Empowered by Innovation NEC

Atelier
Profilage de codes de calcul

Introduction

Laurent Gatineau
Support applicatif
NEC HPC Europe

Ecole Centrale de Paris

11 juin 2014

Plan

- Qu’est-ce que le profilage de code ?
- Quand faire du profilage ?
- Terminologie du profilage
- Différents niveaux de profilage
- Outils de profilage

Auteur: Laurent Gatineau

Qu'est-ce que le profilage de code ?

Profilage de code (wikipedia):

> Analyser l'exécution d'un code afin d'en connaître son comportement.

Buts du profilage:

- Pour optimiser les temps de calcul.
- Pour réduire la consommation mémoire / disque.
- Afin d'estimer le comportement d'un code sur une architecture différente.
- Comprendre le comportement d'un compilateur, processeur…

Auteur: Laurent Gatineau

Quand faire du profilage ?

- Le plus tôt possible…
  - Pour faire le choix des structures de données.
  - Pour faire les choix algorithmiques.
  - Pour faire les choix d'un solveur.

- Après les phases de tests, quand le code de calcul tourne correctement…
  => Peut donner des pistes d'optimisations / améliorations.

Auteur: Laurent Gatineau

Quand faire du profilage ?

- Lors d'un changement d'architecture ou de système, et que les temps de calcul sont plus longs…
  => Peut donner des pistes sur la cause des ralentissements.

- Quand les temps de calcul deviennent trop long…
  => Mais là, c'est peut-être déjà trop tard…

- Attention, il faut savoir:
  - Quoi mesurer (ou au moins regarder une chose à la fois).
  - Quand s'arrêter… Se fixer des objectifs / limites.

Auteur: Laurent Gatineau

Terminologie du profilage

- Profilage / Profiling: Analyse de l'exécution d'un code.
- Profileur / Profiler: Outils de profilage.
- Point chaud / Hot spot: Portion d'un code consommateur en temps.
- Graphe d'appels / Call-graph: Permet de savoir dans quel ordre s'appellent les fonctions d'un code.
- Echantillonnage / Sampling: Méthode de profilage statistique.

Auteur: Laurent Gatineau

Terminologie du profilage

- Instrumentation: Méthode de profilage par l'ajout d'instructions dans le code.
- Compteur matériel / Hardware counter: Permet de compter des instructions ou des évènements matériel (défaut de cache, défaut de branchement, …).
- CPI (Cycles Par Instruction / Cycles Per Instruction): unité de mesure permettant de connaître l'efficacité d'un code.
- TLB (Translation Lookaside Buffer): cache permettant d'accélérer la conversion des adresses virtuelles en adresses physiques.

Auteur: Laurent Gatineau

Différents niveaux de profilage

- Instrumenter son code:
  - Mise en place de « timers »
  - Phases d'initialisations, phases de calcul, d'entrées / sorties, communications.
  - Facile et pratique.

```
Pair  time (%) = 125.575 (96.204)
Neigh time (%) = 0.973892 (0.746104)
Comm  time (%) = 2.55522 (1.95756)
Outpt time (%) = 0.00164243 (0.00125827)
Other time (%) = 1.42418 (1.09107)
```

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR01C2R4       82656   168     2     2            1128.93             3.33484e+02
WR01C2R4       84960   160     2     5            1002.30             4.07912e+02
WR01C2R4      218720   160     4     7            2991.70             2.33164e+03
--------------------------------------------------------------------------------
```


Auteur: Laurent Gatineau

Différents niveaux de profilage

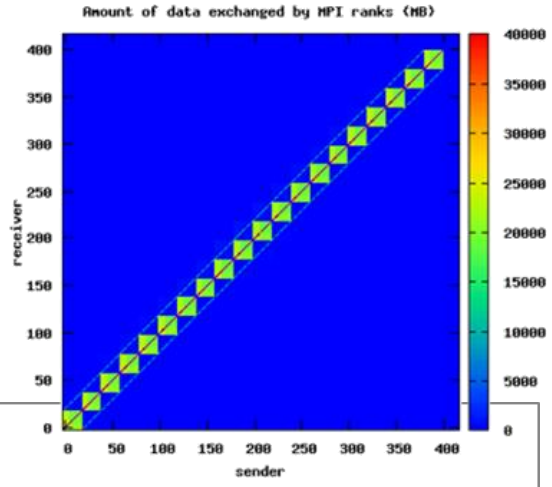
- Profilage simple afin d'identifier les « hot-spots »:
  - Connaître les fonctions consommatrices de temps de calcul.
  - Connaitre le temps passé dans les librairies (calculs / communications, …).

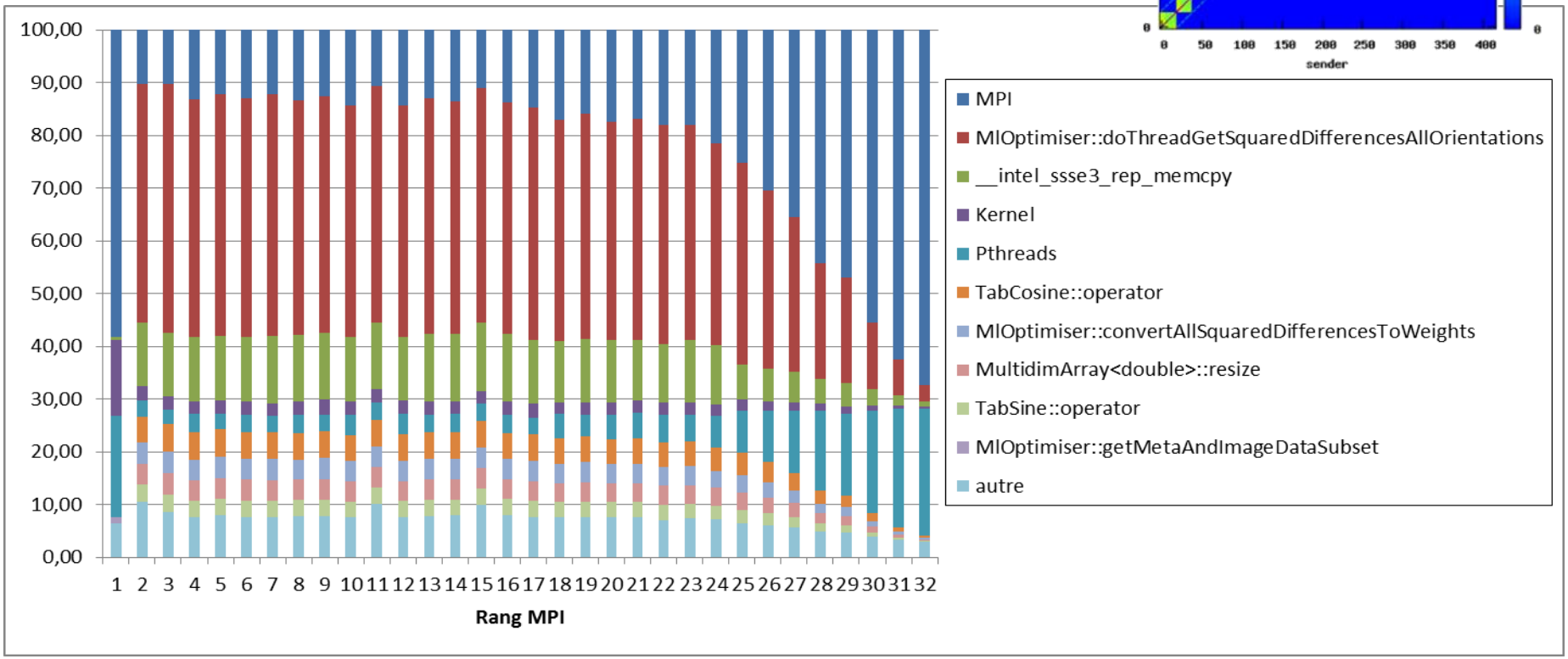
```
$ perf report --input perf_0000.dat -n --stdio --sort comm,dso
# Events: 155K cycles
#
# Overhead  Samples    Command         Shared Object
# ........ ..........  .......  .....................
#
    89.74%     139255     xhpl  libmkl_avx.so
     5.35%       8309     xhpl  xhpl
     4.08%       6339     xhpl  libmpi_dbg.so.4.1
     0.74%       1146     xhpl  [kernel.kallsyms]
     0.06%         93     xhpl  libc-2.12.so
     0.03%         40     xhpl  libmkl_core.so
     0.00%          6     xhpl  libmkl_intel_lp64.so
     0.00%          3     xhpl  ld-2.12.so
     0.00%          1     xhpl  libmkl_sequential.so
     0.00%          1     xhpl  libpthread-2.12.so
```


Auteur: Laurent Gatineau

Différents niveaux de profilage

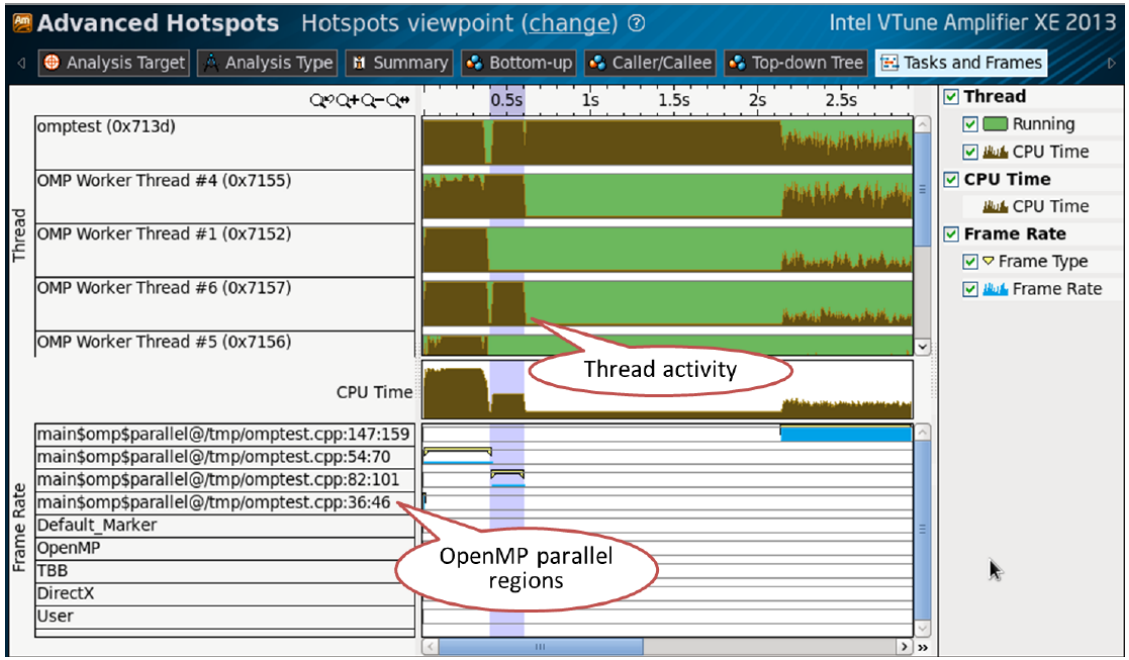
- Profilage MPI:
  - Matrices de communications.
  - Identifications des synchronisations.
  - Equilibrage de charge.





Différents niveaux de profilage

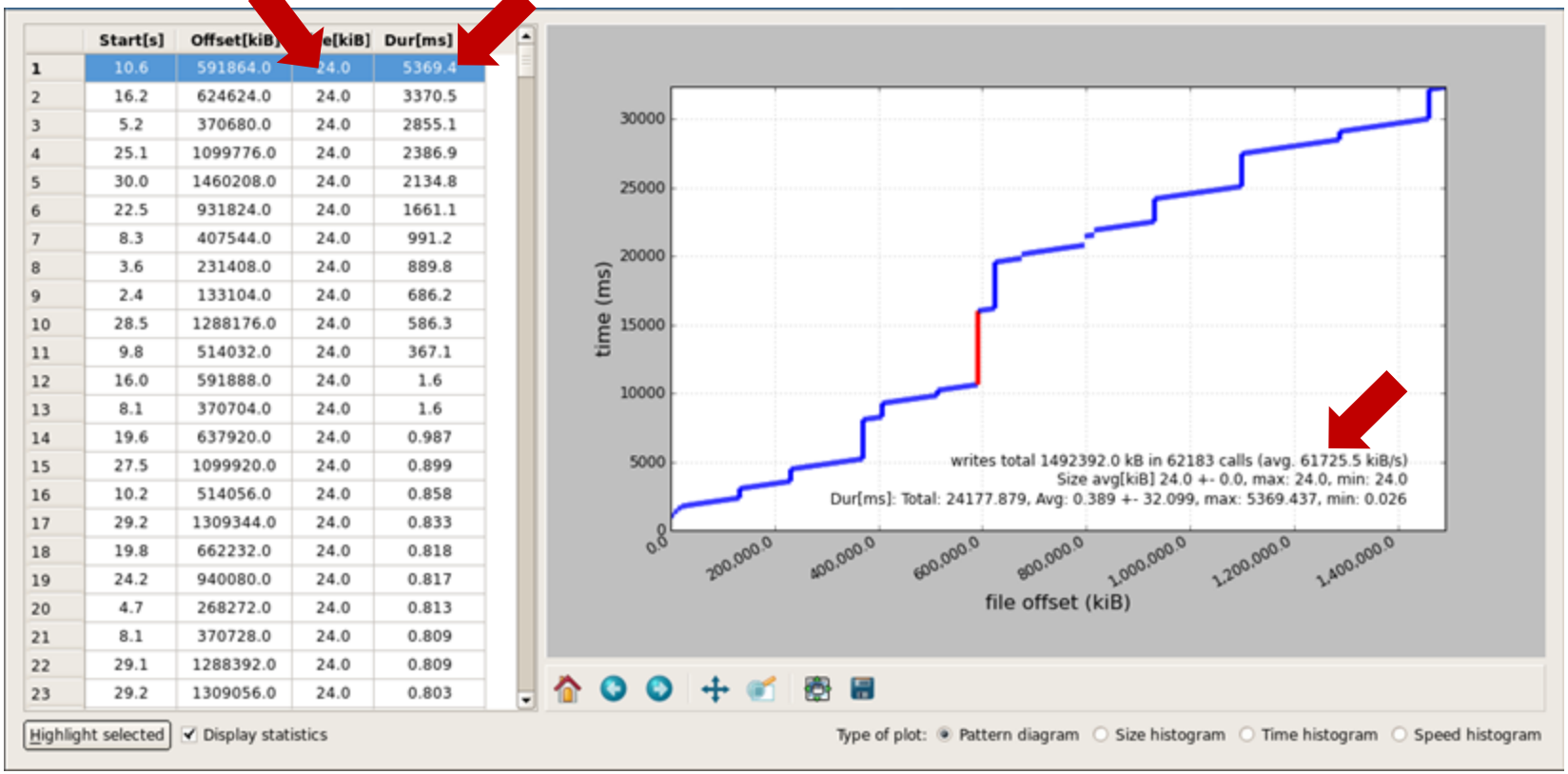
- Profilage OpenMP:
  - Coût des synchronisations et des opérations de réductions.
  - Coût de l'ordonnanceur.
  - Coût de la gestion du cache (« false-sharing », …).
  - Equilibrage de charge.



Auteur: Laurent Gatineau

Différents niveaux de profilage

- Profilage des entrées / sorties disques:
  - Taille des entrées / sorties.
  - Type (séquentielles, aléatoires, en arrière, …)



Auteur: Laurent Gatineau

Différents niveaux de profilage

Profilage via les compteurs matériels

- Calcul du CPI.
- Gestion du cache / des accès mémoire.
- Taux de vectorisation.
- Gestion du « pipeline » d'instructions.
- …

Auteur: Laurent Gatineau

Outils de profilage

- Pour débuter:
  - GPROF: instrumentation du code et échantillonnage.

- OpenSource, basé sur PAPI:
  - TAU (Tuning and Analysis Utilities):
    - http://www.cs.uoregon.edu/research/tau/home.php
    - Graphique et ligne de commande.
    - Séquentiel, Parallèle (MPI & MultiThread).
  - Scalasca (SCalable performance Analysis of LArge SCale Applications):
    - http://www.scalasca.org/
    - Graphique.
    - Séquentiel, Parallèle (MPI & MutliThread).

Outils de profilage

- OpenSource, profilage MPI (statistiques):
  - mpiP (Lightweight, Scalable MPI Profiling):
    - http://mpip.sourceforge.net/
    - Résultats sous format texte.
    - A ajouté lors de l'édition de lien.
  - IPM (Integrated Performance Monitoring):
    - http://ipm-hpc.sourceforge.net/
    - Résultats sous format texte et graphique.
    - Profile CPU.
    - Sans recompilation.

Auteur: Laurent Gatineau

Outils de profilage

OpenSource, profilage MPI (Tracing):

- OpenMPI:
  - http://www.open-mpi.org/
  - Profile intégré à OpenMPI.
  - Compatible avec VampirTrace (format de fichier OTF).

- MPE / Jumpshot:
  - http://www.mcs.anl.gov/research/projects/perfvis/download/
  - http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/
  - Compatible avec différentes librairies MPI.
  - Jumpshot: interface graphique.

Auteur: Laurent Gatineau

Outils de profilage

OpenSource, profilage statique:

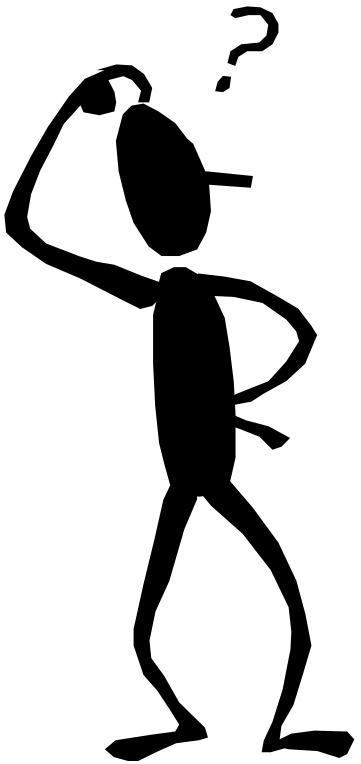
- MAQUAO (Modular Assembly Quality Analyzer and Optimizer):
  - http://www.maqao.org/
  - Analyse statique du code assembleur.
  - Couplage avec instrumentations.
  - Simulateur.
  - Peut permettre de mieux comprendre les accès aux données notamment dans le cas de code OpenMP.

OpenSource, profilage OpenMP:

- ompP (OpenMP Profiler):
  - http://www.ompp-tool.com/
  - Basé sur PAPI.

Auteur: Laurent Gatineau

QUESTIONS ?



Auteur: Laurent Gatineau