Simulating the performance of large-scale architectures on high-performance computers

Ian Jones

Oak Ridge National Laboratory, 2010
● Work carried out on XSIM, an HPC simulator at ORNL
● Useful in investigating performance and scalability of applications when run on HPCs
● Several existing simulators include JCAS, BigSim and MuPi
Aims/Objectives:

- Implement network model
- Implement path-finding for different topologies
- Account for message size and bandwidth
- Enable user specific customisation
- Implement fault tolerance and injection
Network class:

- Stores information about the topology by accepting and parsing user arguments
- Analyses MPI message source and destination and calculates the latency due to time taken to traverse network
- Analyses MPI message size and calculates the latency due to bandwidth
- Latency is calculated mathematically. Topology designs for: star, ring, mesh, torus, twisted torus, tree

- Discriminates cores which are on the same/different processors, by passing appropriate arguments.
– **Initial function extracts parameters from argument and validates**

– **Primary function called every MPI_Receive**
  
  - Identifies network/processor rank
  - Switch statement identifies network type
  - *Appropriate function called to calculate latency*
  - Primary function factors in correct bandwidth
  - Result returned and added to message time
Implementation 2/4

- Type-specific function requires appropriate arguments and source/dest rank
  - Star: $2 \times$ network latency multiplier
  - Ring: Absolute difference between source/dest but may vary depending upon loop-around
  - Tree: Source ranks recursively divided by degree to find common ancestor
Implementation 3/4

- Mesh: Breaks down ranks into Euclidean co-ordinates, to determine the network location
- Latency of route is calculated by summing absolute differences of the individual co-ordinates of source and destination
Implementation 4/4

- Torus: Same as mesh except dimensions have possibility of wrapping around
- Twisted Torus: Tests every single dimension both ways and takes the 'best' option, then repeats until destination found
Testing 1/3

- General Performance Overview
Testing 2/3

- Variable Tuning
- Hybrid Topologies

![Graph 1: Average Latency with Nested Topologies](image1.png)

![Graph 2: Average Latency with Twisted Torus-Nested Topologies](image2.png)
Limitations and Critique

- Twisted torus algorithm is not 100% accurate or bug-free in all situations, problems with implementation
- No accounting for traffic, congestion and any subsequent re-routing of messages
- Fault injection and fault tolerance not implemented or tested
- No variation of parameters, whole network uses a standard defined by user
Future Work

- Implementation of overlay networks and translation onto virtual network (broadcast)
- Possible conversion to data structure method to track exact path of messages and allow for upgrade for purposes of fault injection, congestion ID and message re-routing
- More in-depth testing of hybrid topologies
- Optimistic PDES implementation, extended MPI support, performance metric gathering
Thankyou

Questions?