AutoTune: Automatic Online Tuning

Renato Miceli[1], Anna Sikora[2], and Houssam Haitof[3]

[1] Irish Centre for High-End Computing, Grand Canal Quay, Dublin 2, Ireland
renato.miceli@ichec.ie
[2] Universitat Autònoma de Barcelona, Computer Architecture and Operating Systems Department, Barcelona, Spain
ania@caos.uab.cat
[3] Technische Universität München, Institut für Informatik, Garching, Germany
haitof@in.tum.de

Abstract. Performance analysis and tuning is an important step in programming multicore- and manycore-based parallel architectures. While there are several tools to help developers analyze application performance, no tool provides recommendations about how to tune the code. AutoTune will extend Periscope, an automatic online and distributed performance analysis tool developed by Technische Universität München, with plugins for performance and energy efficiency tuning. The resulting Periscope Tuning Framework will be able to tune serial and parallel codes with and without GPU kernels; in addition, it will return tuning recommendations that can be integrated into the production version of the code. The whole tuning process, consisting of both automatic performance analysis and automatic tuning, will be executed online, i.e., during a single run of the application.

1 Introduction

The pervasiveness of multi- and many-core processors makes any computer system a parallel system. Embedded devices include multiple specialized cores for application and OS processing, media decoding and I/O. Servers consist of multiple multicore processors organized in a NUMA architecture. Parallel desktop machines extend the server architecture by having GPGPUs attached to the main processor, each offering extreme memory bandwidth to their own graphics memory to feed its numerous cores. High-performance computing systems couple hundreds of thousands of nodes, each possibly equipped with accelerators (GPGPUs, FPGAs), connected via a high-performance network in a distributed memory architecture. Programming these parallel architectures requires careful co-optimization of the following interrelated aspects:

- Energy consumption: Energy reduction has become a major issue on HPC architectures, since their power costs almost reach the purchase price over a life time. Careful application-specific tuning can help to reduce energy consumption without sacrificing an application's performance.

- Interprocess communication: The overall scalability of parallel applications is significantly influenced by the amount and the speed of communication required. Reducing the communication volume and exploiting the physical network topology can lead to great performance boosts.
- Load balancing: Implicit/explicit process synchronization and uneven distribution of work may leave a process idle waiting for others to finish. The computing power within all parallel processes must be exploited to the fullest, otherwise program scalability may be limited.
- Data locality: Frequent accesses to shared or distant data creates a considerable overhead. Reducing the contention for synchronization resources and ensuring data locality can yield significant performance improvements.
- Memory access: Even the best arithmetically-optimized codes can stall a processor core due to latency in memory access. Careful optimization of memory access patterns can make the most of CPU caches and memory bandwidth on GPGPUs.
- Single core performance: To achieve good overall performance each core's compute capabilities need to be optimally exploited. By providing access to the implementation details of a targeted platform, application optimizations can be specialized accordingly.

Application tuning with respect to these areas is an important step in program development. Developers analyze an application performance to identify code regions that may be improved. Then they perform different code transformations and experiment setting parameters of the execution environment in order to find better solutions. This search is guided by experience and by the output of performance analyses. After the best set of optimizations has been found for the given – and supposedly representative – input data set a verification step with other input data sets and process configurations is executed.

The research community and vendors of parallel architectures developed a number of performance analysis tools to support and partially automate the first step of tuning process, i.e. application analysis. However, none of the current tools supports the developer in the subsequent step of tuning, i.e. application optimization. The most sophisticated tools can automatically identify the root cause of a performance bottleneck but do not provide developers with hints about how to tune the code.

Therefore, the AutoTune Project's goal is to close the gap in the application tuning process and simplify the development of efficient parallel programs on a wide range of architectures. The focus of this project is on automatic tuning for multicore- and manycore-based parallel systems ranging from parallel desktop systems with and without GPGPU accelerators to petascale and future exascale HPC architectures. To achieve this objective, AutoTune aims at developing the Periscope Tuning Framework (PTF), the first framework to combine and automate both analysis and optimization into a single tool. AutoTune's PTF will:

- Identify tuning alternatives based on codified expert knowledge.

– Evaluate the alternative online (i.e. within the execution of the same application), reducing the overall search time for a tuned version.
– Produce a report on how to improve the code, which can be manually or automatically applied.

2 Related Works

The complexity of today's parallel architectures has a significant impact on application performance. In order to avoid wasting energy and money due to low utilization of processors, developers have been investing significant time into tuning their codes. However, tuning implies searching for the best combination of code transformations and parameter settings of the execution environment, which can be fairly complicated. Thus, much research has been dedicated to the areas of performance analysis and auto-tuning. The latest gathered ideas can be grouped into the following categories:

– Profiling tools, such as gprof and OMPP.
– Tracing tools, such as Vampir.
– Self-tuning libraries for linear algebra and signal processing, which generates and optimizes code execution for the architecture under use, such as ATLAS, FFTW, OSKI and SPIRAL.
– Tools that analyze compiler optimizations and search for their optimal combination, based on either iterative search or machine learning techniques.
– Auto-tuners, which search a space of application-level parameters that are believed to impact application performance, such as Active Harmony.
– Utilities to exploit dynamic clock frequency and voltage level scaling on modern processors, based on the ACPI specification, implemented by Intel's SpeedStep and AMD's Cool'n'Quiet.
– Frameworks that combine several approaches, such as SCALASCA, which profiles and analyzes traces to determine MPI wait time; Periscope, which performs program analysis and searches for performance problems while the application is executing; MATE, which searches for bottlenecks by applying a performance model and dynamically tuning the application during its execution; and Parallel Active Harmony, an auto-tuner for scientific codes that applies a search-based approach.

Performance analysis and tuning are currently supported via separate tools. AutoTune aims at bridging this gap and integrating support for both steps in a single tuning framework. It will not only improve the execution of individual applications but also provide with an optimized version of the code.

3 Approach: The Periscope Tuning Framework (PTF)

AutoTune is developing the Periscope Tuning Framework (PTF) as an extension to Periscope. It will follow Periscope's main principles, i.e. the use of formalized

expert knowledge and strategies, automatic execution, online search based on program phases, and distributed processing. Periscope will be extended by a number of online and semi-online tuning plugins; online plugins perform transformations to the application and/or the execution environment without requiring the application to restart, while semi-online plugins may require the application to be restarted.

The tuning process in PTF starts with the source files, written in C/C++ or Fortran using MPI and/or OpenMP, being pre-processed. PTF will instrument and analyze the code statically. This pre-processing phase in Periscope will be extended to support HMPP and OpenCL codes and parallel patterns. The tuning is then started via the front-end either interactively or in a batch job.

Periscope's analysis strategy will become part of a higher-level tuning strategy, which will control the sequence of analysis and optimization steps. The analysis will guide the selection of a tuning plugin as well as the actions it will perform. After the plugin execution ends the tuning strategy might restart the same or another analysis strategy to continue on further tuning. Each plugin will have its own tuning strategy – which may employ expert knowledge or machine learning –, and combined to the analysis strategies in Periscope they will perform a multi-aspect application tuning.

Each of the tuning plugins will be controlled by a specific plugin strategy, meant to guide the search for a tuned version. The search space will be restricted based on the output of the previous analyses as well as by the output of each plugin executed. Selecting tuning actions ends with a number of possibilities that have to be evaluated experimentally.

Once the tuning process is finished, a tuning report will be generated to document the tuning actions recommended by PTF. PTF will generate it such that the developer can integrate the tuning recommendations for production runs. These actions may then be integrated into the application either manually or automatically such that subsequent production runs will be more efficient.

The concrete output of the AutoTune Project is the Periscope Tuning Framework and its tuning plugins for:

- GPU programming with HMPP and OpenCL;
- Single-core performance tuning;
- MPI tuning;
- Energy efficiency tuning.

The framework will have an extensible architecture so that its functionality can be easily extended by additional plugins. The effectiveness of PTF and the software quality will be demonstrated with real-world parallel applications on parallel architectures with and without GPU accelerators.

Acknowledgements. The authors thank the European Union for supporting this project under the Seventh Framework Programme, grant no. 288038.