OMNeT++ Community Summit 2017, University of Bremen, Sept 7-8.

# Running Simulation Campaigns in the Cloud

Attila Török

# Motivation

Problems:

- Larger simulation studies can take a long time to complete
- This lowers productivity, hinders research
- Quick iteration is desirable during model development

Opportunities:

- Tremendous amount of computational power available in the cloud
- Easy to access, low cost, no commitment
- Simulation runs are usually independent, well suited for execution in the cloud

# What is a Cloud?

Computing capacity sold as a service

- Many computing nodes, lots of memory, high bandwidth
- Virtualized access to resources (CPU, memory, storage, network)
- Billing based on metered usage

# Cloud Service Providers

Many providers:

- Amazon (AWS)
- Microsoft (Azure)
- Google (Cloud Platform)
- Digital Ocean
- etc...

# Typical Pricing

Surprisingly cheap, free trial periods available

E.g. Amazon Web Services:

- One hour of a powerful box costs **$0.80 - $0.90**
  - 16 thread CPU at 2.9GHz with 30 GB RAM
- Free Tier includes 750 hours each month for a year
  - Single core, 1 GB RAM, 10% average usage
- Network traffic is about **$0.01 / GB**
- Data storage is around **$0.025 / GB / month**

# Running Simulations in the Cloud

# Our proposed solution

- Based on AWS services
- Drop-in replacement for `opp_runall`, no change to simulations

```
$ opp_run_aws ./routing -c MeshExperiment --redis-host 174.57.3.18
```

- Tutorial available: [https://omnetpp.org/doc/opp-docker-tutorial/](https://omnetpp.org/doc/opp-docker-tutorial/)

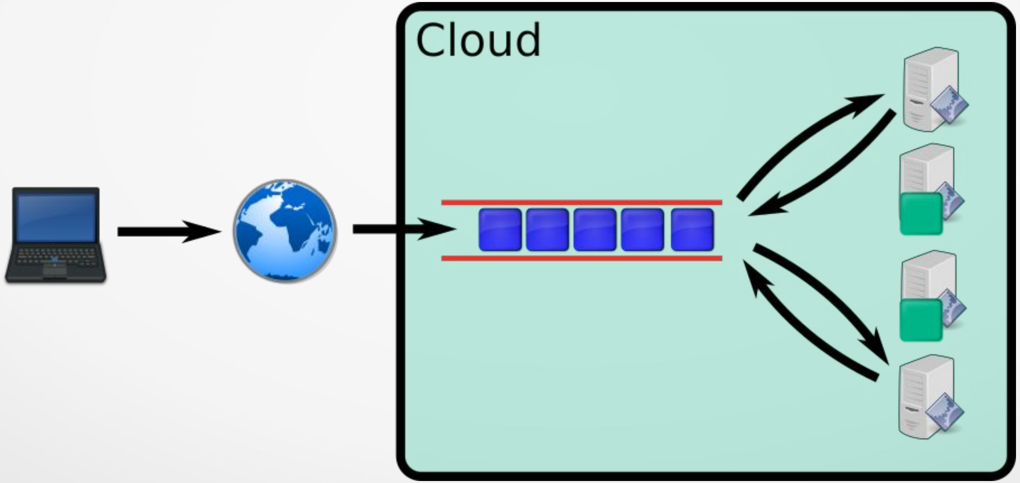
# Technologies

We will use the following technologies:

- Job queue
- Docker containers
- Container Service on Amazon Web Services
- Python (for glue code)

# Job Queues

- Jobs are submitted into the queue through the Internet
- Workers take jobs from the queue, process them, then store the results



# Simulation Jobs

- Each run in a Configuration is enqueued, possibly with a run filter
- The inputs of each job are:
  - Model code
  - Configuration name
  - Run number
- Output of each job is the results of the run it performs
  - scalars, vectors, event logs, output logs - practically the “results” folder
- Workers run in Docker containers

# What is Docker?

- Lightweight alternative to virtualization technologies
- Makes running the same application binaries anywhere possible
  - No need to install the right version of dependencies, etc...
- Works on Linux, Mac, Windows, etc.
- Widely supported by numerous cloud services

# Docker Images

- A template from which containers are created
- Images contain:
  - A file system: All libraries, config files, the application itself
  - Metadata: Which program needs to be started, values for environment variables, etc...
- Hosted and distributed on Docker Hub
- An image may be based on an existing one
  - For example: ubuntu < apache < my-website

# Docker Example

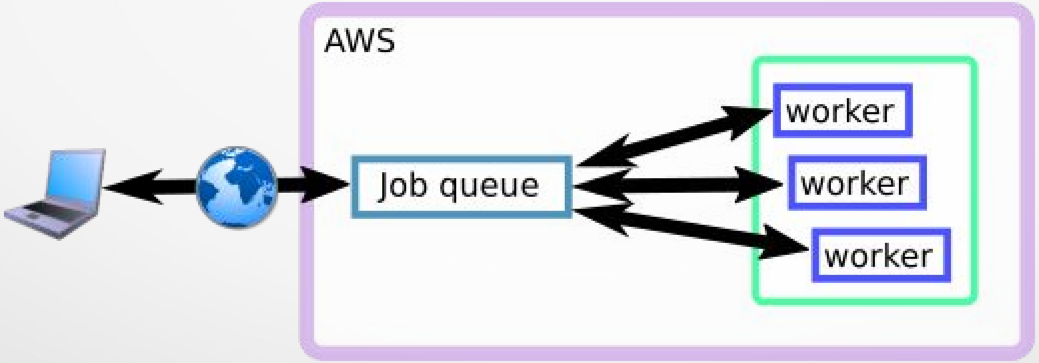
```
attila@inspiron ~ $ docker run -ti hello-world
Hello from Docker!
```

. . .

```
attila@inspiron ~ $ docker run -ti ubuntu
root@c6ba3f91876d:/# ps
  PID TTY          TIME CMD
    1 ?        00:00:00 bash
   13 ?        00:00:00 ps
root@c6ba3f91876d:/#
```

# Architecture

- Containers running on AWS ECS
- One for the Queue, using Redis and RQ
- The rest are workers, their image contains
  - OMNeT++ installation
  - Some Python code as the job itself



# Preliminary Results

We ran a small simulation:

- Custom configuration of the `routing` sample, 36 runs
- **8 minutes** on my laptop with 8 cores
- **3.5 minutes** on 10 single core workers on AWS
    - Well within the Free Tier limits

# Thank You!

The tutorial is at:

https://omnetpp.org/doc/opp-docker-tutorial/