Performance at Exascale

Bernd Mohr
Jülich Supercomputing Centre
b.mohr@fz-juelich.de

Matthias S. Mueller, Wolfgang E. Nagel
Center for Information Services and HPC
{matthias.mueller,wolfgang.nagel}@tu-dresden.de

Introduction

Exascale systems will consist of complex configurations with a huge number of potentially heterogeneous components. Deep software hierarchies of large, complex software components will be required to make use of such systems. While the software layers are designed to be transparent, they are typically not transparent with respect to performance. This *performance intransparency* will result in escalation of unforeseen problems to higher layers, including the application. This is not a really new problem, but certain properties of an exascale system significantly increase its severity and significance.

- At this scale, there always will be failing components in the system with a large impact on performance. A "real-world" application will never run on the exact same configuration twice.
- Load balancing issues limit the success even on moderately parallel systems, and the challenge of locality will become another severe issue which has to be addressed by appropriate mechanisms and tools.
- Dynamic power management, e.g., at hardware level inside a CPU, will result in performance variability between cores and across different runs. The alternative to run at lower speed without dynamic power adjustments may not be an option in the future.
- The unknown expectation of the application performance at exascale will make it difficult to detect a performance problem if it is escalated undetected to the application level.
- The ever growing higher integration of components into a single chip and the use of more and more hardware accelerators makes it more difficult to monitor application performance and move performance data out of the system unless special hardware support will be integrated into future systems.

Altogether this will require a integrated and collaborative approach to handle performance issues and correctly detect and analyze performance problems.

Performance Analysis

A large number of approaches for performance analysis exist that have successfully applied at small and medium scale. The large amount of performance data may seem to impede the use at exascale. However, this is not the case as long as features like memory size and I/O capabilities scale with compute power. An instrumented application is nothing but an application with modified demands on the system executing it. This makes current approaches for performance analysis still feasible in the future as long as all involved software components are parallel and scalable. In addition to increased scalability techniques like automatic analysis, advanced filtering techniques, on-line monitoring, clustering and analysis as well as data mining will be of increased importance. A combination of various techniques will have to be applied. The following considerations are key for a successful approach to performance at exascale:

- Failover or more general the operation with failed components should be performance neutral.
- An exascale system has to be capable to monitor the performance of components, not just the functionality.
- Hardware and software components need to provide sufficient performance details for analysis if a performance problem unexpectedly escalates to higher levels.
- Metrics beyond FLOPs need to be developed to identify and quantify performance problems, to measure the sustained performance and the gap to the attainable peak performance.
- Programming models should be designed with performance analysis in mind. Part of that could be a (standardized) hidden control mechanism in the runtime system that will be able to dynamically control – in time and space – the generation of performance data if requested.
- Performance analysis in the presence of "noise" requires inclusion of appropriate statistical descriptions.
- Performance analysis needs to incorporate techniques from the areas of signal processing and data mining.