Journal of Computational Science 27 (2018) 329–344

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Synapse: Synthetic application profiler and emulator

Andre Merzky*, Ming Tai Ha, Matteo Turilli, Shantenu Jha

RADICAL Laboratory, Electric and Computer Engineering, Rutgers University, New Brunswick, NJ, USA

ARTICLE INFO

Article history:
Received 27 January 2018
Received in revised form 1 June 2018
Accepted 26 June 2018
Available online 2 July 2018

Keywords:
Emulation
Profiling

ABSTRACT

Motivated by the need to emulate workload execution characteristics on high-performance and distributed heterogeneous resources, we introduce Synapse. Synapse is used as a proxy application (or "representative application") for real workloads, with the advantage that it can be tuned in different ways and dimensions, and also at levels of granularity that are not possible with real applications. Synapse has a platform-independent application profiler, and has the ability to emulate profiled workloads on a variety of resources. Experiments show that the automated profiling performed using Synapse captures an application's characteristics with high fidelity. The emulation of an application using Synapse can reproduce the application's execution behavior in the original runtime environment, and can also reproduce those behaviors on different run-time environments.

1. Introduction

A large body of research in high-performance and distributed computing is concerned with the design, implementation and optimization of tools, runtime systems and services in support of scientific applications. These tools, systems and services, often subsumed under the term "middleware", require extensive and continuous development, testing and optimization based on various, real-life applications and production-grade infrastructures.

The use of scientific applications for developing middleware present three main challenges: large amount of deployment requirements; domain-specific knowledge for successful and representative execution; and limited portability and scalability. Synthetic applications—also known as Skeletons, Representative, or Artificial Applications—help to address these challenges by working as proxies of real-life applications.

Synthetic applications capture the relevant properties of applications, minimizing deployment requirements and eliminating the need of domain-specific knowledge for their execution. Further, they enable tuning application parameters relevant to scientific middleware development. Often, parameters like precise runtime, number of computational cycles, memory footprint or I/O patterns cannot be precisely tuned in real-life applications.

A tradeoff in the design and implementation of synthetic applications for use as proxy applications is the need to be simple and general-purpose and to be able to emulate the behavior of each application with the highest level of accuracy and fidelity possible. Achieving this level of accuracy and fidelity is particularly challenging when an emulation is used on multiple heterogeneous resources. It is even more challenging when the resources used for emulation are different from the resource on which the application(s) was profiled.

In response to these requirements and tradeoffs, we have developed Synapse: a *SYNthetic Application Profiler and Emulator*. Synapse is primarily motivated by the need for automated and system-independent application profiling in computational science, where the multitude and generality of applications and platforms are more important than cycle-level accuracy and fidelity. Accordingly, Synapse is designed to provide uniform profiling capabilities across a range of application types, tools and services, while achieving a sufficient level of accuracy and fidelity.

Synapse acts as a proxy application to circumvent the limitations and complexity of scientific applications. For example, scientific applications are not infinitely malleable because of the fixed and often discrete physical sizes of input systems; they have limited tunability as parameters can be modified only in discrete steps over a limited range of values. Synapse can profile an application for given parameter values *and* can emulate the execution behavior of the same application for different parameter values.

Synapse is designed to "profile once, emulate anywhere": Synapse determines the application's resource consumption by running a sample-based, black-box profiler of the application on a machine with specific tools and permissions, and replays the observed resource consumption patterns on an arbitrary machine. In emulation mode, Synapse attempts to consume the same amount of system resources (clock cycles, memory reads/writes, packet

* Corresponding author.
E-mail address: andre@merzky.net (A. Merzky).

https://doi.org/10.1016/j.jocs.2018.06.012
1877-7503/

sent/received) as the original application, while preserving the overall pattern, granularity and sequence of the respective operations.

Synapse provides basic kernels functions to emulate how different system resources are consumed. While these kernels can be controlled to consume the specified amount of system resources precisely, they are less precise on how those resources are consumed, i.e., in what chunkiness, granularity and order. Synapse provides the ability for users to write their own kernels to control more tightly how system resources are consumed. Thus, even when different applications consume the same amount of some system resource, Synapse can be tuned to consume system resources in a way that better reflects how a specific application consumes the same resource.

This paper presents the design of Synapse and progress towards an implementation that is robust and usable. Further, we presents six experiments to validate that Synapse's automated profiling indeed captures the application characteristics with fidelity. The experiments also show that Synapse's emulation reproduces the application characteristics in the original runtime environment, as well as on different resources and runtime systems.

While Synapse, and in particular its profiler, is not designed to achieve the same accuracy as other established approaches (e.g., Vtune [1]), our experiments support the claim that Synapse's emulation has sufficient fidelity, generality and tunability to make it a useful instrument for the development of middleware to support computational science research.

In Section 2 we outline three application and systems use cases that have motivated the development of Synapse. In Section 3, we discuss the design and architecture of Synapse, followed by a discussion of selected implementation details in Section 4. Experiments are discussed in Section 5, followed by future and related work.

2. A case for system-independent profiling and emulation

The development of tools for computational science and large-scale computer science experiments needs proxy applications that provide flexible and tunable capabilities as well as being portable across resource types. We outline three use cases for proxy applications, each highlighting a different requirement.

2.1. High-performance task-parallel computing

Traditionally, high-performance computing (HPC) systems have been optimized to support mostly monolithic workloads. The workload of many scientific applications however, are comprised of spatially and temporally heterogeneous tasks that are often dynamically inter-related [2]. These workloads can benefit from being executed at scale on HPC resources, but a tension exists between their resource requirements and the capabilities of HPC systems and their usage policies. We addressed this tension by developing RADICAL-Pilot [3], a scalable and interoperable pilot system [4].

RADICAL-Pilot provides a runtime system designed to support a large number of concurrent tasks with low start-up overhead. RADICAL-Pilot is agnostic to the specific properties of the executed tasks, supporting many-node parallelism as well as single core tasks, and both short and long running tasks. Many components of RADICAL-Pilot are designed and parameterized to provide balanced performance while being as agnostic as possible to task and resource properties. For example, RADICAL-Pilot's task execution component, the RP Agent, has to be engineered for optimal resource utilization while maintaining full generality in many different dimensions, like MPI/non-MPI; OpenMP/multi-threaded/single-threaded; CPU/GPU/accelerators; single-node/multi-node; homogeneous/heterogeneous tasks and clusters; or different batch systems.

We can support the design and testing of RADICAL-Pilot by tuning the properties of a single proxy application instead of refactoring multiple scientific applications. For example, we can profile a single-threaded scientific application and then emulate it as a mixed OpenMPI or MPI proxy application on an arbitrary resource. Analogously, we can increase the amount of memory required by the same proxy application to a specific value, even if the science problem of the profiled application does not require that amount of memory.

2.2. Abstractions and middleware for distributed computing

In spite of significant progress in scientific distributed computing over the past decade, there are few general-purpose abstractions that support a principled development of middleware for large-scale and distributed execution of applications. As a consequence, many software point-solutions exist that are tailored to specific workloads or resource types but few middleware are capable of supporting arbitrary applications on arbitrary types of resources. The DOE AIMES project contributed to address this problem by defining general-purpose abstractions and developing a middleware for distributing the execution of large-scale applications across multiple resources [5].

AIMES assumed the use of third-party tools to manage dependences among tasks of scientific application. As a consequence, the main challenges in generalizing the capabilities of the AIMES middleware to different applications on different types of resource were primarily related to implementation and deployment. A proxy application that can emulate the execution behavior of actual workloads would play an important role in the validation and extension of base AIMES abstractions and middleware. Proxy applications have the advantage of capturing relevant application properties without exposing the complexity of running these applications on distinct platforms.

2.3. Toolkits for computational science

Many scientific applications in the field of molecular sciences, computational biology [6], astrophysics [7], weather forecasting [8], and bioinformatics [9] are increasingly reliant on ensemble-based methods to make scientific progress. Ensemble-based applications vary in the degree of coupling and dependency between tasks, and in heterogeneity across tasks. In spite of the apparent simplicity of running ensemble-based applications, the scalable and flexible execution of a large set of tasks is non-trivial.

As a consequence of complexity and many degrees-of-freedom, the challenges and the growing importance and pervasiveness of ensemble applications, we designed and implemented Ensemble Toolkit [10]. Similar to the previous two use-cases, a proxy application would provide a lightweight and highly tunable workload so as to simplify and design Ensemble Toolkit for general purpose workloads. In addition, a proxy application would provide the ability to vary the duration and number of task instances between different stages of the application and change the coupling between tasks; this is an important characteristics of applications used for advanced sampling [11].

3. Synapse scope and architecture

A finer-grained analysis of the aforementioned use cases, results in the following requirements on the profiling and emulation stages of Synapse.