Understanding Congestion in High Performance Interconnection Networks

SUMMER OF CODES 2018

PHILIP TAFFET
Outline

M.S. thesis work: Understanding Congestion in High Performance Interconnection Networks Using Sampling

More recent projects: Simulating networking experiments before we run them
High level goal

Enable tools that help developers analyze and tune the communication performance of their applications.

Use a novel sampling method to capture information about where, when, and why congestion is occurring.
Idea

A distributed network monitoring scheme based on reservoir sampling provides a practical and powerful approach for quantitatively understanding network traffic and network congestion

- Samples the links through which a packet passes
- Samples congested links the packet encounters along its route

Uses a constant amount of storage per packet

Captures both network-wide and time-varying behavior
CODES’s Role

Programmable HPC networking hardware doesn’t exist (publically?)
Add a few lines of code to the fat tree CODES model to implement the algorithm
Works with both synthetic traffic patterns and TraceR-based OTF trace replay
Post-process data outside of CODES to avoid needing reversibility
Implemented several new synthetic traffic patterns in model-net-synthetic-fattree
Reservoir sampling for measuring network traffic and congestion

Packet
Source: 0, destination: 99, other headers
Traffic
Link: — hop count: 1
Congestion
Link: — hop count: 0
Packet data
Reservoir sampling for measuring network traffic and congestion

Packet
Source: 0, destination: 99, other headers
Traffic
Link: — hop count: 2
Congestion
Link: — hop count: 0
Packet data

Compute Node 0
Leaf Switch 1
Switch 2
Leaf Switch 3
Compute Node 99
Reservoir sampling for measuring network traffic and congestion

Packet
Source: 0, destination: 99, other headers
Traffic
Link: — hop count: 3
Congestion
Link: — hop count: 0
Packet data
Reservoir sampling for measuring network traffic and congestion

Packet
Source: 0, destination: 99, other headers

Traffic
Link: hop count: 4

Congestion
Link: hop count: 1

Packet data

Leaf Switch 1
Switch 2
Leaf Switch 3
Packet
Compute Node 0
Compute Node 99

Link is congested
Reservoir sampling for measuring network traffic and congestion

Table stored on Compute Node 99:

<table>
<thead>
<tr>
<th>Link</th>
<th>Weighted Traffic Count</th>
<th>Weighted Congested Count</th>
<th>Congested fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>+4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Traffic

Congestion

Link: hop count: 4

Link: hop count: 1

Link is congested

Packet
Reservoir sampling for measuring network traffic and congestion

Table stored on Compute Node 99:

<table>
<thead>
<tr>
<th>Link</th>
<th>Weighted Traffic Count</th>
<th>Weighted Congested Count</th>
<th>Congested fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>~1000</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>~1000</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>~1000</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>~1000</td>
<td>1000</td>
<td>1000</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Reservoir sampling for measuring network traffic and congestion

- **Packet**
  - ×1000

- **Link**
  - Weighted Traffic Count
  - Weighted Congested Count
  - Congested fraction

<table>
<thead>
<tr>
<th>Link</th>
<th>Weighted Traffic Count</th>
<th>Weighted Congested Count</th>
<th>Congested fraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>~1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>~1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>~1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>~1000</td>
<td>1000</td>
<td>1000</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Example congested fraction plot

Darker and thicker links have a higher congested fraction (more congested).

Links from leaf switches to compute nodes shown as row of boxes.
Interpreting congestion measurements

**Goal:** distinguish between problems due to
- Communication pattern
- Logical to physical mapping
- External interference

**Rules**

Endpoint congestion: a problem with the communication pattern

Congested inbound links without 100% utilization: External interference

Other congestion in network interior: logical to physical mapping
How is this different from prior work?

Traffic and congestion information is associated with a packet and a link

Packet-centric monitoring enables association with a program context

Link-centric measurement enables us to pinpoint links
  - Carrying network traffic
  - Experiencing congestion

No centralized collection server needed

Measurement volume per packet is $O(1)$, not $O(\text{path length})$
Experiments

Experiments use either of two fat tree networks
- Full bisection bandwidth with ~3500 nodes
- Half bisection bandwidth (tapered) with ~4600 nodes
- Full machine, but short duration (10 µs – 10 ms)

Four point stencil pattern (synthetic)
- Reservoir sampling diagnosed a poor mapping, guided mapping improvement efforts

Incast (many to one) reduction pattern (trace)
- Reservoir sampling diagnosed a poor communication pattern

Ring exchange pattern (synthetic)
- Reservoir sampling diagnosed that another job was the source of the congestion
How can we store this in a packet?

In CODES, adding metadata to a packet is easy and free. In real life, not so much...

Layout of an InfiniBand packet [InfiniBand Arch. Spec. v1]:

In the BTH? 6 bits reserved for future use

After the Variant CRC? Unlimited (?) bits available
Probabilistic approach

Inspired by work about detecting memory leaks [Bond, ASPLOS 2006]
  ◦ Pinpoint a frequent allocation site that leaks

Use a hash function that returns 1 bit

Instead of storing a link ID, the switch stores Hash(packet ID, link ID) in the packet

Upon receipt of the packet, test each potential link ID
  ◦ Compute Hash(packet ID, link ID)
  ◦ If it matches the hash value stored in the packet, increment the count for that link ID
  ◦ If it doesn’t decrement the count

We “store” the link ID with a single bit
Tradeoff between false positives and loss of detail

With a high false positive threshold, even though this technique loses a lot of detail, the diagnosis is still clear.

With more packets, we can better distinguish signal from noise.

Requiring a lot of packets is not a surprise for developers that are used to sampling-based CPU profiling.
Only testing against links that a packet might have traversed dramatically reduces the error:

- Use the source address.
More recent work

Preparing for a DAT on Quartz at LLNL

Testing the impact of congestion, mapping on the overall performance of several applications

Does Quartz’s 2:1 taper dramatically impact the performance of actual applications?
  ◦ Is there room to taper future machines even more?
  ◦ Does Omni-path’s congestion control scheme work in a real environment?

Which experiments are interesting to run?
Workflow

Pick applications, problems that take a few minutes to run

Use Score-P to collect traces

Setup TraceR mapping files for experiments, SLURM scripts for actual runs

Run in simulation

Unfortunately, no results yet.
My perspective on using CODES

Reversible computation is normally not as scary as I thought it would be

Scaling CODES is tricky
  ◦ I often just run single-threaded
  ◦ Load balancing may help: I have more trouble with load imbalance than with excessive rollbacks

PDES model makes it difficult to implement explicit congestion notification scheme (e.g. IB CC)

TraceR documentation is sparse but Nikhil is very helpful

TraceR mapping specification is ugly
My perspective on using CODES

Reversible computation is normally not as scary as I thought it would be

Scaling CODES is tricky
  ◦ I often just run single-threaded
  ◦ Load balancing may help: I have more trouble with load imbalance than with excessive rollbacks

PDES model makes it difficult to implement explicit congestion notification scheme (e.g. IB CC)

TraceR documentation is sparse but Nikhil is very helpful

TraceR mapping specification is ugly