Predicting Performance Characteristics of Distributed Simulator for Scale-free Networks

Christopher L. Hood, George F. Riley

Georgia Institute of Technology

13 May 2015

Introduction

Long run time $\Longleftrightarrow$ $\boxed{\textbf{Large-scale}}$ $\Longleftrightarrow$ **Parallel simulation**

- Parallel and distributed simulations maximize usage of available computational resources in an effort to minimize total run time.
- We only care about distributing computation when the run time is going to be significant, i.e. large-scale simulations.
- Can we use information about the model to infer the best way to set-up the parallel simulation?

Terminology

Simulation model

- `ns3::Node`
- `ns3::Channel`
- `ns3::NetDevice`
- etc.

Simulator executive

- `ns3::SimulatorImpl` and children
- `ns3::Scheduler` and children
- etc.

"Glues" the two together: `ns3::Event`.

Description of Problem

Given a description of an experiment and a history of previous experiments and their associated performance metrics (run time, memory usage, energy consumption, etc.), can we accurately **predict** how to set-up the simulator executive in order to achieve the best performance?

i.e. can we form a **performance characterization** for how to set-up the simulator executive based on the model configuration?

Synchronization Algorithms in ns-3

- Event causality constraint $\implies$ synchronization.
- Optimistic vs. $\boxed{\text{conservative}}$ synchronization.
- Synchronous (Allgather) vs. asynchronous (null-message).
- Naïve characterization: do not take into account how the synchronization actually works.

Ways We Can Set-up the Executive in ns-3

- $\boxed{\text{Allgather vs. null-message.}}$
- Different scheduling schemes (not considered).
- "Intelligent" automatic segregation of model components across LPs (not currently viable in ns-3).

Simulation Meta-data in ns-3

- `ns3::MpiStatsAggregator` to assist in gathering simulation meta-data, e.g. various wallclock intervals, number of events scheduled, number of messages sent, etc.
  - `::AddClock (clkName, clockType)`
  - `::{Reset,Start,End,Pause}Clock (clock)`
  - `::{Add,Set,Get}Stat (...)`
  - Leverages `ns3::GlobalValue` so values are accessible outside of the `mpi` module.
  - Leverages MPI windowed communications for asynchronous access of values from any LP.
- Abstract a `ns3::WallClock` class:
  - `ns3::SystemWallClockMs` uses system clock.
  - `ns3::TscWallClock` leverages x86 CPU clock counter.

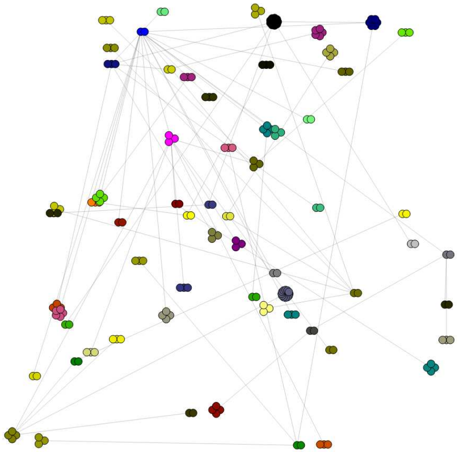
Learning a Performance Characterization

- Problem:
  - Many ways to set-up the executive (many independent variables).
  - Infinite number of ways to configure the model.
  - The model configurations we care about (large-scale models) take a long time to simulate.
- Solution: constrain both the number of independent variables and model configuration and see if we can find a characterization then.
  - Independent variable: synchronization algorithm $\in$ {Allgather, null-message}.
  - Model configuration: constrain topology, network size, traffic, etc.

Experimental Set-up

- Number of router nodes, $N_R \in \{64, 128, 256, 512, 1024, 4096, 8192, 65536\}$.
- Number of LPs, $N_A \in \{64, 128, 256, 512, 1024\}$.
- Mean link delay, $E[\ell] \in \{1, 10, 100\}$ ms.
- Traffic: on-off UDP traffic bursting at 50% duty cycle.
- Random topology generation using BRITE.

Example Topology—64 router nodes, 64 LPs



Running the Experiments

- Using the different combinations of number of routers, LPs and different mean link delays, as well as using different random seeds for the topology, 450 unique model configurations created.
- For each model configuration, experiment run using both Allgather and null-message synchronization, total of 900 different experiments.
- Wallclock run time measured for each experiment.
- Run on LLNL's Cab, Intel Xeon 2.6 GHz, 16 cores and 32 GB memory per node.

Recast as a Classification Problem

- For each pair of experiments corresponding to a unique model configuration (there are 450), there are two components:
  1. The data vector $\mathbf{x}_i$ consisting of features (simulation meta-data).
  2. The label z_i, which is 1 if the null-message algorithm produced a faster run-time than Allgather and -1 if the opposite.
- Then, the goal become thus: given pairs of (x_i, z_i) for i in some subset (the training data set), and given only the x_j for j in a non-intersecting subset (the testing data set), predict what z_j is (compute the prediction $\hat{z}_j$).
- Simply used random permutations (half and half) of the 450 data points as the training and testing data sets.

Recast as a Classification Problem

- In this work, the following features are used:

$$\mathbf{x}_i = \begin{bmatrix} \text{number LPs} \\ \text{number routers} \\ \text{lookahead mean} \\ \text{lookahead std.} \\ \text{lookahead min.} \\ \text{lookahead max.} \\ \text{outdegree mean} \\ \text{outdegree std.} \\ \text{outdegree max.} \end{bmatrix} \in \mathbb{R}^9$$

- Note the following definitions:
  - outdegree: number of edges from a router to other routers
  - mean, std., min., and max.,: the sample mean, standard deviation, minimum, and maximum of a certain measurement over all LPs in an experiment.

Performance Characterization

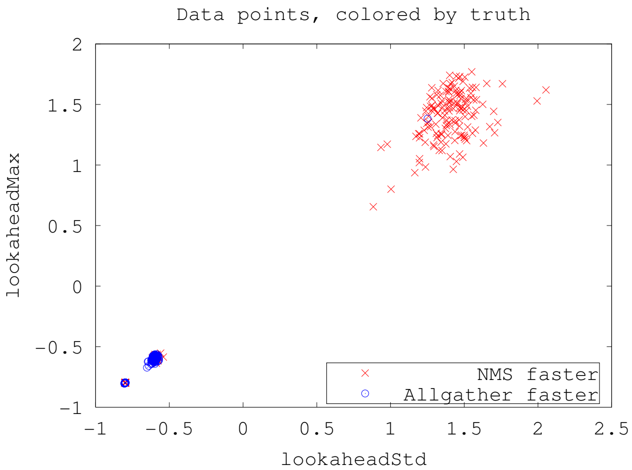
How to compute the estimate $\hat{z}_j$ given $\mathbf{x}_j$ for j in the testing data set?

$\Longrightarrow$ simplest to use a linear classifier:

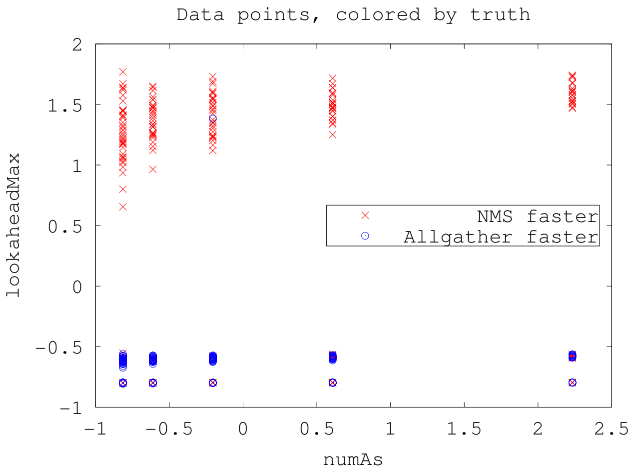
$$\hat{z}_j = \mathrm{sign}(\mathbf{w}^T \mathbf{x}_j + b)$$

where $\mathbf{w}$ and b are determined from the training data.

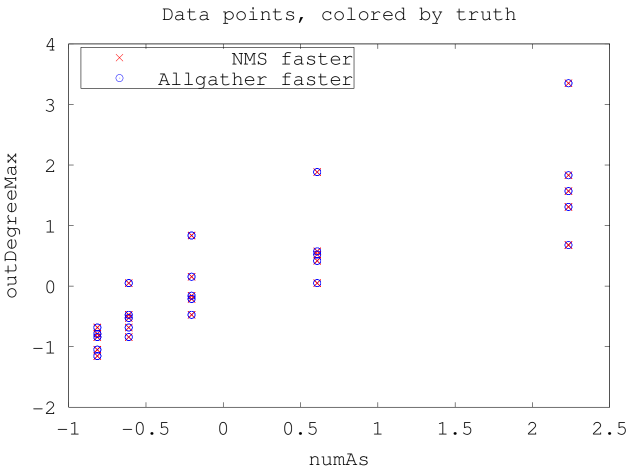
Is There an Apparent Pattern?



Is There an Apparent Pattern?



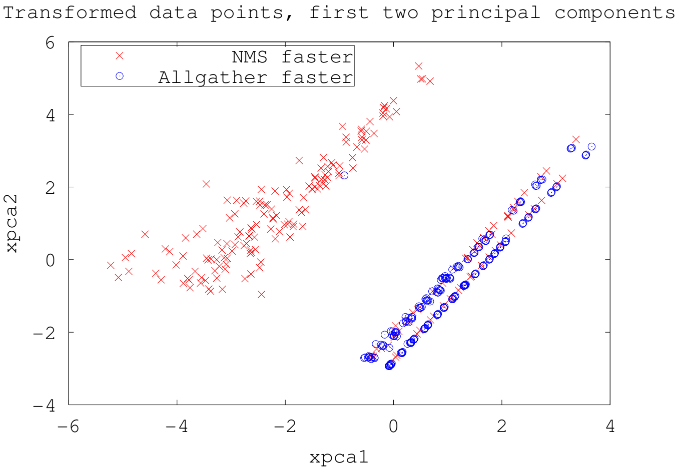
Is There an Apparent Pattern?



Principal Components Analysis

- Given the training data feature vectors $\mathbf{x}_i$, apply a linear transformation
$$\tilde{\mathbf{x}}_i = P\mathbf{x}_i$$
so that the first feature of $\tilde{\mathbf{x}}_i$ represents the highest variance, the second feature the second-highest variance whilst being orthogonal to the first feature, etc.
- Consequences:
    - Each feature of $\tilde{\mathbf{x}}_i$ is uncorrelated with the others.
    - Provides a principled way of dimensionality reduction, which helps prevent over-fitting.

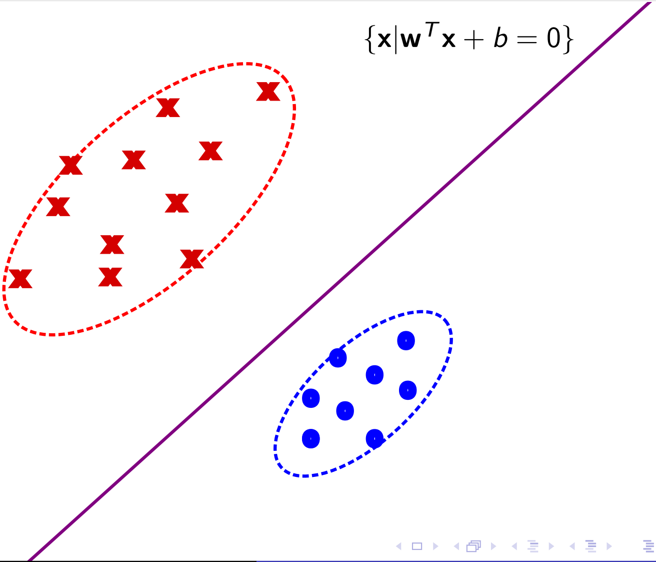
Is There an Apparent Pattern?



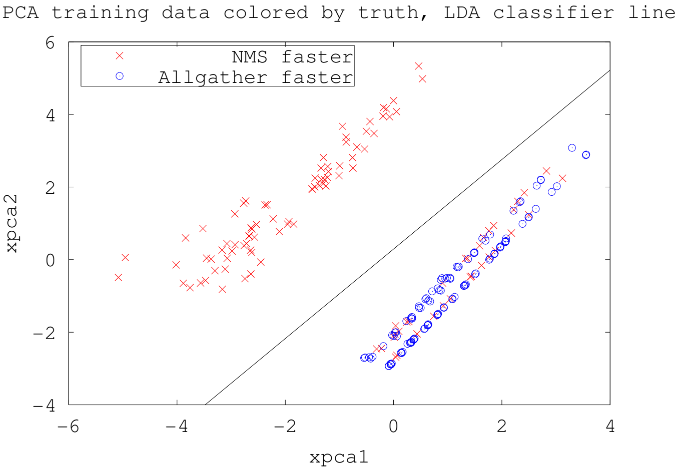
Performance Characterization

- Simple but principled way to compute $\mathbf{w}$ and b from the training data: linear discriminant analysis.
- Assume the training data points corresponding to each label come from Gaussian distributions of identical covariance.
- Compute $\mathbf{w}$ and b by constructing hyperplanes of equal likelihood.

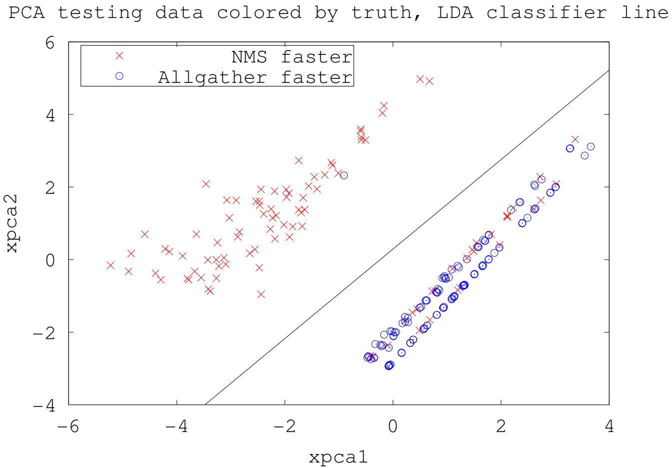
LDA



Training Data Results



Testing Data Results: 93% Classification Success



Conclusions

- Strict constraints on model configuration $\Longrightarrow$ high degree of success in performance characterization.
- More general characterization may or may not exist.
- Results of experiments not immediately useful, but framework is indispensible.

Next Steps

- Repeat procedure using a wider range of model configurations, using more continuously-varying parameters.
  - Larger and finer range of number of LPs, number of routers, link delay, etc.
  - Different (possibly non-random) topologies.
  - Different traffic configuration.
- Include testing data points from model configurations outside of the range of training configurations (see if we can predict via extrapolation).
- Integrate training data gathering and decision framework into ns-3 codebase so we don't have to do everything in Matlab.

Questions

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]

Intel 64 and IA-32 Architectures Software Developer's Manual, January 2015.

M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *SIGCOMM Comput. Commun. Rev.*, vol. 29, pp. 251–262, August 1999.

B. P. Swenson, J. S. Ivey, and G. F. Riley, "Performance of conservative synchronization methods for complex interconnected campus networks in ns-3," in *Proceedings of the 2014 Winter Simulation Conference*, WSC '14, Savannah, Georgia, pp. 3096–3106, 2014.

B. P. Swenson and G. F. Riley, "Simulating large topologies in ns-3 using brite and cuda driven global routing," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '13, Cannes, France, pp. 159–166, 2013.

D. M. Nicol, "The cost of conservative synchronization in parallel discrete event simulations," *J. ACM*, vol. 40, pp. 304–333, April 1993.

A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, October 1999.

B. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, vol. 6, pp. 1617–1622, December 1988.

K. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *IEEE Transactions on Software Engineering*, vol. SE-5, pp. 440–452, September 1979.

R. E. Bryant, "Simulation of packet communication architecture computer systems," tech. rep., Cambridge, MA, USA, 1977.

R. M. Fujimoto, *Parallel and distributed simulation systems*.

New York, NY: John Wiley & Sons, 2000.

S. Theodoris and K. Koutroumbas, *Pattern Recognition*.
Burlington, MA: Academic Press, 4th ed., 2009.

B. University, "Boston university representative internet topology generator," 2002.

A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: Universal topology generation from a user's perspective," January 2001.

ns-3 Project, *ns-3 Manual*, 3.22 ed., 2015.

ns-3 Project, *ns-3 API Documentation*, 3.22 ed., 2015.

ns-3 Project, *ns-3 Model Library*, 3.22 ed., 2015.

Message Passing Interface Forum, *MPI: A Message-Passing Interface Version 3.0*, September 2012.