# Architecture and Implementation of SAFE

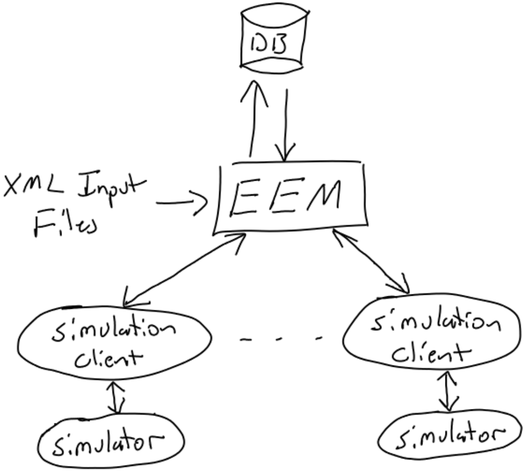
Bryan C. Ward

Bucknell University
Department of Computer Science

November 5, 2010

1. Architecture
   - Experiment Execution Manager
   - Simulation Client
2. Progress
   - Implementation
   - Polling Queues Simulator
   - ns-3 integration
3. Future Work
   - Goals
4. RFC

# SAFE Overview



# Design Point Generation

- Build design points based on the XML input.
- From the XML parser we expect:
  - Map with factors as keys, available levels as values.
  - List of restrictions each in the form of a map.
- Design points are constructed **Just In Time**.
  - Server initialization time reduced.
  - Allows clients to get started faster.

# MRIP - Akaroa

**MRIP - as seen in Akaroa:**

- Run independent simulations on separate processors.
- Must run the **same design point** using different seeds.
- Report results to a central server in **real time**.
- Server determines when all simulator instances should terminate.

# MRIP - SWAN-Tools

**MRIP - as seen in SWAN-Tools:**

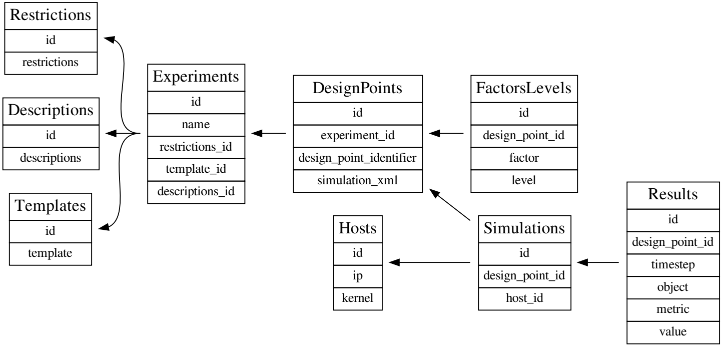
- Run independent simulations on separate processors.
- Different instances of the same design point run using different PRNG seeds.
- Can run **multiple design points concurrently**.
- Results are sent to the server **upon completion** of simulation execution.

# MRIP - SAFE

**MRIP - as seen in SAFE:**

- Run independent simulations on separate processors.
- Central server maintains state for **multiple design points**.
- **One or more processors can work on a design point at a time**.
- Design points dispatched using a **round robin** like algorithm.

# Database Schema



| Restrictions |
|---|
| id |
| restrictions |

| Descriptions |
|---|
| id |
| descriptions |

| Templates |
|---|
| id |
| template |

| Experiments |
|---|
| id |
| name |
| restrictions_id |
| template_id |
| descriptions_id |

| DesignPoints |
|---|
| id |
| experiment_id |
| design_point_identifier |
| simulation_xml |

| FactorsLevels |
|---|
| id |
| design_point_id |
| factor |
| level |

| Hosts |
|---|
| id |
| ip |
| kernel |

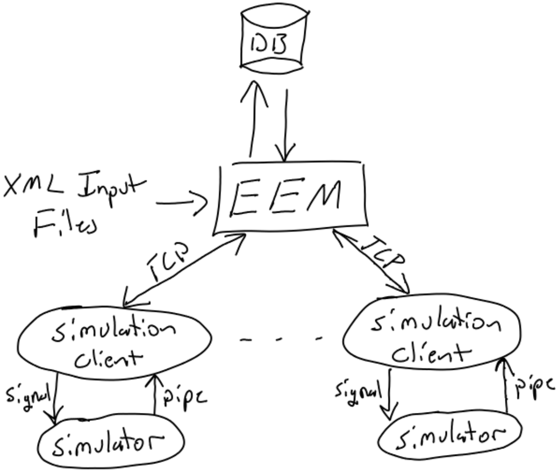
| Simulations |
|---|
| id |
| design_point_id |
| host_id |

| Results |
|---|
| id |
| design_point_id |
| timestep |
| object |
| metric |
| value |

# Run Length Detection

- EEM responsible for run length detection.
- User must specify metrics of interest which are used for run length detection.
- Terminate when all confidence intervals are appropriately bounded.
- Notify all clients of that design point to terminate.

# IPC Overview



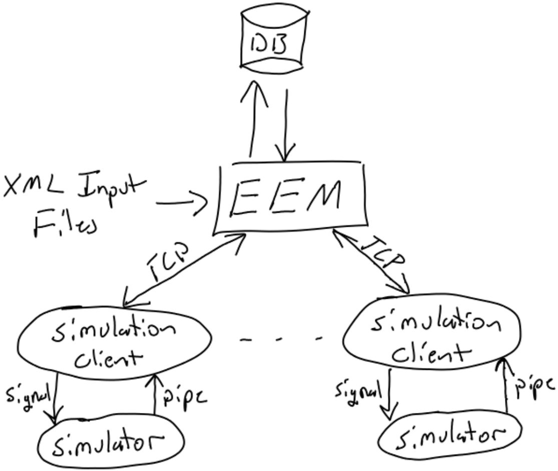
# EEM - Client IPC

- Communicates with the EEM via a TCP socket.
- Notify the EEM at startup of system information.
- Notify the EEM when ready for new design point.
- Listen for design point and for later instructions.
- Listen to simulator for samples and relay to EEM.

# Client - Simulator IPC

- Listen for samples on a pipe.
- Don't want to listen on `STDOUT`, or `STDERR`.
- Create new pipe with `fork`, `dup`, `exec`.
- Simulator can now `fdopen(3,'w')` and write to the pipe.
- Send signal to simulator upon notification of termination.
- This gives a **flexible architecture:**
  - Output of simulator can change as long as reflected in simulation client.

# Architecture Review



# Implementation

- Built using event driven python framework Twisted.
- Documentation: `http://www.eg.bucknell.edu/safe`
- Project page: `http://redmine.eg.bucknell.edu/perrone/projects/framework`

# Polling Queues Simulator

- Configured through XML straight from the EEM.
- Samples written to client pipe.
- Avoids some of the complexities of ns-3 while still testing basic functionality.
- Basic features of SAFE are working!

# ns-3 integration

- Static model.
- Configured using `ConfigStore` and XML from EEM.
- Data written on pipe opened in `main()`.
- Basic proof of concept of this workflow working.

# Wouldn't it be nice...

- ConfigStore is great for experiments which change attributes.
- How do we configure simulations with different topologies?
- Compare with SSF, simulation model specified through DML, which is far easier to generate.
- Can BRITE help with this?

# ns-3 integration

- Properly terminate simulations.
  - Can't send signal to child process, that's `waf`.
- Proper handing of results depends upon the **Data Collection Framework** which is still in development.

# Transient Detection

- Transient Detection similar to Akaroa.
- Client side or Server side?
  - Client side doesn't store transient samples
  - Server side incurs more traffic and more server side processing.

# Results Access API

- Provide programmatic way of accessing results.
- Do not require users to write SQL.
- Architecture of this API still under development.

# Plotting Utility

- Web Based.
- Builds upon work from the Results API.
- Confidence intervals are included by default.
- Export to pdf and other formats for inclusion in documents.
- Build from experience with SWAN-Tools.

# Request for Comments (Suggestions or Otherwise)

- Auxiliary results (routing tables, structs, blobs)
- Time stamp data type (double?)
- Transient detection client side or server side?
- How do we configure simulations with different topologies?
- Can we integrate with BRITE? If so, how?