Analysis and Extrapolation of CPU processing Rate for the Simulation of Parallel Applications

Frédéric Suter

September 19, 2013

1 Motivations

Analyzing and understanding the performance behavior of parallel applications on various compute infrastructures is a long-standing concern in the High Performance Computing community. When the targeted execution environments are not available, simulation is a reasonable approach to obtain objective performance indicators and explore various hypothetical scenarios. In the context of applications implemented with the Message Passing Interface (MPI), two simulation methods have been proposed, on-line simulation and off-line simulation, both with their own drawbacks and advantages. In off-line simulation, a trace of a previous execution of the application is "replayed" on a simulated platform. The main advantage when compared to on-line simulation is that replaying the execution can be performed on a single computer. This is because the replay does not entail executing any application code, but merely simulating computation and communication delays. While the simulation of communications is a well-covered area, with the proposition of accurate models [1], that of computations usually remains rather simple. Indeed, all off-line simulators of MPI applications rely on time-stamped traces, i.e., each traced event is associated to a date, obtained on a homogeneous and at scale platform. It is then possible to apply a uniform scaling when simulating a target platform with slower/faster processors. But, as a result, trace acquisition is not easily scalable since a homogeneous platform may not be available at the required scale.

To solve the above trace acquisition scalability problem, we proposed the Time-Independent Trace Replay Framework [3, 2]. This is achieved by eliminating time-stamps altogether from application traces. An application then corresponds to a list of *actions* performed by each process of an MPI application. An action is described by the *rank* of the process that performs it, a *type* (compute, send, or receive), a *volume* (number of instructions or number of bytes), and some action-specific parameters (e.g., the rank of the receiving process for a one-way communication).

While this framework allows for the acquisition of large traces on any platforms, e.g., heterogeneous multi-clusters, it raises an important question: *how to instantiate the simulator with an appropriate CPU processing rate for the target without any timing information?* In our previous work, we computed an *average* instruction rate on a *calibration instance* of the studied application. This rate was then used by the simulator. However, this method is a rough estimation and hinders the accuracy of our simulation.

2 Subject

The main objective of this internship is to design a efficient way to calibrate the CPU processing rate for the off-line simulation of MPI applications. To achieve this, the candidate will have to study an existing analysis of the distribution of the processing rate versus the number of instructions computed by each action in the trace. It shown great differences that may explain the inaccuracy. Then s/he may start from a simple, but promising, idea that was proposed. Actions are first grouped according to the number of instructions they execute by intervals of fixed size. Then an average processing rate is computed for each interval and used in the simulation. The problem of this approach is that it requires unavailable information to be effective. The candidate will then have to:

- Acquire execution traces on the Grid'5000 experimental platform[1]. Some traces are already available, but more might be needed;
- Conduct a thorough statistical analysis of the distribution of the processing rates. The R language is likely to be used at this step;
- Propose a way to extrapolate information gathered on calibration instances to target, i.e., large instances of the studied application(s);
- Conduct a validation study of the proposed approach by comparing simulation results obtained thanks to the SimGrid toolkit [4] to actual experimentations.

To favor an Open Science approach, the candidate will be encouraged to document and made available all the processes and data used during the internship. This may be eased by tools such as the org mode of the Emacs editor.

3 Environment

The candidate will be hosted by the research team of the IN2P3 Computing Center (CC IN2P3)[2]. CC IN2P3 is a service and research unit belonging to CNRS (USR 6402). A major French research infrastructure, it is responsible for providing computing and data storage resources for researchers involved in corpuscular physics experiments. The main services offered by CC-IN2P3 are the storage and processing of large volumes of data and the transfer of these data over very high- speed international networks. CC-IN2P3 is one of eleven centers worldwide engaged in the primary processing of LHC data and one of only four centers that will provide storage and data processing capacity for all four LHC experiments. Since 2008, CC-IN2P3 has a research team whose members are associated to the AVALON team[3] of the Laboratoire de l'Informatique du Parallélisme (UMR 5668) at the École Normale Supérieure de Lyon.

[1] http://www.grid5000.fr
[2] http://cc.in2p3.fr
[3] http://avalon.ens-lyon.fr/

References

[1] Clauss PN, Stillwell M, Genaud S, Suter F, Casanova H, Quinson M. Single Node On-Line Simulation of MPI Applications with SMPI. *Proc. of the 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Anchorage, AK, 2011.

[2] Desprez F, Markomanolis GS, Quinson M, Suter F. Assessing the Performance of MPI Applications Through Time-Independent Trace Replay. *Proceedings of the 2nd International Workshop on Parallel Software Tools and Tool Infrastructures (PSTI)*, Taipei, Taiwan, 2011; 467–476.

[3] Desprez F, Markomanolis GS, Suter F. Improving the Accuracy and Efficiency of Time-Independent Trace Replay. *Proc. of the 3rd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, Salt Lake City, UT, 2012.

[4] Casanova H, Legrand A, Quinson M. SimGrid: a Generic Framework for Large-Scale Distributed Experiments. *Proc. of the 10th IEEE International Conference on Computer Modeling and Simulation*, Cambridge, UK, 2008.