Rexss Scalability with SST/macro:

Analyzing the Performance of a Distributed Fault-Resilient Software Package

Kathryn Dahlgren, UC Santa Cruz, Ph.D. Computer Science, est. June 2020
Francesco Rizzi (8351)*, Karla Morris (8956)*, Bert Debusschere (8351)*
July 29, 2015
*Sandia National Laboratories, CA
Introduction

- **What is Rexssss?** [1]
  - Resilient Extreme-Scale Scientific Simulations [software]
  - PDE solver
  - Generalized Additive Schwarz
  - Divides iterative solution method into a number of smaller problems
  - Spreads the problems over isolated segments of hardware
  - Communication between the individual computation segments eventually converges to a solution (if the problem has a solution)

- **What is SPMD?** [2]
  - Single-Process, Multiple-Data
  - One of the two Rexssss solver implementations
  - Every MPI rank solves a portion of the PDE within a single subdomain
  - Combines the smaller solutions to find the final PDE solution
Introduction (cont’d)

- What is SST/macro?
  - Structural Simulation Toolkit with macroscale components
  - Simulates the performance of a given software on user-specified architectures
  - Supercomputer runs (on the cheap!)
    - Test different parameter and hardware configurations at-scale on smaller local machines
Introduction (cont’d)

- **Task:**
  - Develop a template emulator for the Rexssss PDE solver

- **Goals:**
  - Re-create known performance bottlenecks
  - Explore more optimal communication configurations
Introduction (cont’d)

- Rexsss SPMD Steps:
  - sampling() = \(\alpha\) seconds
  - buildBoundaryMaps() = \(\beta\) seconds
  - solveAndUpdate() = ???
  - computeRMSEError() = \(\gamma\) seconds
  - print() = \(\delta\) seconds
  - prepareNextIteration() = \(\epsilon\) seconds

- *Where \(\alpha, \beta, \gamma, \delta, \epsilon\) are finite positive rational numbers directly proportional to \(6.24 \times 10^{-6}\) by a positive integer greater than or equal to 1.

- *\(6.24 \times 10^{-6}\) is a semi-arbitrary constant representing a severe overestimate of the time required to perform a basic instruction.
Methods

1. Replace non-communication functionality with \textit{SST\_compute(\# seconds)} statements.

   Example:
   
   \begin{verbatim}
   for(int i = 0 ; i < someBoundaryFromTheOriginalCode ; i++)
   {
     // solution = performAComputation();
     // saveSolution(solution);
     SST_compute(2*0.00000624);
   }
   \end{verbatim}
Methods (cont’d)

2. Replace data sent between neighboring MPI ranks with objects simulating the memory requirements of the original data.
   - Boost MPI is currently used to serialize objects for MPI communications
   - SST/macro currently works best with standard MPI
   - Replace original Boost MPI objects with vectors of doubles supported by standard MPI
2. Implement communication patterns:

- **Base case:**
  - All MPI ranks send and receive a communication from 8 nearest neighbors.

- **Pattern 1: All-to-Root**
  - All ranks send 1 extra communication to root.

- **Pattern 2: Hierarchical**
  - 25% of the ranks act as regional authorities:
    - Receive 1 extra communication from all constituents.
    - Send 1 extra communication to root.

- **Pattern 3: Root-to-All**
  - Root sends a broadcast to all ranks.
Methods (cont’d)

- Test cases
  - Variables:
    - World size = 144, 576, 2304, 9216
    - Either 1, 10, or 50 iterations of each communication type
Results

Current Rexssss SPMD Weak Scaling

Weak Scaling SPMD Skeleton Per Communication Pattern

*Root-to-All 10x and 50x results are practically identical to corresponding All-to-Root results.
Conclusion

- Current skeleton communication patterns approximate known bottle-necks.

- A hierarchical communication scheme would improve SPMD communication efficiency.

- Tests with the SST/macro skeleton offer guidance for future investigations highlighting the costs associated with specific communication patterns.

- SST/macro is an effective tool for analyzing and optimizing software scalability.
Future work

- Refine the skeleton to predict Rexssss performance on NERSC more accurately:
  - Adjust the computation section estimates with actual execution times from profiling data.
  - Validate SST execution time predictions based upon actual NERSC execution times.

- Refine the skeleton to reveal how small changes to communication algorithms effect efficiency.
  - Rather than re-implementing the entire SPMD package with new communication algorithms.

- Determine which hardware and problem configurations optimize SPMD performance.
THANK YOU!

Bert Debusschere  Francesco Rizzi  Karla Morris
Questions?
References
