Rexsss PDE Solver Performance Analysis: Scalability Estimates Using the SST/macro Architecture Simulator

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

Abstract

The future of extreme-scale computing promises to magnify the impact of soft and silent faults as sources of inaccuracy in simulation and modeling solutions obtained from parallel distributed algorithms. The development of resilient computational tools represents an essential strategy for understanding the best methods for absorbing the impacts of soft and silent faults without sacrificing solution integrity. The Rexsss (Resilient Extreme-Scale Scientific Simulations) project pursues the development of fault resilient algorithms for solving PDEs on distributed systems [1]. Performance analyses using SST/macro (Structural Simulation Toolkit with maroscale components) reveal how current implementations of the algorithms scale to simulated NERSC systems without consuming tremendous numbers of CPU hours on NERSC.

Introduction

• The Rexsss scalability analysis focuses upon the SPMD (Single-Program, Multiple-Data) implementation.
• Computations for all solutions over a given subdomain occur within a single MPI process. Each MPI process is responsible for one subdomain [2].
• SST/macro simulates the performance of a given software on NERSC’s Edison system.
• Task:
  1. Develop a template SST/macro skeleton for Rexsss SPMD
• Goals:
  1. Re-create known performance bottle-necks
  2. Explore more optimal communication configurations
• Pursuing scalability analyses provides insight into how well the current algorithms satisfy intended applicability on extreme-scale architectures and identify better communication configurations.
• Results compare the SPMD skeleton performance to known performance behaviors.

Rexsss SPMD Steps:
- sampling() = α seconds
- buildBoundaryMaps() = β seconds
- solveAndUpdate() = ?? seconds
- computeRMSError() = γ seconds
- print() = δ seconds
- prepareNextIteration() = ε seconds

Methods

1. Replace all non-communication functionality with SST_compute(# seconds) statements.
2. Replace messages passing Boost objects with vectors of doubles simulating the memory requirements of the original data.
   a. SST/macro currently best supports standard MPI.
   b. Implement communication patterns:
      a. (base) All MPI ranks send & receive a communication from 8 nearest neighbors.
      b. (All-to-Root) All ranks send 1 extra communication to root.
      c. (Hierarchical) 25% of the ranks act as regional authorities (RA):
         o Receive 1 extra communication from all constituents.
         o Send 1 extra communication to root.
      d. (Root-to-All) Root sends a broadcast to all ranks.
3. Test case variables:
   a. World size = 144, 576, 2304, 9216
   b. Either 1, 10, or 50 iterations of each communication type

Results

Current Rexsss SPMD Weak Scaling:

Weak Scaling SPDST SST Skeleton:

*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.

References