Intel Parallel Studio: Vtune

C.Berthelot
Christophe.Berthelot@atos.net

Laboratório
Nacional de
Computação
Científica

Atos

Agenda

- Introduction
  Bottelneck
  Gprof

- Vtune Amplifier XE
  Introduction
  The Software Optimization Process
  Vtune: GUI
  Sum up
  Labs: Demo

Introduction

Bootelneck

In software engineering, a bottleneck occurs when the capacity of an application or a computer system is severely limited by a single component. The bottleneck has lowest throughput of all parts of the transaction path.
As such, system designers will try to avoid bottlenecks and direct effort towards locating and tuning existing bottlenecks. Some examples of possible engineering bottlenecks are: a processor, a communication link, disk IO, etc.
Tracking down bottlenecks (sometimes known as "hot spots" - sections of the code that execute most frequently - i.e. have the highest execution count) is called performance analysis. Reduction is usually achieved with the help of specialized tools, known as performance analyzers or profilers. The objective being to make those particular sections of code perform as fast as possible to improve overall algorithmic efficiency.

Introduction

Gprof

How ?

- Compile and link with `-p` and `-g`
- You can use `GMON_OUT_PREFIX` for MPI code
- To summarize the information: `gprof -s a.out GMON_OUT_PREFIX.*`

Example benchmark bt from NAS (9 tasks)

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 26.21      9.10     9.10 67697001     0.00     0.00  binvcrhs_
 17.93     15.32     6.22      603     0.01     0.02  y_solve_cell_
 11.79     19.41     4.09      603     0.01     0.01  z_solve_cell_
 11.60     23.43     4.03 67697001     0.00     0.00  matmul_sub_
  9.86     26.85     3.42      202     0.02     0.02  compute_rhs_
  7.78     29.55     2.70      603     0.00     0.01  x_solve_cell_
  3.20     30.66     1.11 67697001     0.00     0.00  matvec_sub_
```

Vtune Amplifier XE

- Introduction
  - Bottelneck
  - Gprof
- Vtune Amplifier XE
  - Introduction
  - The Software Optimization Process
  - Vtune: GUI
  - Sum up
  - Labs: Demo

Vtune Amplifier XE

Advanced level

Vtune Amplifier XE

Find performance bottlenecks with advanced profiling technologies:

- Event-Based, System-Wide Sampling with little impact on program execution (typically $< 1\%$).
- Call Graph Profiling offers a pictorial view of program flow to help you quickly identify critical functions.

Atos

Vtune Amplifier XE

The Software Optimization Process

Identify Hotspots

- Identify the Hotspots
- Determine Efficency
- Identify Architectureal Reason for Inefficency

Vtune Amplifier XE

Optimize issue

Three questions

- **Why?** Why you should be concerned about this potential problem
- **How ?** Which profile and metric to use inside Vtune.
- **What now ?** Try to give suggestions to try some optimizations

Vtune Amplifier XE

Identify the Hotspots

What ?

Hotspots are where your application spends the most time ;-)

Why ?

You have to look where you lost a lot of your time

How ?

The good event is

$$CPU\_CLK\_UNHALTED.THREAD \tag{1}$$

This counter measures unhalted clockticks on per thread basis. If you use Hyperthreading this event will count 2 ticks for each tick of the CPU's clock.

Vtune

amplxe-cl -collect general-exploration – ./a.out

Vtune Amplifier XE

Determine Efficency of the hotspot

Three ways

- % Pipeline Slots Retired/Cycle
- Changes in CPI (Cycles per Instruction)
- Code examination

Vtune Amplifier XE

% Retired Pipeline Slots/Cycle

Why

This help you to understand how efficiently your application is using the processors

How ?

$$\frac{UOPS\_RETIRED.RETIRE\_SLOTS}{CPU\_CLK\_UNHALTED.THREAD}$$

What Now, for a given hotspot ?

- If $> 90\%$ retiring (0.9 or higher) is good. Go to efficiency method 3 (code examination)
- Between 50 and 90% for client apps investigating stall reduction
- Less than 60% for server apps consider stall reduction

Atos

Vtune Amplifier XE

Efficiency: Changes in Cycles per Instruction: CPI

Why ?

A measure of efficiency that can be use to campare two runs

How ?

General exploration profile (snb-general-exploration)

$$\frac{CPU\_CLK\_UNHALTED.THREAD}{INST\_RETIRED.ANY}$$

What now ?

- CPI is a ratio, if the code size changes for a binary, CPI will change. In general, if CPI reduces as a result of optimizations, that is good, and if it increases, that is bad.
- Optimized code may actually lower the CPI, and increase stall % but it will increase the performance. CPI is just a general efficiency metric the real measure of efficiency is work taking less time.

Vtune Amplifier XE

Efficiency Method 3: Code Examination

Why ?

The two first methods look how long it takes instruction to execute. The other type of inefficiency is to execute too many instructions

How?

With VTune capability to mixt source and disassembly vith viewer

What now ?

This method involves looking at the disassembly to make sure the most efficient instruction streams are generated. This can be complex and can require an expert knowledge of the Intel instruction set and compiler technology.

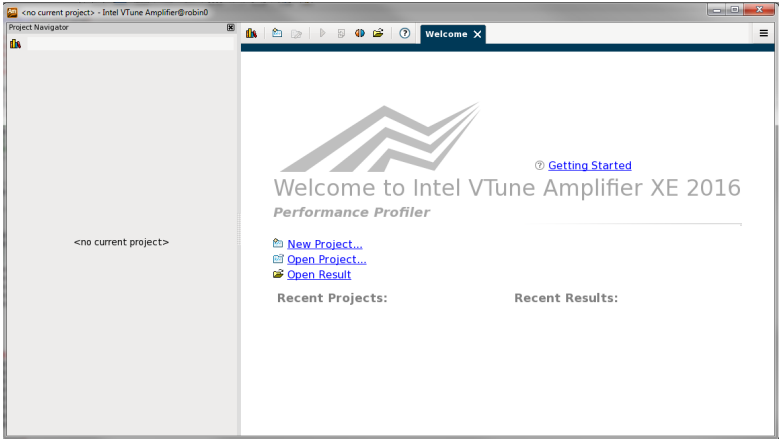
Atos

Vtune Amplifier XE

First step

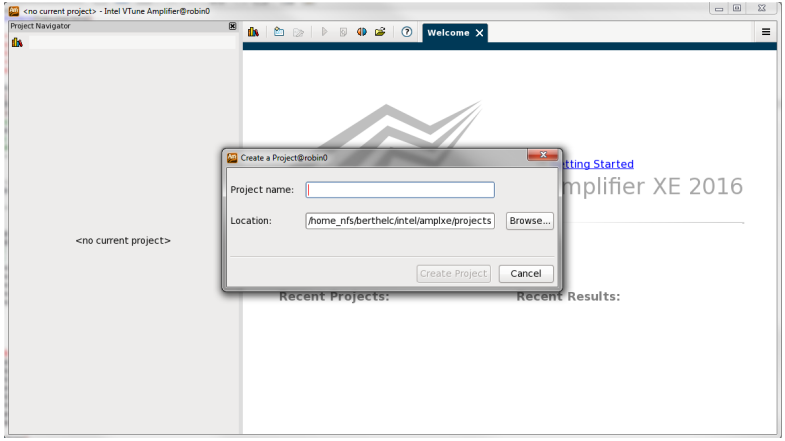
Load Env

- source . /opt/intel/parallel_studio_xe_YYYY.XX.YY/psxevars.sh
- Run amplxe-gui



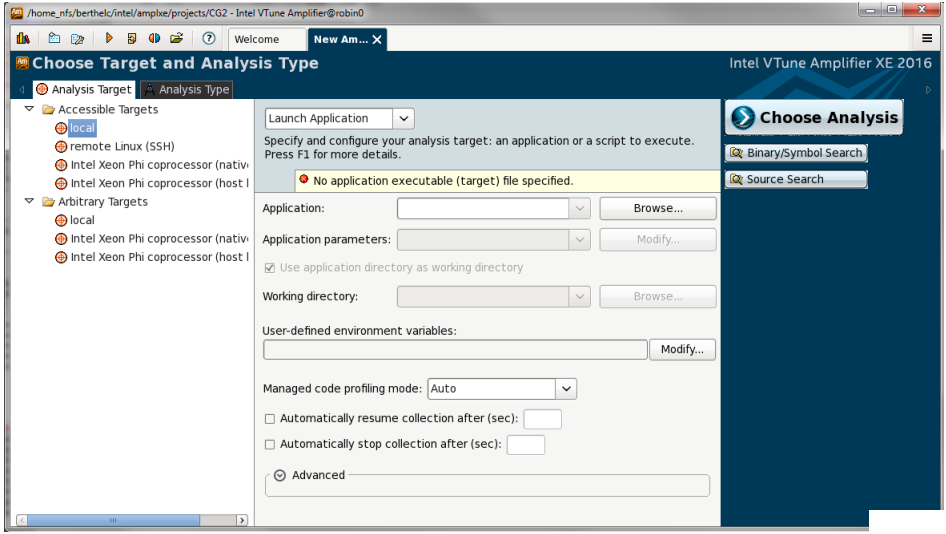
Vtune Amplifier XE

New project



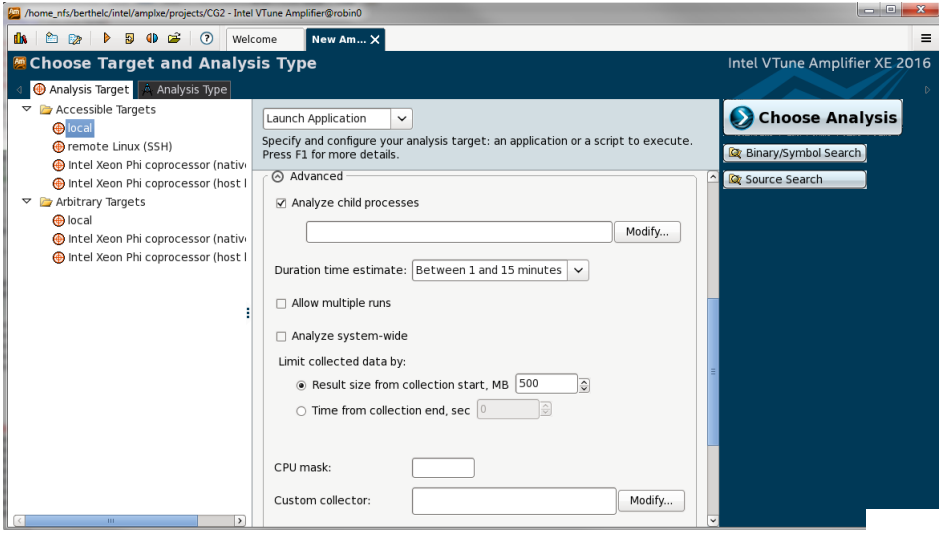
Vtune Amplifier XE

Select Target and options



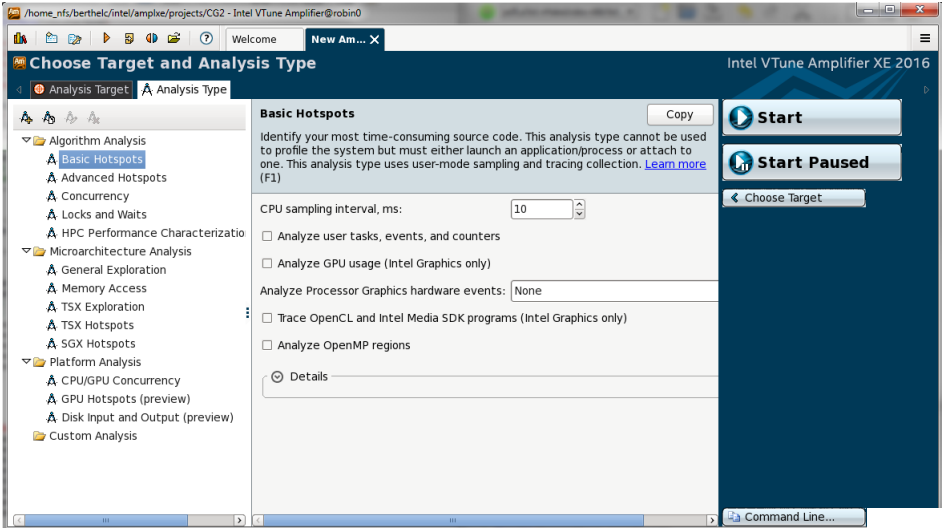
Vtune Amplifier XE

Select Target and options (advanced)



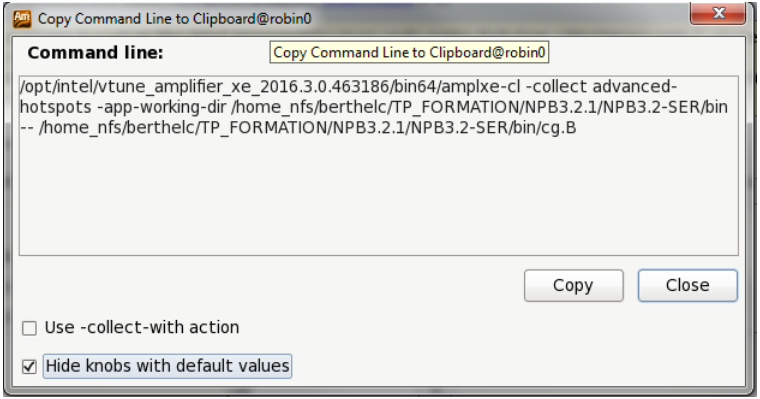
Vtune Amplifier XE

Select a new analysis



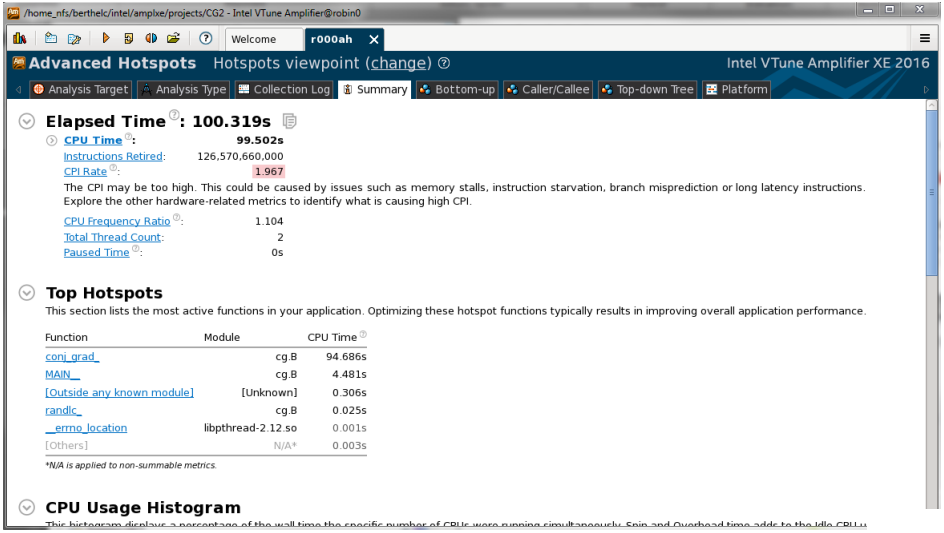
Vtune Amplifier XE

Command line to use inside batch



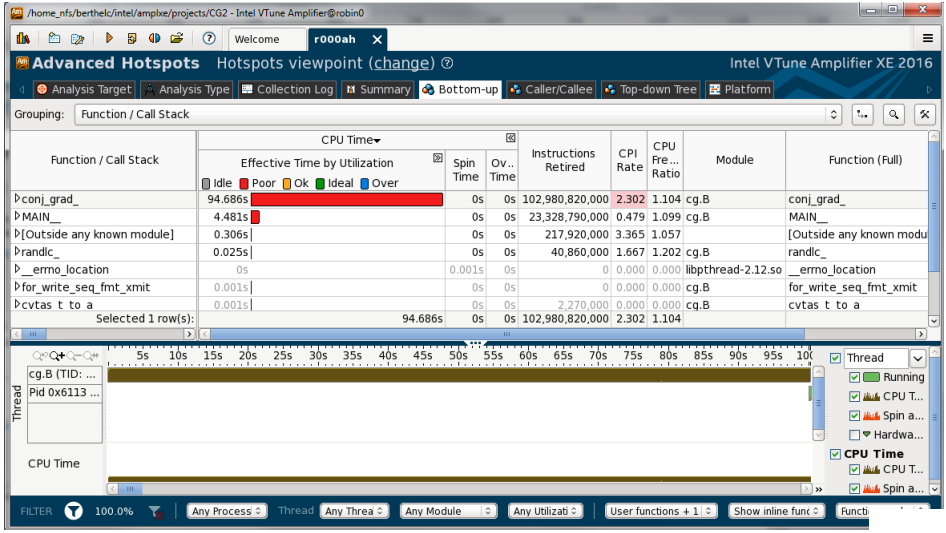
Vtune Amplifier XE

First windows after the run



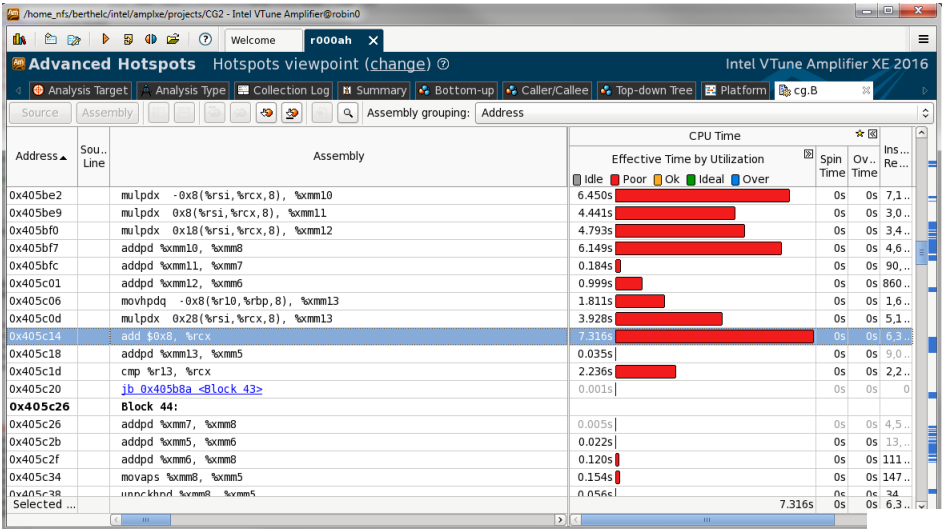
Vtune Amplifier XE

Hotspots



Vtune Amplifier XE

Low level: ASM view



Vtune Amplifier XE

Vtune and MPI

Introduction

You can use vtune with **Intel** MPI. It does not work with all MPI. For other application see paper Analyzing MPI programs with Intel VTune Amplifier XE and Intel Inspector XE tools

How

```
mpirun -n <N> -gtools "<abbr>-cl -r my_result -collect <analysis type>:MPIRANK"  my_app [my_app_ options]
```

The list of analysis types available can be viewed using `amplxe-cl -help collect`. The most simple to start with vtune is to use `hotspot` as analysis.

Vtune Amplifier XE

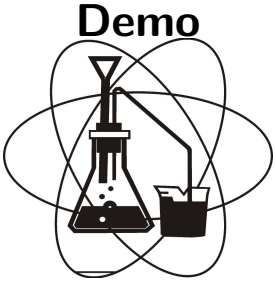
Positives points

- Vtune : easy to use (first level)
- First level of profiling you don't have to know information about processor, you have to believe the tool
- Vtune works with MPI and with slurm

Difficulties points

- To extract all informations have to understand μ-arch
- To extract all informations have to known some information about ratio or build you own.

Vtune Amplifier XE



Vtune Amplifier XE

Labs: Hotspots

Use module to set your env

- load parallel stutio XE : source /opt/intel/parallel_studio_xe_YYY.XX.ZZZ/psxevars.sh
- Set export VISUAL=gedit
- Extract /opt/intel/parallel_studio_xe_YYYY.XX.ZZZ/vtune_amplifier_xe_YYYY/samples/en/C++/tachyon_vtune_amp_xe.tg
- Compile : make

Labs: Hotspots

First Run

- Run amplxe-gui
- New Project
- New Analysis (Hotspot)
- Fin hotspot
  - Create New Project
  - load binary :tachyon_find_hotspots
  - parameter data/balls.dat
  - select hotspot
  - Run application

Labs: Hotspots

Code modification

- Edit file
- Modification of memory access
- Compile (make)
- New Analysis (Hotspot)
- Run application

Vtune Amplifier XE

Labs: Hotspots

Compare

- Load 2 files
- Compare results

Vtune Amplifier XE

Vtune and MPI

- Go on TP_HPCToolkit/NPB3.2.1/NPB3.2-MPI
- Compile `make CG CLASS=B NPROCS=16`
- Run code with vtune on rank 0 :
  `mpirun -gtool "amplxe-cl -collect hpc-performance -r result:0" -n 16 ./cg.B.16`
- Load result inside Vtune GUI

Vtune Amplifier XE

Vtune and OpenMP

- Go on TP_HPCToolkit/NPB3.2.1/NPB3.2-OMP
- Compile `make CG CLASS=B`
- Run code with numact all on node 0, and CPU on node 0 / meme en node 1l
- Load result inside GUI


BE CONTINUED

COPYRIGHT NOTICE