Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

Damian Dechev[1,2], Amruth Dakshinamurthy[1]

[1]Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816

[2]Scalable Computing R&D Department, Sandia National Laboratories, Livermore, CA 94551

`dechev@eecs.ucf.edu, amruth.rd@knights.ucf.edu`

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- Motivation
- Challenges
- Key Technology Components
- Technicalities
- Methodology
- Status of the Framework
- Experimental Setup
- Conclusions
- Future Work
- References

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Motivation**
  - Computing on a massive scale
    - Large Hadron Collider, Weather Forecasting, Research in Biology, Energy
    - Particle Accelerator in LHC generates more than 2 GB of data every ten seconds
  - Applications and algorithms need to keep up the pace
  - Predicting the most effective design of a multi-core exascale architecture
  - Optimizing and fine-tuning the software application to efficiently execute in such a highly concurrent environment.

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Need of the Hour?**
  - Hardware/Software co-design
  - Methodology depends on a bi-directional optimization of design parameters
  - Software requirements drive hardware design decisions and Hardware design constraints motivate changes in the software design
  - Need an environment to enable co-design practices to be applied to the design of future extreme-scale systems
  - Prototyping ideas for future programming models and software infrastructure for these machines

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Challenges**
  - Difficult to develop a software model based on a complex scientific HPC application
  - Such applications often include a large number of HPC computational methods and libraries, sophisticated communication and synchronization patterns, and architecture-specific optimizations
  - Difficult to analyze and predict the runtime statistics for domain-specific applications using heuristic algorithms

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Key Technology Components**
  - Structural Simulation Toolkit (SST) Simulator
  - ROSE Compiler
  - Software Skeleton Model of the Large-Scale Parallel Application

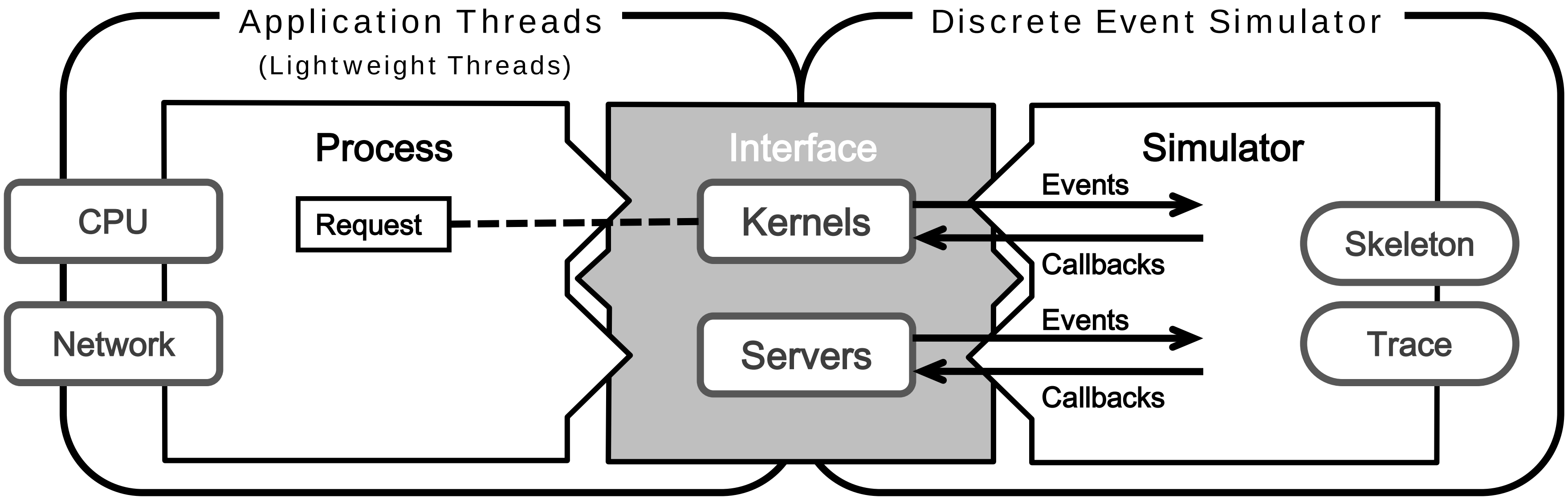
University of Central Florida
Sandia National Laboratories
Lawrence Livermore National Laboratory

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **SST Simulator**
  - **Structural Simulation Toolkit (SST) macro discrete event simulator is an open-source simulation framework offered by Sandia National Laboratories for the development of extreme-scale hardware and software**
  - **Enables the coarse-grained study of distributed memory applications in highly concurrent systems and is implemented in C++**
  - **Provides a parallel simulation environment based on MPI and, several network and processor models.**
  - **Enables to derive detailed information about the effect of various design parameters on execution time**
    - **memory bandwidth, latency, number of nodes, processors per node, network topology and other design parameters**

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **SST Simulator**

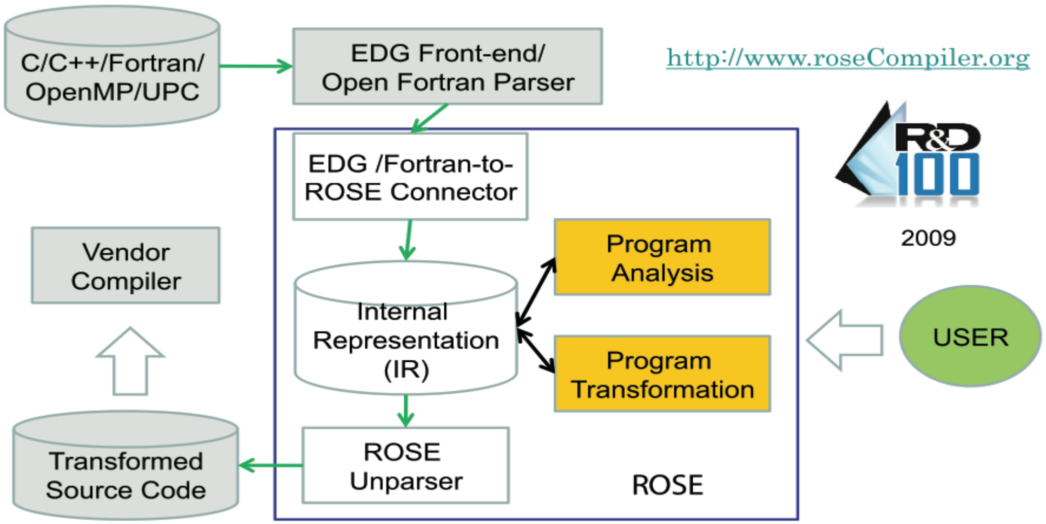


Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **SST Simulator**
  - Makes use of extremely lightweight application threads, allowing it to maintain simultaneous task counts ranging into the millions
  - Lightweight application threads perform MPI operations
  - Task threads create communication and compute kernels, then interact with the simulator's back-end by pushing kernels down to the interface layer
  - The interface layer generates simulation events and handles the scheduling of resulting events to the simulator back-end
  - The interface layer implements servers to manage the interaction with the network model in the context of the application
  - Supports two execution modes
    - Skeleton application execution
    - Trace-driven simulation mode
  - The processor layer receives callbacks when the kernels are completed

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **ROSE Compiler**
  - ROSE compiler is an open source-to-source compiler infrastructure for building a wide variety of customized analysis, optimization and transformation tools
  - Enables rapid development of source-to-source translators from C, C++, UPC, and Fortran codes to facilitate deep analysis of complex application codes and code transformations



Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **ROSE Compiler**
  - The front-end section contains tools for the construction of the AST
  - The mid-end section contains tools for the processing and transformation of the AST
    - Interfaces to operate on AST nodes
    - Traversal mechanism – Preorder and Postorder
    - Attributes mechanism
    - AST rewrite mechanism
    - Loop analysis and optimization
    - AST Merge mechanism
  - The back-end section contains tools for generating source code from the AST

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

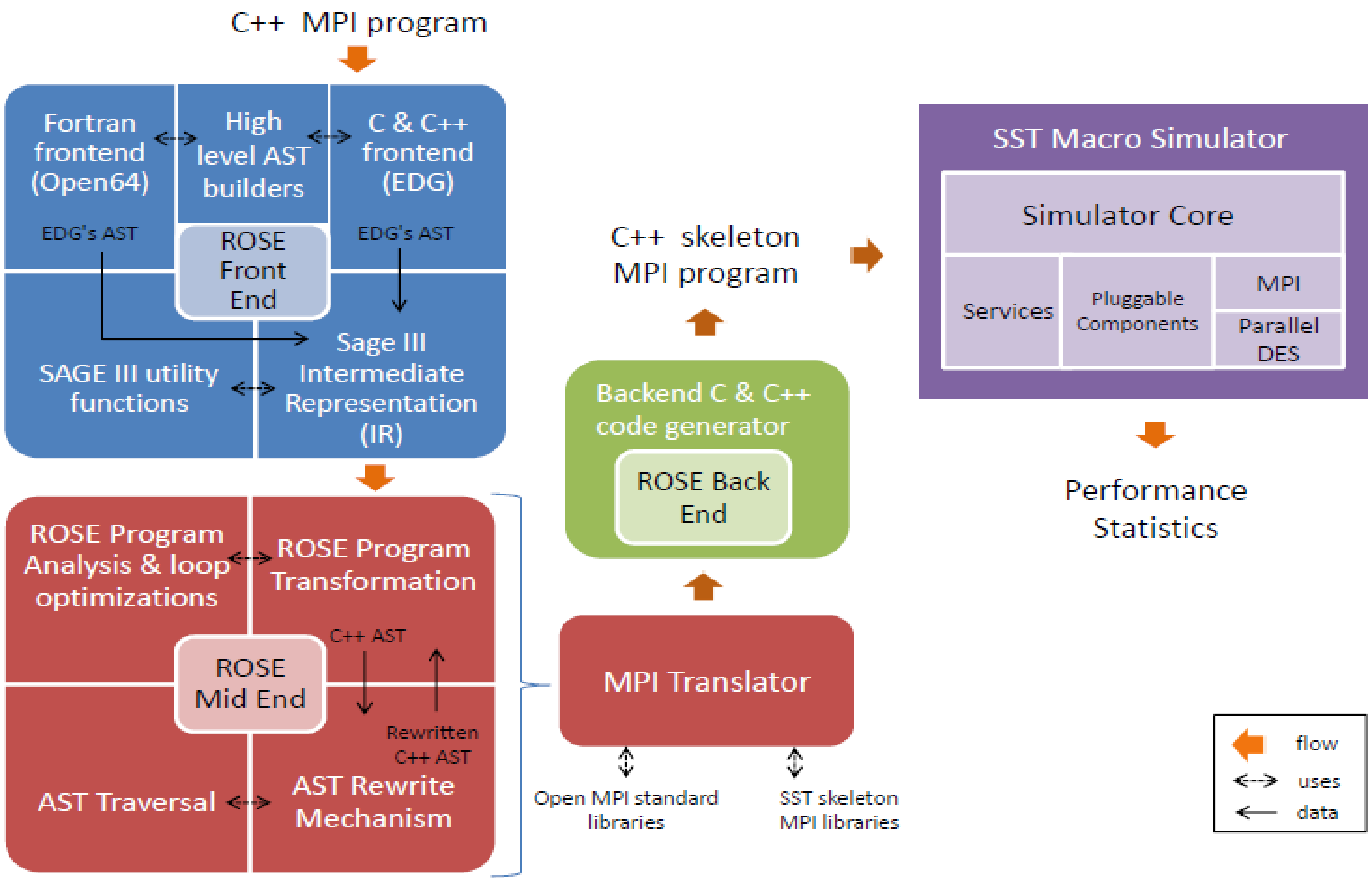
- **Software Skeleton Model**
  - Skeleton Model is an abstract form of the full scale application similar to the native MPI implementation with the exception of the syntax of the MPI calls
  - Captures control flow and communication patterns of an application
  - Constructed by removing parts of computations that do not affect the application's state and replacing them with system calls such as compute(…), which reduce simulation time dramatically
  - Can be executed on the SST/macro simulator for extreme-scale studies.
  - Much less expensive than running the full application
  - Allows the simulation of applications and architectures at levels of parallelism that are not obtainable by the most powerful supercomputers today

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Technicalities**
  - Need to match MPI expression patterns on the AST provided by the ROSE infrastructure to derive a backward slice
    - A backward slice is a collection of all program statements and expressions that affect a given point in the user code
  - Our slicing algorithm operates on the Program Dependence Graph
    - A PDG represents both data dependence edges and control dependence edges
  - Matching is done in the ROSE translator by making String comparisons of the input code statements within AST nodes to that of desired SST/macro statements and finally rewriting portions of the AST nodes
    - AST, a tree representation of the source code flow graph where each node denotes a construct occurring in the source code

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Methodology**



Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Methodology**
  - C++ full scale program utilizing standard MPI calls is fed to ROSE front end.
    - Jacobi Relaxation algorithm implemented in C++ using MPI
  - The front end parses the input source code and generates EDG's AST, the Edison Design Group's compiler intermediate representation
  - This AST is traversed internally to generate a new AST called Sage III Intermediate representation

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Methodology**
  - The new SAGE III AST is the input to our translator
  - The translator applies the program transformation and AST Rewrite/Traversal modules provided by ROSE
  - Using ROSE compiler extension modules: Slicing module and Transformation module
  - Slicing module performs program slicing by abstracting the application code and modeling only the relevant software components (MPI calls)
  - Operates on the Program Dependence Graphs to remove computation parts from the program

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Methodology**
  - Transformation module specifies the desired output based on the sliced-out Abstract Syntax Tree (AST)
  - Operates on the sliced-out AST to perform transformation of native MPI calls to the ones compatible for SST simulator
  - The backend C++ source generator uses this rewritten AST and unparses it to generate C++ source code
  - The skeleton program will only provide information about MPI calls and their associated argument lists

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Status of the Framework**
  - SST MPI declarations are inserted into native mpi.h header file in advance to generate SST MPI function identifiers
  - Avoids inserting SST MPI function symbols into symbol table during transformation phase
  - Most of the SST MPI calls return a type timestamp. So normal MPI functions calls are transformed into expressions
  - Argument reordering has been handled since method signatures of standard MPI call vary from those of SST MPI call
  - Currently, the framework handles MPI routines from Environment Management, Point-to-Point Communication and Collective Communication groups(there are few exceptions)
  - Slicing has been done using Program Dependence Graph

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Experimental Setup**
  - We are able to derive software skeleton models from C++ implementations of Jacobi Relaxation and Poisson's equation solutions
  - The derived skeleton model has to inherit from sstmac::mpiapi to gain access to the SST MPI interface and should implement the run method
  - SST simulator uses a simple C++ driver that sets up the simulation environment and runs the application
  - The driver constructs network topology and interconnect model.
  - The driver constructs an instance of the skeleton application and calls the run method to start the simulation
  - Analysis of derived Skeleton Models on SST simulator is planned

Automatic Extraction of Software Models for Exascale Hardware/Software Co-Design

- **Conclusions**
  - Addresses many issues concerning hardware/software co-design by allowing a feedback between hardware design and software development
  - Use of cycle accurate simulation tools provides the data and insights to estimate the performance impact on an HPC applications when it is subjected to certain architectural constraints
  - Allows for the evaluation and optimization of C++ applications using the MPI programming model
  - Allows us to employ new programming models and algorithms in the design of large-scale parallel applications

- **Future Work**
  - Our plan is to extract skeleton models for applications provided by Mantevo project https://software.sandia.gov/mantevo/
  - Mantevo applications are small self-contained proxies for real applications that embody essential performance characteristics
  - We plan to extend our work to include other programming models, in addition to MPI.
  - We plan to enhance our present framework to extract skeleton models for Fortran applications.

- References
  - ROSE Compiler, http://www.roseCompiler.org
  - Sandia's Computational Software Site, https://software.sandia.gov
  - Message Passing Interface Forum, http://www.mpi-forum.org/
  - SST: The Structural Simulation Toolkit http://sst.sandia.gov/
  - C. L. Janssen, H. Adalsteinsson, S. Cranford, J. P. Kenny, A. Pinar, D. A. Evensky, J. Mayo. A Simulator for Large-Scale Parallel Computer Architectures. International Journal of Distributed Systems and Technologies
  - Tae-Hyuk Ahn, Damian Dechev, Heshan Lin, Helgi Adalsteinsson and Curtis Janssen. Evaluating Performance Optimizations of Large-Scale Genomic Sequence Search Applications Using SST/macro. SIMULTECH 2011
  - Exascale Computing Software Study: Software Challenges in Extreme Scale Systems

Questions?