Outils de développement et d'optimisation

Jeudi 8 octobre 2009

Romaric DAVID
david@unistra.fr
Direction Informatique
Département Expertise pour la Recherche

Plan

Introduction

- Un des rôles des centres de calcul ⇔ Faciliter le développement et la mise au point d'applications
  - Outils de développement fondamentaux
  - Connus des utilisateurs ⇒ doivent être connus du gestionnaire
- Panorama de ces outils, du développement à l'optimisation

Plan

Introduction

Les familles de compilateurs

Les familles de MPI

Analyse de performances MPI

IDE pour le développement

Débogueurs

Conclusion

Familles de compilateurs

1ère distinction : prix

Nom / Version	Architecture	Prix indicatif
Gcc 4.4	Toutes	0
Intel 11.1	X86_64/IA64	1000€/2 jetons, 2385€/5 jetons
Portland 9.2	X86_64	2800€/2 jetons
Open64*	X86_64, ARM, ..	0
Nag Fortran 5.2	X86_64	280€ (Node locked)
Absoft Fortran 10.2	X86_64	1025€/2 jetons, 2150€/5 jetons

Certains de ces compilateurs sont couplables avec des IDE du même éditeur

*Outil de recherche

Familles de compilateurs

Les processeurs devant massivement multi-coeurs, que proposent les compilateurs pour leur exploitation automatique (OpenMP, Parallélisation automatique)?

Nom / Version	OpenMP	Autres fonctionnalités
Gcc 4.4	OMP 3	Vectorisation
Intel 11.1	OMP 3	Auto-parallélisation, Profil
Portland 9.2	OMP 3	Auto-parallélisation, GPU
Hmpp	(cf Intel / Gcc)	GPU (Nvidia, ATI, CAL)

Familles de compilateurs - Bibliothèques

Associé indispensable des compilateurs = les bibliothèques mathématiques. Quelle bibliothèque pour quel compilateur ?

Nom / Version	OpenMP	Autres fonctionnalités
Bibliothèque	Origine	Compilateurs compatibles
ACML	AMD	Gnu, Intel, Portland, Nag, Open64, Absoft
MKL	Intel	Intel 10/11, Gcc >= 4.2, PGI 7.1.6
Atlas	Communauté	GPU (Nvidia, ATI, CAL)
Nag	Nag	Nag, Intel, PGI

Familles de compilateurs - cohabitation

Certaines applications ont besoin d'un compilateur précis. Ex : Gaussian sur x86_64 se compile avec PGI, sur Ia64 avec Intel

Certaines bibliothèques sont liées à un compilateur précis. Ex : MPI ?

Compilateur par défaut pour vos utilisateurs ?

Cohabitation : commande *module*

Familles de compilateurs - module

Scripts tcl facilitant la manipulation de PATH, LD_LIBRARY_PATH, MANPATH, ...

`module list` : modules chargés dans l'environnement utilisateur

`module avail` : modules disponibles sur le système

`module load/remove/purge`

Intérêt : Description de dépendances entre logiciels, conflits

Familles de compilateurs - module

http://modules.sourceforge.net/

Nécessite de ré-écrire les scripts d'initialisation des applications (LD_LIBRARY_PATH, …)

Exemple ?

Familles de compilateurs - conclusion

Lors de l'achat de compilateurs, vérifier les produits annexes (bibliothèques)

Demandes précises des utilisateurs

Peut engendrer multiplication des tâches

S'inspirer de l'environnement des grands centres ?

Plan

Introduction

Les familles de compilateurs

Les familles de MPI

Analyse de performances MPI

IDE pour le développement

Débogueurs

Conclusion

Familles de MPI

MPI : Standard de primitives de passage de messages, 1994

Conçu pour machines à mémoire distribuée

MPI-2 : 1997, évolution du standard

Associé à plusieurs langages de programmation

Problèmes à la mise en production : couplage avec le gestionnaire de batch,prise en compte des réseaux haut-débit

Familles de MPI

Prise en compte des sytèmes de batch ?

Facilité non seulement à démarrer, mais surtout à contrôler (terminer, interrompre) les processus pendant l'exécution

Prise en compte des réseaux haut-débit ?

Une même primitive peut se réaliser « pour de vrai » de différentes manières

Familles de MPI – Envoi de données

Exemple de mécanismes pour MPI_Send() :

- TCP/IP : Socket, Write, Read
- Myrinet (MX) : mx_isend, mx_recv
- RDMA : put + flow_control
- SM : memcpy

Familles de MPI

Nombreuses implémentations

Nom / Version	Architecture	Fonctionnalités
Intel MPI ٣.٢	X٦٤_٨٦/IA٦٤	MPI ٢-, SM, RDMA, Sockets
HP MPI ٢.٣.١	X٦٤_٨٦/IA٦٤	MPI2, SM, RDMA, Quadrics, Sockets
OpenMPI 1.3.3	Toutes (sources)	MPI2, SM, RDMA, Quadrics, Sockets
MPICH١.١.١ ٢p١	Toutes (sources)	MPI ٢, SM, Sockets, BG
MVAPICH ١.٤ ٢	Toutes (sources)	RDMA
MPICH ١.٢.٧	Toutes (sources)	SM, TCP

Ces implémentations se couplent avec certains gestionnaires de batch.
Au minimum PBS (Torque)

Elles sont incompatibles (binairement) entre elles

Pourquoi ?

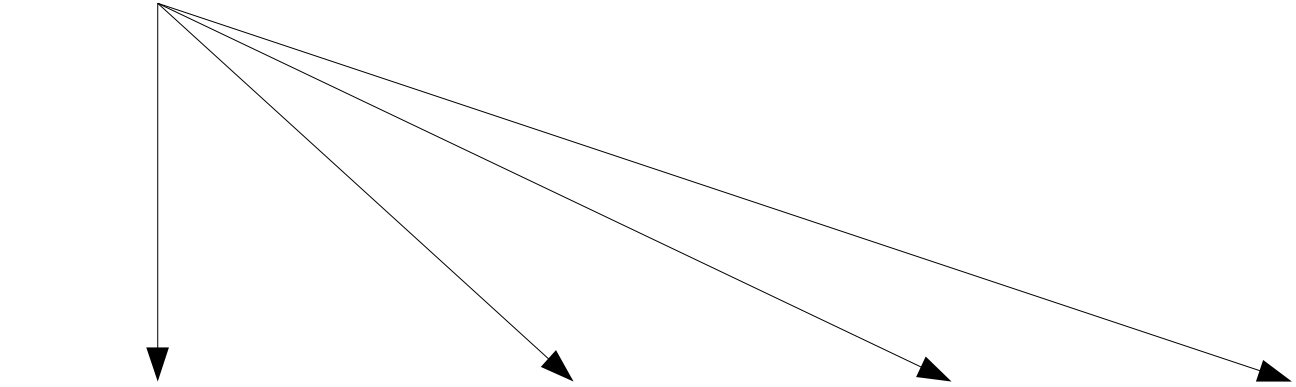
Familles de MPI – Choix

- Fonction des programmes utilisateurs.
- Par exemple ADF fourni avec HP-MPI sur X86_64
- Fonction de l'environnement de batch. Est-ce que mon environnement de batch prévu sait cohabiter avec l'implémentation x,y,z de MPI ?
- Retour d'expérience sur Open MPI

Familles de MPI – OpenMPI

Architecture de OpenMPI

Composants modulaires




```
mpirun -mca pls rsh -mca pls_rsh_agent ssh
```

⇒ Utilisation d'un lanceur de type rsh avec ssh

Familles de MPI – OpenMPI

OpenMPI essaye automatiquement de déterminer comment lancer des processus

Examen de l'environnement utilisateur

Utile pour le batch

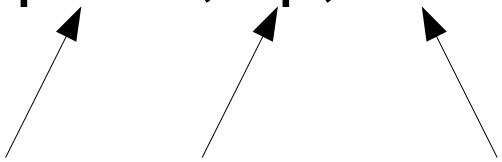
Environnements reconnus :

- Torque (appelé TM)
- Sun Grid Engine (via rsh)
- Autres environnements : hack rsh

Familles de MPI – OpenMPI

Prise en compte des réseaux d'interconnexion dans OpenMPI

mpirun -mca btl openib,tcp,self



Inf niband Tcp/ip Interne à un processus

Utilisation de plusieurs réseaux
Choisit le plus adapté / disponible

Familles de MPI – OpenMPI

Où trouver OpenMPI ?

- Dans les paquetages de sa distribution :
  - Réglages du compilateur (OMPI_CC, OMPI_F77)
  - Version ? (actuellement 1.3.3)
- Compiler à la main
  - Penser aux options bien pratiques comme `--enable-mpirun-prefix-by-default`
  - Avantage : fait référence au compilateur que vous avez choisi
- À partir des paquetages sources OFED

Familles de MPI – OpenMPI

Comment l'utiliser ensuite ?

RAPPEL : Le compilateur à utiliser pour les codes utilisateur devient mpif90, mpicc, …

Pourquoi ?

- Prise en compte des bons chemins d'#include
- Édition de liens avec l'ensemble de bibliothèques
- Patch Makefile fournis par les utilisateurs

Familles de MPI – Mvapich2

Fork de mpich2 spécialisé Infiniband. Gère TCP/IP

Checkpoint/restart pour les processus lancés par rsh

Familles de MPI – Évolutions

- Placement des processus sur les coeurs (PALPI)
- Checkpoint/restart
- Tolérance aux pannes

Plan

Introduction

Les familles de compilateurs

Familles de MPI

Analyse des performances MPI

IDE pour le développement

Débogueurs

Conclusion

Analyse des performances MPI - Préambule

Il est possible d'utiliser gprof pour analyser les performances de codes parallèles

- Variable d'environnement GMON_OUT : donne le préfixe des noms de fichiers qui seront produits
- Fichiers GMOUN_OUT.pid

Analyse des performances MPI - Principe

Chaque fonction de la bibliothèque MPI est présente sous la forme de 2 symboles :

- MPI_something, appelé habituellement
- PMPI_something, appelé pour de vrai

Le symbole MPI_something est un weak symbol. Si un autre symbole de même nom est présent, ce dernier prend le dessus

- Exécution : génération de *fichiers de traces* contenant l'historique du déroulement du programme

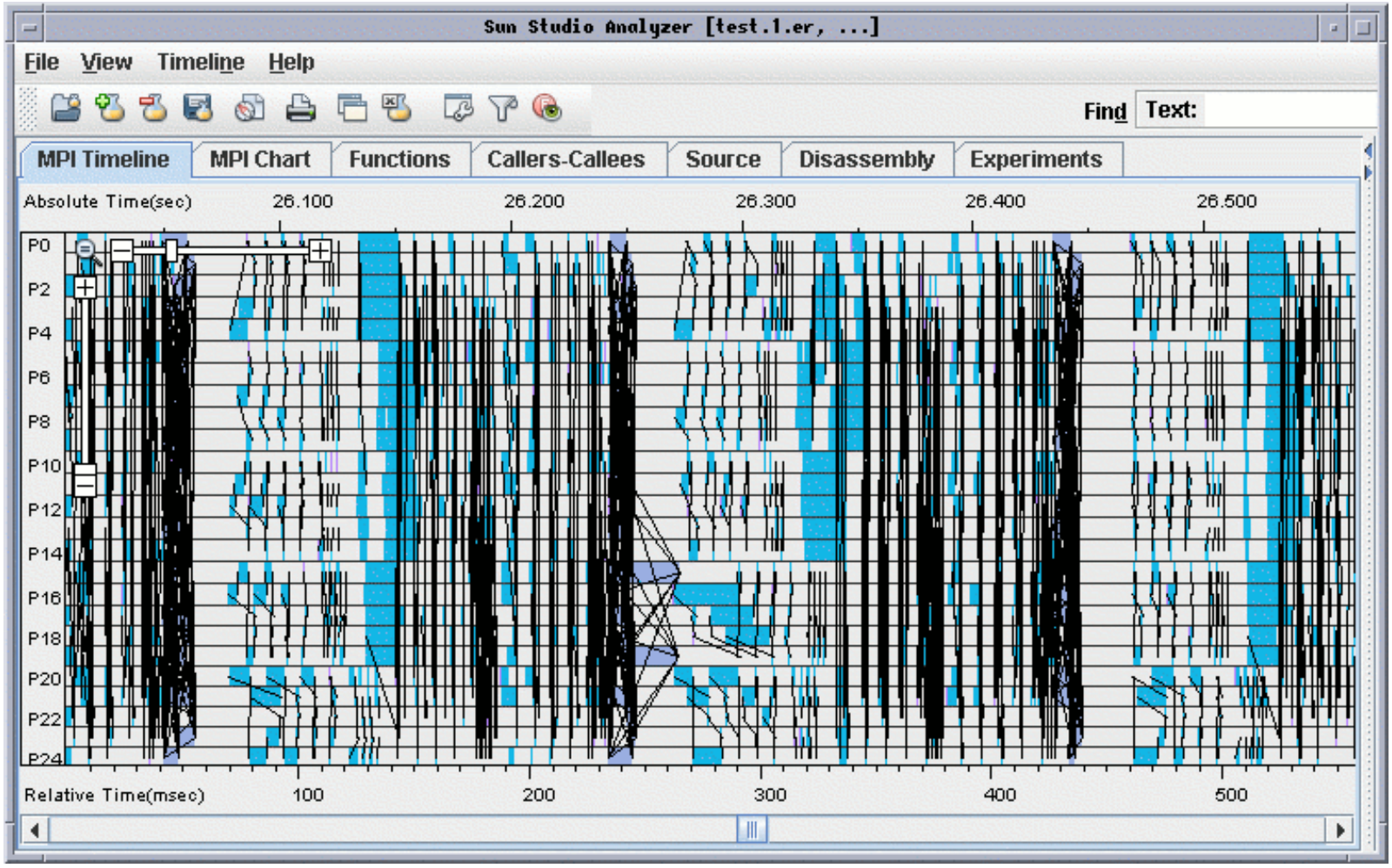
Analyse des performances MPI – Outils disponibles

- Intel trace collector/analyzer 7.2 (Payant)
- Tau (Libre)
- Outil *collect* des Sun Performance Tools (Gratuit)
- Scalasca 1.2 (MPI, OpenMP) www.scalasca.org (Libre)

Fichiers de trace, dont la visualisation est complexe dès qu'un grand nombre de processeurs ont été sollicités pour l'exécution ⇒ outils d'aide à la visualisation

Ex : Cube de Scalasca

Analyse des performances MPI – Outils disponibles



Timeline avec Sun Studio

Analyse des performances MPI – Keep it simple

- MpiP 3.1.2 : lightweight MPI profiling (http://mpip.sourceforge.net/)
- But : distinguer temps MPI / temps non MPI
- Compilation avec bibliothèques mpiP
- Lié à Mpich ?
- Encore maintenu ?

Analyse des performances MPI – Outils disponibles

Que faire d'autres avec les traces ?

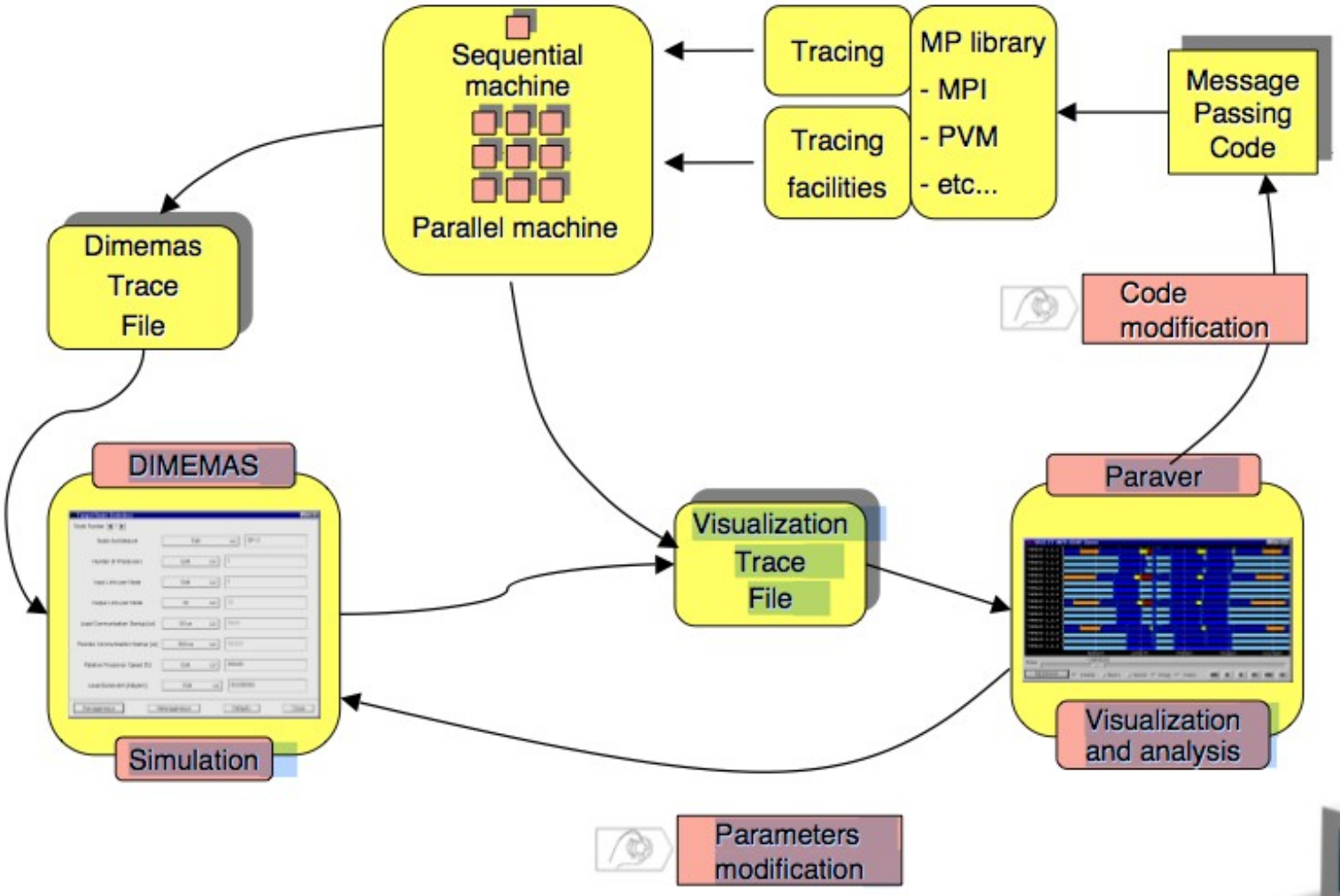
- Utilisation pour prédiction de performances d'applications : Dimemas 2.3 (http://www.bsc.es/plantillaA.php?cat_id=497)
- Prévoir passage à l'échelle de codes
- Prévoir achat de machines : ai-je besoin de réseau haut-débit ?

Analyse des performances MPI – Dimemas

Fonctionnement :

- Exécution réelle du programme à analyser, collecte de traces
- Interprétation des traces par Dimemas
- Cycles de :
  - Modification des paramètres (machine, réseau)
  - Simulation par Dimemas
  - Examen des résultats
- Modification de code

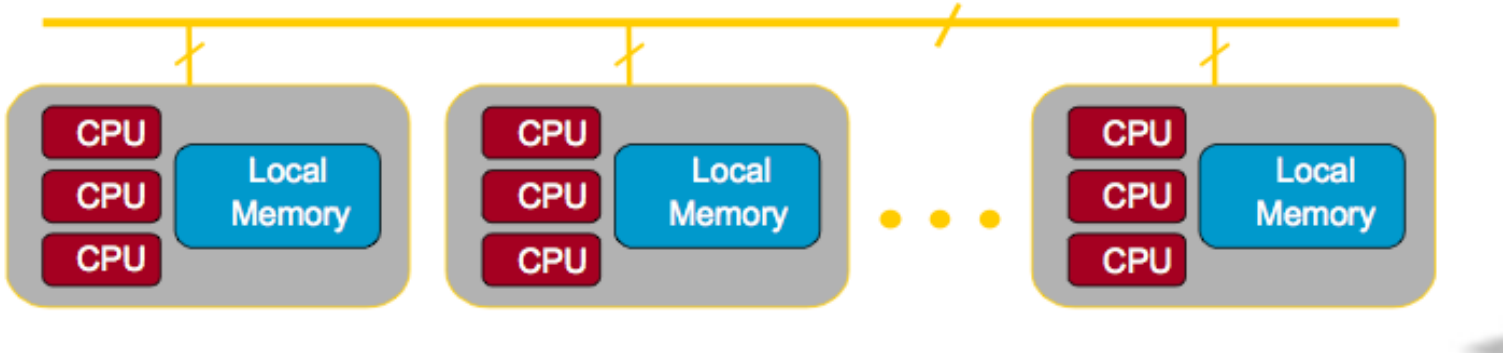
Analyse des performances MPI – Dimemas



dimemas@cepba.upc.es

Analyse des performances MPI – Dimemas

Modèle de machine suivant Dimemas :



Modèle de réseau abstrait :

- Bande Passante
- Type et nombre des liens (half/full duplex)
- Débit de bissection, points de contention

Analyse des performances MPI – Pistes

Communications non bloquantes :

- Implémenté par MPI (progess thread ?)
- Consommation CPU sur les coeurs ?

Opérations collectives :

- Habituellement conseillé de les utiliser
- Suffisant ?

Changer de MPI ?

Ré-écrire le code ?

Plan

IDE pour le développement

- De nombreux éditeurs proposent un IDE avec leurs compilateurs
- Eclipse : standard de fait dans le monde du libre
- Développement de programmes parallèles avec Eclipse ?

IDE pour le développement

- Eclipse s'enrichit de *plugins*
- G-Eclipse : développement sur environnements hétérogènes www.geclipse.eu
- PTP 2.1 : Parallels Tools Project : www.eclipse.org/ptp/
- Contenus de PTP :
  - Environnement de lancement de processus
  - Débogueur parallèle
    SDM : Scalable Debug manager

IDE pour le développement - Installation

Installation de PTP :

- Pré-requis : Eclipse 3.4, OpenMPI / Mpich2
- Eclipse CDT ⇒ C/C++ uniquement

L'installation de PTP nécessite la compilation (binaire) de SDM et des *proxys* pour le lancement des programmes

Plan

Introduction

Les familles de compilateurs

Les familles de MPI

Analyse de performances MPI

IDE pour le développement

Débogueurs

Conclusion

Debogueurs

- PDT vu précédemment
- Totalview (Etnus)
- DDT (Alinea)
- Mpigdb
- g-eclipse

Debugging - Totalview

- Débogueur « historique » de codes parlallèles
- Actuellement version 8.7
- Débogage local et à distance (via serveur de debogage)
- Ligne de commande ou GUI
- Fonctionnalités avancées :
  - Files de messages MPI
  - Examen des types structurés des langages
  - Visualisation des données pendant la session de debug
  - Insertion de patchs dans le code
- Éléments de prix : 2300 € pour 16 processus

Debugging - DDT

- Concurrent à Totalview
- Actuellement version 2.4.1
- Fonctionnalités identiques à Totalview

Debugging – OpenMPI et MPICH2

- Outils plus simple
- Options passées à mpirun
- Surcouches autour de gdb
- Ligne de commande

Debugging - g-eclipse

- *Plugin* à Eclipse
- Destiné aux développements sur grille (lancement des processus)

Plan

Conclusion

- Vaste choix d'outils
- Outils de base : logiciel libre mûr
- Outils évolués (debug) : reste du chemin à parcourir
- Achats mutualisés de logiciels

Conclusion - Workflow

On vous soumet un programme...

- Le compiler avec « vos » compilateurs
- Y insérer les bibliothèques optimisées
- En première analyse de performances : gprof (//)
  - « Je le savais »... dit l'utilisateur
  - Ensuite, Traces
- Debug
  - Est-ce que le programme séquentiel tourne ?
  - En parallèle, achat de débogueur ?