Performance analysis tools: Intel® Application Performance Snapshot and ITAC

Dr. Luigi Iapichino
luigi.iapichino@lrz.de
About the presenter

Dr. Luigi Iapichino
Scientific Computing Expert,
Leibniz Supercomputing Centre

- Member of the PRACE High-Level Support Team

- Team leader of the Application Lab for Astrophysics (LRZ AstroLab)

- Expert in computational astrophysics and simulations

Email: luigi.iapichino@lrz.de. WWW: http://iapichino.userweb.mwn.de
Outline until 12:00

- Overview on code profiling through tools.
- Intel® Application Performance Snapshot (APS).
- Intel® Trace Analyzer and Collector (ITAC).
- Live demo and short break.
- Intel® VTune™ Amplifier XE.
- Intel® Advisor.
- Live demo.
Optimizing Performance On Parallel Hardware

Ignore if you are not targeting clusters.

Cluster Scalable?

Tune MPI

Effective threading?

Vectorize

Memory Bandwidth Sensitive?

Optimize Bandwidth

Thread

Figure: courtesy Intel
Which tool do I use in my project? A roadmap to optimisation (and to the next hours)

We will focus on tools developed by Intel, available to users of the LRZ systems.

Code modernisation is an iterative process.

Where to start from?

How to assess easily any potential in performance tuning?

It is useful to get an early overview.
**Application Performance Snapshot (APS)**

- Quick and easy performance overview
  - Does the code need performance tuning?

- MPI & non-MPI applications:
  - Distributed MPI with or without threading
  - Shared memory applications

- Suggestion on the next tool for in-depth analysis.

- APS produces a simple text/HTML output, to be opened into an editor/browser.
### APS output in detail

#### Setup and system

#### Underlined definitions with explanations

#### Performance bottlenecks in red

---

**Application Performance Snapshot**

- **Elapsed Time**: 497.95s
- **CPI**: 2.00
- **MPI Time**: 53.85% of Elapsed Time (268.1ns)
- **MPI Imbalance**: 38.71% of Elapsed Time
- **I/O Bound**: 0.28% (AVG 45.39, PEAK 5.07)

### Performance summary and recommendations

**Your application is MPI bound.**

This may be caused by high busy wait time inside the library (imbalance), non-optimal communication schema or MPI library settings. Use **MPI profiling tools** like **Intel® Trace Analyzer and Collector** to explore performance bottlenecks.

<table>
<thead>
<tr>
<th></th>
<th>Duration</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI Time</td>
<td>53.85%</td>
<td>&lt;10%</td>
</tr>
<tr>
<td>OpenMP Imbalance</td>
<td>0.90%</td>
<td>&lt;10%</td>
</tr>
<tr>
<td>Back-End Stalls</td>
<td>19.76%</td>
<td>&lt;20%</td>
</tr>
<tr>
<td>SIMD Instr. per Cycle</td>
<td>0.00%</td>
<td>&lt;1</td>
</tr>
<tr>
<td>I/O Bound</td>
<td>0.28%</td>
<td>&lt;10%</td>
</tr>
</tbody>
</table>

### Main properties for characterising the HPC performance

- **Back-End Stalls**
  - L2 Hit Bound: 3.56% of cycles
  - L2 Miss Bound: 11.67% of cycles

- **SIMD Instr. per Cycle**
  - 0.00%

- **FP Instruction Mix**
  - % of Packed SIMD Instr.: 28.05%
  - % of Scalar SIMD Instr.: 71.95%
APS workflow

Load Amplifier module; source environment

Run application:
[mpirun <mpi options>] aps <application and args>

Generate report on result folder
aps-report <result folder>

Generate CL reports with detailed MPI statistics
we will not discuss this today
First step (most of the times): MPI layer of the parallelisation

- Cluster Scalable?
  - Yes: Tune MPI
  - No: Effective threading?
    - Yes: Vectorize
    - No: Thread

- Memory Bandwidth Sensitive?
  - Yes: Optimize Bandwidth
  - No: Intel® Trace Analyzer & Collector (ITAC) or Intel® MPI Tuner

Profiling tools

- Intel® VTune™ Amplifier
- Intel® Advisor
- Intel® VTune™ Amplifier
Intel® Trace Analyzer and Collector (ITAC)

- Graphical tool for understanding MPI application behaviour.
- Quickly finding bottlenecks (serialisations, load balancing, communication hotspots).
- Achieving high performance for parallel cluster applications running on Intel architectures.
- Improve weak and strong scaling.

In the rest of this first tutorial we will go through the main functionalities of this tool.
Collecting the MPI trace

One of ITAC’s strengths is the event-based tracing, which allows to:

➢ predict the MPI program behaviour;
➢ record the timing sequence of program states;
➢ collect information (times, order) about the exchange of messages.

Among the multiple methods for trace collection, one of the most useful is just to run the code giving the `-trace` option to `mpirun`.

This will generate, among others, a `.stf` trace file.

Beware of the trace size!
The summary page reminds APS

Resources and setup

Time spent in different code parts

Where to go from here

Top MPI functions

To optimize node-level performance use:
Intel® VTune™ Amplifier for:
- algorithmic level tuning with hpc-performance and threading efficiency analysis;
- microarchitecture level tuning with general exploration and bandwidth analysis;
Intel® Advisor for:
- vectorization optimization and thread prototyping.

Use the following command lines to run these tools for the most CPU-bound rank:
Intel® VTune™ Amplifier:
mpirun -gtool "amplxe-cl -collect hpc-performance -r result:6" -n 32 ./P-Gadget3/P-Gadget3 ipoC-test.par 2
Intel® Advisor:
mpirun -gtool "advice-cl -collect survey:6" -n 32 ./P-Gadget3/P-Gadget3 ipoC-test.par 2

Show Summary Page when opening a tracefile
Trace analysis

Charts → Event timeline

- Toolbar buttons

- Trace map: MPI function activity over time (zoomable)

- Event timeline: MPI in red, application in blue.

- Black lines: send/receive operations. Blue lines: collective operations
Trace analysis – Ungrouping MPI functions

Which MPI functions are called in the application?

Right-click on MPI in the Event Timeline and select Ungroup Group MPI
Measuring MPI times

Charts → Function profile
right click → Ungroup MPI

Application time: non-MPI run time.
Times are accumulated over all ranks.
Measuring MPI times - definitions

ITAC provides accumulated timings for all MPI routines $T_{mpi,acc}$ and the remaining computation $T_{comp,acc}$ (named: Group Application).

For many analyses, average (~ wallclock) times are often needed:

$$T_{comp}[p] = T_{comp,acc} / p$$
$$T_{mpi}[p] = T_{mpi,acc} / p$$

The averages can be directly shown by ITAC using the Function Profile Settings and checking $T_{self} / process$. 
According to the conventions just introduced, the wall clock run time is $T[p]$: 

$$T[p] = T_{\text{comp}[p]} + T_{\text{mpi}[p]}$$

Speedup and Efficiency can now be calculated for the compute and total time separately:

$$S_{\text{comp}[p]} = T_{\text{comp}[1]} / T_{\text{comp}[p]} = T[1] / T_{\text{comp}[p]}$$


$$E_{\text{comp}[p]} = S_{\text{comp}[p]} / p$$

Load balance

Function profile → Load balance

Useful to detect serialisations and inefficient parallelism.

Among the MPI and Application parts, what is more important (or more unbalanced)?

Ungrouping
Message profile

Charts → Message profile

It can be useful for finding patterns in the communication between MPI processes.
What is to be found in the MPI timing provided by ITAC?

There are two main contributions:

➢ Load imbalances and dependencies in the algorithm;
➢ Network transfer times due to interconnect limitations. They depend on latency L and bandwidth BW. For a data volume V:

\[ T_{\text{trans}}[V] = L + \frac{1}{BW} \times V \]
Ideal network simulator

One of ITAC’s functionalities to study this point is the ideal network simulator.

This starts from a real trace file as basis, and simulates as the transfer times are set to zero (meaning $L = 0$ and $BW = \infty$ for the simple model).

Compute times (non MPI) will stay the same.

Within the simulator, the MPI time will shrink but not vanish. The wait times and unbalance will remain exposed.

Advanced → idealisation to create an idealised trace.
Timeline comparison: real vs. idealised

View → compare
Imbalance diagram

Advanced → Imbalance diagram

(test on 8 KNL nodes and 4 MPI tasks per node)
Imbalance diagram – breakdown mode

Advanced → Imbalance diagram → Breakdown mode

Relative weight of the single MPI functions

Data volume of the call
Summary

Intel® Application Performance Snapshot: first overview of an application, planning of the next optimisation steps.


Intel® Trace Analyzer and Collector: analysis and modernisation of the MPI layer of the code.


Intel tutorial on ITAC: https://software.intel.com/en-us/node/561850

Acknowledgements

Thanks to Heinrich Bockhorst and Fabio Baruffa (Intel) for providing me with training material, as starting point for this tutorial.