FatTreeSim: Modeling Large-scale Fat-Tree Networks for HPC Systems and Data Centers Using Parallel & Discrete Event Simulation

Ning Liu, Adnan Haider, Xian-He Sun, Dong (Kevin) Jin

Outline

- Why do we choose to model fat-tree networks?
  - Introduction/Motivation

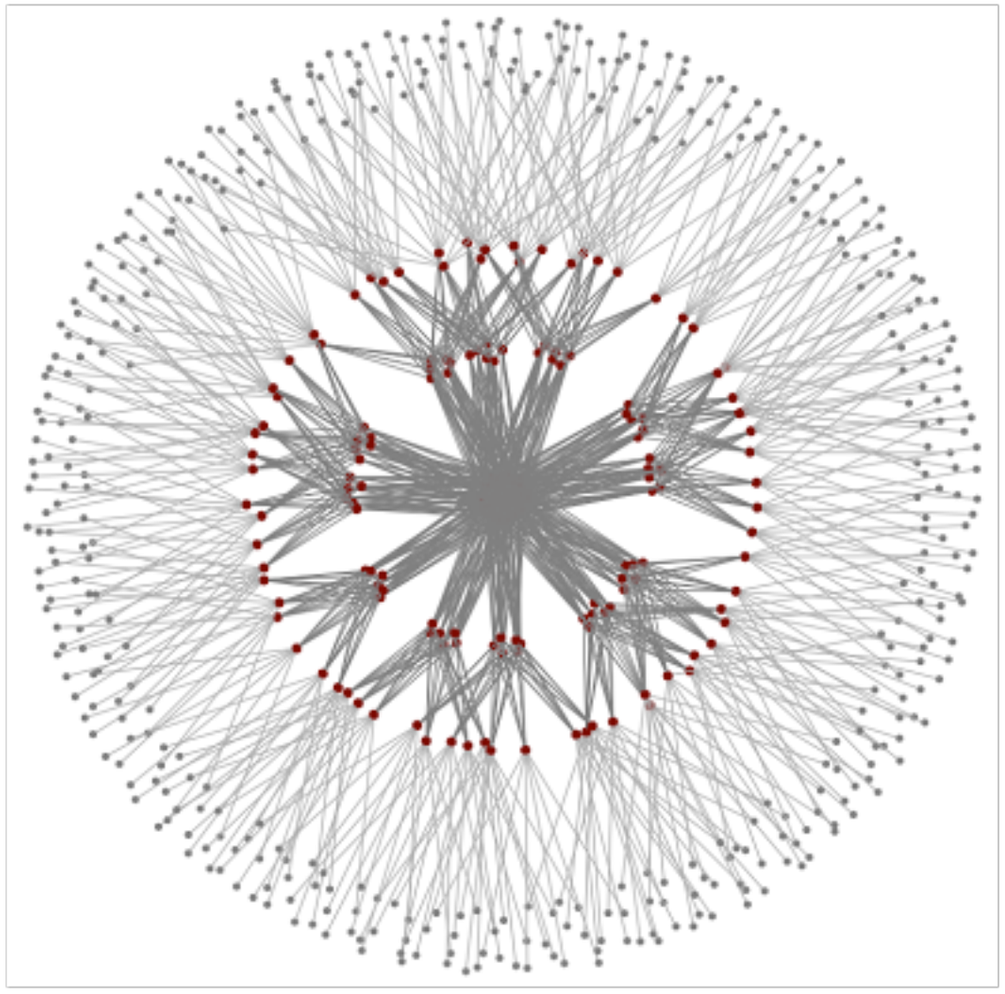
- How do we design and implement FatTreeSim?
  - Design/Implementation

- How do we evaluate the system?
  - Evaluation/Conclusion

Introduction

- Fat-tree networks
  - Invented by Charles E. Leiserson of MIT
  - Widely used in Datacenters
  - Will be used in next generation supercomputers.

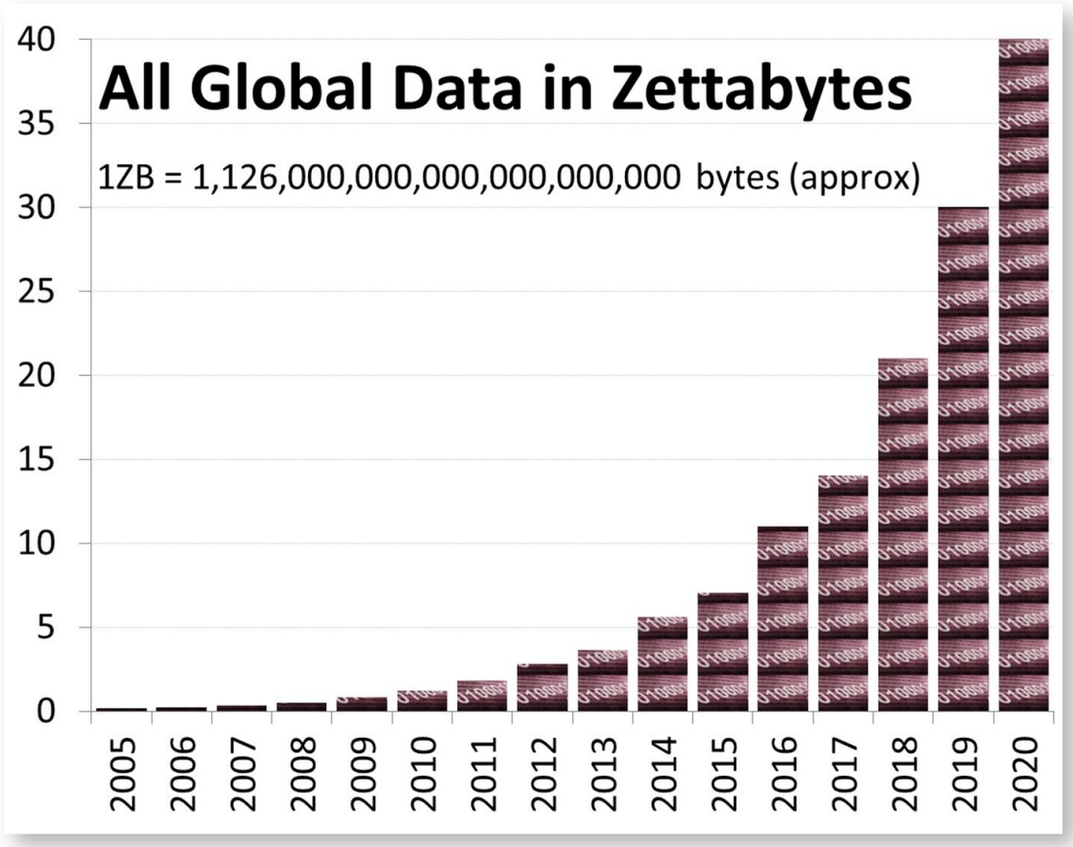
- Many issues rises as fat-tree network grows to extreme-scale
  - scalability/fault tolerance/load balance etc.



3-level fat-tree · 432 servers, 180 switches, degree 12
[1] https://reproducingnetworkresearch.wordpress.com/2012/06/04/jellyfish-vs-fat-tree/

Motivation

- Big data
  - Most data are stored and processed in datacenters
  - Most traffic (75%) is internal traffic
- There is a pressing need to understand the performance of fat-tree network at scale
- Redesign the architecture and algorithms



Global data growth
[2] http://www1.unece.org/stat/platform/display/msis/Big+Data

Motivation cont'd

- Next generation supercomputers: OLCF SUMMIT
  - A collaboration between OLCF, IBM, Mellanox and NIVIDIA
  - An investment of over 300 million dollars
  - Adopt fat-tree as the interconnection network provided by Mellanox
  - FatTreeSim can assist in evaluating the network performance, serve as as the platform for building app models

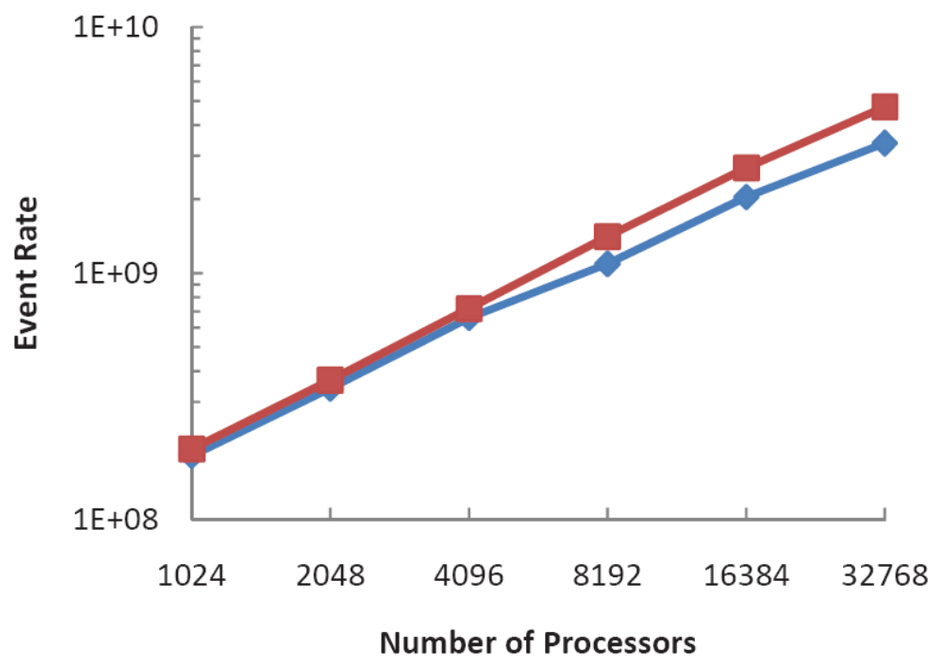
Why do we build FatTreeSim?

- Support the design of datacenters and HPC systems
  - Understand the design constraints and trade-offs
  - Characterize the challenges to the scalability of extreme-scale system
  - Explore various possibilities at extreme-scale in a time and budget efficient manner

- Support the design of parallel & distributed applications
  - Predict and optimize the performance at extreme-scale
  - Qualitatively analyze the interactions between system software and hardware and the impact on applications

Background: ROSS

- ROSS: **R**ensselaer **O**ptimistic **S**imulation **S**ystem
  - Designed in C, the interface is lean
  - Features optimistic simulation using reverse computation
  - Runs on supercomputers like ALCF Blue Gene series
  - Used by many other projects

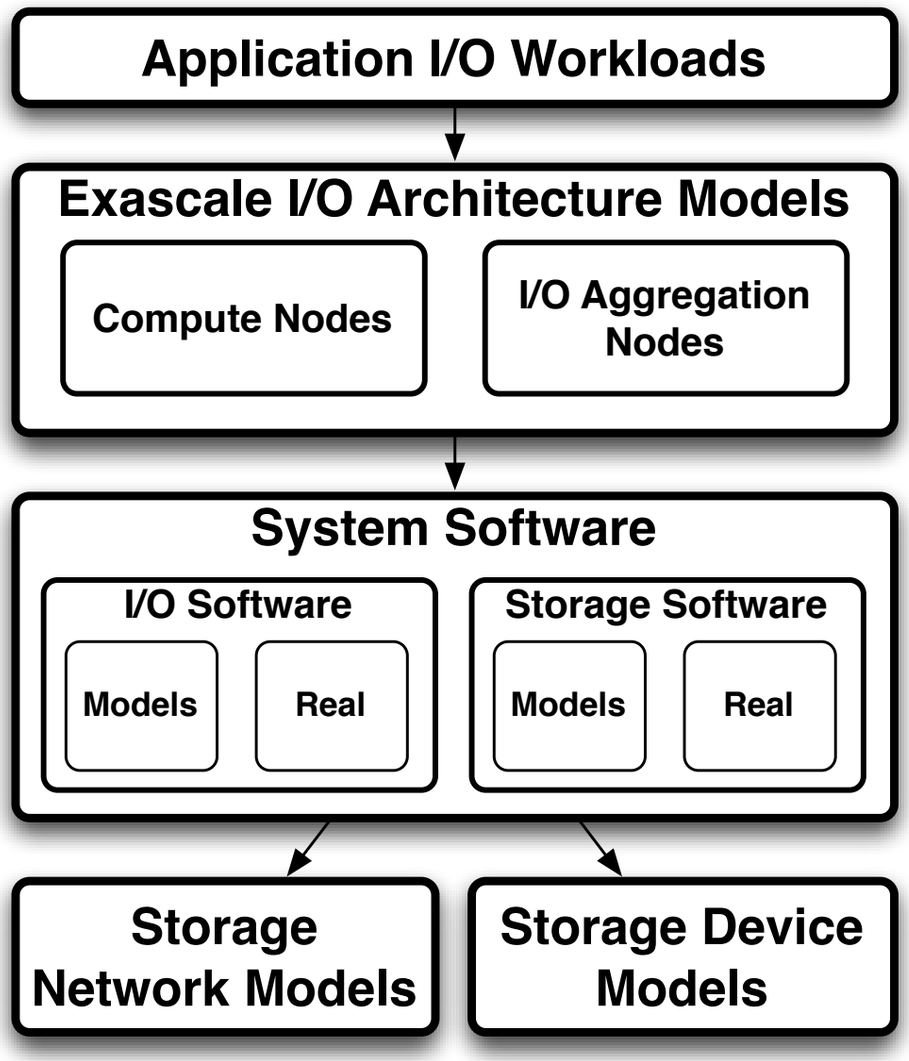


Background: CODES

- CODES: Enabling **Co-D**esign of Multilayer **E**xascale **S**torage Architectures

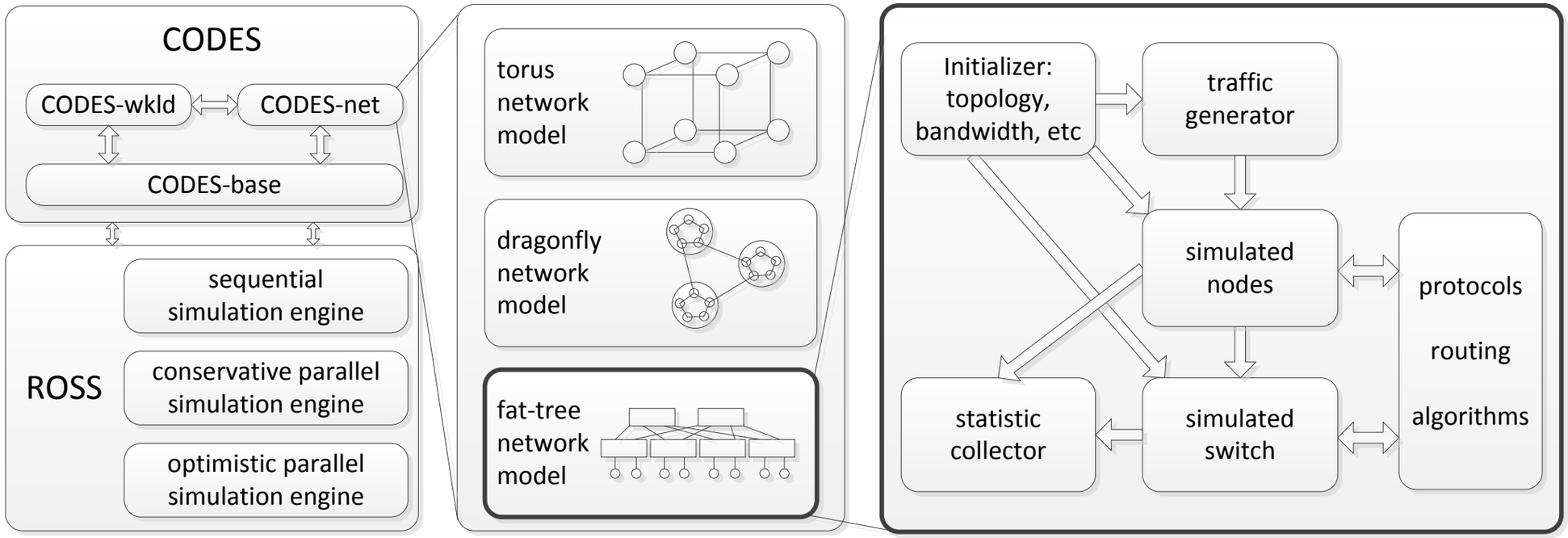
- CODES Goal:
  - Develop a simulation framework for evaluating exascale storage design challenges

- CODES components:
  - CODES-net/CODES-wkld/ CODES-lsm/CODES-base



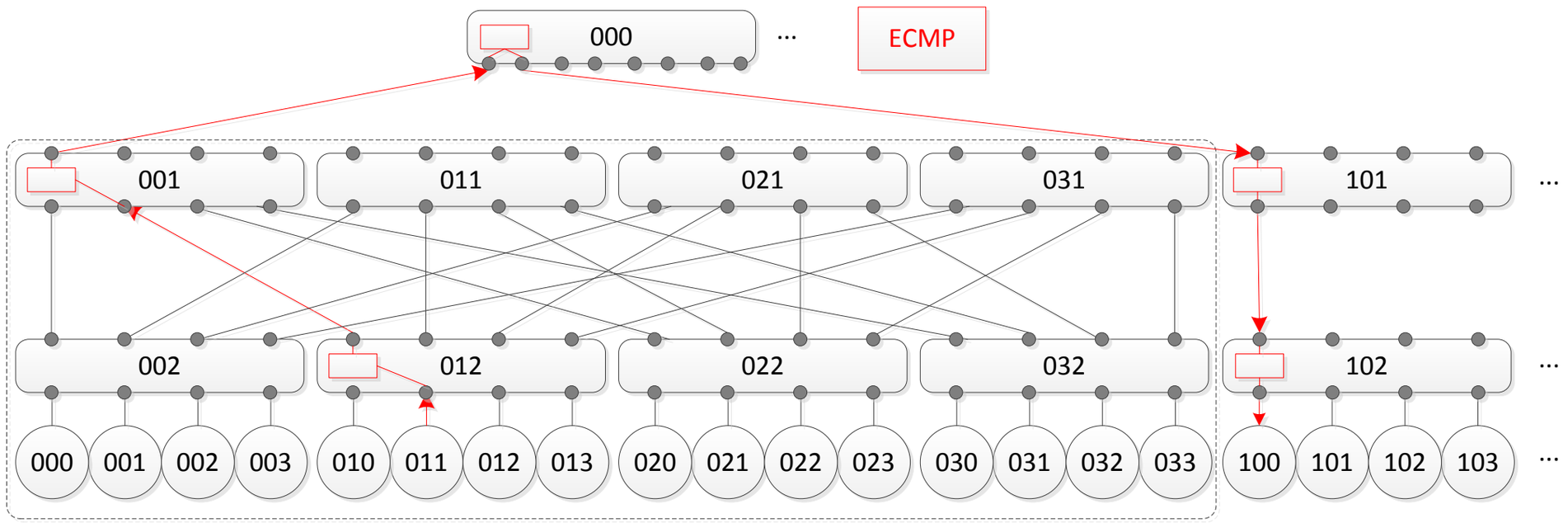
FatTreeSim in CODES

- CODES is built on ROSS
  - Leverage the parallel simulation engine and other functionalities
- FatTreeSim
  - Is a part of CODES-net and in parallel with other network modules



Design

- Use LPs to model switches and servers
- Use events to model packets flow
- Implement ECMP in switch LP



Selected Procedure

- We use different procedures to model system behaviors in fat-tree networks

- We use random destination and nearest neighbor to represent a variety of traffic patterns in datacenters and supercomputer

procedure GT ▷ generate packet stream
  t =processing delay
  $\tau = rng(I)$
  if RandomDestinationTraffic **then**
    $dst = rng(maxnodeID)$
    Generate packet (header contains dst)
  else if NearestNeighborTraffic **then**
    $dst = neighborID$
    Generate packet (header contains dst)
  else
    Unsupported traffic
  end if
  Call NSP procedure with t
  Call GT procedure with τ
end procedure

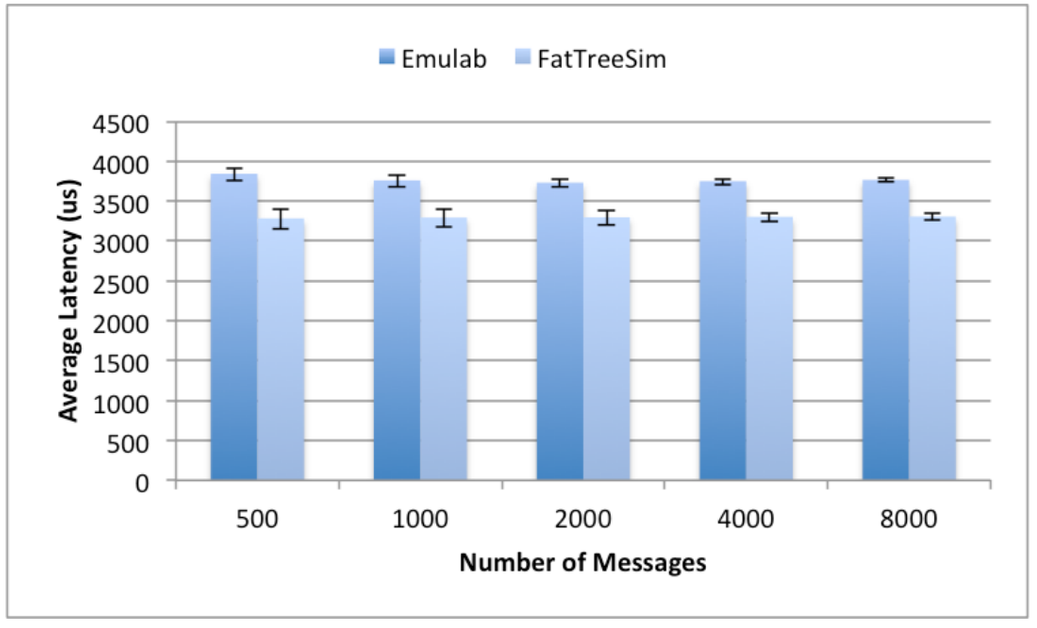
Emulab

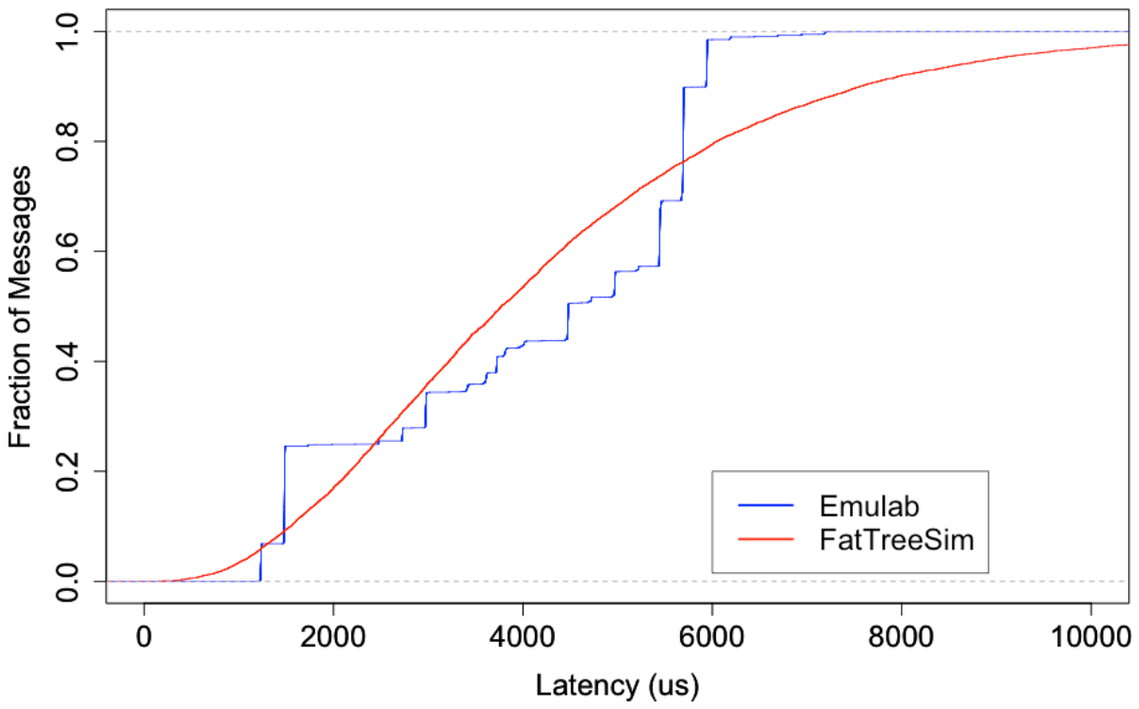
- Emulab is a network testbed, giving researchers a wide range of environments in which to develop, debug, and evaluate their systems.

- An emulated experiment allows you to specify an arbitrary network topology, giving you a controllable, predictable, and repeatable environment, including PC nodes on which you have full "root" access, running an operating system of your choice.

Evaluation on Emulab

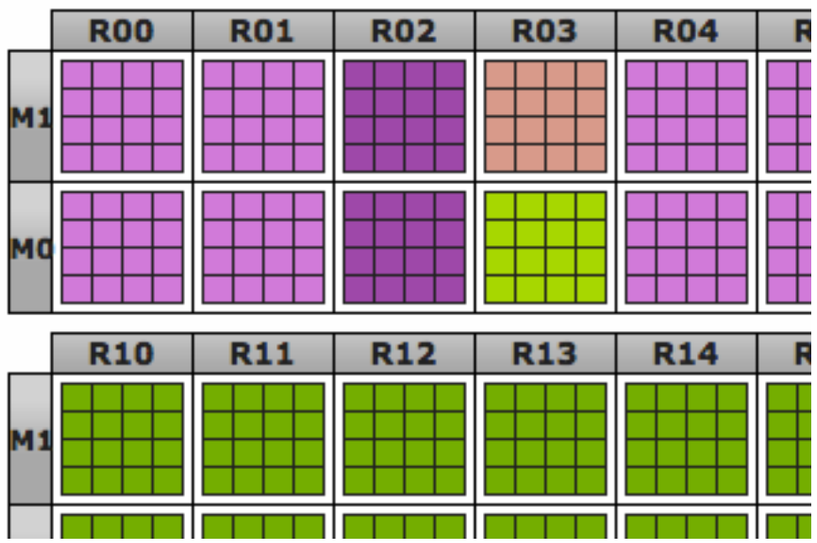
- Traffic pattern is random destination and nearest neighbor.
- Configuration is 4-port 2 tree, 4-port 3-tree, and 8-port 3-tree.



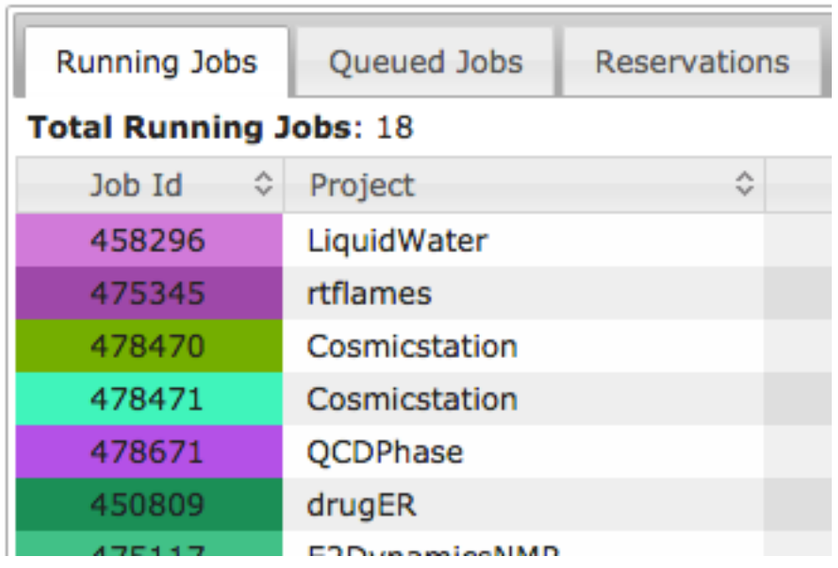


Blue Gene/Q: Mira

- Facts about Mira:
  - DOE supercomputer located at Argonne National Lab, Chicago
  - Mira ranks 5th as of Nov. 2014 in the top 500 list
  - Deliver a peak rate of 10 PFlop/s
  - Total number of cores is 0.78 million

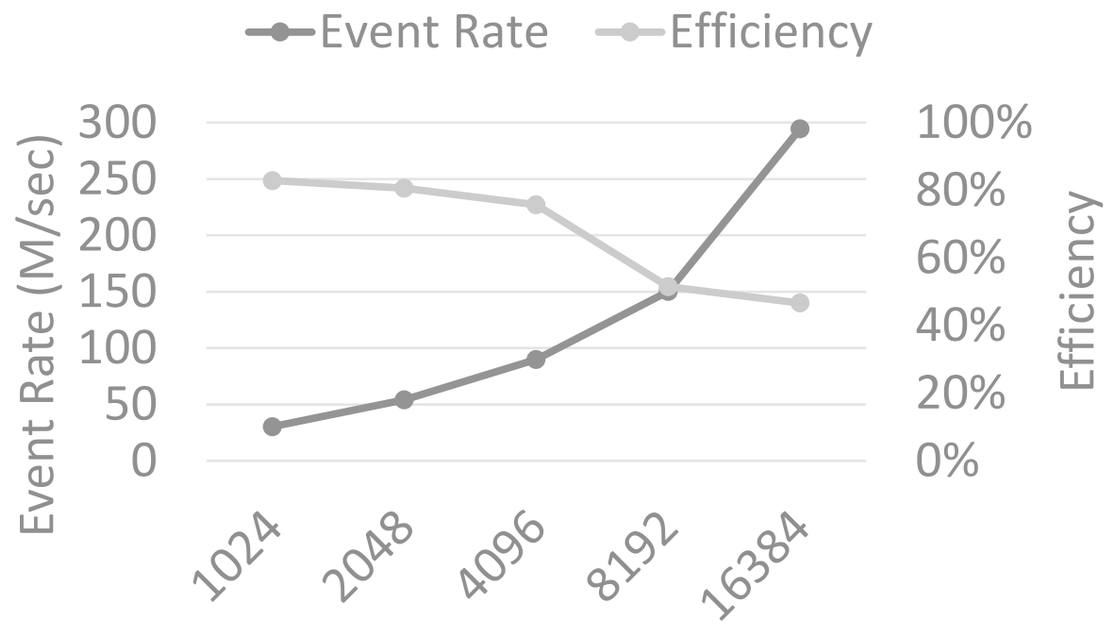


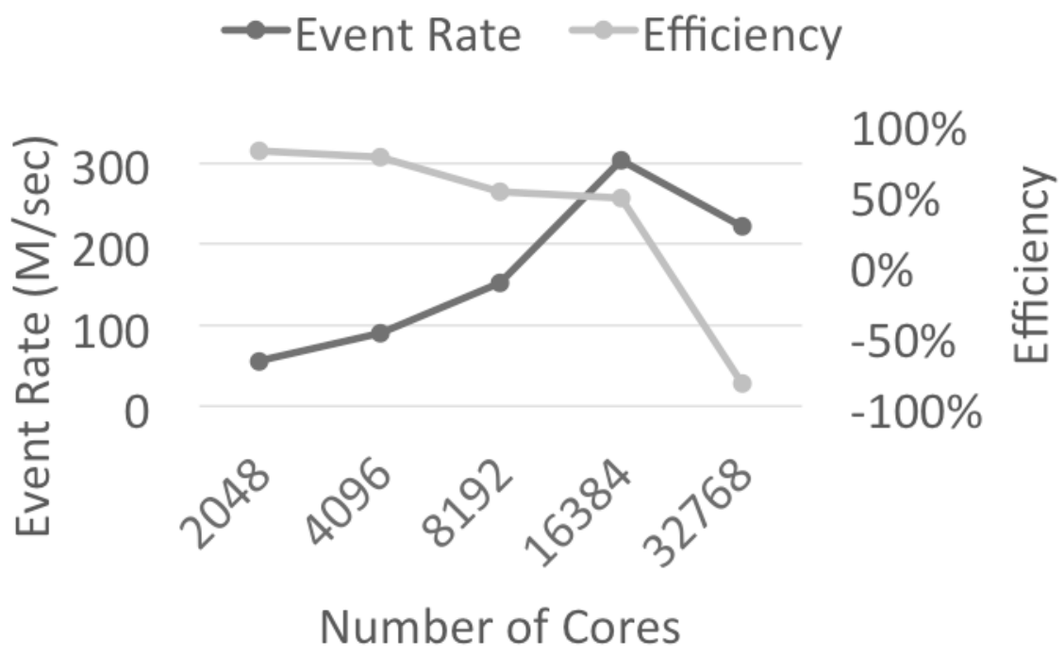
- Run FatTreeSim with Mira:
  - Both ROSS and CODES can run on BG series supercomputers
  - Scalability and load balance are our concerns



Evaluation on BG/Q

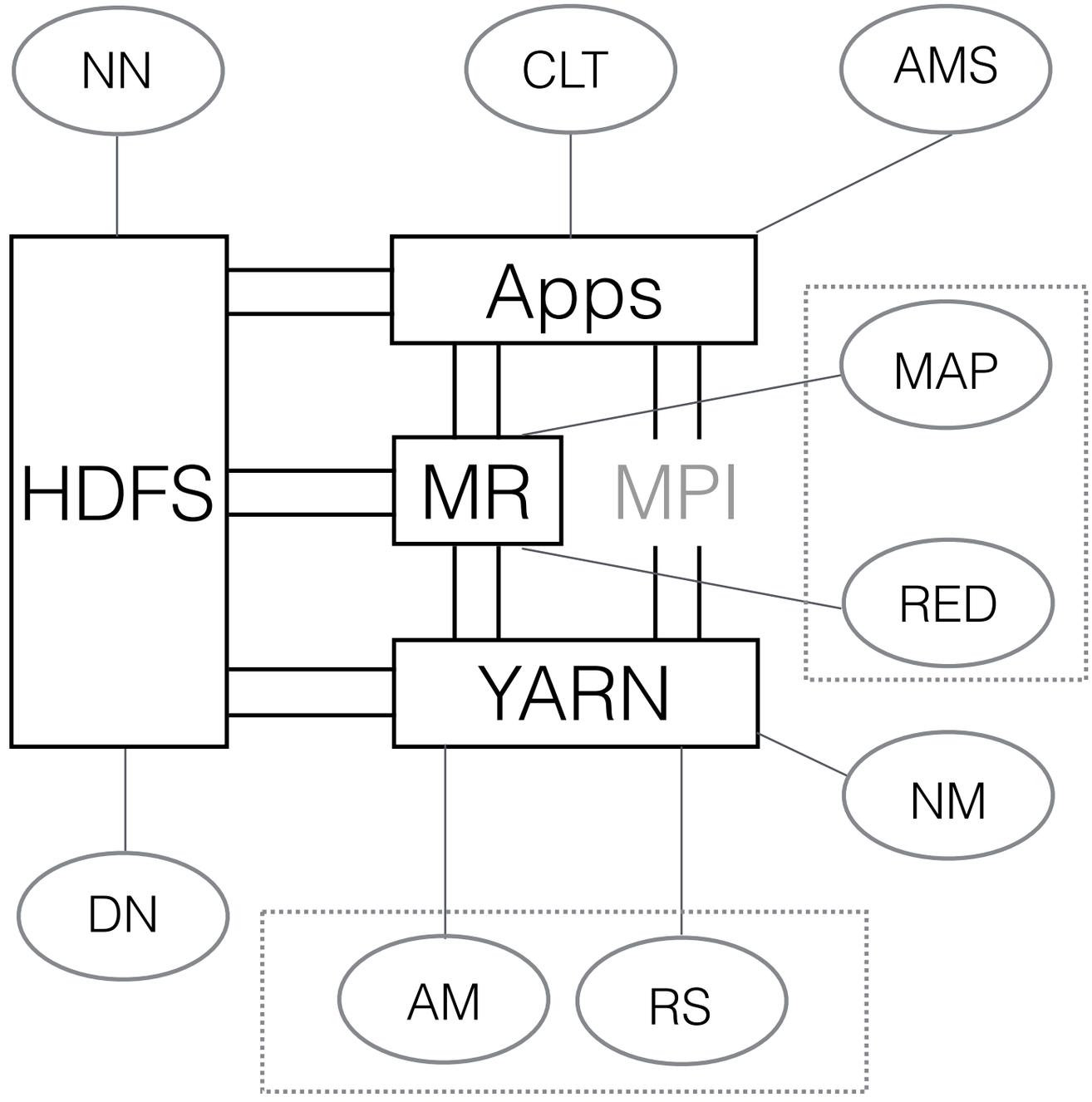
- Traffic pattern is random destination. Packet arrival rate is 1600 ns.
- Demonstrate near linear scalability in c8 mode, and observe a performance drop in 16K cores in c16 mode.





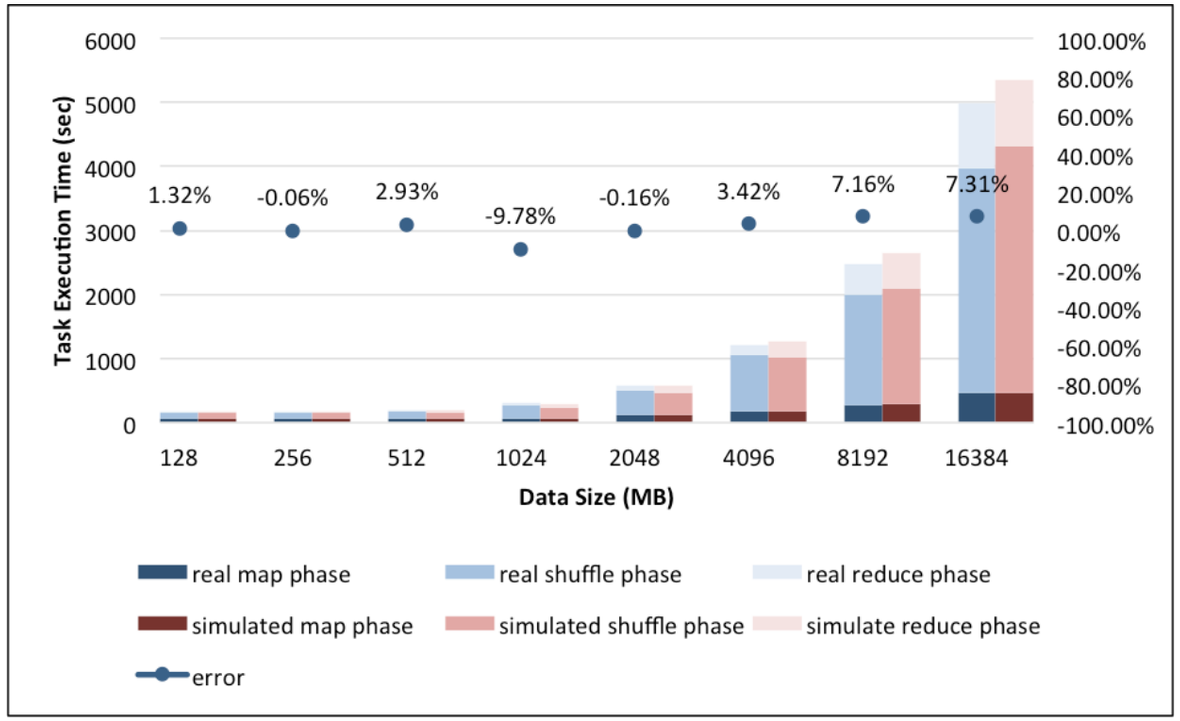
YARNsim

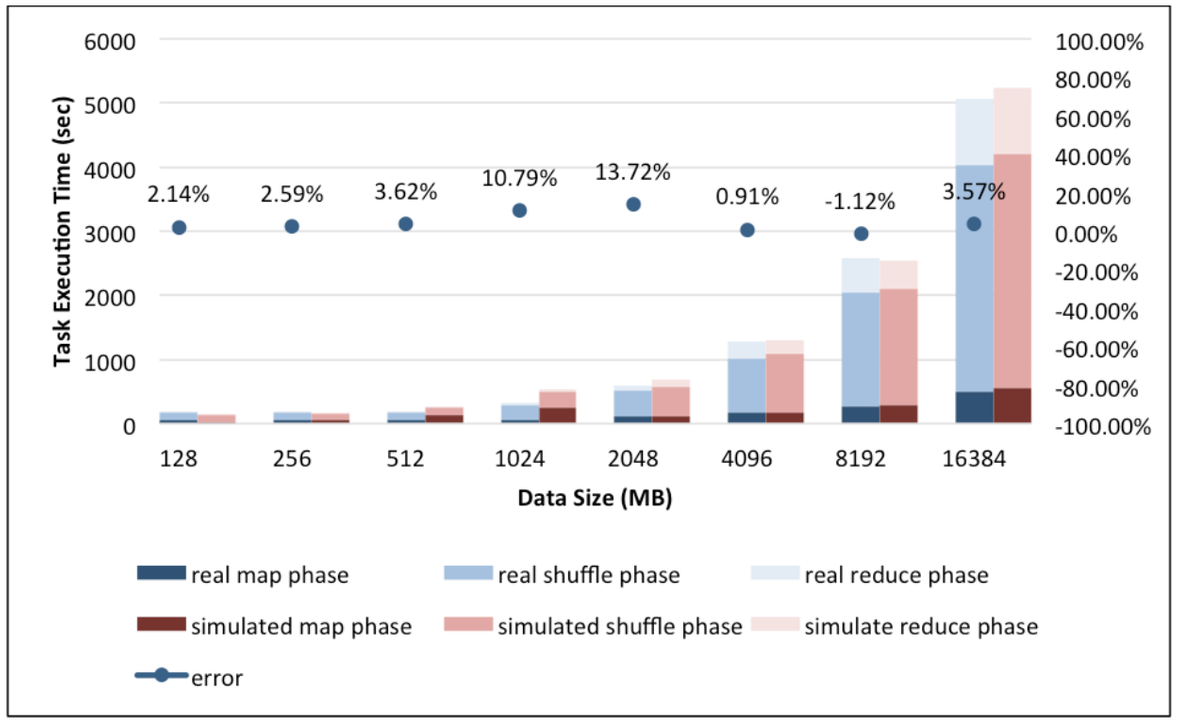
- A simulation system for Hadoop YARN
- Still in development
- Can simulate basic Hadoop and HDFS services
- Paper published in CCGrid 2015



Evaluation on YARNsim

- Demonstrate FatTreeSim can be used by YARNsim
- Hadoop benchmarks: Wordcount and Terasort
- Achieve good accuracy for basic benchmark tests





Conclusion and Future work

- FatTreeSim accomplished goals:
  - It serves as one CODES network module
  - It is accurate as verified in Emulab using real traffic
  - It scales to 32K cores on ALCF BG/Q system, peak event-rate is 305 M/s, total nodes is 0.5 million
  - It is accurate as verified in YARNsim system using Hadoop benchmarks and a bio-application

- FatTreeSim to-dos:
  - test dynamic routing algorithms, e.g. Hedera
  - model large-scale datacenter using FatTreeSim
  - model large-scale Hadoop applications and explore them using FatTreeSim

Acknowledgment

- Many thanks to
  - Dr. Christopher Carothers — Rensselaer Polytechnic Institute
  - Dr. Jonathan Jenkins — Argonne National Laboratory
  - Dr. Misbah Mubarak — Argonne National Laboratory
  - Dr. Robert Ross — Argonne National Laboratory