Predicting Performance Characteristics of Distributed Simulator for Scale-free Networks

Christopher L. Hood, George F. Riley

Georgia Institute of Technology

13 May 2015
Long run time $\iff$ Large-scale $\iff$ 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.
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. 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

- 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.
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 $\in \{\text{Allgather, null-message}\}$.
- 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 $x_i$ 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:

\[ x_i = \begin{bmatrix} \text{number LPs} \\ \text{number routers} \\ \text{lookahead mean} \\ \text{lookahead std.} \\ \text{lookahead min.} \\ \text{lookahead max.} \\ \text{outdegree mean} \\ \text{outdegree std.} \\ \text{outdegree max.} \end{bmatrix} \in \mathbb{R}^9 \]

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

$\implies$ simplest to use a linear classifier:

$$\hat{z}_j = \text{sign}(w^T x_j + b)$$

where $w$ and $b$ are determined from the training data.
Is There an Apparent Pattern?

Data points, colored by truth

lookaheadMax

lookaheadStd

NMS faster

Allgather faster

Christopher L. Hood, George F. Riley
Georgia Tech ECE
Is There an Apparent Pattern?

Data points, colored by truth

lookaheadMax
numAs

NMS faster
Allgather faster

Christopher L. Hood, George F. Riley
Georgia Tech ECE
Is There an Apparent Pattern?

Data points, colored by truth

- **NMS faster**
- **Allgather faster**

Christopher L. Hood, George F. Riley  
Georgia Tech ECE
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?

Transformed data points, first two principal components

- NMS faster
- Allgather faster

Christopher L. Hood, George F. Riley
Georgia Tech ECE
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

\{ x \mid w^T x + b = 0 \}
Training Data Results

PCA training data colored by truth, LDA classifier line

- NMS faster
- Allgather faster
Testing Data Results: 93% Classification Success

PCA testing data colored by truth, LDA classifier line

NMS faster
Allgather faster
Conclusions

- Strict constraints on model configuration $\implies$ 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.
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
Introduction

Experiments

Results

Conclusions

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


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


