Graphite – Getting Started

Getting up and running with Graphite
Outline

• System Requirements & Dependencies
• Getting & Building Graphite
• Simulating Your First Application
• Adding Your Own Application
• Benchmarks
System Requirements

- Operating System
  - Debian 5 (Lenny)

- Graphite has not been tested on other OSes
  - Issues usually arise from different syscalls that result from different compiler toolchains.

- Lightweight Solution:
  - Try installing the Lenny business card CD image on a VM.
Dependencies

• Intel PIN version 27887

• Libraries (g++, make, boost)

  ```
  $ apt-get update
  $ apt-get install build-essential
  $ apt-get install libboost1.35-dev libboost-filesystem1.35-dev libboost-system1.35-dev
  ```

• Git

  ```
  $ apt-get install git-core
  ```
Outline

• System Requirements & Dependencies
• Getting & Building Graphite
• Simulating Your First Application
• Adding Your Own Application
• Benchmarks
Getting Graphite

• Tarball method

$ wget http://github.com/mit-carbon/Graphite/tarball/master -O graphite.tar.gz

• Git method

$ git clone git://github.com/mit-carbon/Graphite.git
Building Graphite

• Makefile.config

<table>
<thead>
<tr>
<th>TARGET_ARCH = x86_64 (or IA_32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOST_VERSION = 1_35</td>
</tr>
<tr>
<td>PIN_HOME = /path/to/pin</td>
</tr>
</tbody>
</table>

• Done!

$ make
Outline

• System Requirements & Dependencies
• Getting & Building Graphite
• Simulating Your First Application
• Adding Your Own Application
• Benchmarks
Simulating Your First Application

- Toy applications available under /tests/apps/
  - eg. /tests/apps/hello_world/

- Graphite’s build system makes building apps a breeze:

  ```
  $ make hello_world_app_test
  ```
Simulating Your First Application

• Toy applications available under /tests/apps/
  – eg. /tests/apps/hello_world/

• Graphite’s build system makes building apps a breeze:

  $ make hello_world_app_test
Hello, world!
Simulation Results

• Results in output_files/sim.out

• Includes results collected for:
  – Core Models
  – Cache Models
  – Memory Models
Wallclock Times (μs)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>=</td>
<td>simulator start time</td>
</tr>
<tr>
<td>start</td>
<td>=</td>
<td>start of main() loop</td>
</tr>
<tr>
<td>stop</td>
<td>=</td>
<td>end of main() loop</td>
</tr>
<tr>
<td>shutdown</td>
<td>=</td>
<td>simulator end time</td>
</tr>
</tbody>
</table>

Simulation timers:
- start time: 1507035
- stop time: 1631016
- shutdown time: 1725671
Simulation timers:
start time 1507035
stop time 1631016
shutdown time 1725671

<table>
<thead>
<tr>
<th>Tile 0</th>
<th>TS 0</th>
<th>MCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1329</td>
<td>119</td>
<td>0</td>
</tr>
<tr>
<td>9705</td>
<td>1441</td>
<td>0</td>
</tr>
</tbody>
</table>

Core Performance Model Summary
Instructions
Completion Time
Average Frequency
Recv Instructions
Recv Instruction Costs
Sync Instructions
Sync Instruction Costs
Branch predictor stats
type
num correct
num incorrect

Network summary
Network model 0
type
num packets received
num bytes received
average latency (in clock cycles)
average latency (in ns)
Activity Counters
Switch Allocator Traversals
Crossbar Traversals
Link Traversals

Network model 1
Activity Counters
Switch Allocator Traversals
Crossbar Traversals
Link Traversals

Network model 2
Activity Counters
Switch Allocator Traversals
Crossbar Traversals
Link Traversals

Simulated Times
Time in ns the core runs.
<table>
<thead>
<tr>
<th>Completion Time</th>
<th>9763</th>
<th>1711</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Frequency</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Recv Instructions</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Recv Instruction Costs</td>
<td>10</td>
<td>0</td>
<td>18239</td>
</tr>
<tr>
<td>Sync Instructions</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sync Instruction Costs</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Branch predictor stats
num correct | 124 | 6 | 0 |
num incorrect | 59 | 3 | 0 |
type | one-bit (1024) | one-bit (1024) | one-bit (1024) |

Network summary
Network model 0
num packets received | 0 | 0 | 3 |
num bytes received | 0 | 0 | 246 |
average latency (in clock cycles) | nan | nan | 12.6667 |
average latency (in ns) | nan | nan | 12.6667 |
Activity Counters
Switch Allocator Traversals | 6 | 0 | 0 |
Crossbar Traversals | 64 | 0 | 0 |
Link Traversals | 64 | 0 | 0 |

Network model 1
num packets received | 0 | 0 | 0 |
num bytes received | 0 | 0 | 0 |
average latency (in clock cycles) | nan | nan | nan |
average latency (in ns) | nan | nan | nan |
Activity Counters
Switch Allocator Traversals | 0 | 0 | 0 |
Crossbar Traversals | 0 | 0 | 0 |
Link Traversals | 0 | 0 | 0 |

Network model 2
num packets received | 98 | 26 | 24 |
num bytes received | 7546 | 1874 | 1784 |
average latency (in clock cycles) | 4.10204 | 10.3846 | 11.6667 |
average latency (in ns) | 4.10204 | 10.3846 | 11.6667 |
Activity Counters
Switch Allocator Traversals | 74 | 14 | 24 |
Crossbar Traversals | 708 | 196 | 336 |
Link Traversals | 708 | 196 | 336 |
<table>
<thead>
<tr>
<th>Activity Counters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch Allocator Traversals</td>
</tr>
<tr>
<td>Crossbar Traversals</td>
</tr>
<tr>
<td>Link Traversals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network model 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>num packets received</td>
</tr>
<tr>
<td>num bytes received</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shmem Perf Model summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>num memory accesses</td>
</tr>
<tr>
<td>average memory access latency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cache Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache L1-I</td>
</tr>
<tr>
<td>num cache accesses</td>
</tr>
<tr>
<td>miss rate</td>
</tr>
<tr>
<td>num cache misses</td>
</tr>
<tr>
<td>Cache L1-D</td>
</tr>
<tr>
<td>num cache accesses</td>
</tr>
<tr>
<td>miss rate</td>
</tr>
<tr>
<td>num cache misses</td>
</tr>
<tr>
<td>Cache L2</td>
</tr>
<tr>
<td>num cache accesses</td>
</tr>
<tr>
<td>miss rate</td>
</tr>
<tr>
<td>num cache misses</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dram Perf Model summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>num dram accesses</td>
</tr>
<tr>
<td>average dram access latency</td>
</tr>
<tr>
<td>average dram queueing delay</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Queue Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue Utilization(%)</td>
</tr>
<tr>
<td>Analytical Model Used(%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dram Directory Cache Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Addresses</td>
</tr>
<tr>
<td>Average set size</td>
</tr>
<tr>
<td>Set index with max size</td>
</tr>
<tr>
<td>Max set size</td>
</tr>
<tr>
<td>Set index with min size</td>
</tr>
<tr>
<td>Min set size</td>
</tr>
</tbody>
</table>
Distribution

• Graphite simulations can be distributed.
  – Shared file system
  – SSH permissions

• Define process map in carbon_sim.cfg:

```plaintext
[general]
num_processes = 2

[process_map]
process0 = "server1.csail.mit.edu"
process1 = "server2.csail.mit.edu"
```
Outline

• System Requirements & Dependencies
• Getting & Building Graphite
• Simulating Your First Application
• Adding Your Own Application
• Benchmarks
Adding Applications

• Create app in /tests/apps/app_name/
  – include source code and header files

• Create makefile

<table>
<thead>
<tr>
<th>TARGET = app_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCES = app_name.cc</td>
</tr>
<tr>
<td>include ..//../Makefile.tests</td>
</tr>
</tbody>
</table>

• Done!

$ make app_name_app_test
Running Outside the Build System

• Compile your application with the following flags:

```
-static -u CarbonStartSim -u CarbonStopSim
-L${GRAPHITE_HOME}/lib
-L${GRAPHITE_HOME}/os-services-25032-gcc.4.0.0-linux-ia32_intel64/intel64
-L${GRAPHITE_HOME}/contrib/orion
-pthread -lcarbon_sim -lorion -los-services -lboost_filesystem-mt -lboost_system-mt -pthread -lstdc++ -lm
```

• Set environment variables:

```
$ export GRAPHITE_HOME = path/to/graphite
$ export PIN_HOME = path/to/pin
```
Running Outside the Build System

• Set process index
  – Each instance of Graphite needs a process index.
  – Scripts in /tools/ spawn instances ordered by index.
  – For single machine:
    
    ```
    $ export CARBON_PROCESS_INDEX = 0
    
    $ ${{PIN_HOME}}/intel64/bin/pinbin -mt -t
    ${{GRAPHITE_HOME}}/lib/pin_sim -c
    ${{GRAPHITE_HOME}}/carbon_sim.cfg
    -- [PATH/TO/YOUR/APPLICATION]
    ```

• Done!
Outline

• System Requirements & Dependencies
• Getting & Building Graphite
• Simulating Your First Application
• Adding Your Own Application
• Benchmarks
SPLASH Benchmarks

• SPLASH-2 under /tests/benchmarks/
  – Integrated into Graphite build system.
  – Easy!

$ make barnes_bench_test
Parsec Benchmarks

• Parsec 2.1
  – Download and point $PARSEC_HOME to the parsec home directory.

• Build Parsec and add the Graphite configuration

```
$ ./bin/parsecmgmt -a build -p tools
$ ./bin/bldconfadd -n graphite -c gcc -f
```
• Download Graphite configuration file:

```
-O ${PARSEC_HOME}/config/graphite.bldconf
```

• Build with Graphite configuration:

```
$ ./bin/parsecmgmt -a build -p apps -c graphite
$ ./bin/parsecmgmt -a build -p kernels -c graphite
```
• Specify path in /tests/Makefile.parsec:

```
PARSEC_HOME = /path/to/parsec/
```

• Done!

```
$ make blackscholes_parsec
```
• Parsec applications tested at large core counts:
  – blackscholes
  – canneal
  – fluidanimate
  – streamcluster
  – swaptions

• Other Parsecs yet to be debugged.
Regression Suite

• If you develop on Graphite, remember to check that Graphite still works!

$ make regress_quick
Happy Simulating!