SST-GPU: An Execution-Driven CUDA Kernel Scheduler and Streaming-Multiprocessor Compute Model

M. Zhang1, M.Khairy1, T.Rogers1, and C. Hughes2

1AALP research Group, Purdue University

2Center for Computing Research, Sandia National Laboratories

Session Type: Tutorial

Requested Duration: Half-day

Attendee Benefit

Tutorial participants will be introduced to key facets of conducting reproducible simulations of High Performance Computer (HPC) architectures and infrastructures. Through a blended process of presentation, tutorial examples, and interactive hands-on lab exercises, attendees will be exposed to the Structural Simulation Toolkit (SST) framework and the new GPGPU-Sim component. A high-level description of the integration can be found in "SST-GPU: An Execution -Driven CUDA Kernel Scheduler and Streaming-Multiprocessor Compute Model," SAND Report, February 2019 (M. Khairy *et al*.).

Relevance to IISWC

As the science of supercomputing and HPC evolves, there is an urgent need to understand and quantify the performance and compositional value of key technologies. Modeling and simulation techniques are well positioned to serve this purpose. However, there is an urgency for tools and techniques to become more unified; enabling and fostering collaboration and repeatability among academic, industrial, and government partners. SST is a parallel discrete event-driven simulation framework that provides an infrastructure to produce simulators capable of modeling a variety of high performance computing systems at many different scales. Currently used by a wide variety of government agencies and computer manufacturers to design and simulate HPC architectures, and supported by a Python and C++ code base and a large array of customization options, SST offers the HPC community powerful, highly customizable tools to create and integrate models for evaluating current and novel HPC architectures, interconnect networks, and infrastructures.

Target Audience

It is anticipated that all members of the HPC community, including industry partners, academic researchers, and students with a focus in computer architecture and simulation, will gain knowledge and experience in a collaborative platform with easy integration of their research models. Faculty members and industrial partners will also recognize the benefit of this useful simulation tool suite. This tutorial nominally includes 75% introductory and 25% intermediate content; it is highly recommended that participants have basic knowledge of the Linux command line (e.g., create folders, and manipulate files), have some coding knowledge (C++, Python), and have a fundamental knowledge of computer architecture.

Detailed Description of Courses and Objectives

As components, architectures, and systems become increasingly complex, simulation has taken on a pervasive role in the process of realizing a complex engineering endeavor. Often, simulation is the foremost method of understanding the intricacies of novel high-performance architectures, emerging technologies, and the interconnect topologies while gathering crucial information about energy consumption, network efficiency, and software execution. In parallel with the increased use of simulations, there has been a growth in the number of available models for specific applications, fomenting a growing urgency for interoperability, consistency, and communication between simulation tools and their developers. To ease the use and interoperability of larger simulated systems, a standard communication methodology between models is needed.

The tutorial will introduce, explain, and expand on several key concepts. SST will be introduced as a framework for modeling and simulation. After a brief introduction, key concepts such as the simulation environment and module creation will be discussed. An overview of the simulator framework will be presented, followed by an in-depth discussion of its features and their application. Using SST as a framework, the presenters plan to show:

* Intuitive framework. SST presents a clean, concise simulation framework. As a component-based event-driven parallel simulator, SST offers a lightweight interface with excellent scalability and customization options.
* Clear expandability. SST offers a modular interface to customize novel simulators, while keeping the back end free from interference. This ensures compatibility and allows for customization of hardware while easing scalability.
* Open source development. SST is developed with a mind for open source, utilizing the interests of the community to expand its growing base of components.

The presenters will also (1) explain the need for tools that enable repeatable experimentation and allows easy collaboration and, (2) introduce the new GPU component in SST.

Previous versions of this tutorial

This tutorial was presented in a half-day format during SC16 and a full-day session at ISCA in June 2017. Additionally, expanded versions of these tutorials have been presented during LPS hosted sessions on the University of Maryland’s campus. Similarly, parts of the tutorial have also been given in a graduate computer architecture course at the University of Pittsburgh and at Boise State University. The additions in this session are the GPU component and an improved component interface.

Sample Schedule

Session 1 (8:30-10:00am)

1. Modeling and Simulation Overview
   1. Tutorial roadmap
   2. Motivation
2. SST Overview
   1. Motivation, infrastructure, use cases
   2. Components, extensibility, open source
3. Hands-on Exercise: Introduction to SST
   1. Simulating a simple processor
      1. Configuration files & tools
      2. Running SST simulations

Session 2 (10:30am-12:00pm)

1. GPGPUSim Overview
2. Hands-on Exercise: Introduction to GPU Component