Graphite: Directory Structure

Directory structure of code base and important files.
Overview

• Graphite: ~350 source files in 40 directories
  — Not counting 3rd-party tools and benchmarks

• High-level structure of the code and key files

• Many less important files and directories are omitted
/ (Top Level)

- carbon_sim.cfg – Default simulator configuration file
- common – Code for simulator backend
- contrib – External tools that are merged with Graphite (e.g., McPAT, DSENT)
- lib – Simulator compiled libraries
- pin – Front-end for PIN
- tests – Location of tests and benchmarks
- tools – Various helper scripts and tools
- Obsolete: docs (mostly), qemu
- Others: results
/common

• config
• mcpat
• tile
• misc
• network
• system
• transport
• user
• **config** – Config file parser, **but not** the main Config class for a simulation. (Confusing, we know.)

- mcpat
- tile
- misc
- network
- system
- transport
- user
• config

• **mcpat** – Code for interfacing Graphite and McPAT.

• tile

• misc

• network

• system

• transport

• user
• **config**

• **tile** – Code for a single tile in a simulation. Largely a container class for other components (cores, memory subsystem, etc.)

• **misc**

• **network**

• **system**

• **transport**

• **user**
/common

• config
• mcpat
• tile
• **misc** – Miscellaneous code.
• network
• system
• transport
• user
/common

- config
- mcpat
- tile
- misc
- **network** – Network code, including models.
- system
- transport
- user
/common

- config
- mcpat
- tile
- misc
- network
- **system** – Contains simulator-specific code that has no physical counterpart; *e.g.* MCP, LCP, and SimThread.
- transport
- user
• config
• core
• mcpat
• misc
• network
• system
• transport – Code to provide basic communication between cores and processes in a simulation. Multiplexes sockets and shared memory.
• user
• config
• core
• mcpat
• misc
• network
• system
• transport
• **user** – User-level (*i.e.*, application) interface to simulator backend. For example, explicit messages on the network or new instructions. This is how hooks to the backend are supplied.
• Simulator configuration

• Logging

• Contains common abstractions
  – Threads
  – TLS
  – Locks

• These abstractions have different implementations depending on how the simulator is used!
  – *E.g.*, pintools can not use pthreads
Shared abstractions

Idiom

• Declare base class with create method

• Backend (/common) provides a default implementation, declared with weak binding

• Frontend can supply another implementation that overrides the backend during linking

Example

```
/common/misc/int.h

class Int
{
  int _i;
  Int(int i) : _i(i) {} 
public:
  static Int * create();
}

/common/misc/int.c

__attribute__((weak)) Int * Int::create()
{
  return new Int(0);
}

/pin/pin_int.c

// Overrides above declaration if this is linked with executable!
Int * Int::create()
{
  return new Int(-1);
}
```
• **SimThread** – Thread to manage asynchronous requests to a given core

• **MCP** and **LCP** – Centralized system calls, thread spawning, etc..

• Top-level **Simulator** container class to manage per-process state
  – Initialization and shutdown
  – **TileManager**
  – **SimThreadManager**
/pin

• Contains Pin frontend that traps application events and forwards them to the backend

• Key files:
  – handle_syscalls.{cc,h}
  – instruction_modeling.{cc,h}
  – pin_sim.cc – Contains many Pin callbacks and entry point to entire simulation.
  – redirect_memory.{cc,h}
  – routine_replace.{cc,h} – Trap into backend; e.g. messaging routines, thread spawning, new instructions
  – thread_start.{cc,h}
  – /pin/lite directory – Equivalent functionality for lite mode
Contains benchmarks and correctness tests.

- **unit** – Unit tests

- **apps** – Higher-level correctness and performance tests (matrix multiply, jacobi relaxation)

- **benchmarks**
/tools

• **tests.py** – Batch run many simulations.

• **regress directory** – Scripts and config files for regression tests

• **parse_output.py** – Examples of ways to parse and analyze sim.out files

• **scripts/progress_trace.*** – Scripts to show simulated clock time for all cores in a simulation.

• ...and more
Conclusion

• More detailed information is available at