Technical Report:

Overview of different profiling tools

Abstract: This technical report will try to give an overview of different profiling tools, both open-source and commercial. Some of them are currently available in FinisTerraeII supercomputer, so a brief usage of this tools will be shown too.

This report will give a point of reference to start to use this tools, and at the same time, a little information about when it is recommended the use of each one, languages and parallel standards supported and the different results we can get using them.

Document Id.:	CESGA-2017-003
Date:	**December 15th, 2017**
Task:	
Responsible:	**Diego Mairena Díaz and José Carlos Mouriño**
Status:	**Final**

Overview of different profiling tools

Galicia Supercomputing Centre

Diego Mairena Díaz
José Carlos Mouriño Gallego

Index

List of Figures

List of Tables

1 Introduction

Currently, there are a large number of tools which allow us to obtain different information about the behaviour of an application, and at the same time, get some clues to improve its performance. In this technical report, we will try to give an overview of different profiling tools, some of them already available in FinisTerraeII supercomputer.

This toolsets will allow us to detect many memory problems or threading bugs, or check where an application spends more time and help us to get more performance, both in sequential and parallel applications.

We have based this study in some tools used by POP (Performance Optimization and Productivity), a centre of excellence in computing applications, tools recommended by Prace's Best Practice Guide – Haswell/Broadwell published this year and other interesting profiling tools, both open-source and commercial tools.

This report try to give the general vision for the features of each application and have a quick access to their usage and specific documentation. The advanced usage of a tool will depend on each case and the specific information we want to obtain.

The structure of this report is as follows: In next Section, we will analyse the open-source tools used by POP, and then, some commercial applications are presented in Section 3, like Intel profiling tools suite. In Section 4, we will see other useful open-source tools. Finally, some conclusions and guidelines are given in Section 5.

2 POP Open Source Tools

In this section, we analyse different open source tools developed by POP partners and collaborators, like BSC, JSC, INRIA and the Universities of Dresden and Oregon. All this tools are used by POP to provide performance optimization and productivity services for different final users.

2.1 JSC performance tools

The Jülich Supercomputing Centre develops different tools for scalable performance analysis of large-scale parallel applications that are deployed on most of the largest HPC systems. Some of this tools are Scalasca, Score-p, Cube or Extra-p, which we analyse below.

2.1.1 Scalasca

Scalasca is a software tool that supports the performance optimization of parallel programs by measuring and analysing their runtime behaviour. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronization – and offers guidance in exploring their causes.

It is specifically designed for large-scale systems, and it provides in-depth studies of concurrent behaviour via event traces. Some of its features are:

- ➢ Localization of wait states and their root causes on large processor configurations
- ➢ Identification of the critical path
- ➢ Support C, C++ and Fortran applications
- ➢ Support for MPI, OpenMP, Pthreads, and hybrid MPI+OpenMP/Pthreads
- ➢ Support different supported platforms, including Linux-based clusters and Intel Xeon Phi

To generate measurements which can be used as input for the Scalasca Trace Tools, user applications first need to be instrumented. All the necessary instrumentation of user routines, OpenMP constructs and MPI functions should be handled by the Score-P instrumenter (see section 2.1.2). Once Scalasca measures are done, to visualize and examine this results, Cube browser is used (see section 2.1.3).

As brief summary, this is the Scalasca measurement and analysis workflow:

- ➢ Run Score-P instrumented target application to produce runtime summary

- Generate targeted event traces of critical code regions for closer investigation of concurrent behaviour
- Examine trace analysis results using Cube graphical user interface

Scalasca documentation is available in:

User Guide:

https://apps.fz-juelich.de/scalasca/releases/scalasca/2.3/docs/UserGuide.pdf

Web site documentation:

http://www.scalasca.org/software/scalasca-2.x/documentation.html

2.1.2 Score-p

The Score-P measurement infrastructure is a highly scalable and easy-to-use tool suite for profiling, event tracing, and online analysis of HPC applications. It offers the user a maximum of convenience by supporting a number of analysis tools. Score-p contains the code instrumentation functionality supporting various methods and it performs the run-time data collection in the parallel environment.

Score-p supports C, C++ and Fortran applications, and MPI, OpenMP, Pthreads, and hybrid MPI+OpenMP/Pthreads.

Currently, it works with Periscope, Scalasca, Vampir, and Tau (some of this tools are analysed in this document) and it is open for other tools. Score-P comes together with the new Open Trace Format Version 2, the Cube4 profiling format and the Opari2 instrumenter.

As a quick approximation, Score-p take care of all necessary instrumentation of user and MPI functions. This is done by using *scorep* command, that needs to be prefixed to all the compile and link commands usually employed to build de application. Thus, an application executable that is normally generated as:

```
mpifort app.f90 -o app
```

Will now be build by:

```
scorep mpifort app.f90 -o app
```

For deeper usage of Score-p, documentation is available in:

User manual: https://silc.zih.tu-dresden.de/scorep-current.pdf

Cheatsheet: https://www.vampir.eu/public/files/pdf/spcheatsheet_a4.pdf

2.1.3 Cube

Cube, which is used as performance report explorer for Scalasca and Score-P, is a generic tool for displaying a multi-dimensional performance space consisting of the dimensions performance metric, call path, and system resource. Each dimension can be represented as a tree, where non-leaf nodes of the tree can be collapsed or expanded to achieve the desired level of granularity. In addition, Cube can display multi-dimensional Cartesian process topologies.

Cube documentation is available in:

User Guide:

https://apps.fz-juelich.de/scalasca/releases/cube/4.3/docs/CubeGuide.pdf

Web site documentation:

http://www.scalasca.org/software/cube-4.x/documentation.html

2.1.4 Extra-p

Extra-P is an automatic performance-modeling tool that supports the user in the identification of scalability bugs.

Extra-P uses measurements of various performance metrics at different processor configurations as input to represent the performance of code regions (including their calling context) as a function of the number of processes. All it takes to search for scalability issues even in full-blown codes, and run a manageable number of small-scale performance experiments. Launching Extra-P, it is possible to compare the asymptotic or extrapolated performance of the worst instances to the expectations. Besides the number of processes, it is also possible to consider other parameters such as the input problem size.

Extra-P generates not only a list of potential scalability bugs but also human-readable models for all performance metrics available such as floating-point operations or bytes sent by MPI calls that can be further analysed and compared to identify the root causes of scalability issues.

Extra-p documentation is available in:

Tutorial slides (theory):

https://apps.fz-juelich.de/scalasca/releases/extra-p/slides/InsightfulAutomaticPerformanceModelingTutorialPartI.pdf

Tutorial slides (practical usage):

https://apps.fz-juelich.de/scalasca/releases/extra-p/slides/InsightfulAutomaticPerformanceModelingTutorialPartII.pdf

2.1.5 TAU

TAU is a portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, UPC, Java or Python. It is capable of gathering performance information through instrumentation of functions, methods, basic blocks, and statements as well as event-based sampling. All C++ language features are supported including templates and namespaces.

The API also provides selection of profiling groups for organizing and controlling instrumentation. The instrumentation can be inserted in the source code using an automatic instrumentor tool based on the Program Database Toolkit (PDT), dynamically using DyninstAPI, at runtime in the Java Virtual Machine, or manually using the instrumentation API.

TAU's profile visualization tool, ***paraprof***, provides graphical displays of all the performance analysis results, in aggregate and single node/context/thread forms. It is possible to identify sources of performance bottlenecks in the application using the graphical interface. In addition, TAU can generate event traces that can be displayed with the Vampir (see section 3.3), Paraver (see section 2.3.2) or JumpShot trace visualization tools.

TAU documentation is available in:

User Guide: https://www.cs.uoregon.edu/research/tau/docs/newguide/bk01.html

Web site documentation: https://www.cs.uoregon.edu/research/tau/docs.php

2.2 INRIA performance tools

Inria, the French National Institute for computer science and applied mathematics, provides a tool called SimGrid, to profile and understand the behaviour of different systems and applications.

2.2.1 SimGrid

SimGrid is a scientific instrument to study the behaviour of large-scale distributed systems such as Grids, Clouds, HPC or P2P systems. It can be used to evaluate heuristics, prototype applications or even assess legacy MPI applications. It provides ready to use models and API to simulate many different distributed systems: clusters, wide-area and local-area networks, peers over DSL connexions, data centers, etc.

SimGrid is a tool which has different usages:

- Grid Simulator: Accurate yet fast simulation models
- P2P Simulator: Highly scalable simulations (several millions of nodes on a single machine)

- MPI Simulator: Realistically simulates unmodified MPI programs
- Cloud Simulator: SimGrid provides an libvirt-like interface for a Cloud simulation
- SimGrid in other domains: It was used too as a volunteer computing simulator, fog computing simulator or MapReduce simulator.

SimGrid produces traces that can easily be visualized and post-processed with state-of-the-art tools such as Pajé and Viva.

SimGrid documentation is available in:

User Manual: http://simgrid.gforge.inria.fr/documentation.php

Tutorials: http://simgrid.gforge.inria.fr/tutorials.php

2.3 BSC performance tools

The performance analysis tools developed at BSC provide a detailed analysis that allows understanding of an application's behaviour as well as identifying performance critical issues. Paraver is a trace-based performance analyser to explore and extract information, using the obtained results of Extrae. In addition, Dimemas allows a fast evaluation of what-if scenarios for MPI applications.

2.3.1 Extrae

Extrae is a dynamic instrumentation package to trace programs compiled and run with the shared memory model (like OpenMP and pthreads), the message passing (MPI) programming model or both programming models.

Extrae generates trace files that can be later visualized with Paraver, and this combined use offers an enormous analysis potential, both qualitative and quantitative. With these tools the actual performance bottlenecks of parallel applications can be identified. The microscopic view of the program behaviour that this tools provide is very useful to optimize the parallel program performance.

Extrae documentation is available in:

User manual:

https://tools.bsc.es/sites/default/files/documentation/pdf/extrae-3.5.2-user-guide.pdf

2.3.2 Paraver

Paraver is a flexible parallel program visualization and analysis tool based on an easy-to-use wxWidgets GUI. Paraver was developed responding to the need of having a qualitative global perception of the application behaviour by visual inspection and then to be able to focus on the detailed quantitative analysis of the problems. Paraver provides a large amount of information useful to decide the points on which to invest the programming effort to optimize an application.

Some Paraver features are:

- Detailed quantitative analysis of program performance
- Concurrent comparative analysis of several traces
- Fast analysis of very large traces
- Support for mixed message passing and shared memory (network of SMPs)
- Customizable semantics of the visualized information

Paraver traces supports the same programming interfaces as Extrae: MPI, OpenMP, pthreads, OmpSs and CUDA.

Paraver documentation is available in:

Tutorial guidelines: https://tools.bsc.es/tutorial_guidelines

2.3.3 Dimemas

Dimemas is a performance analysis tool for message-passing programs. It enables the user to develop and tune parallel applications on a workstation, while providing an accurate prediction of their performance on the parallel target machine. The Dimemas simulator reconstructs the time behaviour of a parallel application on a machine modelled by a set of performance parameters. Thus, performance experiments can be done easily. The supported target architecture classes include networks of workstations, single and clustered SMPs, distributed memory parallel computers, and even heterogeneous systems.

Dimemas generates trace files that are suitable for Paraver enabling the user to conveniently examine any performance problems indicated by a simulator run.

The analysis module performs critical path analysis reporting the total CPU usage of different code blocks, as well as their importance for the program execution time. Based on a statistical evaluation of synthetically perturbed traces and architectural parameters, the importance of different performance parameters and the benefits of particular code optimizations can be analysed.

Dimemas main goals can be summarized as:

- Development and tuning

- Benchmarking, training and features
- Predict behaviour of target architecture
- Forecast effects of code optimization
- Integrated with visualization tool
- Extensive application characterization

Dimemas documentation is available in:

Tutorial manual:

https://tools.bsc.es/sites/default/files/documentation/introduction_dimemas.pdf

Tutorial guideline: https://tools.bsc.es/tutorial_guidelines

3 Commercial Tools

In this section, we will analyse some Intel tools, all of them available in FinisTerraeII supercomputer, as well as some Allinea tools. Also, we will see a brief overview of Vampir software, POP tool which only has a time-limited demo version available for free.

3.1 Intel

Intel provides a set of tools allowing minimizing development, tuning and testing time and effort. Intel parallel studio comes whit different tools like Intel-XE Advisor, Intel-XE Inspector, Intel VTune-Amplifier or Intel Trace Analyser and Collector, which allow to optimize and understand different applications, including OpenMP and MPI ones.

3.1.1 Intel XE-Advisor

Intel Advisor provides two tools to help ensure your Fortran, C and C++ applications realize full performance potential on modern processors, such as Intel Xeon Phi processors:

- Vectorization Advisor is a vectorization optimization tool that lets you identify loops that will benefit most from vectorization, identify what is blocking effective vectorization, forecast the benefit of alternative data reorganizations, and increase the confidence that vectorization is safe.
- Threading Advisor is a threading design and prototyping tool that lets you analyse, design, tune, and check threading design options without disrupting your normal development.

On modern processors, it is crucial to both vectorize – with Intel Advanced Vector Extensions (Intel AVX) or single instruction, multiple data (SIMD) instructions – and thread software to realize the full performance potential of the processor.

Some of Intel XE-Advisor’s features are:

- Check loop-Carried dependencies or memory accesses patterns for marked loops
- Cache-aware roofline analysis
- Analyse performance and scalability: compare alternative designs
- Check correctness

This tool provides a GUI to work with it, at the same time it can be used through command line. In FinisterraeII supercomputer, it is available loading the following modules:

```
intel/2016 advisor/2016.1
```

```
intel/2018 advisor/2018.1
```

And the GUI executable is “advixe-gui” (it must be run in a compute --x11 session).

Intel XE-Advisor tutorial for vectorization, threading and MPI, as well as all documentation, are available in:

https://software.intel.com/en-us/get-started-with-advisor

https://software.intel.com/en-us/articles/advisor-tutorials

Tutorial files are available in following FinisTerraeII paths:

/opt/cesga/intel/2018/advisor/samples/en/C++

/opt/cesga/intel/2018/advisor/samples/en/Fortran

3.1.2 Intel XE-Inspector

Intel Inspector is a dynamic memory and threading error checking tool for users developing serial and multithreaded applications on Windows and Linux operating systems.

It helps to find errors early, when they are less expensive to fix. Intel Inspector is an easy-to-use memory and threading error debugger for C, C++, and Fortran applications that run on Windows and Linux. No special compilers or builds are required, just use a normal debug or production build. It can be used through the graphical user interface or automate regression testing with the command line.

Some of the Intel-XE Inspector features are:

- Standalone GUI and command line operational environments
- Pre-set analysis configurations (with some configurable settings), as well as the ability to create custom analysis configurations to help to control analysis scope and cost
- Visibility into individual problems, problem occurrences, and call stack information, with problem prioritization and filtering by inclusion and exclusion to help focus on items that require special attention
- Interactive debugging capability, so it is easy to investigate problems more deeply during analysis
- A wealth of reported memory errors, including on-demand memory leak detection
- Data race, deadlock, lock hierarchy violation, and cross-thread stack access error detection, including error detection on the stack

In FinisterraeII supercomputer, it is available loading the following modules:

```
intel/2016 inspector/2016.1
intel/2018 inspector/2018.1
```

And GUI executable is inspxe-gui (it must be run in a compute --x11 session).

Intel XE-Inspector tutorial and documentation are available in:

https://software.intel.com/en-us/get-started-with-inspector-linux

https://software.intel.com/en-us/articles/inspector-tutorials

Tutorial files are available in the following FinisTerraeII paths:

/opt/cesga/intel/2018/inspector/samples/en/C++

/opt/cesga/intel/2018/inspector/samples/en/Fortran

3.1.3 Intel VTune-Amplifier

Intel VTune Amplifier can be used for analysis of local and remote target systems. Use this tool to analyse the algorithm choices, find serial and parallel code bottlenecks, understand where and how an application can benefit from available hardware resources, and speed up the execution.

Intel VTune Amplifier provides advanced profiling capabilities with a friendly analysis interface. And for media applications, it is also possible to get powerful tools to tune OpenCL and the GPU.

Some of the Intel VTune-Amplifier features are:

- Algorithm analysis (Basic hotspots, advanced hotspots, concurrency analysis, locks and waits analysis)
- Microarchitecture analysis (general exploration analysis, memory access)
- Platform analysis (system overview, CPU-GPU concurrency, GPU Hotspots)
- Compute-intensive applications analysis (HPC performance characterization, Analyse OpenMP regions, explore OpenMP and MPI efficiency metrics)

In FinisterraeII supercomputer, it is available loading the following modules:

```
intel/2016 vtune/2016.3
```

```
intel/2018 vtune/2018.1
```

The GUI executable is amplxe-gui (it must be run in a compute --x11 session).

Intel Vtune-Amplifier tutorial and documentation are available in:

https://software.intel.com/en-us/get-started-with-vtune-linux-os

https://software.intel.com/en-us/articles/intel-vtune-amplifier-tutorials

Tutorial files are available in the following FinisTerraeII paths:

/opt/cesga/intel/vtune_amplifier_xe/samples/en/C++

/opt/cesga/intel/vtune_amplifier_xe/samples/en/Fortran

3.1.4 Intel Trace Analyser and Collector

Intel Trace Analyser and Collector is a graphical tool for understanding MPI application behaviour, quickly finding bottlenecks, improving correctness, and achieving high performance for parallel cluster applications based on Intel architecture.

Intel Trace Collector enables user to collect statistics for different applications, while Intel Trace Analyser provides powerful capabilities for visualizing and analysing the collected data.

Some of its benefits are:

- Visualize and understand parallel application behaviour
- Evaluate profiling statistics and load balancing
- Analyse performance of subroutines or code blocks
- Learn about communication patterns, parameters, and performance data

- Identify communication hotspots
- Decrease time to solution and increase application efficiency

The usage of this tools consist on two steps. First at all, it is necessary to generate the traces of the application. There are two ways to do this:

- Compiling the application with "`-g -trace`" flags and running it directly
- Compiling the application in release mode and running it with itac module and this environment variable:

  ```
  export LD_PRELOAD=/opt/cesga/intel/itac/9.1.2.024/slib/libVT.so
  ```

Once traces are generated, it is possible to see them using the GUI command of Intel Trace Analyser `traceanalyzer` (it must be run in a compute --x11 session[1]).

This is an example of a script to submit in FinisTerraeII supercomputer and get the traces (.stf files) for hdf5 in release mode:

```
#!/bin/bash
#SBATCH -p cola-corta
#SBATCH -n 12
#SBATCH -t 00:05:00
module load intel/2016 impi/5.1 itac/9.1 hdf5/1.8.16
export
LD_PRELOAD=/opt/cesga/intel/itac/9.1.2.024/intel64/slib/libVT.so
srun h5perf -e 1M -B 16K -x 16K -X 16K -p 12 -P 12
```

This script will generate different files for each process, and the main file, h5perf.stf.

Then, just starting an X11 session and running the GUI command, it is possible to visualize the behaviour of this application:

```
$ compute --x11
$ module load intel/2016 impi/5.1 itac/9.1
$ traceanalyzer h5perf.stf
```

Figure 1 shows an overview of h5perf by process. In blue, the time spent in sequential parts, and in red, the time spent in parallel parts, and the communications between them.

At the same time, it also shows the total time in collective operations by process (box on the bottom right). It is possible to show the call tree and the call graph too.

1 Finisterrae connection with the ability to open remote graphical windows. In Linux with X11 forwarding and in Windows with a local X server.

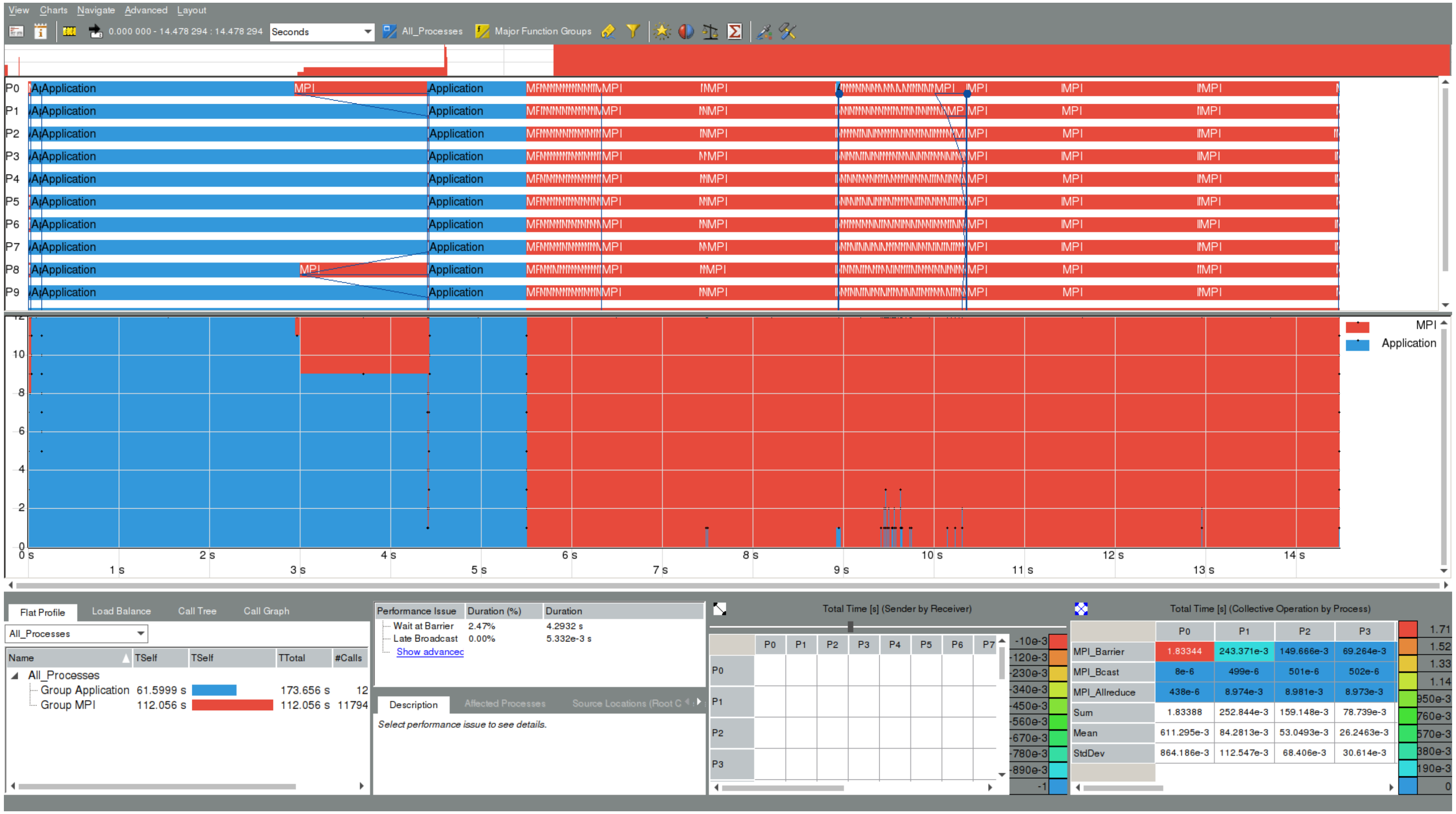
In FinisterraeII supercomputer, Intel Trace Analyser and Collector is available loading the following modules:

```
intel/2016 impi/5.1 itac/9.1
intel/2018 impi/2018 itac/2018.1
```

Intel Trace Analyser tutorial and documentation are available in:

https://software.intel.com/en-us/analyzing-mpi-apps-with-itac-and-vtune

https://software.intel.com/en-us/get-started-with-itac-for-linux



Name	TSelf	TSelf	TTotal	#Calls
All_Processes				
Group Application	61.5999 s	🟦	173.656 s	12
Group MPI	112.056 s	🟥	112.056 s	11794

Performance Issue	Duration (%)	Duration
Wait at Barrier	2.47%	4.2932 s
Late Broadcast	0.00%	5.332e-3 s
Show advancec		

Select performance issue to see details.

Total Time [s] (Collective Operation by Process)

	P0	P1	P2	P3
MPI_Barrier	1.83344	243.371e-3	149.666e-3	69.264e-3
MPI_Bcast	8e-6	499e-6	501e-6	502e-6
MPI_Allreduce	438e-6	8.974e-3	8.981e-3	8.973e-3
Sum	1.83388	252.844e-3	159.148e-3	78.739e-3
Mean	611.295e-3	84.2813e-3	53.0493e-3	26.2463e-3
StdDev	864.186e-3	112.547e-3	68.406e-3	30.614e-3

Figure 1: Overview of h5perf behaviour using Intel Trace Analyser and Collector

3.2 Allinea

Allinea Software, now part of Arm, provides a set of tools for build and running applications to maximum effect, and at the same time, to development and profile them. Some Allinea profiling tools are MAP and Performance Reports, both of them could be available soon in FinisTerraeII supercomputer.

3.2.1 Allinea Tool: MAP

MAP is a profiler for parallel, multithreaded or single threaded C, C++, Fortran and F90 codes. It provides in-depth analysis and bottleneck pinpointing to the source line. It's designed to be able to profile pthreads, OpenMP or MPI for parallel and threaded code. It profiles with no relinking, instrumentation or code changes required.

MAP exposes a wide set of performance problems and bottlenecks by measuring:

- Computation - with self and child and call tree representations over time
- Thread activity - to identify over-subscribed cores and sleeping threads that waste available CPU time for OpenMP and pthreads
- Instruction types (for x86_64) - to show use of vector-units or other performance extensions
- Synchronization, communication and workload imbalance for MPI or multi-process usage
- I/O performance and time spent in I/O - to identify bottlenecks in shared or local file systems

MAP documentation and tutorials are available in:

User Guide: https://developer.arm.com/docs/101136/latest/map

Video demos and tutorials:

https://developer.arm.com/products/software-development-tools/hpc/arm-forge/arm-map/video-demos-and-tutorials-for-arm-map

3.2.2 Allinea Tool: Performance Reports

Performance Reports are a non-intrusive performance tool for High Performance Computing (HPC) and scientific software. They analyse the applications running on the system to seek out inefficiencies and pinpoint exactly where to focus optimization work.

A report measures overall information about the computation, communication and

I/O - and provides detail for each of these areas:

- The time spent in various categories of instruction: memory access, numeric operations, floating point operations
- I/O - time and the effective performance (transfer rate) of read and write operations to storage
- Memory - the mean and peak usage of memory per node
- Communication - MPI time and performance for collective and point-to-point operations
- Threads - the time spent in computation and synchronization, the physical core utilization and system load
- GPUs - the utilization and memory use of NVIDIA CUDA GPUs
- Energy - the energy pack add-on reports energy usage and peak power - for system, CPU and any NVIDIA GPUs

Performance Reports documentation is available in:

User Guide: https://developer.arm.com/docs/101137/latest/introduction

3.3 Vampir

Vampir provides an easy-to-use framework that enables developers to quickly display and analyse arbitrary program behaviour at any level of detail. The tool suite implements optimized event analysis algorithms and customizable displays that enable fast and interactive rendering of very complex performance monitoring data.

The combined handling and visualization of instrumented and sampled event traces generated by Score-P enables an outstanding performance analysis capability of highly-parallel applications.

Some of its features and functions are:

- Easy to use performance analysis framework for parallel programs
- Graphical data representation enables detailed understanding of dynamic processes on massively parallel systems
- In-depth event based analysis of parallel run-time behaviour and interprocess communication
- Identification of performance problems and bottlenecks
- Linux-based PCs and Clusters, SGI, IBM, SUN, NEC, HP, Apple, ...

Vampir documentation and a use case are available in:

https://www.vampir.eu/tutorial/manual

https://www.vampir.eu/tutorial/a_use_case

4 Other Open Source Tools

In this section, we will see an overview for some interesting open-source profiling tools, which can give us useful information about our application or the system where they are running.

4.1 Valgrind

Valgrind is an instrumentation framework for building dynamic analysis tools. There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile programs in detail.

The Valgrind distribution includes six production-quality tools:

- Memcheck: it detects memory-management problems, and is aimed primarily at C and C++ programs. When a program is run under Memcheck's supervision, all reads and writes of memory are checked, and calls to malloc/new/free/delete are intercepted.
- Cachegrind: Cachegrind is a cache profiler. It performs detailed simulation of the I1, D1 and L2 caches in a CPU and so can accurately pinpoint the sources of cache misses in the code.
- Callgrind: it is an extension to Cachegrind. It provides all the information that Cachegrind does, plus extra information about callgraphs.
- Massif: Massif is a heap profiler. It performs detailed heap profiling by taking regular snapshots of a program's heap.
- Helgrind: Helgrind is a thread debugger which finds data races in multithreaded programs.
- DRD: DRD is a tool for detecting errors in multithreaded C and C++ programs.

Valgrind is available in FinisTerraII as a system tool or loading:

```
intel/2016 with impi/5,1 or impi/2017 and valgrind/3.11.0
```

Valgrind documentation is available in:

Manual: http://valgrind.org/docs/manual/manual.html

Quick start: http://valgrind.org/docs/manual/quick-start.html

4.2 ompP

ompP is a profiling tool for OpenMP applications. ompP's profiling report becomes available immediately after program termination in a human-readable ASCII format. ompP supports the measurement of hardware performance counters using PAPI and it supports productivity features such as overhead analysis and detection of common inefficiency situations.

To instrument an application and link it with ompP's monitoring library simply prefix any compile or link command with kinst-ompp. For example:

```
icc -openmp example.c -o myapp
```

Becomes:

```
kinst-ompp icc -openmp example.c -o myapp
```

Then, simply running the application, an ASCII file will be generated, where it is possible to see the different parallel regions, a callgraph or different overheads.

ompP documentation is available in:

Manual: http://www.ompp-tool.com/downloads/ompp-manual.pdf

4.3 Likwid

Likwid is a simple to use toolsuite of command line applications for performance oriented programmers. It works for Intel and AMD processors on the Linux operating system.

Some of its relevant features are:

- Print thread, cache and NUMA topology
- Configure and read out hardware performance counters on Intel and AMD processors
- Pin a ghreaded application (pthread, Intel and gcc OpenMP to dedicated processors)
- Micro benchmarking platform
- Wrapper to start MPI and Hybrid MPI/OpenMP applications (Supports Intel MPI, OpenMPI and MPICH)
- Sweep memory of NUMA domains and evict cachelines from the last level cache

Likwid documentation is available in:

https://github.com/rrze-likwid/likwid/wiki

4.4 gperftools

gperftools is a collection of a high-performance multi-threaded malloc() implementation, plus some performance analysis tools.

It provides a thread-caching malloc (at first faster than glibc malloc), which assigns each thread a thread-local cache. This way, small objects are moved from central data structures into a thread-local cache as needed, and periodic garbage collections are used to migrate memory back from a thread-local cache into the central data structures. Large objects are allocated directly from central heap using a page-level allocator.

This thread-caching malloc includes a heap checker and a heap profiler as well. The first one is used to detect memory leaks in C++ programs, and the second one to explore how C++ programs manage memory. gperftools also includes a cpu profiler.

All this analysis tools can be used simply linking the right library in the executable (`-ltcmalloc`) or in applications already compiled, using `LD_PRELOAD` flag. For each tool, it is necessary to export specific environment variables too.

This tool is available in FinisTerraeII supercomputer loading `gperftools/2.5` module.

Gperftools documentation is available in:

https://github.com/gperftools/gperftools/wiki

4.5 Darshan

Darshan is a scalable HPC I/O characterization tool. Darshan is designed to capture an accurate picture of application I/O behaviour, including properties such as patterns of access within files, with minimum overhead. It can be used to investigate and tune the I/O behaviour of complex HPC applications. In addition, Darshan's lightweight design makes it suitable for full time deployment for workload characterization of large systems.

Darshan only instruments MPI applications, however, it captures both MPI-IO and POSIX file access. It also captures limited information about HDF5 and PnetCDF access. Darshan provides a collection of tools for parsing and summarizing log files produced by it instrumentation too.

To instrument a compiled application, simply export the environment variable LD_PRELOAD with the right path to libdarshan.so library. In FinisterraeII, job should be run as follow:

```
srun -n 4 --export=LD_PRELOAD=/path-to/libdarshan.so mpi-app
```

This will generate a log file which can be analysed with the different Darshan utils.

Darshan is available in FinisterraeII loading the following modules:

```
intel/2016 with impi/5.1, impi/2017 or openmpi/1.10.2 and darshan/3.1.2
gcc/5.3.0 with impi/5.1 or openmpi/1.10.2 and darshan/3.1.2
```

Specific documentation for runtimes and utils is available in:

http://www.mcs.anl.gov/research/projects/darshan/documentation/

5 Conclusions

As we have seen, there are a great variety of tools to improve the performance of an application and for understanding its behaviour. Deeper usage of this tools requires some time, but the benefits obtained make up for the effort.

Depending on our interest, some tools are more appropriated than others, but combining them we can get a great increase of the performance and a deeper knowledge of a specific application or system.

As guideline, this could be an example of the optimization workflow for a generic application:

- Find all memory related problems, like memory leaks or illegal memory access, and fix them (Intel Inspector, Valgrind, gperftools, ...)
- Find and vectorize loops and compare the gain using different options (Intel Advisor)
- If the analysed application is a sequential one, it is possible to predict where are the best parts of the code to add OpenMP parallelism, and run a performance and scalability analysis.
- Find hotspots and try to reduce time in these areas (ompP, Intel Vtune, ...)
- Find and fix different threading errors, like data races or deadlocks (Valgrind, Intel Inspector, ...)
- If parallelism is added, re-run a memory related tool to check memory problems again.

➢ Analyse OpenMP regions and check OpenMP and MPI metrics (ompP, Scalasca, TAU, Intel Vtune-Amplifier, ...)

➢ Detect parallelism related problems and bottlenecks to improve the application performance (MAP, Extrae combined with Paraver, Intel Vtune combined with Intel Trace Analyser, ...)

As a summary, Table 1 contains a brief description of all analysed tools and a quick access to their reference manuals. Only applications with * symbol are currently available in FinisTerraeII supercomputer.

Application	Usage	Support	Manual
Scalasca	Scalasca is a software tool that supports the performance optimization of parallel programs by measuring and analysing their runtime behaviour. The analysis identifies potential performance bottlenecks – in particular those concerning communication and synchronization – and offers guidance in exploring their causes	C, C++, Fortran MPI, OpenMP, Pthreads hybrid MPI+OpenMP/pthreads Intel Xeon Phi	https://apps.fz-juelich.de/scalasca/releases/scalasca/2.3/docs/manual/
Score-p	The Score-P measurement infrastructure is a highly scalable and easy-to-use tool suite for profiling, event tracing, and online analysis of HPC applications. Score-P offers the user a maximum of convenience by supporting a number of analysis tools	C, C++, Fortran MPI, OpenMP, Pthreads hybrid MPI+OpenMP/pthreads Intel Xeon Phi	https://silc.zih.tu-dresden.de/scorep-current.pdf
Cube	Cube, which is used as performance report explorer for Scalasca and Score-P, is a generic tool for displaying a multi-dimensional performance space consisting of the dimensions, performance metric, call path, and system resource	C, C++, Fortran MPI, OpenMP, Pthreads hybrid MPI+OpenMP/pthreads Intel Xeon Phi	https://apps.fz-juelich.de/scalasca/releases/cube/4.3/docs/CubeGuide.pdf
Extra-p	Extra-P is an automatic performance-modeling tool that supports the user in the identification of scalability bugs	C, C++, Fortran MPI	http://www.scalasca.org/software/extra-p/documentation.html

Application	Usage	Support	Manual
TAU	TAU Performance System is a portable profiling and tracing toolkit for performance analysis of parallel programs	C, C++, Fortran UPC, Java	https://www.cs.uoregon.edu/research/tau/docs.php
SimGrid	SimGrid is a scientific instrument to study the behaviour of large-scale distributed systems such as Grids, Clouds, HPC or P2P systems. It can be used to evaluate heuristics, prototype applications or even assess legacy MPI applications	C, C++ Java MPI	http://simgrid.gforge.inria.fr/tutorials.php
Extrae	Extrae is a dynamic instrumentation package to trace programs compiled and run with the shared memory model, the message passing programming model or both	C, C++, Fortran MPI, OpenMP/pthread CUDA	https://tools.bsc.es/sites/default/files/documentation/pdf/extrae-3.5.2-user-guide.pdf
Paraver	Paraver is a flexible parallel program visualization and analysis tool based on an easy-to-use GUI	C, C++, Fortran, CUDA OpenMP/pthread, MPI	https://tools.bsc.es/tools_manuals
Dimemas	Dimemas is a performance analysis tool for message-passing programs. It enables the user to develop and tune parallel applications on a workstation, while providing an accurate prediction of their performance on the parallel target machine	MPI	https://tools.bsc.es/sites/default/files/documentation/introduction_dimemas.pdf
***Intel-XE Advisor**	Intel Advisor provides two tools to help ensure applications realize full performance potential	C, C++, Fortran Threads Intel Xeon Phi	https://software.intel.com/en-us/get-started-with-advisor
***Intel-XE Inspector**	Intel Inspector is a dynamic memory and threading error checking tool for users developing serial and multithreaded applications	C, C++, Fortran Threads	https://software.intel.com/en-us/get-started-with-vtune

Application	Usage	Support	Manual
***Intel Vtune-Amplifier**	Use this tool to analyse the algorithm choices, find serial and parallel code bottlenecks, understand where and how your application can benefit from available hardware resources, and speed up the execution	C, C++, C#, Fortran Java, Python MPI, OpenMP/pthread Intel Xeon Phi OpenCL	https://software.intel.com/en-us/get-started-with-vtune-linux-os
***Intel Trace Analyser and Collector**	Intel Trace Analyzer and Collector is a graphical tool for understanding MPI application behaviour, quickly finding bottlenecks, improving correctness, and achieving high performance for parallel cluster applications based on Intel architecture	C, C++, Fortran MPI Intel Xeon Phi	https://software.intel.com/en-us/get-started-with-itac-for-linux
MAP	MAP provides in-depth analysis and bottleneck pinpointing to the source line	C, C++, Fortran MPI, OpenMP/pthread	https://developer.arm.com/docs/101136/latest/map
Performance Reports	Performance Reports analyse the applications running on your system to seek out inefficiencies and pinpoint exactly where to focus optimization work	MPI, threads GPUs	https://developer.arm.com/docs/101137/latest/introduction
Vampir	Vampir provides an easy-to-use framework that enables developers to quickly display and analyze arbitrary program behavior at any level of detail	MPI, OpenMP/ Pthreads CUDA, OpenCL, OpenACC	https://www.vampir.eu/tutorial/manual
***Valgrind**	Valgrind is an instrumentation framework for building dynamic analysis tools. There are Valgrind tools that can automatically detect many memory management and threading bugs, and profile your programs in detail	C, C++, Fortran Python, Java, Perl	http://valgrind.org/docs/manual/manual.html
ompP	ompP is a profiling tool for OpenMP applications. It supports the measurement of hardware performance counters using PAPI and it supports productivity features such as overhead analysis and detection of common inefficiency situations	C, C++, Fortran OpenMP	http://www.ompp-tool.com/downloads/ompp-manual.pdf

Application	Usage	Support	Manual
Likwid	Likwid is a simple to install and use toolsuite of command line applications for performance oriented programmers	Intel and AMD processors	https://github.com/RRZE-HPC/likwid/wiki
***gperftools**	gperftools is a collection of a high-performance multi-threaded malloc() implementation, plus some performance analysis tools.	C++ Threads	https://github.com/gperftools/gperftools/wiki
***Darshan**	Darshan is designed to capture an accurate picture of application I/O behaviour, including properties such as patterns of access within files, with minimum overhead.	C, C++, Fortran MPI	http://www.mcs.anl.gov/research/projects/darshan/documentation/

Table 1.- Summary Table