Running Ensemble Simulations with CODES/Swift/CoRtEx

Matthieu Dorier
Argonne National Laboratory
How to setup a large number of runs?

- Study of various networks
  - Many topologies: Dragonfly, Slimfly, Torus, Fat Tree, etc.
  - Many parameters: link bandwidth, buffer size, etc.
  - Many variants of collective algorithms
  - Many job placement strategies
  - Many routing strategies

- Typically, we want to run MANY experiments varying such parameters

- This is a difficult task
  - Hand-written scripts to submit hundreds of thousands of jobs?
  - How to know which jobs will require more resources?
  - How to keep track of the jobs that completed?
  - How to make the whole campaign reproducible?
Two tools helping CODES ensemble simulations

- The CoRtEx library
- The CODES-ESW framework
CoRtEx: Collective Runtime Extension

- **Goal**: translate MPI collective calls into a series of point-to-point calls

- **Features**:
  - Reads DUMPI trace files
  - Translates any event into a series of other events using C functions or Python scripts
  - Can generate events without a DUMPI trace using Python scripts
  - Provides a set of collective to p2p translation as implemented in Mpich

- [https://xgitlab.cels.anl.gov/mdorier/dumpi-cortex](https://xgitlab.cels.anl.gov/mdorier/dumpi-cortex)

- Now integrated into CODES!
import cortex

class MyTranslator:

    def MPI_Bcast(self, thread, **args):
        dtype = args['datatype']
        root = args['root']
        comm = args['comm']
        count = args['count']

        print "MPI_Bcast called in Python, root = ", root

        if not (cortex.MPI_COMM_WORLD == comm):
            print "Communicator is not MPI_COMM_WORLD, not translating"
            return

        if thread != root:
            s = cortex.MPI_Status()
            cortex.MPI_Recv(thread, count=count, datatype=dtype, source=root, tag=1234, comm=comm, status=s)
        else:
            size = cortex.comm_world_size()
            for i in range(size):
                if i != thread:
                    cortex.MPI_Send(thread, count=count, datatype=dtype, dest=i, tag=1234, comm=comm)
import cortex

def GenerateThings(thread):
    print "In the generator"
    if thread == 0:
        cortex.MPI_Send(thread, count=1, datatype=cortex.MPI_INT, dest=1, tag=1234, comm=cortex.MPI_COMM_WORLD)
    elif thread == 1:
        s = cortex.MPI_Status()
        s.nbytes = 4
        s.source = 0
        s.tag = 1234
        s.cancelled = False
        s.error = 0
        cortex.MPI_Recv(thread, count=1, datatype=cortex.MPI_INT, source=0, tag=1234, comm=cortex.MPI_COMM_WORLD, status=s)
The CODES-ESW framework

Running many instances of CODES/CoRtEx using Swift/T
Overview of the CODES-ESW workflow

1. codes.py computes the tasks to spawn and the required parallelism
2. Tasks are spawned using launch_turbine (subworkflow)
3. codes.py generates the input files for the given tasks
4. run-codes-par.sh is called on multiple swift workers and setup env
5. CODES is executed

https://bitbucket.org/mdorier/codes-esw
Example configuration file (YAML)

- codes:
  binary: /home/mdorier/codes-work/install/bin/model-net-mpi-replay
  timeout: 1200
  network: dragonfly
  parameters:
    packet_size: [ 512, 1024, 2048, 4096 ]
    routing: prog-adaptive
    seed: 0
    local_vc_size: [ 8192, 16384, 32768, 65536 ]
    local_bandwidth: [ 2.0, 4.0, 8.0, 16.0, 32.0 ]
    alloc_size: [ 128, 512, 1024, 2048, 3096 ]
    cortex_file: ../scripts/BinomialBcast.py
    cortex_gen: GenerateEvents
    workload_file: none
    bcast_num_iter: [ 2, 4, 8, 16, 20 ]
    bcast_msg_size: [ 1024, 4096, 8192 ]
  tasks: 1
  template: ../templates/dragonfly.conf.tpl

Default value if none provided

Default values can be computed based on other parameters

./run.sh 32 hosts.txt config/bcast.yml
CODES-ESW output

- A set of directory names with a number (e.g. "00001234"), containing
  - the configuration files used as CODES input
  - a YAML file summarizing all the input parameters
  - the resulting CODES and ROSS output files/directories
Result: a framework to run many CODES instances (and supporting fault-tolerance)
Conclusion

- CoRtEx: a library for translating MPI collective events into point-to-point events
  - Supporting Python scripts
  - and DUMPI traces
- CODES-ESW: a framework based on Swift/T for running ensemble simulations using CODES
  - Supporting fault-tolerance
  - Easy description of a design-space using a YAML file and template configurations
- Publication (on a new MPI feature, MPI_Comm_launch, used by CODES-ESW)
  - EuroMPI 2017, Launching MPI applications inside MPI applications (Matthieu Dorier, Justin Wozniak, Rob Ross) - submitted