Summer of CODES 2015

Robert Ross, Phil Carns, Kevin Harms, John Jenkins, Shane Snyder, Misbah Mubarak

Argonne National Laboratory

Christopher Carothers, Justin LaPre, Elsa Gonsiorowski, Caitlin Ross, Noah Wolfe, Mark Plagge

Rensselear Polytechnic Institute
Team CODES (ANL)

Rob Ross

Phil Carns

Kevin Harms

John Jenkins

Shane Snyder

Misbah Mubarak
Team ROSS (RPI)

Chris Carothers

Justin LaPre

Caitlin Ross

Noah Wolfe

Elsa Gonsiorowski

Mark Plagge
Workshop Goals

- Bring together the increasing number of CODES/ROSS users
  - In attendance: ANL, RPI, IIT, UIUC, Tsukuba University Japan
  - Others interested in or already collaborating!
- Present research using simulation with the CODES/ROSS frameworks
  - Identify common research interests, areas
- Receive feedback from the community
- Hack on code!
Workshop Agenda

- [Link](http://press3.mcs.anl.gov/summerofcodes2015/workshop-proceedings/)
- 4 Sessions today
  - Mostly Research
- 1 Session + Hackathon tomorrow
  - Development-centric
Research Overview
What’s being done with CODES?

- **Networking**
  - Torus, Dragonfly
  - FatTree

- **Storage**
  - I/O Protocols
  - Data placement
  - Fault Detection / Response

- **HPC**
  - Trace extrapolation / replay

- **Grid**
  - Workflow processing / management

(non-exhaustive)

Data/control flow in example replicated storage system
CODES Projects @ ANL/RPI - Networking

- Research questions:
  - What torus dimensionality makes sense at scale?
  - Effect of routing algorithms on extreme-scale topologies (e.g., dragonfly)?

- Experimentation scale: up to 50 million nodes, packet-level simulation

Groups are “virtual” routers: each router in group connected to subset of routers in other groups
CODES Projects @ ANL - Distributed Storage

Key algorithmic design aspects explored at large-scale by simulation

Group Membership
- Detect member entry, exit
- Disseminate membership updates

Fault Tolerance
- Distribute objects+replicas
- Devise recovery plan on error
- Rebuild cluster to full redundancy

Replication Protocol
- Propagate user data across system
- Report operation completion/failure given resiliency/durability constraints
CODES Projects @ ANL - Distributed Storage

Key algorithmic design aspects explored at large-scale by simulation

Fault Tolerance  Group Membership  Replication Protocol

Key questions:
- Centralized/synchronized vs. decentralized approach?
- How fast do membership updates propagate through the system?
- How much network traffic are we willing/able to incur?

Simulation explores the feasibility of epidemic-style protocols in an HPC/datacenter deployment, in particular SWIM*.

CODES Projects @ ANL/RPI - Workflow Processing

Application-driven design space explored at large-scale by simulation

High-Energy Physics (HEP)
- Experimental/observational data processing pipeline @ Fermilab
- Special-purpose hardware/software stack doing petabyte-scale filtering/analysis

MG-RAST
- Metagenomics workflow processing system @ ANL
- Distributed, VM-based compute, centralized data cataloging/storage w/REST interface
CODES Projects @ ANL/RPI - Workflow Processing

Key algorithmic design aspects explored at large-scale by simulation

Key questions:

- How best to distribute the control/data planes?
  - Proxy servers, hierarchical server topology
- How best to schedule jobs in the face of heterogeneous resources?
- How to configure existing peta-byte scale storage systems?
  - Increasing cache life times, trying different cache policies
- How to quantify the value for deploying new hardware?
  - Adding more archival devices for e.g. tapes.

Simulation explores trace-driven replay of MG-RAST workflows in different architectural configurations.
CODES Projects in the Wild
(presented today)

- Large-scale HPC workload replay (Bilge Acun, UIUC)
- Networking
  - FatTree simulation (Ning Liu, IIT)
  - Scheduling WAN transfers (Xin Wang, IIT)
  - Topology-aware HPC job scheduling (Xu Yang, IIT)
- Storage (Yuki Kirii, Hiroki Ohtsuji)
- Frontiers in ROSS research
  - ROSS + Charm (Eric Mikida, UIUC)
Resources

- CODES website: http://www.mcs.anl.gov/projects/codes/
- ROSS website: https://github.com/carothersc/ROSS
- Getting started:
  - CODES: (codes-base repository)
    - doc/GETTING_STARTED
    - doc/codes-best-practices.tex
    - doc/example and doc/example_heterogeneous for detailed examples showing usage of (nearly) every feature
  - ROSS: check out the ROSS wiki
    https://github.com/carothersc/ROSS/wiki/_pages
- CODES repositories
  - codes-base (git clone git://git.mcs.anl.gov/radix/codes-base)
  - codes-net (git clone git://git.mcs.anl.gov/radix/codes-net)
- Mailing list: http://lists.mcs.anl.gov/mailman/listinfo/codes-ross-users
Anatomy of a CODES simulation

Simula>on
  (PDES)
Simula>on
  software
stack	
  view

Simulation software
stack view

Workflows
  Workloads

Services

Protocols

Hardware

Simulation (PDES)

CODES-centric
simulation view

Application
workloads

Workload
generator

Simulation
models

Simulation
configuration

Namespacing

Configuration

Communication

Network/storage/utility
models
CODES models: Networking/storage

- Network models:
  - analytic – based on LogGP [1]
  - packet-level simulation of torus [2], dragonfly [3] topologies at extreme scale (Misbah Mubarak)
  - models are decoupled from higher levels via model-agnostic API (modelnet)

- Storage model:
  - seek/rate histogram by access size (need for reverse computation precludes use of other models such as DiskSim)


CODES configuration

- ROSS provides simulator kernel configuration at command line.
  - Functionality for mapping LP-IDs to LP implementations
  - LP configuration left to users
- Structured configuration format with “sections” (think JSON, libconfuse)
- Usage model slanted towards large #s of homogenous/symmetric components (clusters, HPC/data center systems)
- Support for parameterizing otherwise equivalent simulation entities
- Informs LP namespace management, network modeling
- Screen capture is a (heavily commented) server pinging setup

Main components:
- LP specification
- LP-specific configuration (e.g., hardware capabilities)
- ROSS simulator parameters
- Arbitrary other sections

```
# the LPGROUPS set is required by all simulations using codes. Multiple groups
# can be entered (only one is here for our example), each consisting of a set
# of application- and codes-specific key-value pairs.
LPGROUPS
{
  # in our simulation, we simply have a set of servers, each with
  # point-to-point access to each other
  SERVERS
  { # required: number of times to repeat the following key-value pairs
    repetitions="16";
    # application-specific: parsed in main
    server="1";
    # model-net-specific field defining the network backend. In this example,
    # each server has one NIC, and each server are point-to-point connected
    modelnet_siminenet="1";
  }
  # required by CODES: miscellaneous parameters used in the simulation that
  # don't fit in group definition.
  PARAMS
  { # ROSS-specific parameters:
    # message_size: ROSS expects you to upper bound your event message size.
    # Going over this size will crash or otherwise destroy your simulation.
    message_size="256";
    # - pe_mem_factor: this is a multiplier to the event memory allocation that
    # ROSS does up front (multiplier is per-PE). Increase this
    # (or change the associated mem_factor variable in
    # codes-base) if you have a (very) large event population.
    pe_mem_factor="512";
    # model-net-specific parameters:
    # - individual packet sizes for network operations
    # (each “packet” is represented by an event)
    # - independent of underlying network being used
    packet_size="512";
    # - order that network types will be presented to the user in
    # model_net_set.params. In this example, we're only using a single
    # network
    modelnet_order="(siminenet)"
    modelnet_scheduler="fcs"; # first come first serve
  }
  # custom parameter sets can also be added - this one is used to define the
  # rounds of communication the servers will undergo
  server_pings
  { num_pings=5;
    payload_size=4096;
  }
```

7/28/15
LP Namespacing

- ROSS addressing: global LP-ID, PE-specific IDs, meaning of LP-IDs left to user
- In other tools:
  - SST – explicit “links” through which components communicate
  - OmNet – explicit specification of input/output “ports” between “modules”
  - SimGrid – MPI-style message passing driven by creation of “tasks” (MPI overlay via SMPI)
- CODES – addressing LP API driven by LP configuration
- Lookup LP relative to
  - Group name
  - Repetition within group
  - Offset within repetition
  - (optional) annotation
    - Specific to annotation or annotation-independent
- Note: LP placement in ROSS is static. CODES places LPs with the goal that nearest-neighbor LPs w/in a group are mapped to nearest-neighbor PEs / MPI ranks.