Performance Optimizations and Modeling for Large-Scale Heterogeneous Computing Systems

Ashwin M. Aji, PhD Candidate, Virginia Tech (aqji@cs.vt.edu)
Advisor: Wu-chun Feng (feng@cs.vt.edu)

Performance Optimizations:
- How do we write optimal code for large CPU-GPU systems?
  - Within a GPU: Thread, memory and cache management
  - Across nodes: CPU-GPU co-scheduling
- Performance Modeling:
  - Have we reached the maximum achievable performance?

Performance Modeling:
- Across nodes: Lower bound (e.g., Amdahl's law and iso-efficiency metric)
- Within a node: Scheduling

Case Study: Epidemiology Simulation

Deaths in 2002:
- From injuries: 5.2 x 10^6
- From communicable diseases: 18.4 x 10^6

Are we prepared?
- Can we handle a moderate outbreak?
- Not enough vaccine or production capacity to immunize the population
- Logistics – Who should get the limited supplies? How to get the supplies to everyone?

Solution: EpiSimdemics (Barrett et. al., Supercomputing 2008)

MPI-ACC
- Extension of MPI for accelerators (GPUs)
- Productivity Goal
  - Extend the rich data transfer interface of MPI for CUDA, OpenCL and other GPU models
- Performance Goal
  - Pipeline the data movement between device, host and the network using architecture-specific enhancements
    - NVIDIA's GPU Direct
  - Future architectures:
    - Zero-copy data movement if accelerators have direct network access

GPU Communication and Memory
- MPI Send/Receive (CPU only)
- GPU Send/Receive (CUDA only)
- Network

Goal of MPI-ACC
- Lower Bound:
  - MPI_Send(s_acc_buf, ...)
  - MPI_Recv(c_acc_buf, ...)

GPU-GPU Latency Benchmarks
- Manual Blocking
- Lower Bound

Future Work
- Performance Optimizations
  - Extend MPI-ACC to enable CPU-GPU co-scheduling within a node
- Performance Modeling
  - Extend CampProf to other GPU architectures and any kernel
  - Provide a generic model for CPU-GPU task distribution