# Overview of common GC3Pie use cases

Sergio Maffioletti <sergio.maffioletti@uzh.ch>

*S3IT: Services and Support for Science IT*

University of Zurich

September 12–13, 2016

# What is GC3Pie?

GC3Pie is . . .

1. An *opinionated* Python framework for defining and running computational workflows;
2. A *rapid development toolkit* for running user applications on clusters and IaaS cloud resources;
3. The worst name ever given to a middleware piece. . .

As *developers*, you're mostly interested in this part.

# Uses of GC3Pie: parameter sweep

You have a simulation code that is dependent on a number of parameters.

Run the code for all possible combinations of parameters.

Then collect all the outputs and post-process to get a statistical overview.

# Uses of GC3Pie: model calibration

You have a simulation code that is dependent on a number of parameters.

Run the code for all possible combinations of parameters, and find the ones that “best” approximate a given experimental result.

# Uses of GC3Pie: parallel processing

Run the same program over and over again, feeding it different input files each time.

Then collect all the outputs and post-process to get a statistical overview.

(At times, you chop a large input file into pieces and process each one separately instead.)

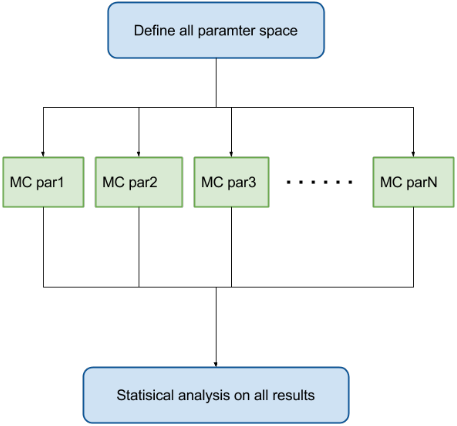
# Uses of GC3Pie: Our example

## Function to generate sample paths for assets assuming geometric Brownian motion:

For example, say we want to simulate the price evolution of a given asset using a Matlab Monte-Carlo simulation using a large parameter space - say 10’000 - for the initial conditions.
If we run 1 Monte-Carlo Matlab simulation and loop over all initial parameter configurations it could take along as (2.4hours * 10’000) = 24’000 cpu/hours (so you get your results in 1’000 days... ).
However, if we split this job into individual parameters evaluation and ran 100 jobs in "parallel" on a cluster, our run-time is reduced to only 240 hours.

# Uses of GC3Pie: workflows

Orchestrate execution of several applications: some steps may run in parallel, some might need to be sequenced.



# A typical high-throughput script structure

1. Initialize computational resources
2. Prepare programs and inputs for submission
3. Submit tasks
4. Monitor task status (loop)
5. Retrieve results
6. Postprocess and display

# What GC3Pie handles for you

1. Resource allocation (e.g. starting new instances on ScienceCloud)
2. Selection of resources for each application in the session
3. Data transfer (e.g. copying input files in the new instances)
4. Remote execution of the application
5. Retrieval of results (e.g. copying output files from the running instance)
6. De-allocation of resources