Proposition de TER L3

Une commande « make » compatible avec les clusters

Sujet

Mettre en place un cluster (appelé encore grappe de serveurs ou ferme de calcul) consiste à mettre en place plusieurs ordinateurs indépendants en réseau qui vont apparaître comme un seul ordinateur ayant plus de performances (puissance du processeur, dimension de l'espace de stockage, quantité de mémoire vive, etc.). Chaque ordinateur du réseau est appelé nœud de calcul. Un des objectifs évident de la mise en place de clusters est d'avoir plus de puissance de calcul et on les utilise généralement pour des applications qui nécessitent beaucoup de temps de calcul.

Sur une machine donnée avec un processeur possédant plusieurs cœurs, l'utilisation des différents cœurs par les processus est complètement transparent pour l'utilisateur qui n'a pas à se soucier de répartir ses processus sur les différents cœurs. Dans un cluster, la répartition des processus sur les nœuds de calcul n'est pas transparente. L'utilisateur doit en effet utiliser des commandes spécifiques au système de d'ordonnancement de tâches du cluster (plusieurs systèmes existent). Certes, le système d'ordonnancement de tâches offre une barrière d'abstraction à l'utilisateur, qui n'a pas à se soucier de la communication entre les machines constituant le cluster, mais il doit explicitement utiliser des commandes pour distribuer ses tâches sur les nœuds du cluster.

L'objectif de ce TER est de récupérer un peu de transparence dans l'utilisation des clusters sur une commande en particulier, à savoir la commande (Linux) « make ». Cette commande est extrêmement pratique car outre l'automatisation de la compilation de fichiers, elle permet plus généralement d'automatiser n'importe quel traitement en ligne de commande. L'option « -j », suivie d'un nombre n, est particulièrement intéressante car elle permet de lancer n processus (de compilation ou de traitement, suivant ce que l'on met dans le fichier de configuration « Makefile ») simultanément. Ces différents processus sont distribués automatiquement par l'ordonnanceur sans que l'utilisateur n'ait à spécifier quoique ce soit. Une telle option n'est malheureusement pas compatible avec les clusters (il faut en effet embarquer de nombreuses notions de réseaux pour la partie communication entre machines du cluster) et notre objectif sera de la porter pour qu'elle

puisse distribuer les processus non seulement sur les différents cœurs, mais aussi sur les différents nœuds de calcul du cluster.

Pour réaliser notre portage, nous nous placerons dans un cluster en particulier (qui nous permettra de tester notre développement) et donc dans un système d'ordonnancement des tâches spécifique (comme dit précédemment, plusieurs systèmes existent, et il n'y a, pour le moment, pas de standard qui émerge). Le cluster en question sera un de nos clusters locaux, à savoir HPC@LR, hébergé au CINES, et qui utilise le gestionnaire de ressources de calcul SLURM. Notre objectif sera donc de simuler le comportement de l'option « -j » de la commande « make » sur tous les nœuds du cluster (et plus seulement sur les cœurs d'un nœud de calcul) en utilisant les commandes de SLURM qui permettent de gérer l'ordonnancement des processus sur les nœuds du cluster. Pour ce faire, nous modifierons le code la commande GNU « make », qui est écrit en C et qui est disponible (code « open source »).

Travail à réaliser

— Regarder les propositions qui ont été faites pour résoudre ce problème (plusieurs travaux existent sur le sujet, dont certains proposent de nouvelles versions de « make », mais rien n'est standard) ;
— Se familiariser avec le cluster HPC@LR et notamment avec son système d'ordonnancement des tâches SLURM (on essaiera, en particulier, de développer quelques petites applications nécessitant des calculs en parallèle) ;
— Modifier la commande GNU « make » pour qu'elle prenne en charge la distribution des processus sur plusieurs nœuds de calcul du cluster et tester le code correspondant (avec différents « Makefiles »).

Prérequis

— Avoir des notions systèmes et réseaux, notamment sur l'ordonnancement des processus et sur la communication entre machines ;
— Connaître la commande « make » dans son utilisation très basique et savoir programmer un peu en C ;
— Aucune connaissance sur les clusters et leurs systèmes d'ordonnancement des tâches n'est nécessaire.

Remarques additionnelles

L'encadrement du TER sera réalisé par :
— David Delahaye (Université de Montpellier, LIRMM, David.Delahaye@lirmm.fr).