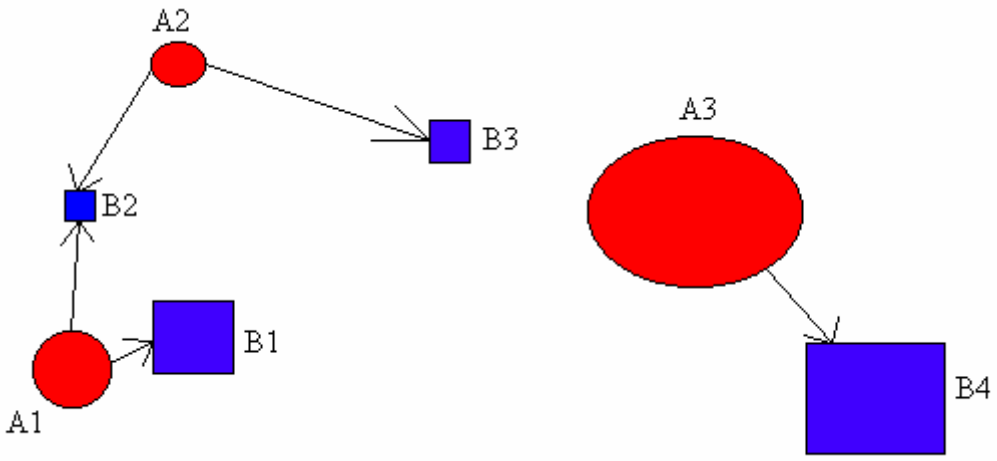
Le «croisement de catalogues» a pour principe de combiner les mesures obtenues pour les mêmes objets astronomiques situés dans des catalogues différents. Deux objets sont associés si ils sont suffisamment « proches » : en pratique, on se fixe une seuil

d'acceptation, et on retient les associations entre deux objets séparé par une distance inférieure à ce seuil. La « distance » peut faire intervenir d'autres que les positions des objets sur le ciel.

Exemple de prototype de la fonction distance :

réel distance(position_de_objet_cat_A, position_de_objet_cat_B, magnitude_de_objet_cat_A, magnitude_de_objet_cat_B)

Exemple d'un croisement de deux catalogues A et B utilisant le prototype de la fonction distance précédente :



Explication :

On croise les astres du catalogue A (les cercles) avec ceux du catalogues B (les carrés). Les flèches noires signifient qu'il y a un croisement effectif entre les deux astres (la distance les séparant est inférieure ou égale au seuil). La taille des symboles est proportionnelle à la magnitude des astres qu'elles modélisent

Observation :

Un objet peut appartenir à deux groupes comme le montre l'astre B2. L'astre B3 et A3 ne sont pas regroupés malgré la faible distance angulaire qui les sépare. En effet, une différence de magnitude trop importante provoque une distance supérieure au seuil.

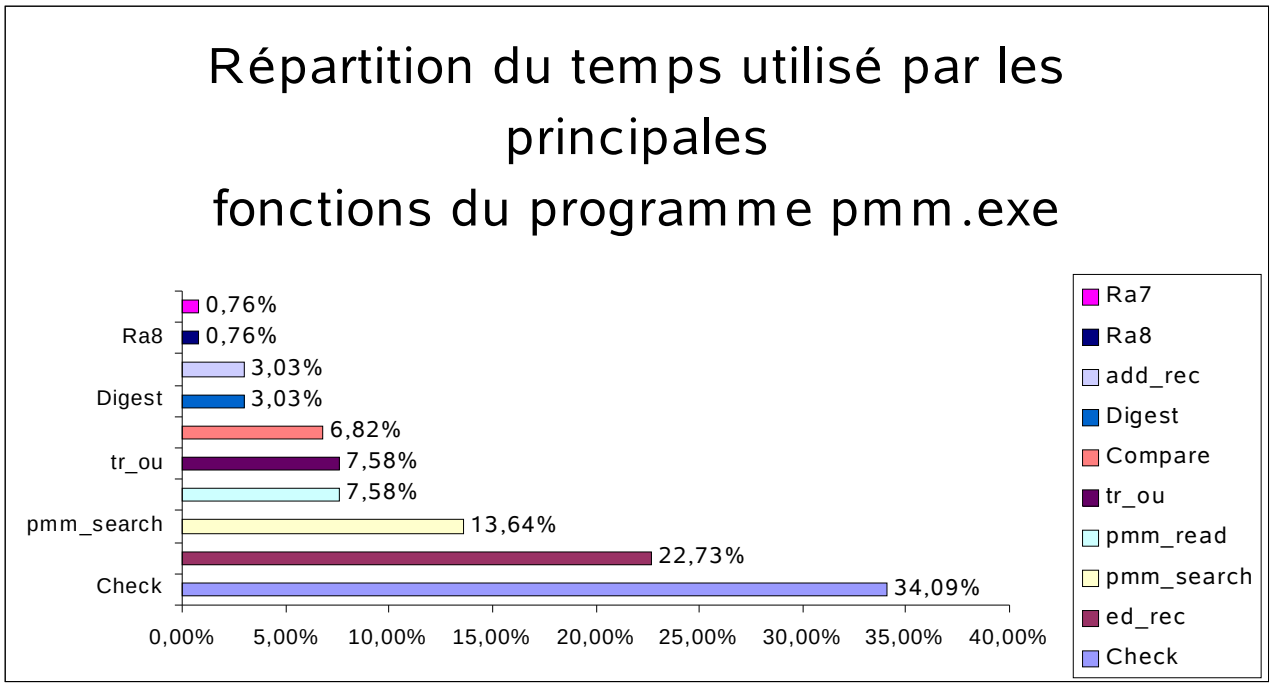
Dans les différentes applications qui mettent en œuvre cette fonction, l'utilisateur choisit la liste des catalogues qu'il souhaite croiser, le seuil d'acceptation et les zones des différents catalogues où le croisement s'effectuera.

3.5.2. Analyse des performances lors d'une requête dans un catalogue :

Le programme pmm.exe est chargé d'extraire les informations du catalogue USNO_A2 et de les traiter. La taille de ce catalogue est de 10 giga-octets.

Lors de cette étude, la commande Unix gprof a été utilisée. Cette commande génère à la volée un fichier texte affichant les statistiques sur le programme mis en argument. Le programme testé recherche tous les astres situés sur une zone du ciel et les ordonne.

Les principaux résultats sont affichés sous la forme d'un histogramme :



Cette répartition montre que seulement quatre fonctions consomment 80% du temps utilisé par le programme pmm.exe. Ce graphe permet de déduire qu'il serait optimal de les paralléliser. Mais, comment ? Paralléliser les algorithmes n'est pas un avantage dans le cadre d'un cluster lorsqu'ils réclament beaucoup d'informations. En effet, les différentes composantes d'un algorithme parallélisé dans différents nœuds, s'échangent les données qu'ils traiteront chacun leur tour. Ces accès réseaux entre nœuds fréquents et longs, risquent de charger le réseau. Cette forte charge freine l'évolution de l'algorithme qui pourrait atteindre une performance moins bonne que dans le cadre d'un fonctionnement en mode séquentiel. Il est donc préférable de répartir les données et non de paralléliser les fonctions d'une applications dans le cadre d'un cluster.

4. Réalisation du cluster CoCat

Toute puissance est faible, à moins que d'être unie.
« le vieillard et ses enfants » **de jean de la fontaine**

4.1. Présentation de la distribution CLIC :

Le projet CLIC est un projet RNTL (Réseau National de recherche et d'innovation en Technologies Logicielles) dont les partenaires sont ID-IMAG (laboratoire de recherche), BULL et Mandrakesoft. Ce projet a pour but de réaliser une sur-couche du système d'exploitation Mandrake 9.0 permettant la gestion d'un cluster. Le projet est pour le moment en phase initiale, c'est à dire qu'il contient les principaux outils GNU pour gérer le fonctionnement d'un cluster (outils de mise en oeuvre des processus, gestion des processus, gestion de l'espace des différents disques durs, etc...) ainsi que les scripts d'installation automatiques.

En effet, cette distribution contient :

- Les scripts permettant d'utiliser le protocole PXE (voir l'annexe V : le déploiement des nœuds au sein d'un cluster).
- La librairie MPI et les applications qui l'utilisent.
- La librairie PVM.
- L'environnement PBS.
- Les différents serveurs (le serveur DNS, le serveur NIS, etc...).
- Les outils d'administration du cluster.

4.2. Installation de la distribution CLIC:

Cette installation s'effectue à l'aide de l'image ISO de CLIC qui est téléchargée sur le site de Mandrakesoft (*http://clic.mandrakesoft.com/download-fr.html*). Puis, cette image est gravée sur un CD. L'installation se déroule en trois phases :

- Installation du serveur.
- Installation du golden node.
- Déploiement de l'image du golden node sur les autres nœuds du cluster.

4.2.1. Installation du serveur :

Cette installation s'effectue en deux étapes. La première étape se déroule à l'aide d'une interface graphique. Cette interface permet de configurer :

- la langue utilisée,
- les différentes partitions,
- le mot de passe du serveur,
- le pilote de la carte vidéo.

Exemple d'interfaces graphiques du programme d'installation du serveur :

Après cette installation graphique, le noyau du système et le boot de démarrage sont installés. Dans la seconde étape, on configure les différentes composantes du serveur CLIC. Lors de cette installation en mode texte, on paramètre, en tapant la commande «setup_auto_server» :

- Le nombre de nœuds
- Les différents services du serveur :
    - Service DNS (permet de faire correspondre un nom à une adresse réseau).
    - Service NIS (permet de gérer la connexion et l'authentification d'un utilisateur).
    - Service DHCP (permet d'attribuer à la volée une adresse IP à des ordinateurs qui le lui demandent).
    - Service PXE (permet à un client de s'amorcer à l'aide d'une image téléchargée à partir du serveur).
    - Configuration MAC (configuration de l'adresse statique et du réseau).
    - Configuration Postfix (configuration du gestionnaire de la boîte électronique).
    - Service Maui (service permettant de synchroniser les différents nœuds du cluster).

4.2.2. Installation du golden node :

Le golden node est le nœud qui servira de références aux autres nœuds. En effet, les machines qui composent le cluster ont une architecture homogène. Cette caractéristique permet de configurer un seul nœud et de déployer son image au sein des autres nœuds. Pour cela, le Bios de chaque nœud est configuré pour démarrer avec la carte réseau PXE*. Nous exécutons un script shell dans un environnement du serveur afin que son serveur PXE oblige les nœuds à utiliser l'image Ka (voir annexe V : le déploiement des nœuds au sein d'un cluster) :

Exécution du script shell :

```
[root@servcl] setup_pxe_server setup linux
        Entering setup mode
        - Patching /var/lib/tftpboot/PXEClient/pxelinux.cfg/default
        - Patching /var/lib/tftpboot/X86PC/linux/pxelinux.cfg/default
        PXE Server is now set to boot on "linux" image
```

L'installation du "golden node" commence par l'intermédiaire du CD de la distribution. Elle s'effectue par le biais d'une interface graphique. Le mot de passe, les différents packages du système et le type du réseau sont configurés.

Exemple d'interface graphique du programme d'installation du « golden node » :

Il faut maintenant que le « golden node » redémarre par l'intermédiaire du boot local. Pour cela, on lance la commande « setup_pxe_server setup local » :

Exemple d'exécution :

```
[root@servl] setup_pxe_server setup local
        - Entering setup mode
```

* Pour plus d'informations, voir l'annexe VI : la technologie PXE

```
- Patching /var/lib/tftpboot/PXEClient/pxelinux.cfg/default
- Patching /var/lib/tftpboot/X86PC/linux/pxelinux.cfg/default
PXE Server is now set to boot on "local" image
```

On reboot le “golden node”.

4.2.3. Déploiement des nœuds*:

Pour déployer les nœuds, nous utilisons le script *setup_ka_deploy* afin que le serveur KA situé dans le « golden node » démarre. Ensuite, nous configurons le serveur à l’aide du script « ka-d ».

Syntaxe de ka-d :

```
ka-d.sh -n NUMBER_OF_NODES -r lilo -p hda desc
“NUMBER_OF_NODES” indique le nombre de noeuds à dupliquer.
« desc » est un fichier qui décrit les partitions des nœuds à dupliquer.
```

Maintenant, nous changeons l’image de boot avec l’image Ka ; Cette image a été créée lors de la configuration du « golden node ». Elle contient toutes ses caractéristiques logicielles (mot de passe, disposition des partitions, etc….) .

Exécution de la commande « setup_pex_server setup ka »

```
[root@servcl] setup_pxe_server setup ka
- Entering setup mode
- Patching /var/lib/tftpboot/PXEClient/pxelinux.cfg/default
- Patching /var/lib/tftpboot/X86PC/linux/pxelinux.cfg/default
PXE Server is now set to boot on "ka" image
```

Nous démarrons les autres nœuds qui téléchargent l’image Ka. Ainsi, les nœuds comportent les mêmes caractéristiques que le « golden node ». Nous changeons enfin, les images des nœuds afin qu’il redémarre avec le nouveau boot local.

Exécution de la commande « setup_pxe_server setup local » :

```
[root@servclk] setup_pxe_server setup local
Entering setup mode
```

*Pour savoir « Comment se produit le déploiement des nœuds ? », voir l’annexe V : Le déploiement des nœuds au sein d’un cluster.

- Patching /var/lib/tftpboot/PXEClient/pxelinux.cfg/default

- Patching /var/lib/tftpboot/X86PC/linux/pxelinux.cfg/default

PXE Server is now set to boot on "local" image

4.3. Les outils d'administration de la distribution CLIC

4.3.1. Les outils Berkeley :

4.3.1.1. Auth :

Auth est un outil qui permet d'obtenir et de vérifier l'identité d'un utilisateur. Cette identité est mémorisée sous la forme d'une signature cryptée par l'intermédiaire d'une clef RSA. Auth est donc un serveur qui stocke ces différentes signatures cryptées. Les autres outils d'administration effectuent des requêtes sur ce serveur afin de connaître les droits des utilisateurs sur les informations qu'ils traitent ou sur les nœuds où elles sont traitées.

4.3.1.2. gexec :

Gexec est un outil qui distribue les différents processus au sein des nœuds du cluster. Il gère les entrées et les sorties de chaque processus du cluster. Il permet de leurs envoyer un message et d'évaluer la robustesse de chaque nœud.

Exemple d'utilisation avec la commande Unix 'hostname':

Le nombre situé en début de ligne indique le nombre de noeuds qui ont répondu à la requête.

[root@servcl addons] gexec -n 0 hostname

0 node1.clic.com

1 node2.clic.com

2 node3.clic.com

3 node4.clic.com

4.3.1.3. pcp :

Pcp est un outil qui permet de dupliquer des fichiers dans les noeuds du cluster. L'exemple suivant montre comment copier le fichier /etc/authd_pub.pem dans les noeuds node1, node2, node3 et node4.

Exemple d'utilisation :

```
[root@servcl addons] pcp /etc/auth_pub.pem /etc/auth_pub.pem node1 node2 node3 node4
Write succeeded on node1.clic.com (192.168.200.1)
Write succeeded on node2.clic.com (192.168.200.2)
Write succeeded on node3.clic.com (192.168.200.3)
Write succeeded on node4.clic.com (192.168.200.4)
```

4.3.2. Les outils «clusterit» :

Dsh : lance une commande sur les différents nœuds du cluster.

Dshbak : effectue les même fonctions que dsh mais elle affiche la sortie de chaque nœud.

Barrier : synchronise un processus sur le nombre de nœuds mis en argument.

Jsd : synchrone les nœuds du cluster.

Jsh : lance les démons de synchronisation des différents nœuds.

Run : lance un processus sur un nœud au hasard.

Cseq : lance une commande sur chaque nœud, un par un.

Cpcq : copie un fichier situé sur l'un des nœuds du cluster.

Pdf : affiche l'espace disque libre au sein du cluster.

Prm : efface un ou plusieurs fichiers du cluster.

Rvt : un émulateur pour le gestionnaire de fenêtres X Windows. Il permet d'obtenir une interface graphique qui facilite la gestion du cluster.

Dvt : lance une fenêtre interactive sur le cluster.

Clustersed : partitionne une fichier du cluster.

Pconsole : console qui permet d'envoyer des commandes au cluster.

4.3.3. Les scripts de Mandrakesoft pour la distribution CLIC :

4.3.3.1. Le script «sauvegarde» :

«Sauvegarde» est un script qui permet à un utilisateur d'archiver un répertoire sous la forme d'un fichier tar.

Exemple d'utilisation (archivage du répertoire racine de l'utilisation guibo) :

```
[guibo@serverd guibo]$ sauvegarde guibo ~/
- Saving /home/nis/guibo/
tar: Removing leading /' from member names
home/nis/guibo/
home/nis/guibo/tmp/
home/nis/guibo/chess2.tga
.....

- Backup SUCCESS in /home/backup/guibo directory
- Setting read-only and mode undelete on file
```

4.3.3.2. Le script «adduserNis» :

«adduserNis» est un script permettant d'ajouter un utilisateur au service NIS du serveur CLIC. Cet ajout permet à l'utilisateur d'utiliser tous les noeuds du cluster.

Exemple d'utilisation (ajout de l'utilisateur guibo) :

```
root@servcl tmp]#
    adduserNis
    ------------------------------------------------------------
    Add New user in NIS environment where users with an uid > 500
    are NIS user
    ------------------------------------------------------------
    Login: guibo Group(s) [users] (You are member of mpi, pbs,
    pvm by default): Adding guibo in mpi group Adding guibo in pvm
    group Adding guibo in pbs group Comment (ex: James Bond):
    Antoine Ginies
    ------------------------------------------------------------
    Login: guibo Group: users Comment: Antoine Ginies passwd
    guibo: Changing password for user guibo.  New UNIX password:
    BAD PASSWORD: it's WAY too short Retype new UNIX password:
    passwd: all authentication tokens updated successfully.
    gmake[1]: Entering directory /var/yp/clic.com' Updating
    passwd.byname...  Updating passwd.byuid...  Updating
    group.byname...  Updating group.bygid...  Updating
    netid.byname...  gmake[1]: Leaving directory
    /var/yp/clic.com' - Creating ssh key for user guibo
```

Generating public/private dsa key pair. Your identification has been saved in /home/nis/guibo/.ssh/id_dsa. Your public key has been saved in /home/nis/guibo/.ssh/id_dsa.pub. The key fingerprint is:
de:15:3a:f8:b2:43:6a:f5:27:51:70:54:09:6c:88:e5
guibo@servcl.clic.com - authorize user to ssh himself - Setting .rhosts file - Creating /home/backup/guibo - Setting permission on file

4.3.3.3. Le script «deluserNis» :

Ce script permet de supprimer un utilisateur du service NIS du serveur Clic.

Exemple d'utilisation (suppression de l'utilisateur guibo) :

root@servcl tmp]# deluserNis
- Remove user from NIS Map

User: guibo
- Deleting user
- Updating NIS table
gmake[1]: Entering directory /var/yp/clic.com'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating netid.byname...
gmake[1]: Leaving directory /var/yp/clic.com'

4.3.4. La commande URPMI :

Cette commande permet d'utiliser les fichiers rpm dans le cadre d'un cluster. En effet, les fichiers rpm contiennent un logiciel ainsi que son programme d'installation. Ainsi, la commande urpmi décompresse et déploie les logiciels contenus dans les fichiers rpm. Puis ensuite, elle configure les nœuds afin que le fonctionnement de cette application, soit optimal.

Syntaxe de la commande URPMI :

urpmi --parallel group_name package

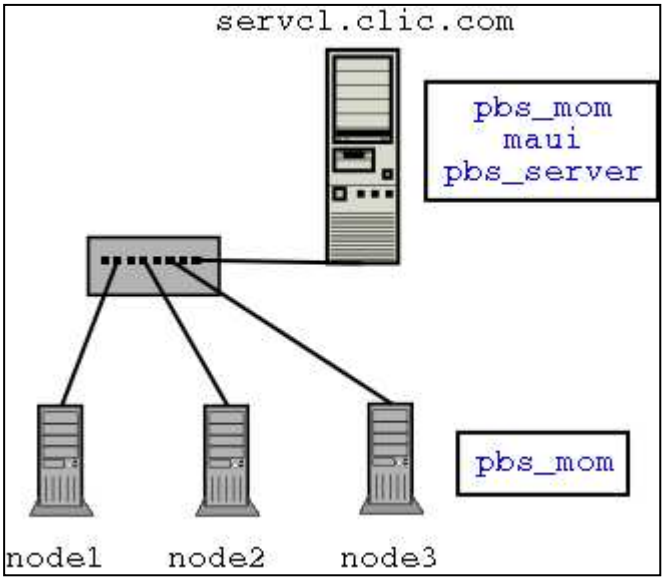
-- *parallel* : option obligatoire dans le cadre d'une installation sur un cluster.
group_name : le nom du domaine du cluster.
Package : nom du fichier rpm.

4.3.5. OpenPBS :

OpenPBS est un environnement de processus. Il est composé de trois parties :

- Le serveur (pbs_server) a pour fonction de créer, de recevoir et de modifier les processus fonctionnant sur le cluster. De plus, il peut les protéger d'une instabilité provenant de l'un des nœuds du cluster.
- L'exécuteur (pbs_mom) place les processus dans les différents nœuds du cluster. Le démon pbs_mom les place seulement lorsqu'il a reçu une copie du processus provenant du serveur.
- Le gestionnaire a pour fonction de savoir quand et ou fonctionne chaque processus. Toutefois, dans la distribution CLIC, c'est le démon Maui qui occupe cette responsabilité.

Configuration des emplacements des différents démons au sein d'un cluster :



4.4. Problèmes rencontrés lors de l'installation de la distribution CLIC :

Lors de mise en oeuvre du cluster CoCat, divers problèmes d'installation ont été rencontrés. Leur origine sera expliquée ainsi que les solutions qui ont été apportées afin de les résoudre.

4.4.1. Installation du pilote de la carte réseau Broadcom BCM 4400 :

4.4.1.1. Problème :

L'installation de la distribution CLIC s'effectue en trois parties. La première phase consiste à installer le serveur CLIC. Cette opération se déroule correctement. Malheureusement le pilote de la carte réseau Broadcom n'est pas reconnu par le programme d'installation de CLIC. Le pilote sera alors installé par la suite.

La deuxième phase consiste à installer le golden node. Pour cela, le nœud recherche une image située au sein du serveur CLIC. Le problème survient au moment de l'installation de cette image. En effet, le programme ne peut pas communiquer à l'aide de la carte réseau car il n'a pas son pilote par défaut. Le système d'ajout de pilote ne fonctionne pas ; un bug relatif à la distribution Mandrake 9.0 interdit l'accès au lecteur de disquettes. Ainsi, il est impossible d'installer et de configurer le golden node.

4.4.1.2. La solution et les différentes tentatives :

La première tentative a été d'injecter le pilote de la carte dans l'image de la distribution de CLIC. Mais, le programme d'installation ne le reconnaît pas alors qu'il fonctionne au sein du système d'exploitation Mandrake 9.0.

A la deuxième tentative, certains fichiers de l'image de CLIC ont été remplacés par un noyau de même version dont le bug du fonctionnement du lecteur de disquettes a été résolu. Mais, le programme d'installation refuse d'utiliser un pilote externe.

Lors de la troisième tentative, le noyau de CLIC (le noyau de la Mandrake 9.0) a été remplacé par celui de la version 9.1. En effet, cette version reconnaît le pilote de la carte. Mais, cette tentative a échoué ; cette version n'est pas compatible avec les outils de CLIC.

Nous nous sommes résolus d'acquérir un type de carte réseau conseillé par Mandrakesoft, une 3Com 3c90.

4.4.2. Conflit entre le service DHCP de CLIC et le serveur DHCP du réseau de l'observatoire :

Un problème est survenu lors du démarrage du serveur CLIC. En effet, son service DHCP déroutait tous les paquets IP et remplaçait ainsi le véritable serveur DHCP du réseau interne de l'observatoire. Afin d'éviter ce type de problème, le service DHCP de CLIC est configuré pour fonctionner qu'à l'intérieur du sous-réseau que constitue le cluster. Ainsi, ce service s'occupe que des paquets IP dont le destinataire est un nœud du cluster.

4.4.3. La carte réseau 3Com ne fonctionne pas :

Lors de la première utilisation, la carte réseau 3Com ne se connectait pas au réseau. Ce problème est dû à une mauvaise connexion entre la carte 3Com et la carte mère de la machine. Ce problème provient d'un conflit d'IRQ. Il faut donc en rechercher un emplacement (trouver un slot PCI) afin de garantir un bon fonctionnement.

5. L’avenir de VizieR : incorporation dans une grille de calcul

5.1. Qu’est-ce qu’une grille de calcul ?

Le concept de « grille de calcul » est apparu suite à la demande des scientifiques de disposer d’une très grande puissance de calcul et d’une capacité de stockage de l’ordre du Péta Octet. Le travail coopératif des chercheurs nécessite en permanence l’accès et l’échange d’importantes bases de données à travers l’Internet afin de corréler entre elles leurs informations, établir des statistiques, analyser…

Une « grille de calcul » se veut l'analogue d'un réseau électrique. Celui-ci fournit à chaque utilisateur toutes les ressources dont il a besoin au moyen d'une interface simplifiée à l'extrême (prise de courant) à peu standardisée dans le monde. Toute la complexité du réseau sous-jacent (de la centrale électrique au particulier) est complètement cachée. De plus, l'utilisateur peut faire varier brutalement sa consommation sans démarche préalable (dans le cadre d'un abonnement).

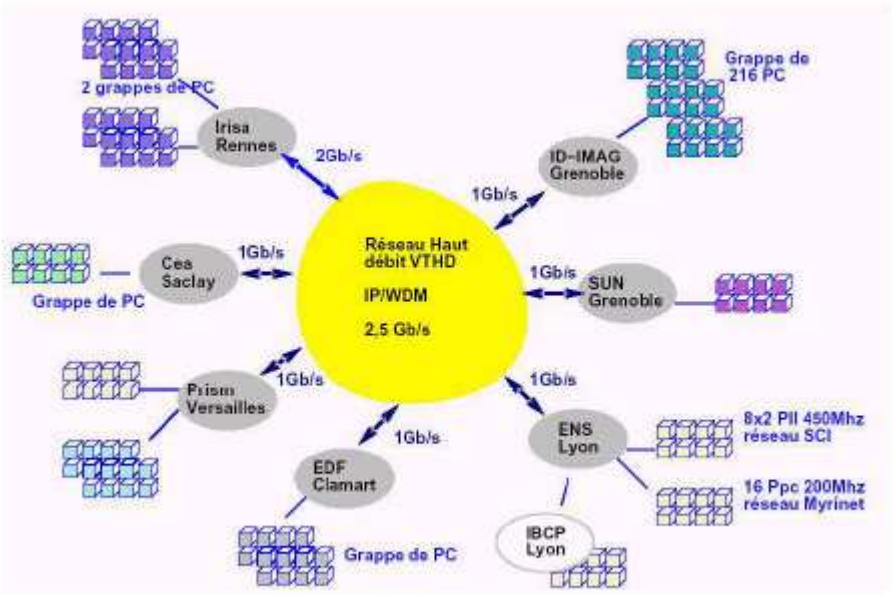
On voit bien les avantages d'un telle grille de calcul : puissance de calcul et capacités de stockage pratiquement illimitées puisque toutes les ressources de la grille peuvent être mobilisées en cas de besoin, calcul et/ou stockage à la demande, possibilité de mettre sans effort en production intensive une application développée localement, meilleure utilisation des ressources disponibles (dans les centres de calcul et dans les laboratoires pour le côté académique, dans les différents sites d'une entreprise), meilleur partage des ressources entre plusieurs disciplines, entre calcul académique et offres commerciales

On peut citer un certain nombre de domaines où cette nouvelle donne est susceptible de modifier en profondeur la problématique des moyens de calcul :

- Le calcul intensif avec les capacités de couplage des ordinateurs pour atteindre de nouvelles dimensions en matière de puissance de traitement ou de capacité mémoire,
- La visualisation,
- Les grandes bases de données.

C'est autour de ces éléments que l'architecture des systèmes informatiques va évoluer.

Exemple d'une grille de calcul : la plate-forme expérimentale du projet RNTL (E-Toile) :



5.2. VizieR au sein d'une future grille de calcul dédiée à l'astronomie :

Au sein d'une grille de calcul dédiée à l'astronomie, le cluster CoCat est une brique essentielle. En effet, cette grille a pour fonction de permettre à un astronome de croiser des informations de natures différentes (mesures, photos, publications, etc…) sans qu'il connaisse leurs provenances. Pour cela, des applications du type Globus permettent de rendre transparents l'acheminement et le traitement des informations.

Ainsi, cette grille permet à un astronome situé à Moscou, d'utiliser une application localisée à New York, avec des données provenant de Strasbourg, qui s'exécute sur des serveurs situés à Londres.

Conclusion

Comme vous avez pu le constater, la distribution CLIC n'est pas totalement mûre. En effet, l'utilisation de matériels un peu trop exotiques, génère de gros problèmes d'installation. De plus, les outils qu'elle propose ne sont pas totalement complémentaires entre eux. Ce manque de complémentarité donne l'impression aux administrateurs et aux utilisateurs qu'un cluster est une accumulation de ressources.

Mais, la mise en place d'un cluster ne se résume pas à son installation. Il faut aussi analyser l'organisation des données ainsi que les processus qui vont s'y exécuter. Cette étape est primordiale. En effet, une mauvaise analyse aurait pour conséquence d'obtenir un système d'informations moins performant que la version de départ.

Après cette analyse et cette installation, nous effectuons la mise en œuvre du projet VizieR au sein du cluster.

Personnellement, la découverte des architectures de types cluster m'a ouvert de nouveaux horizons dans le monde des systèmes d'informations. En effet, cette technologie permet d'en augmenter la puissance de calcul et la masse des données, entreposée, plus facilement que dans une architecture composée d'un serveur multi-processeur. De plus, j'ai découvert des technologies permettant de paralléliser des processus comme PVM et MPI. J'ai aussi entrevu la technologie du déploiement des nœuds d'un cluster.

Le Cluster est l'avenir des PME utilisant des logiciels très gourmands en calcul. La distribution CLIC facilite leur installation et leur gestion, ce qui leur permet de diminuer les coûts ; une personne interne peut effectuer les différentes opérations de configurations.

Pour que la technologie des clusters puisse s'exporter dans le monde industriel, il est important que des distributions comme CLIC gagnent en maturité.

Liens utiles

Site du CDS : cdsweb.u-strasbg.fr/CDS-f.html

Site de Simbad : simbad.u-strasbg.fr/sin-fit.pl

Site de VizieR : vizier.u-strasbg.fr/viz-bin/vizier

Site d'Aladin : aladin.u-strasbg.fr/aladin/gml

Site de LAM/MPI : www.lam-mpi.org

Site de PVM : www.csm.oml.gov/pvm/pvm-home.html

Site du protocole PXE : clic.mandrakesoft.com/documentation/pxe

Site de la distribution CLIC : clic.mandrakesoft.com

Site d'OpenPBS : www-unix.mcs.anl.gov/openpbs/

La documentation officielle de la librairie MPI : www-unix.mcs.anl.gov/mpi/mpich/

Glossaire

Magnitude : C’est une mesure de l’éclat d’un objet.

Mode dégradé majeur : Etat qui a pour conséquence l’arrêt complet du fonctionnement d’un cluster (perte du serveur ou du switch).

Mode dégradé mineur : Etat ayant pour répercussion la diminution des performances du cluster (Exemple : la perte d’un ou plusieurs nœuds).

ANNEXES

Annexe I : Les différents types de clusters.

Clusters Scientifiques :

Typiquement, il s'agit d'un système où l'ensemble des nœuds cumule leur puissance de calcul pour arriver à des performances égales à celles qu'atteignent les supers calculateurs universitaires. En fait, il est considéré de l'extérieur comme étant une machine multiprocesseur à part entière, spécialisée dans la résolution de problèmes scientifiques complexes.

Ce cluster utilise des applications spécialisées dans la paraléllisation de calcul à travers une couche de communication commune. En fait, même si TCP/IP représente le protocole réseau du moment, il diffuse trop de paquets d'overhead, qui ne sont pas forcement nécessaires dans le cas d'un réseau fermé comme un cluster.

A la place, un administrateur pourra utiliser le « Direct Memory Acces » (DMA, similaire à celui utilisé par certains périphériques d'un PC) à travers ses nodes, qui fonctionne comme une forme de mémoire partagée accessible par l'ensemble des processeurs du système. Il pourra aussi utiliser un système de communication dit de low-overhead comme Message Passing Interface (MPI), qui est une API (Application Program Interface) pour développeurs d'applications de calculs parallèles.

En amont de ce type de cluster nous retrouvons tous les utilisateurs en quête de puissance de calcul. C'est le cas des mathématiciens désirant résoudre des problèmes complexes, des généticiens désirant décoder plus facilement le génome humain, des astronomes cherchant à voir encore plus loin dans l'univers, ou encore de simples graphistes modélisant leurs scènes avec des équipements comme Lightwave ou Maya.

Clusters de stockage :

Ce type de système est comparable au cluster scientifique. Toutefois, ce n'est pas une puissance de calcul qui est recherchée ici, mais plutôt une puissance de stockage.
Les concepteurs de tels systèmes sont partis du constat que les entreprises utilisaient de plus en plus d'applications performantes utilisant des flux de données conséquents et donc nécessitant une capacité de stockage supérieure à celle d'un seul disque dur. Le système en clustering a pu heureusement contourner ce problème en offrant une vaste capacité de stockage virtuel.

En fait, physiquement, le fichier est découpé en blocs de taille raisonnable et stocké par morceaux sur plusieurs disques. Virtuellement, on a l'impression que l'espace de stockage ne fait qu'un et que notre fichier est stocké en un seul morceau sur un "disque".

Il s'agit pour ce type de cluster d'utiliser le potentiel des systèmes dits de "stockage combiné", c'est à dire qu'il distribue les données par l'entremise de plusieurs disques répartis sur les nœuds du cluster. Ainsi, tout utilisateur aura le loisir de travailler avec des fichiers de très grandes tailles, tout en minimisant les transferts (si la taille des blocs adoptée reste raisonnable).

Clusters Haute Disponibilité :

Les clusters dits à haute disponibilité ont été créés pour prévenir les failles hardware et software d'une seule machine, et ceci afin de garder l'ensemble des services d'un système disponible le mieux possible. La redondance, le fonctionnement du cluster et l'assurance contre les pertes peuvent être garanties à 99,9 %.

Ainsi, dans ce type de système, si le nœud primaire venait à rencontrer une défaillance, il sera immédiatement remplacé par le nœud secondaire, mis en état de "sommeil" en attendant. Typiquement, ce second nœud n'est ni plus ni moins qu'une image exacte du nœud primaire et ceci afin qu'il puisse usurper l'identité du primaire et garder ainsi l'environnement inchangé pour un utilisateur extérieur.

Il existe certains clusters à haute disponibilité capables de garder de manière redondante jusqu'au fonctionnement d'une application. Ainsi, en cas de disfonctionnement du nœud primaire, le nœud secondaire sera capable de migrer l'exécution de l'application en une poignée de secondes et ceci sans trop perturber l'utilisateur distant. Ce dernier ne ressentira qu'une baisse temporaire du temps de réponse.

Ce type de système fonctionne par l'entremise de la technique dite de heartbeat. En fait le nœud primaire envoie un signal de heartbeat périodique au nœud secondaire afin de lui notifier sa présence. Dès que le nœud secondaire ne reçoit plus ce "battement de coeur", il déclare le nœud primaire comme ne faisant plus partie du cluster et prend son identité complète.

Clusters à répartition de charge :

Les systèmes à répartition de charge permettent de distribuer l'exécution de processus systèmes ou réseaux à travers les nœuds du cluster.

Le serveur se voit ainsi attribuer la tâche de réceptionner le processus et de le répartir sur la machine adéquate. Cette dernière est en fait choisie car sa charge est faible et donc elle peut traiter le processus entrant de manière quasi instantanée. Elle peut aussi être choisie en fonction de sa spécialisation, c'est à dire qu'elle seule pourra traiter la demande sur l'ensemble des nœuds du cluster.

Toutefois, même si les nœuds du cluster n'utilisent pas les mêmes systèmes d'exportations et les mêmes entrées sorties, il existe tout de même une relation commune entre eux, matérialisée sous la forme d'une communication directe entre les machines ou à travers un serveur contrôlant la charge de chaque « workstation ». Pour pouvoir répondre à ce besoin de communication, ce type de cluster utilise des algorithmes spécifiques permettant de distribuer la charge.

Ce type de cluster est surtout largement utilisé dans le domaine du réseau et plus particulièrement sur les services lourds comme les serveurs WEB ou FTP. Ces systèmes requièrent des applications qui examinent la charge courante des nœuds et déterminent quel nœud pourra résoudre de nouvelles requêtes. Ainsi, chaque machine se verra attribuer un processus et donc la qualité de service rendu s'en trouvera meilleure. De plus, il évite les surcharges que peut subir une seule machine destinée à répondre aux requêtes du réseau.

Annexe II : LAM/MPI

LAM (Local Area Multicomputer) est un environnement de programmation et de développement respectant le standard MPI 1.1. Cet environnement permet à des systèmes hétérogènes de travailler ensemble via un réseau. Le logiciel LAM aide à la gestion des processus fonctionnant sur un cluster. Il permet d'optimiser la vitesse des processus, aide au debuggage des applications parallèles et facilite leurs développements.

Qu'est-ce que le standard de communication MPI :

MPI est une bibliothèque d'échanges de messages pour machines parallèles hétérogènes, c'est un standard créé pour le développement d'applications parallèles portables. Une application MPI est un ensemble de processus exécutant chacun son propre code (modèle SPMD) et communiquant via des appels à des sous-programmes de la bibliothèque MPI. Le modèle d'exécution d'une application MPI est le SPMD (*Single Program Multiple Data*), soit l'exécution du même programme pour tous les processus. Les paramètres et les données étant privés à chaque processus, que l'on soit sur une architecture à mémoire distribuée ou partagée. Le partitionnement, l'échange des données, le contrôle de la synchronisation des processus MPI est de la responsabilité de l'utilisateur. On peut donc facilement simuler un modèle maître-esclave.

L'interface MPI permet donc l'exploitation des machines multiprocesseurs par passage de messages. Celle-ci doit être pratique, portable efficace et flexible. La première version finalisée date de mars 1994, le projet a principalement été mené par : le centre de recherche d'IBM T.J. Watson, le NX/2 Centre, le PARAMACS.

Les principaux objectifs de ce consortium étaient de concevoir une bibliothèque permettant les points suivants :

- communications bas et haut niveau
- portabilité quasi totale
- bonne modularité (pour le développement de bibliothèques)
- supportant l'hétérogénéité
- supportant divers groupes et topologies de processus

Remarque : la librairie MPI supporte divers langages comme le fortran 90, le C, le C++ ou encore Java.

Fonctionnement de l'environnement LAM/MPI :

Nous disposons, dans cet environnement, de trois principales fonctions :

- développement parallèle d'une application (mpicc).
- Gestion de la répartition des différents processus (mpirun).
- Gestion de l'environnement LAM.

Mise en oeuvre d'une application paralléle fonctionnant dans l'environnement LAM :

La commande mpicc pemet de créer une application fonctionnant dans cet environnement. En effet, ce compilateur C effectue le lien entre les librairies standards et la librairie MPI.

Exemple de fonctionnement de la commande mpicc (compilation du programme foo):

```
mpicc -c foo.c -o foo
```

La commande mpirun :

La commande mpirun transfère chaque fichier exécutable à une machine du cluster. Par défaut, mpirun réparti les exécutables uniformément. Par exemple, dans le cas d'un cluster composé de cinq processeurs, il répartira six exécutables ainsi : deux pour le premier et un pour les quatre derniers. Cette commande garanti cette pseudo-uniformité qu'à l'initialisation. En effet, au cours du traitement, l'environnement mpi est incapable de transférer un processus d'un processeur supportant une lourde masse de calcul à un processeur plus libre.

Exemple de commande permettant de lancer quatre processus de l'exécutable a.out sur le cluster :

```
mpirun -np 4 a.out
```

Toutefois, il est possible de personnaliser la répartition des exécutables et d'attribuer un nombre de processus émanant d'un même exécutable.

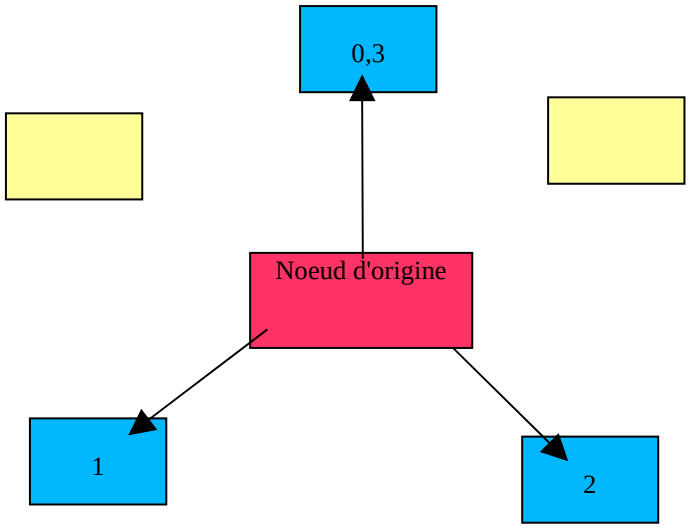
Commande permettant de comfigurer mpirun afin que 3 processus sur sun4 et 2 sur rs6000 :

```
mpirun -arch sun4 -np 2 -arch rs6000 -np 3 program
```

Initialisation d'un processus avec LAM :

Pour lancer une application, on lance notre processus à l'aide de la commande mpirun. Cette commande gère la répartition des différents processus au sein du cluster. La machine où est lancée l'application est appelée «nœud d'origine» (origin node). On appelle les autres machines les nœuds locaux (local node) . Chaque un processus a un rang (numéro d'identification). Le processus père a toujours le rang 0.

Schéma récapitulatif :



Legendes :

n,...,m — Nœud utilisé par mpirun. Ce noeud exécute les processus de rang n, ..., m.

Nœud non utilisé par mpirun.

: Nœud d'origine.

: téléchargement de l'exécutable.

Gestion de l'environnement LAM :

Commandes principales :

Lamboot : Cette commande permet de démarrer l'environnement. En effet, elle lance, sur toutes les machines du cluster, les démons qui gèrent la communication entre processus. Pour des problème de sécurité, l'environnement ne doit pas être lancé avec l'utilisateur root.

Lamclean : Cette commande détruit tous les processus de l'environnement.

Mpitask : liste les différents processus en cours.

Mpimsg : liste les différents messages qui sont en transition.

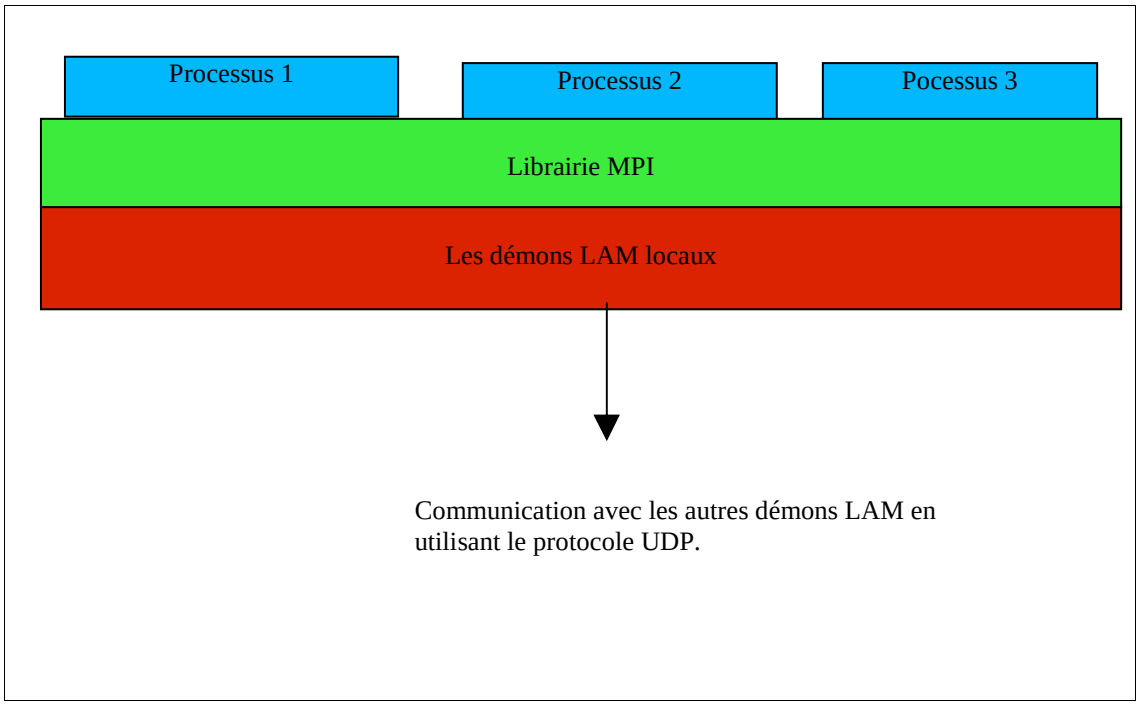
Lamhalt : Cette commande arrête totalement l'environnement en tuant tous les démons précédemment installés par la commande lamboot.

Wipe : Cette commande permet d'arrêter l'environnement correctement lorsque celui-ci est devenu instable.

Fonctionnement d'un processus avec LAM :

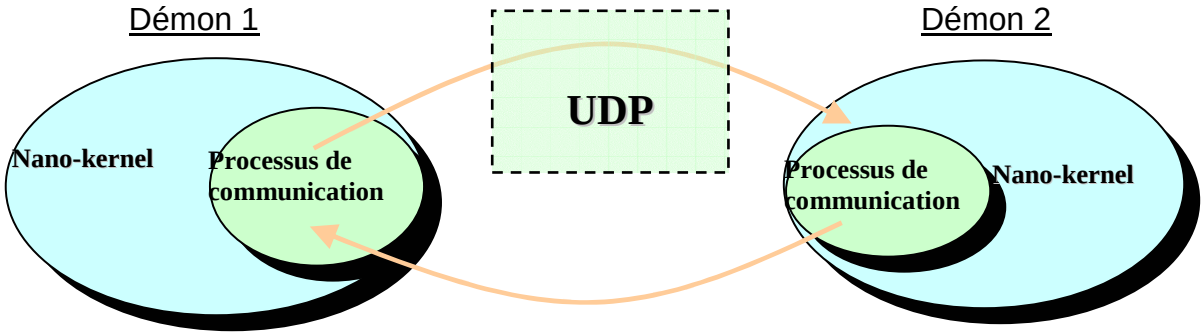
Un processus communique à l'aide d'une interface MPI. Cette interface permet de communiquer avec le handler lamboot qui envoie les différents messages aux lamboots qui s'exécutent sur les machines du cluster.

Schéma :



Fonctionnement des démons de LAM :

LAM se présente comme un simple démon présent sur chacune des machines. Chaque démon est structuré comme un micro-noyau fournissant des services de passage de messages et de synchronisation. Certain processus internes au démon forment un sous-système de communication qui permet le passage de messages vers les démons des autres machines. La communication entre les différents processus se fait via UDP c'est à dire en mode non connecté et par paquets.



La communication se fait point à point : un message va d'un processus vers un autre sans nécessairement passer par le serveur. Les programmes écrits pour LAM sont compatibles avec les autres implémentations de MPI. LAM détecte les erreurs de communication dues à un «crash» d'une machine ou à une rupture de communication. LAM bloque toutes les communications vers la machine posant problème, les processus restants sont informés de la panne de manière asynchrone.

On peut donc gérer ce type d'erreurs dans les applications en supprimant les communications vers les processus ne réagissant plus et en créant de nouveaux.

Annexe III : PVM

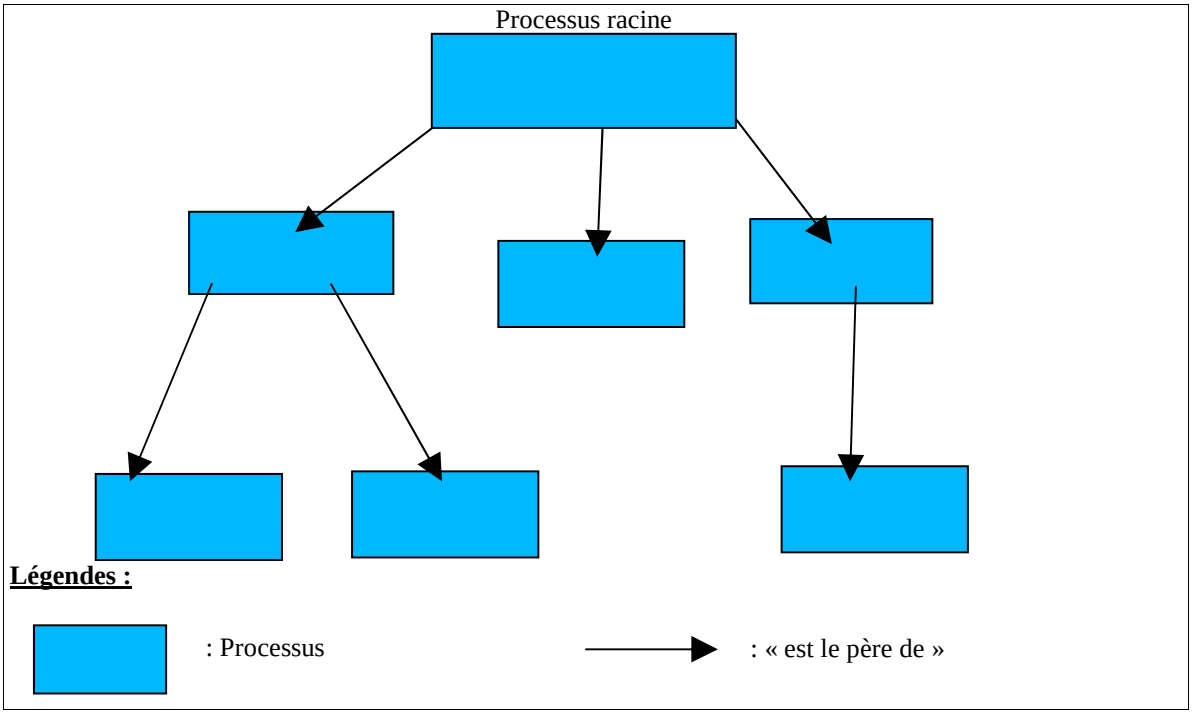
PVM (Parallel Virtual Machine) est une machine virtuelle qui gère des processus s'exécutant au sein d'un cluster homogène ou hétérogène. PVM comporte les fonctionnalités suivantes :

- gestion des nœuds,
- gestion des processus,
- gestion des messages entre processus,
- gestion des groupes de processus,
- détection des processus instables.

Organisation logique des processus au sein de la machine virtuelle PVM :

Dans une machine virtuelle, les processus s'organisent sous la forme d'un arbre. A la racine de cet arbre, il y a un processus qu'on appelle «processus racine». Cette organisation est due à la spécification de PVM. En effet, l'utilisateur peut créer qu'un seul processus et seuls ces mêmes processus peuvent en créer d'autres. Ainsi, les processus ont tous un père sauf le processus «racine». Lorsqu'un fils est créé, il hérite de l'environnement de son père. De plus, la machine virtuelle donne un identifiant unique à chaque processus. Cet identifiant permet de localiser le processus au sein du cluster ainsi que le groupe auquel il appartient.

Schéma récapitulatif :



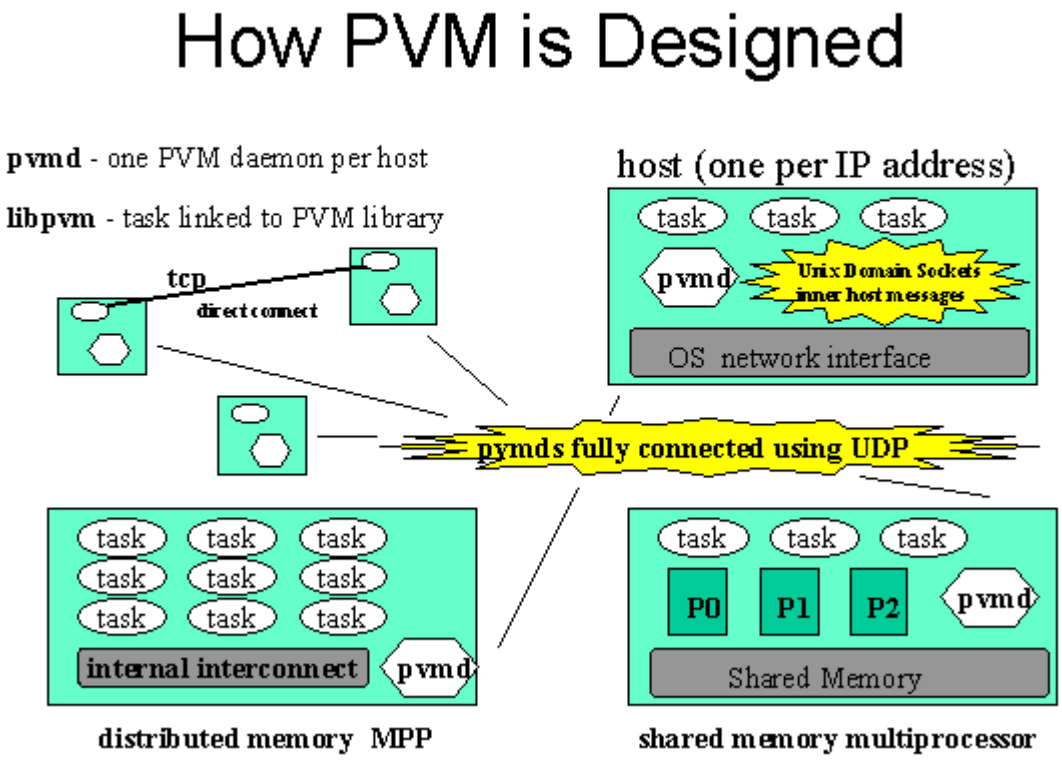
Cette organisation permet à plusieurs machines virtuelles de fonctionner ensemble. Ainsi, une machine virtuelle ne peut pas avoir accès à un processus dont elle n'est pas responsable.

Fonctionnement de la machine virtuelle PVM :

Organisation des machines virtuelles PVM :

PVM est un démon appelé «pvmd». Ce démon gère les messages qui transitent entre processus et contrôle la présence des processus dont il a la charge. Lorsqu'il est lancé sur l'un des nœuds du cluster par un utilisateur, il devient le «pvmd maître». Il lance les démons, sur les autres nœuds, qui deviennent les «pvmd esclave». Seul le maître peut créer des esclaves et peut détecter leur instabilité. Les démons communiquent entre eux à l'aide des protocoles UDP/IP. Mais, pour amorcer la communication avec un nœud, le «démon maître» utilise les protocoles TCP/IP.

Fonctionnement des démons pvmd au sein d'un cluster :



Le «démon maître» :

Le «démon maître» est composé d'un gestionnaire de messages et d'un gestionnaire de ressources (ressources manager RM). Le manager des ressources est un processus PVM qui a pour fonction de gérer l'ensemble des processus ainsi que de synchroniser les différents nœuds du cluster.

Système de communication entre démons :

Lorsqu'un processus envoie un message à un autre processus, il utilise le service offert par son démon local. Ce message est composé de deux parties, l'une permet d'indiquer à la machine virtuelle son type ainsi que son destinataire, l'autre représente le message propre.

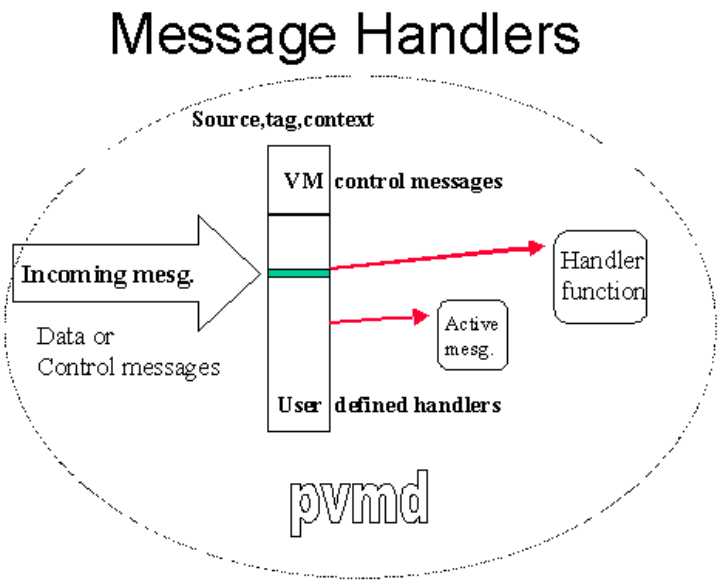
Les processus gérés par PVM peuvent envoyer deux sortes de messages :

- Les messages asynchrones (l'émetteur envoie un message mais si le récepteur n'est pas prêt alors ce message est perdu).
- Les messages synchrones (l'émetteur envoie un message qui est entreposé dans un buffer. Son destinataire peut le réclamer à n'importe quel moment.)

les messages asynchrones :

Ce type de message arrive au démon local responsable du processus destinataire. Il analyse l'en-tête afin de connaître la taille du message ainsi que le numéro destinataire. Puis, il l'achemine au destinataire.

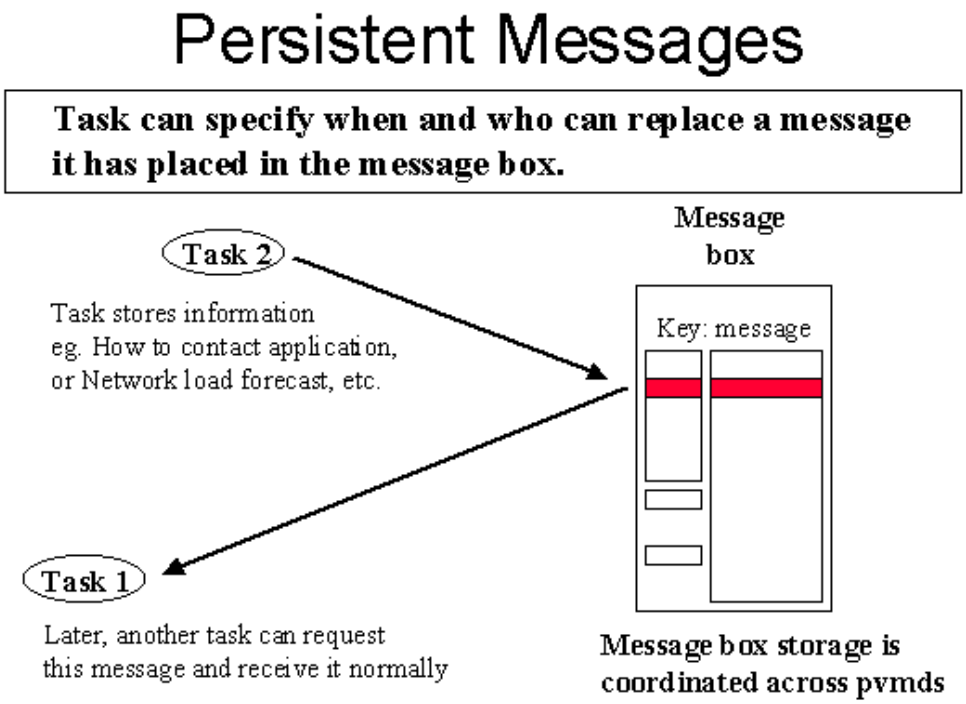
Schéma explicatif :



Les messages synchrones :

Les messages synchrones sont emmagasinés dans une table de hachage dont la clef d'appel est le nom du message et son destinataire. En effet, un processus 1 stocke un message avec un nom et le processus 2 récupère ce message en utilisant le même nom. Cette technique permet à un processus d'effectuer des traitements plus prioritaires que la communication avec d'autres processus. Ce service sert donc de «boite aux lettres» aux différents processus.

Le gestionnaire de messages :



Le mode dégradé :

Perte d'un processus :

Le démon, responsable de ce processus, le relance à nouveau avec un environnement «vierge».

Perte d'un nœud administré par un «démon esclave»:

Au bout d'un certain laps de temps (3 minutes par défaut), les démons s'aperçoivent que l'un d'entre eux ne répond plus. Il modifie automatique la topologie du cluster et relance les processus dont le démon défunt en avait la charge.

Perte du nœud du «démon maître» :

Les «démons esclaves» observent que le «démon maître» (par défaut, au bout de trois minutes) ne répond plus. Ils tuent tous les processus dont ils ont la charges puis, ils se suicident.

Conclusion :

L'environnement PVM a l'avantage d'être portable et assez facile d'utilisation. En effet, la bibliothèque de base comporte que 35 fonctions. De plus, il interface avec de multiples langages C, C++, fortran 77, etc.... Le développement d'application client-serveur demande des outils puissants pour la programmation, le test, le déboggage et la mise au point. PVM est riche en extensions, ce qui facilite le développement de ce type d'application.

Toutefois, PVM est assez lent comparé à la technologie MPI.. Cette lenteur est un grand handicap pour les applications de type calcul distribué. De plus, son ancienneté n'a pas permis à ses concepteurs de mettre en oeuvre une bibliothèque de formes de messages, assez large par rapport à ce que propose la norme MPI..

Annexe IV : La librairie MPI

MPI est une librairie C portable permettant d'échanger des messages entre processus repartis sur un ensemble de processeurs. Tous les fichiers qu'utilisent les différents processus sont supposés être déjà sur les machines. MPI ne gère pas dynamiquement les processus (pas de migration par exemple). Par contre, MPI garantit la préservation de l'ordre des messages si plusieurs messages sont envoyés à la suite.

Un programme, écrit à l'aide de la librairie MPI, est compilé par la commande mpicc. Le nombre de processus est donné au lancement du programme par la commande mpirun. En effet, cette commande utilise le fichier parallel.cfg qui décrit les machines disponibles et les groupes auxquels elles appartiennent. Ainsi, si l'on veut lancer 3 processus ex, on tape :

```
mpicc ex.c -o ex
mpirun -np 3 ex
```

- **Description d'un programme utilisant la librairie MPI :**

Un processus commence toujours par définir un monde commun qui contient l'ensemble des processus du programme parallèle. Ceci se fait par la commande **MPI_Init**.

Il est ensuite alors possible d'accéder au nombre total de processus (**MPI_Comm_size**), au nom (i.e. numéro) du processus courant par **MPI_Comm_rank**, au nom de la machine qui gère le processus courant (**MPI_Get_processor_name**), ...

Le programme finit toujours par **MPI_Finalise**.

- **Les différents types de messages :**

La librairie MPI met en oeuvre six types de messages qui permettent aux différents processus de communiquer entre eux. Les informations échangées entre processus sont des tableaux d'éléments qui peuvent être du type :

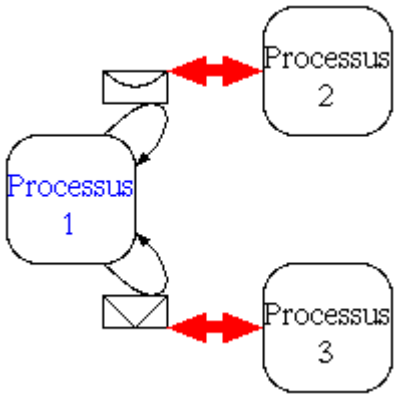
- MPI_CHAR.
- MPI_SHORT.
- MPI_INT.
- MPI_LONG.
- MPI_UNSIGNED_CHAR.
- MPI_UNSIGNED_SHORT.
- MPI_UNSIGNED.
- MPI_INSIGNED_LONG.

- MPI_FLOAT.
- MPI_DOUBLE.
- MPI_LONG_DOUBLE.
- MPI_BYTE (groupe de 8 bits).
- MPI_PACKED (pour gérer d'autres types).

- **Les messages asynchrones non sécurisés :**

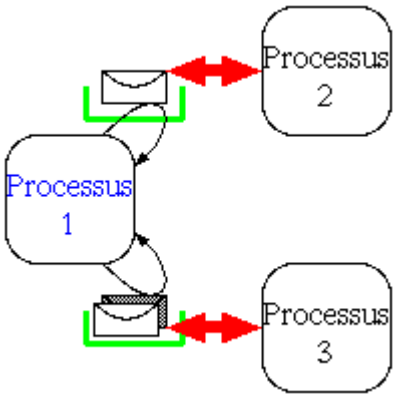
Le processus envoyeur envoie son message par la commande **MPI_Send** et continue son travail (**non bloquant**). Le message est alors bien envoyé, mais pas nécessairement arrivé.

Un processus réceptionne le message par la commande **MPI_Recv** quand il le souhaite. Cette attente bloque le travail du processus réceptionneur. L'échange n'est pas garanti et le message peut être perdu ou écrasé.



- **Les messages asynchrones bufférisés :**

Deux processus veulent échanger un message sécurisé sans rendez-vous. Le processus envoyeur se créé une zone tampon dans laquelle seront mis les messages. Cette zone doit être attachée au processus avant le premier envoi par la commande **MPI_attach,** puis détachée avant la fin du programme par **MPI_detach**. Le processus envoyeur envoie son message par la commande **MPI_Bsend** et continue son travail. Le processus réceptionneur récupère le message par la commande **MPI_Recv** quand il le souhaite. Cette attente bloque le travail du processus réceptionneur. L'échange est garanti et le système assure que le message ne peut être perdu.



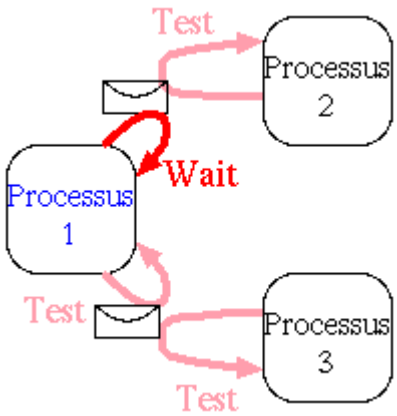
- **Les messages asynchrones sécurisés :**

Deux processus veulent échanger un message sécurisé sans rendez-vous et sans tampon. Le principe est le même qu'avec les procédures simples **MPI_Send** et **MPI_Recv** avec des tests possibles et de façon non bloquante en émission et à la réception. Dans ce cas, l'envoi s'effectue avec **MPI_Isend** et la réception avec **MPI_Irecv**. Un paramètre supplémentaire est ajoute à ces procédures : c'est un indicateur de requête pour effectuer les tests.

Une première procédure de test permet d'attendre qu'une partie de la communication (envoi ou réception) se soit bien passée. On utilise alors **MPI_Wait**. Cette procédure d'attente est alors bloquante.

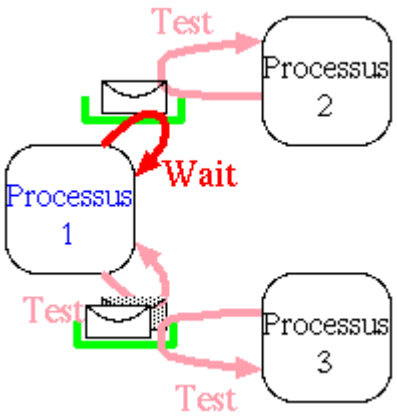
Une seconde procédure de test permet de savoir si une partie de la communication (envoi ou réception) s'est bien passée. On utilise alors **MPI_Test**. Cette procédure est non bloquante.

L'échange est garanti ou le système affiche un message d'erreur.



- **Les messages asynchrones sécurisés bufférisés :**

Même principe que les messages bufférisés simples, avec la possibilité de contrôle à l'aide des commandes **MPI_Test** et **MPI_Wait**.



- **Les messages asynchrones sécurisés en mode réception prête :**

Deux processus veulent échanger un message sécurisé sans rendez-vous (**non bloquant**) et sans tampon, mais l'émetteur doit être sûr que le récepteur est prêt. On utilise la procédure **MPI_Irsend** qui a un équivalent non sécurisé appelé **MPI_Rsend**.

C'est l'écriture du programme qui doit garantir que le réceptionneur est prêt. Si ce n'est pas le cas, la communication n'est pas censée fonctionner et le résultat est non prédictible.

Dans ce modèle d'échange, on cherche à être certain que le message ne sera jamais perdu (écrasé dans le buffer ou refusé pour cause de buffer plein) en économisant la gestion d'un buffer (puisque le récepteur est prêt à consommer le message).

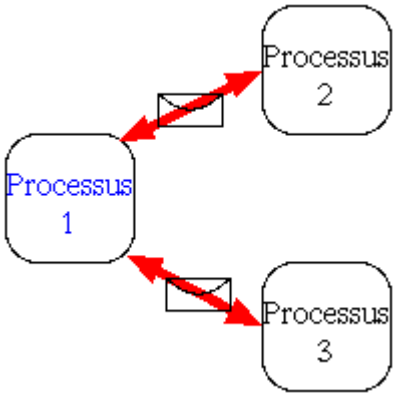
En reprenant l'exemple utilisant **MPI_Isend** et en le modifiant avec **MPI_Irsend**, le programme fonctionne parfaitement même si on oblige l'émetteur à émettre avant que le récepteur ne soit prêt !

Il semble donc préférable et plus clair de choisir soit le mode asynchrone sécurisé soit le mode synchrone.

- **Les messages synchrones :**

Les messages synchrones utilisent le principe du rendez-vous entre deux processus (la procédure d'envoi et la procédure de réception doivent se "rencontrer" a un même moment).

Les procédures simples (**bloquantes**, tout le processus s'arrête en attente de l'achèvement de la communication) et sécurisées (**non bloquantes**, le processus continue de s'exécuter) s'appellent respectivement **MPI_Ssend** et **MPI_Issend**.



- **Principales possibilités de la librairie MPI :**

Optimisation des communications :

Lorsque l'on passe un ordre d'envoi ou de réception, un lien de communication est créé puis détruit à chaque fois. On peut optimiser les communications en créant un lien permanent qui sera réutilisé autant de fois que nécessaire.

Le lien de communication peut être rendu permanent du coté de l'envoyeur, du récepteur ou des deux a la fois.

Tous les types de communication peuvent y recourir.

Un lien permanent est créé par les commandes **MPI_Send_init**, **MPI_Bsend_init**, **MPI_Rsend_init**, **MPI_Recv_init**. Ces commandes utilisent un paramètre qui sert à stocker la requête qui sera utilisée par la future communication.

Une fois créé, le lien permanent doit être activé par **MPI_Start** (pour une requête donnée) ou **MPI_Startall** (pour un tableau de requêtes).

Test et attente de plusieurs communications asynchrones à la fois :

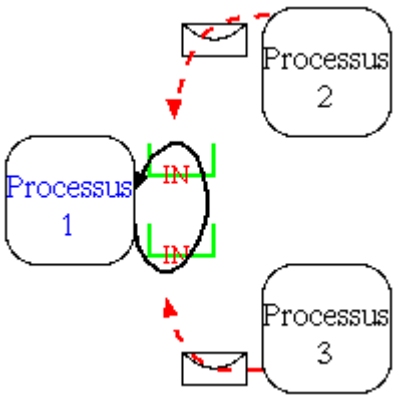
Un processus s'est mis en attente de plusieurs communications asynchrones (**MPI_Irecv**). On veut utiliser des opérations similaires a **MPI_Wait** et **MPI_Test**, mais sur tout ou partie de ces communications.

On dispose de deux familles de procédures :

MPI_Waitany, MPI_Waitall, MPI_Waitsome

MPI_Testany, MPI_Testall, MPI_Testsome

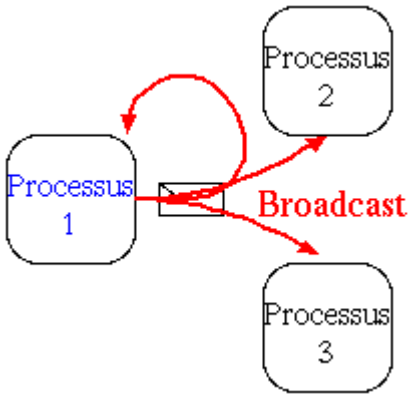
Ces procédures travaillent sur des tableaux de requêtes (type **MPI_REQUEST**) et renvoient des tableaux de statut (type **MPI_STATUS**). La famille des **MPI_Waitxx** et bloquante alors que celle des **MPI_Testxx** ne l'est pas.



Diffusion générale d'un message :

Un processus veut diffuser un message à tous les autres (et à lui-même), il utilise la procédure **MPI_Bcast.** Cette procédure est utilisée à la fois en émission du message et en réception.

Un des paramètres de **MPI_Bcast** indique quel est le processus émetteur.



Synchronisation de processus par barrières :

On souhaite synchroniser les processus d'un groupe à un endroit donné du programme. Ceci peut se faire par le biais d'une barrière posée par l'instruction **MPI_Barrier**.

Quand un processus donné arrive à cette instruction dans le programme, il s'arrête et attend que tous les autres processus du groupe soient arrivés à cette même barrière. Lorsque c'est le cas, chacun des processus peut reprendre son travail.

La topologie logique :

La librairie MPI permet également de définir une topologie logique des processus (cartesian virtual topology). Une topologie logique permet de retrouver chaque processus sans connaître leurs véritables adresses mémoires. Ainsi, l'organisation des processus est totalement transparente aux développeurs.

Conclusion :

MPI permet donc de compiler et d'exécuter un programme parallèle ou distribué. Les processus de ce programme sont définis statiquement. Cette librairie a pour but principal de définir un ensemble de primitives de communication portables sur un grand nombre de machines. Il faut noter qu'avec MPI les processus sont naturellement et statiquement liés à un processeur. Il n'est donc pas simple de se détacher complètement des considérations d'implantation finale.

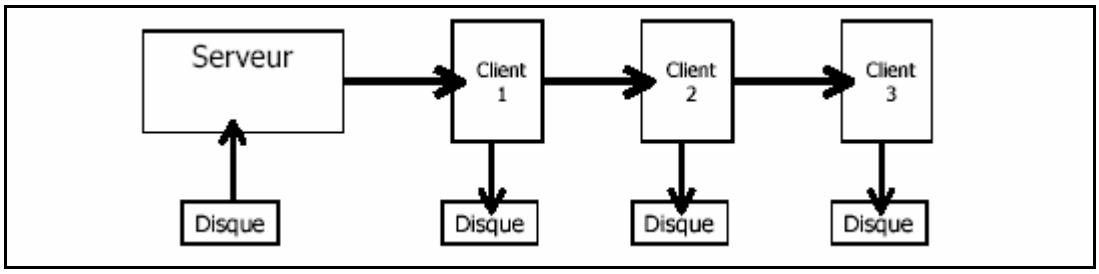
Annexe V : Le déploiement des nœuds au sein d'un cluster

L'installation des nœuds s'effectue par le réseau pour éviter des tâches manuelles répétitives. La taille des données à transférer (plusieurs gigas) impose une exploitation optimale du réseau. L'installation d'une machine vierge oblige à écrire un secteur de boot sur la machine cliente. Pour cela, un serveur fournit une image du système, créée lors de l'installation manuelle d'un premier nœud qui servira de serveur d'installation. Un nœud vierge démarre sur sa carte réseau et charge depuis le réseau en plusieurs étapes les informations et le système d'exploitation qu'il va utiliser. C'est le protocole PXE, implanté dans la bootrom de la carte Ethernet, qui est appelé pour interroger des serveurs DHCP et TFTP qui fourniront les programmes et images d'installation.

BpBatch est un programme capable d'interpréter de petits scripts écrits dans un langage spécifique, donnant ainsi la possibilité de démarrer une machine de plusieurs manières différentes en fonctions de certains paramètres ou de l'intervention d'un utilisateur. BpBatch est notamment capable de partitionner un disque dur, de formater des partitions, de créer des fichiers, faire démarrer la machine sur son disque dur, ou sur une image de disquette, on encore de démarrer un noyau Linux chargé à travers le réseau.

Lorsque plusieurs nœuds doivent être installés en même temps, il est nécessaire d'utiliser une procédure où le serveur d'image ne risque pas la saturation. Pour cela, on met en place une chaîne de diffusion permettant la copie en une seule étape du système de fichiers.

Chaîne d'installation du système d'exploitation :



L'application résultante se nomme Ka et est disponible en téléchargement sous licence GPL. L'installation de machines avec Ka se passe de la manière suivante : on utilise PXE/BpBatch pour démarrer un mini système Linux. Ce mini-système monte son système de fichiers racine par NFS à travers le réseau. Ensuite il crée et formate les partitions qui vont être utilisées pour installer le système d'exploitation, puis lance le client

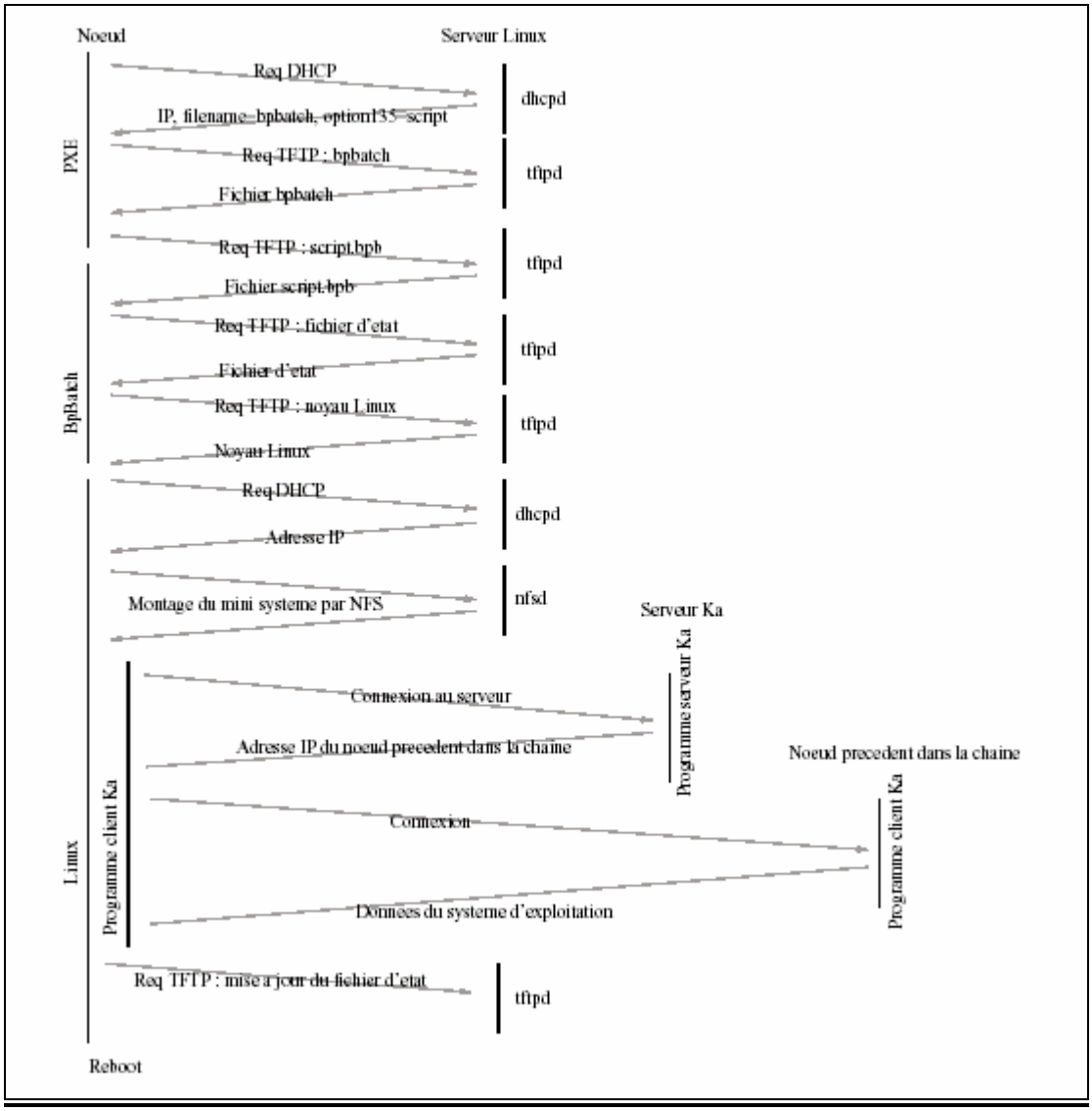
Ka. Le client Ka va se connecter à un serveur Ka situé sur la machine à cloner. Et lorsque toutes les machines à installer se sont connectées au serveur Ka, celui-ci coordonne la création d'une chaîne de connections TCP.

Une fois cette chaîne créée, elle est utilisée pour diffuser vers toutes les machines les données constituant le système à installer. Les données sont alors lues depuis le disque dur de la machine à cloner, envoyées vers les machines à installer et écrites sur le disque dur de celles-ci en continu.

Une fois toutes les données écrites sur le disque dur, la machine peut redémarrer et utiliser le système qu'elle vient d'installer. Afin de ne pas recommencer la procédure d'installation au début lors de ce redémarrage, on maintient à jour sur un serveur l'état d'avancement de l'installation pour chacune des machines. Ce fichier est consulté par BpBatch lors du démarrage afin de choisir l'action à effectuer (démarrer l'installation ou démarrer le système qu'on a installé ?). Ce fichier est mis à jour au fur et à mesure des étapes de l'installation.

Dans notre cas, ce fichier peut contenir trois états différents : système encore à installer, système installé mais secteur de « boot » pas encore écrit, système prêt. Chacun de ces états, correspond à une action : installation de la machine, écriture du secteur de « boot », démarrage de la machine pour la production.

Voici pour résumer un aperçu des échanges faits à travers le réseau lors de la première étape de l'installation.



Annexe VI : La technologie PXE

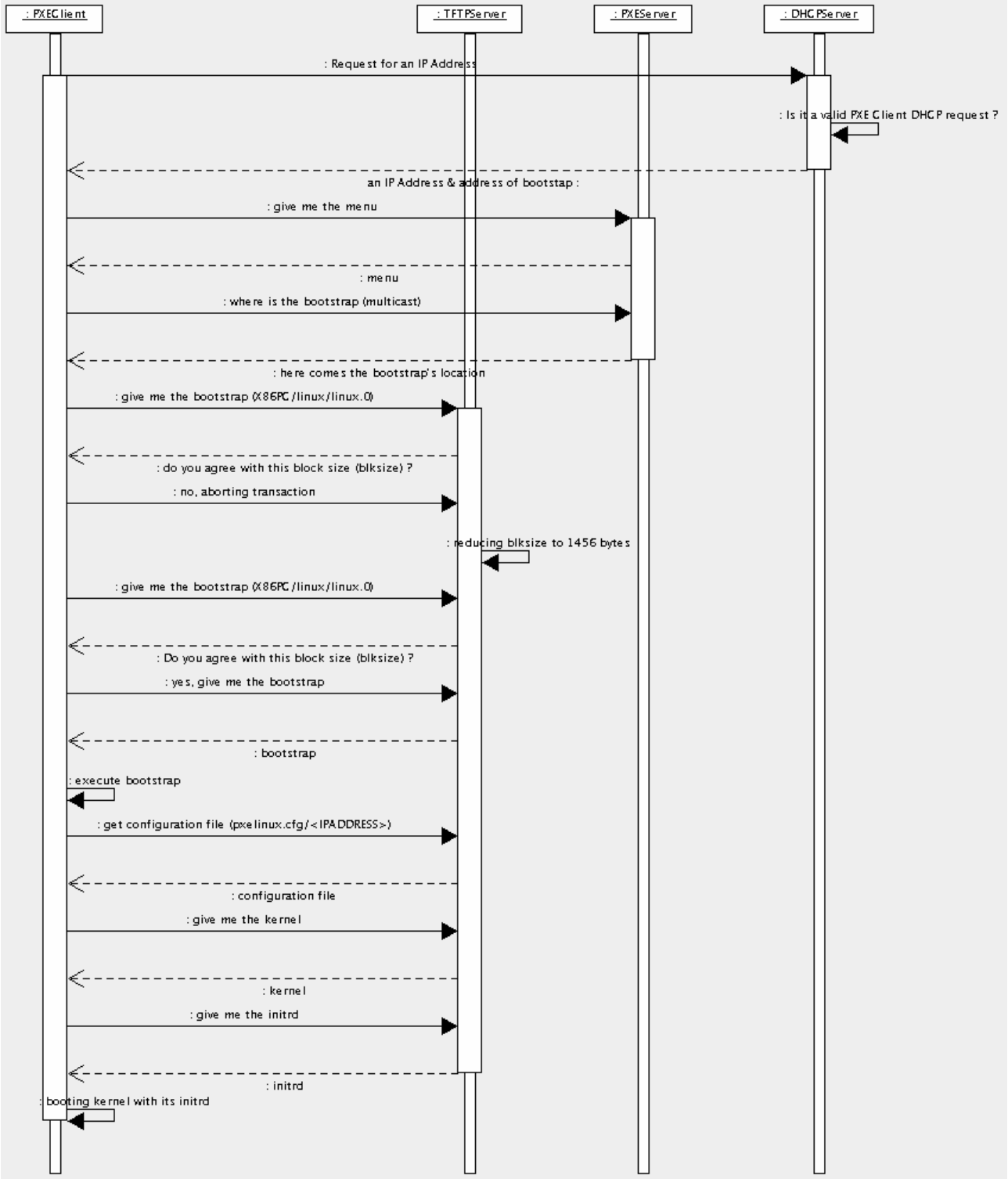
PXE (Pre-boot Execution Environnment) est un protocole spécifié par la société Intel. Ce protocole permet à un ordinateur de booter directement par l'intermédiaire de sa carte réseau. En effet, PXE se situe dans la ROM de ces cartes réseaux, nouvelle génération. Quand l'ordinateur boot, le Bios lit la ROM PXE et l'exécute.

Ce protocole recherche un serveur PXE au sein du réseau. Lorsque cette recherche est effectuée, il affiche un menu qui permet à l'utilisateur de choisir les noyaux qui sont disponibles sur le serveur. Puis, il télécharge et exécute le noyau sélectionné au sein du client.

La machine serveur est constituée de trois serveur :

- Un serveur DHCP qui a pour fonction de donner une adresse IP dynamiquement aux différents clients PXE.
- Un serveur TFTP qui permet de télécharger le noyau du système d'exploitation ainsi que les différents fichiers de configuration.
- Un serveur PXE qui gère le menu et le bootstrap (module d'initialisation avant l'exécution du noyau).

Diagramme de séquence de l'initialisation d'un client PXE :



Résumé

Les performances des ordinateurs de bureau standards à base de processeur Intel ont atteint un tel niveau qu'il est aujourd'hui possible de construire à partir de ces machines des supercalculateurs aussi rapides que les calculateurs dédiés type CRAY, IBM, SP ou SGI Origin 2000. D'autre part, des problèmes technologiques ou simplement les coûts de certains composants (commutateurs pour l'accès à la mémoire par exemple) limitent les possibilités de passage à l'échelle des machines multi processeurs. Dans ce contexte, les clusters (ou grappes) de PC apparaissent comme une très bonne alternative aux autres moyens de calculs.

Le projet CoCat (co-processeur catalogue) n'est pas aussi ambitieux que de mettre en oeuvre un supercalculateur. Dans notre cas, nous nous contenterons d'élaborer un cluster composé de six machines interconnectées par un réseau de 100 Mb. Ce cluster accélérera la gestion des requêtes, sur des catalogues astronomiques, effectuées par les astronomes du monde entier. Ce cluster sera une pièce maîtresse dans VizieR qui est l'un des trois principaux projets du CDS (Centre de Données astronomiques de Strasbourg).

Pour amener à bien ce projet, un état de l'art présente les différents types de cluster ainsi que les différents logiciels qui s'articulent autour. Puis, nous avons préparé la mise en œuvre du cluster en analysant les différentes organisations des données et les processus à paralléliser.

Nous présentons l'installation de la distribution CLIC qui est une sur-couche du système d'exploitation Mandrake 9.0. Lors de cette installation, divers problèmes ont été rencontrés. En effet, le pilote de la carte réseau Broadcom n'était pas compatible avec CLIC. De plus, des problèmes avec le service DHCP sont apparus ainsi qu'avec la carte réseau 3Com. Nous décrivons la boite à outils de cette distribution (ex : les outils « Berkeley », OpenPBS, etc…). Enfin, nous prospectons l'avenir du cluster CoCat au sein d'une grille de calcul dédiée à l'astronomie.