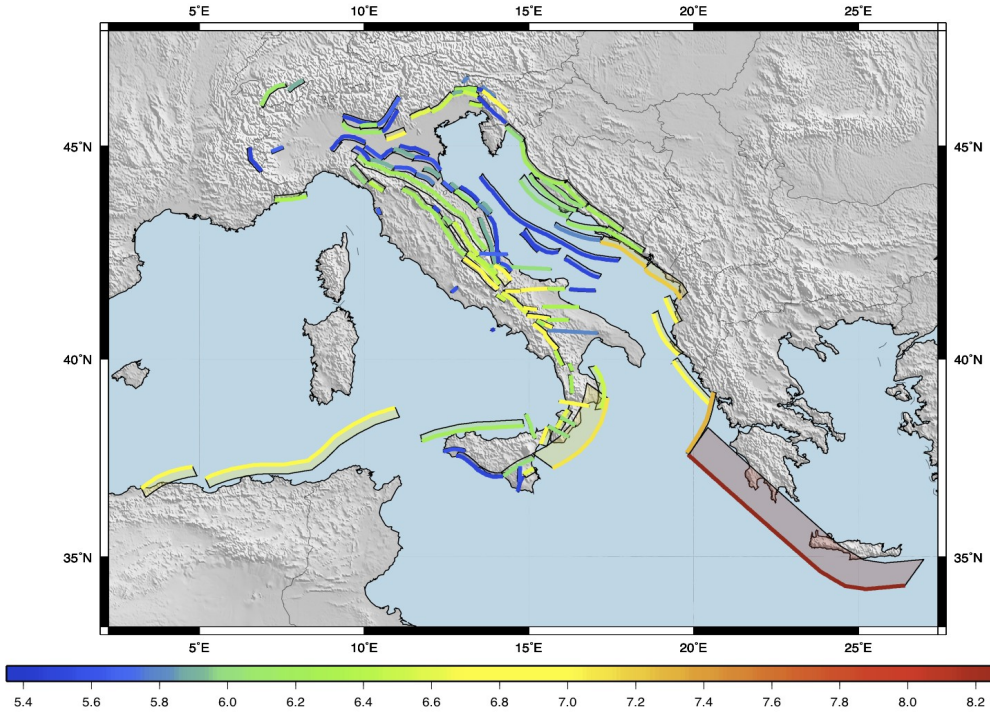
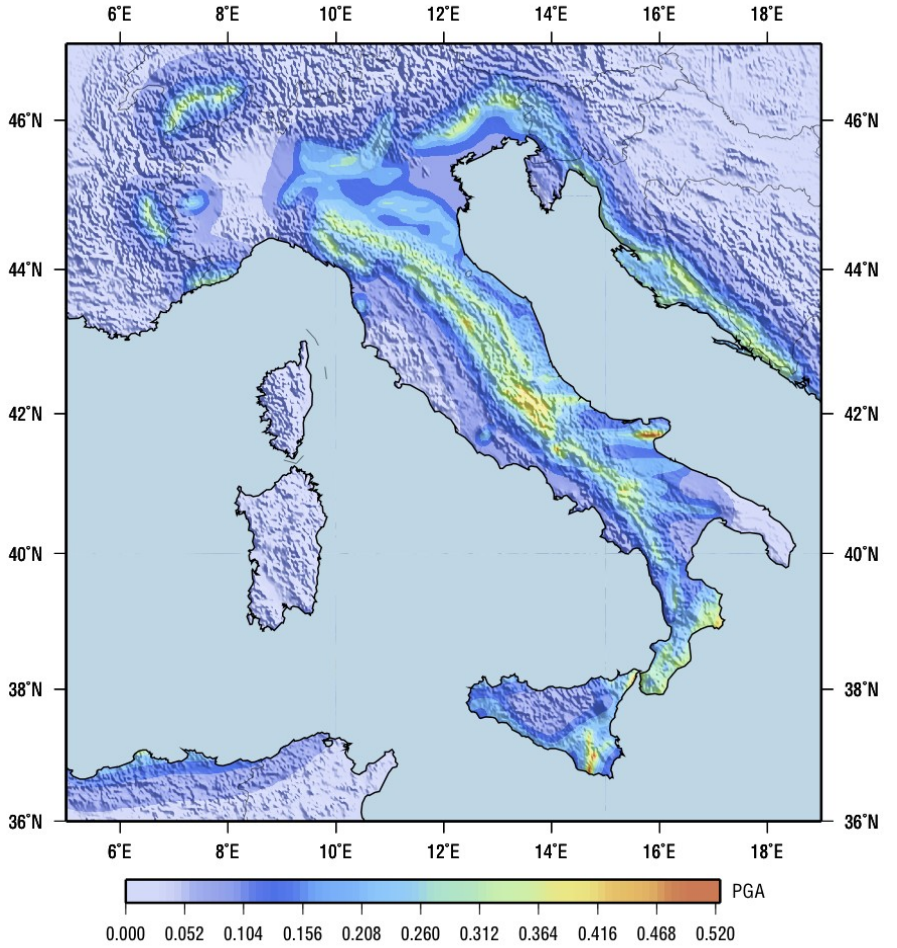
Scaling up OpenQuake

Muharem Hrnjadovic | Senior engineer (mh@foldr3.com)

Overview

- Not a game!
- Seismic hazard and risk





Surprise, surprise: OpenQuake is open!

- Open source
- Open for anybody to use
- Open to outside contributors
- Open processes
  - Blueprints, bugs, milestones and Ubuntu packages on launchpad.net
  - Open software repository on github.com
  - Code reviews
  - irc discussion channel (#openquake @freenode.net)
  - Mailing list (openquake-dev@googlegroups.com)

OpenQuake is elastic

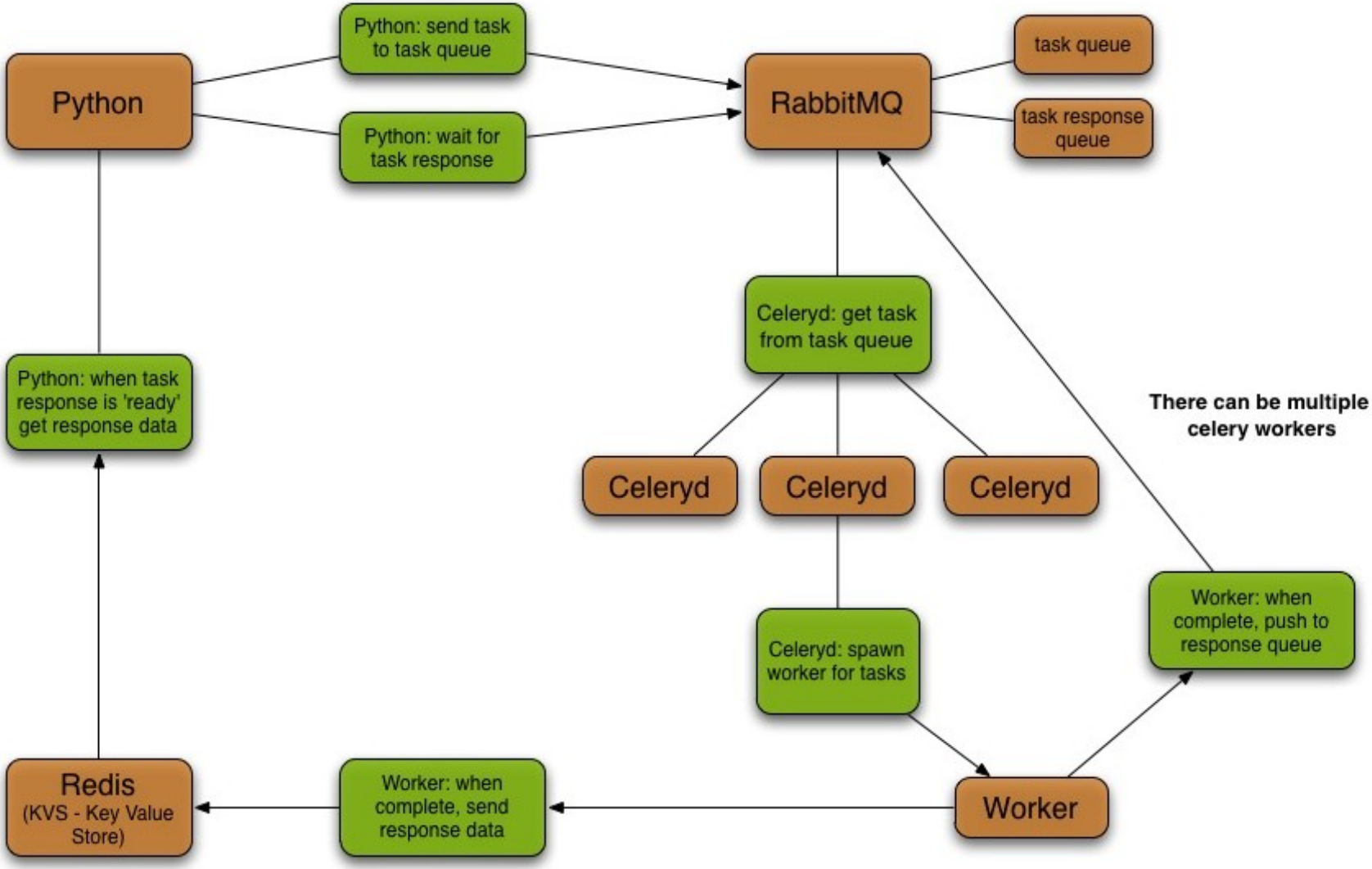
"Capable of adapting to change or a variety of circumstances"

- Scalable
- Potentially deployable in a number ways
  - Single machines
  - **Networks of physical machines**
  - cloud deployment (e.g. EC2)
  - Hybrid networks of physical/virtual machines

Major building blocks

- PostGres, PostGIS
- Python, Java
- Celery
- RabbitMQ
- Redis

How it works (sort of)



The starting point

- Hazard calculation for Europe: 150,000 points
- Computation in stages
  - Compute hazard for **all** points
  - Write results to database for **all** points
  - Serialize **everything** to XML
- Celery
  - 1 point per task
  - Create 150K tasks and let the party begin

The initial problem

- Hazard calculation gets stuck
- Where? Why? At what point?
- 4 machines, 100 processes, RabbitMQ, redis, postgres, celery python, java, erlang, django, postgis
- Key learnings
  - Software must be more “inspectable” (progress counters)
  - Make sure you monitor **all** pieces of your infrastructure
  - Connect to upstream (rabbitmq dev team in our case)
- Big THANKS to the
  - rabbitmq team
  - celery author

  for helping us find the root cause of the blockage!

Optimizations

- Computation in stages, disadvantages:
  - We cannot interrupt a job: all work is lost in case of a crash (weeks of 24x7 CPU time!)
  - Resource consumption:
    - All intermediate computation results need to be held until serialization begins
    - 150,000 result queues for celery tasks
- Improvements
  - Calculate **and** serialize in (configurable) slices
  - Use celery tasks in "ignore_result" mode
  - Interleave calculation and serialization i.e. collect results and persist in parallel

Outlook and closing remarks

- Workers should handle the **entire** workflow
- Use ubuntu's juju for service orchestration and reconfiguration
- Once we learned how to use them properly and appropriately
  - celery
  - rabbitmq
  - redis

  have been working very nicely

Questions?