D1.3 - Design of the Cross-Tier Service Infrastructure for Performance Profiling

Work Package 1

December 2017

ProfiT-HPC Consortium

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)
KO 3394/14-1, OL 241/3-1, RE 1389/9-1, VO 1262/1-1, YA 191/10-1

Abstract

In this project, the goal is to aid especially new users with only little programming experience in understanding the importance of performance profiling and optimisation of an application. For this, we will develop a toolkit which shall provide HPC users with an understandable report, at best in a comprehensive, consistent manner throughout Germany by rolling out the toolkit during the course of the project.

This deliverable summarizes the results of the recent work of work package 1 and 2 (see [3], [4], [1] and [2]). Furthermore a short evaluation of the topic is given, if the considered performance metric collecting tools and monitoring frameworks are applicable without significant changes on Tier 3 systems as well as on Tier 2 clusters.

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
KO 3394/14-1, OL 241/3-1, RE 1389/9-1, VO 1262/1-1, YA 191/10-1

Contents

Introduction

This document represents the final part of work package 1: "D.1.3 - Design of the Cross-Tier Service Infrastructure for Performance Profiling". This document will wrap up the results of all work packages contributing to the design of the developed toolkit. For this, the document will again shortly assess different performance metrics collector daemons and monitoring frameworks and furthermore present a short summary of the results of the online survey. Section 3 will regive the aforementioned results, before Sec. 8 concludes the results to a broader perspective on the projeict and the toolkit development.

Overview

The goal of the ProfiT-HPC project is that the ProfiT-HPC profiling toolkit automatically gathers the performance information for all jobs and presents them in an easy understandable manner. The following three steps establish the basis for this project:

- Creation of an online survey, that was open for 53 national Tier 3 and Tier 2 computing centres and universities, and the evaluation of the answers to get an overview of the on-site installed hardware and software architecture (see Deliverables D1.1 [?] and D1.2 [3]). This overview was an important step in the project, since it is very important for the future acceptance of the performance measurement tool, that the needs and the infrastructures of national compute centres are well understood.
- To get an overview of currently freely available profiling and monitoring tools, a concise overview of performance metrics and performance metric collecting tools was produced, to analyse, with which tools the selected metrics can be collected (see Deliverable D2.1 [4]).
- On the basis of the selected metrics and tools, some for the project concerns most promising tools were examined in detail, as well as the metrics collected by them. Furthermore, frameworks which base on those tools were evaluated further and (when possible) a proof of concept was made.

This document will give a short summary of these three steps and put the results of the according deliverables into a greater context.

The Online Survey

53 computing centres and universities were invited via e-mail to the online survey of which 33 took part in this survey (62 % of all invited institutions). The complete questions and results of this survey can be found in Deliverable D1.2 (see [3]). The following results are the most prominent survey outcomes:

- On most HPC clusters, Intel Xeon CPUs are installed. Approximately one half of all Intel Xeon processors are of the `Haswell` and `Broadwell` generation.
- In almost all institutions some kind of `InfiniBand` interconnect is installed.
- Regarding the parallel file systems, the result was not as clear at all: the three most prominent parallel file systems (`BeeGFS`, `Lustre` and `GPFS`) are all represented with a non negligible value.
- The operating system on all clusters of the participating institutions is Linux and the most prominent Linux distribution is `CentOS` (Version 7). The kernel version 3.10 of Linux is present on approximately one half of the number of clusters.
- `Slurm` is the batch system, which is used on more than one half of the clusters, followed by `TORQUE`, which is installed on one third of all clusters.
- The most used C/C++ and Fortran compiler families are GCC and Intel.
- `OpenMPI` (which is on site on approximately 90 % of all clusters) and `IntelMPI` (installed on approximately 80 % of the clusters) are the dominating MPI implementations.
- One of the most important question of this survey was, which profiling or monitoring tools are already installed on the clusters. Intel VTune was mentioned most of all tools. But since the project goal is, that the tool should be light weighted and IntelVTune is very

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
KO 3394/14-1, OL 241/3-1, RE 1389/9-1, VO 1262/1-1, YA 191/10-1

invasive, this tool is not an option with regard to the goal of this project. There are further tools, which were often stated (`vmstat`, `Sysstat`, `perf` and `Likwid`), which are promising candidates for our needs.

The Evaluation of the Performance Measurement Tools

Another task at the beginning of the project was the identification of those metrics, which are of interest for the project needs and the evaluation of many profiling and monitoring tools, which should collect the metrics of interest (see Deliverable D2.1 [4]).

Two kinds of tools were investigated:

- Tools, which are part of Linux or generally part of a Linux distribution, e.g.
  - `GNU time`,
  - `perf` and,
  - `procfs` (this is a pseudo filesystem, but delivers interesting metrics).
- Tools, which are in general not part of Linux or not part of a Linux distribution, e.g.
  - `Darshan`,
  - `Likwid` and
  - `Open|SpeedShop`.

In both cases the following questions had to be evaluated:

- Collected metrics of a job / process (in the following general metric classes with examples are listed):
  - CPU, e.g. wall clock, user, system and idle time.
  - Hardware counters, e.g. FLOPS, cache misses, branch mispredictions.
  - Memory, e.g. currently and maximum used memory (RSS and memory high water mark).
  - File system, e.g. number of page faults, bytes transferred from or to the file system.
  - MPI, e.g. number of bytes transferred over the network, loadbalancing issues.
- Operational aspects:
  - Is the tool easy to use?
  - Is it possible to collect the performance metrics automatically with the tool?
  - What is the overhead of the collection process?
  - What is the possible impact on the performance metrics collection process?
- Which method is used to collect the performance metrics?
  - Counting,
  - Sampling or
  - Profiling.

Based on this overview, in Deliverable D2.2.1 a selection of the tools from Deliverable 2.1. was investigated further and in more detail concerning practical aspects.

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
KO 3394/14-1, OL 241/3-1, RE 1389/9-1, VO 1262/1-1, YA 191/10-1

A More Detailed Evaluation of Performance Measuring Tools

In the course of the project the following monitoring and profiling tools were identified as possibly useful for the metric collection process:

- `procfs` (which is pseudo file system, but delivers valuable information of the system),
- `vmstat` (monitoring tool, to collect CPU related, memory related and further metrics),
- `GNU time` (a time command with additional measurements of program wide metrics),
- `Sysstat` (monitoring toolkit, including the tools `sar`, `pidstat`, `iostat`),
- `perf` (collects hardware counter values),
- `Likwid` (collects hardware counter values),
- `Darshan` (collects IO information),
- `Intel MPI Statistics Gathering Mode`,
- `Open|SpeedShop`.

While the installation phase it turned out that `Open|SpeedShop` has a lot of dependencies and was not installable straight forward. This was a negative point, since the goal of the project is to create a lightweight toolkit regarding the installation, too. Furthermore `Open|SpeedShop` and `perf` (in the `record` mode) produced a non negligible overhead during the benchmark runs with BQCD and IOR, which was significantly above our defined overhead limit, which is problematic.

All other tools behaved largely well in terms of the collection overhead and the impact on the collection process, and could be used for the project's purposes. In the end, the pseudo filesystem `procfs` is one of the most promising tools for collecting basic metrics, because of the simplicity of collecting many metrics, the relatively low collecting overhead and that it is a part of the Linux kernel.

The other tools will again be evaluated for the next steps of the toolkit, where more detail on the application's performance can be collected with more intrusive tools.

An Evaluation of Performance Measuring Frameworks

Although it is possible to implement all necessary tools of the collecting and storing process within the project, it is more convenient and more efficient to use already developed frameworks for the project purposes, which are mostly based on metric collecting tools.

In this part of the project, some monitoring frameworks were examined, which could be useful for the projects concerns and are partly based on the above mentioned tools.

The following frameworks were investigated further (a more exhaustive description of the different frameworks can be found in Deliverable 2.2.2 [2].):

- `TICK Stack`, which consists of
  - `Telegraf` as the metric collector,
  - `InfluxDB` as the time series database,
  - `Chronograf` as the visualisation tool and

  - `Kapacitor` as the real-time streaming data processing engine.

- `TACCStat` (a performance monitoring tool from the University of Texas),
- `Collectl` (another performance monitoring tool),
- `PerSyst` (a performance monitoring tool, which is used to collect metrics and doing performance analysis on the SuperMUC (Leibniz Supercomputing Centre, Munich)).

In the course of the evaluation it was seen, that the `PerSyst` tool has some very interesting features, but since the porting to another system is not straight forward and needs non trivial adaption to the respective system, the `PerSyst` system is not further considered for our project purposes. Another detail, which stands in contrast to the needs of the toolkit, is the relatively long total time measurement interval (in total ten minutes).

`TACCStat` was not taken into further consideration either, because also this has been developed with special focus to the TACC system which made it difficult to port and install. In addition, the focussed audience for `TACCStat` are the administrators of the system and not the users themselves. Nevertheless it has many good features, but this framework is not recommended for the project purposes.

The `TICK Stack` was relatively easy to install and the plugin concept is flexible and appropriate for our needs. In addition, it already comprises all components that the ProfiT-HPC toolkit needs, while at the same time allowing to replace single components with available components at the installing data centre. This aspect should be most beneficent for the roll-out phase near the end of the project. Furthermore, GWDG has experience with this system in other fields, so the ongoing work will base on this framework.

`Collectl` is similarily customizable as the `TICK Stack` and already includes many aspects needed in the ProfiT-HPC toolkit. Nonetheless, the `Tick Stack` already has more plugins directly available, thus minimizing the development overhead of the ProfiT-HPC toolkit.

Conclusion

In this final document of work package 1, an overview of the results of the online survey and the evaluation of the performance metric collecting tools and monitoring frameworks were given.

In the course of the evaluation phase, several tools were identified, which are suitable to collect special metrics (semi-) automatically and with negligible overhead. The most promising tools are listed in Section 6. Furthermore some monitoring frameworks were investigated regarding the simplicity of the installation and portability process, possible security issues and measurement overhead. The final result of these evaluation efforts was, that in the further project especially the metric collecting tool `Telegraf` with the time series database `InfluxDB` will be used. Since `Telegraf` uses a plugin concept, it is possible to include the selected tools in this framework (e.g. `procfs`, `sar`, etc.).

An intended goal of the ProfiT-HPC project is the deployment of the ProfiT-HPC toolkit on Tier 3 clusters. Furthermore it is desirable, that the toolkit is also installable and usable on Tier 2 systems without the need of significant adaptions to the toolkit. A further advantage of the portability of the toolkit to Tier 2 cluster is, that the user of the ProfiT-HPC toolkit on a Tier 3 system does not need to get used to another performance analysis system. Even the Tier 1 Leibniz Supercomputing Center uses `procfs`, `Likwid` and `sar` (among other tools) for its `PerSyst` monitoring tool, thus, `PerSyst` theoretically also usable in a monitoring and profiling toolkit on Tier 2 systems. Our favoured `TICK Stack` system is installed on many cloud systems with no big problems documented. This is valid also for some Tier 3 HPC clusters but to our best knowledge there are no implementations of the TICK Stack on Tier 2 clusters yet. Since there are no obvious restrictions on the architecture, we conclude, that no significant implementation and configuration issues exist. Further investigations will have to be made on scalability and overhead of the different components of the stack.

If this strategy fails, it is possible to only install parts of this toolkit, because of the envisaged modularity of the ProfiT-HPC Toolkit (performance metric collector, time series database, performance analysis and report generator, visualisation tool). Especially the report generator should be independent of the size of a cluster, which would simplify the work for the user, if they migrate their programs from Tier 3 to Tier 2 systems, which is one of the project goals.

References

[1] ProfiT-HPC Consortium. D2.2.1: Functional specification of the backend – part 1: Pre-selection of the metric collection tools.

[2] ProfiT-HPC Consortium. D2.2.2: Functional specification of the backend – part 2:.

[3] ProfiT-HPC Consortium. Deliverable 1.2 - results of a survey concerning the tier-2 and tier-3 hpc-infrastructure in germany.

[4] ProfiT-HPC Consortium. Deliverable 2.1 - concise overview of performance metrics and tools.

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG)
KO 3394/14-1, OL 241/3-1, RE 1389/9-1, VO 1262/1-1, YA 191/10-1