University of Zurich[UZH]

S3IT: Service and Support for ScienceIT

GC3Pie: orchestrating large scale executions of scientific applications

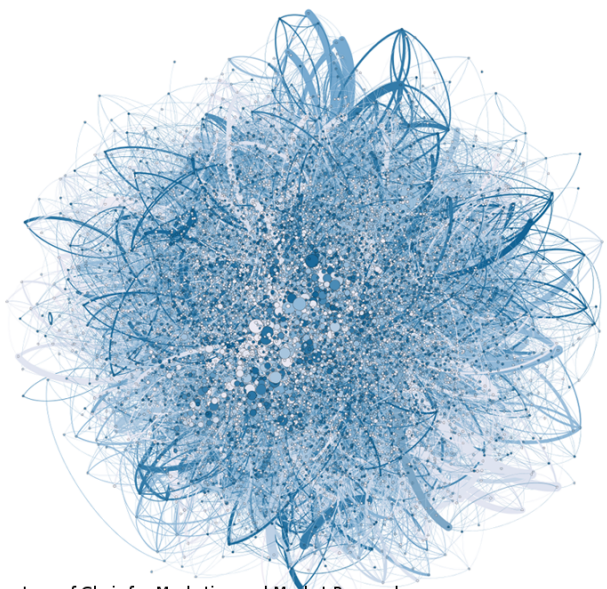
Sergio Maffioletti
S3IT: Service and Support for ScienceIT,
University of Zurich

June.18, 2014

Let's consider an example

Cloud-based Social Network analysis

- Quantitative empirical research and the extended network analysis.
- Challanges are the increasing size of datasets and the complexity of statistical models.



Courtesy of Chair for Marketing and Market Research.
University of Zurich (http://www.socialnetworks.uzh.ch/index.html)

Scaling out from local experiment

- User has defined an R function that performs the statistical computations.
- Function is applied to each element of the network data.
- Need to process 3.5M entries.
- On average the function takes 1-2' to process an entry.

How S3IT did help

```
...
weight_list <-apply(input.edges,
                    1,
                    GetWeight,
                    data=input.data,
                    threads.nodes.posted=input.posted)
...
```

How did GC3Pie help

```
class GWeightApplication(Application):
    """
    Custom class to wrap the execution of the R scripts passed in src_dir.
    """
    application_name = 'gweight'

    def __init__(self, edges_data_filename, **extra_args):
        # Check consistency of input data
        # Build remote command invocation
        # e.g.  Rscript --vanilla $MASTER $WEIGHT $EDGES $DATA $THREADS
        ...

        Application.__init__(
            self,
            arguments = arguments,
            inputs = inputs,
            outputs = outputs,
            stdout = 'gweight.log',
            join=True,
            executables = ['wrapper.sh'],
            **extra_args)
```

How GC3Pie did help

```
./gweight.py -h
usage: gweight [-h] [-V] [-v] [--config-files CONFIG_FILES] [-c NUM]
               [-m GIGABYTES] [-r NAME] [-w DURATION] [-s PATH] [-u URL] [-N]
               [-C NUM] [-J NUM] [-o DIRECTORY] [-l [STATES]] [-k [NUM]]
               [-M [PATH]] [-D [PATH]] [-F [PATH]] [-T [PATH]]
               edges_data
```

Let's see how it works. . .

What is GC3Pie and why do we need something like that ?

Supervised execution

Need to automate the execution of a large number of different applications on a variety of computational resources.

1. **Access** to computational resources
2. **Supervise** execution of collection of jobs
3. **Handling** of error conditions individually
4. **Post-process** and store results

The issues GC3Pie wants to solve

1. **Portability:** Cannot run on a different cluster without rewriting all the scripts.
2. **Code reuse:** Scripts are often very tied to a certain purpose, so they are difficult to reuse.
3. **Heavy maintenance:** the more a script does its job well, the more you'll find yourself adding *generic* features and maintaining requests from other users.

What is GC3Pie ?

GC3Pie is a Python toolkit:

it provides the building blocks to write Python scripts to run large computational campaigns and

to combine several tasks into a dynamic workflow.

What is GC3Pie ?

GC3Pie consists of three main components:

GC3Libs:

Python library for controlling the life-cycle of computational job collections.

GC3Apps:

A collection of driver scripts to run large job campaigns.

GC3Utils:

This is a small set of low-level utilities exposing the main functionality provided by GC3Libs.

GC3Pie for developers

Programming model based on customization of base classes through inheritance (Template method pattern)

Different level of interfaces depending on the control required

SessionBasedScript is the highest level of abstraction

How is GC3Pie different? (I)

GC3Pie runs specific **applications**, not generic jobs.

That is, GC3Pie exposes `Application` classes whose programming interface is adapted to the specific task/computation a scientific application performs.

You can add your own application by specializing the generic `Application` class.

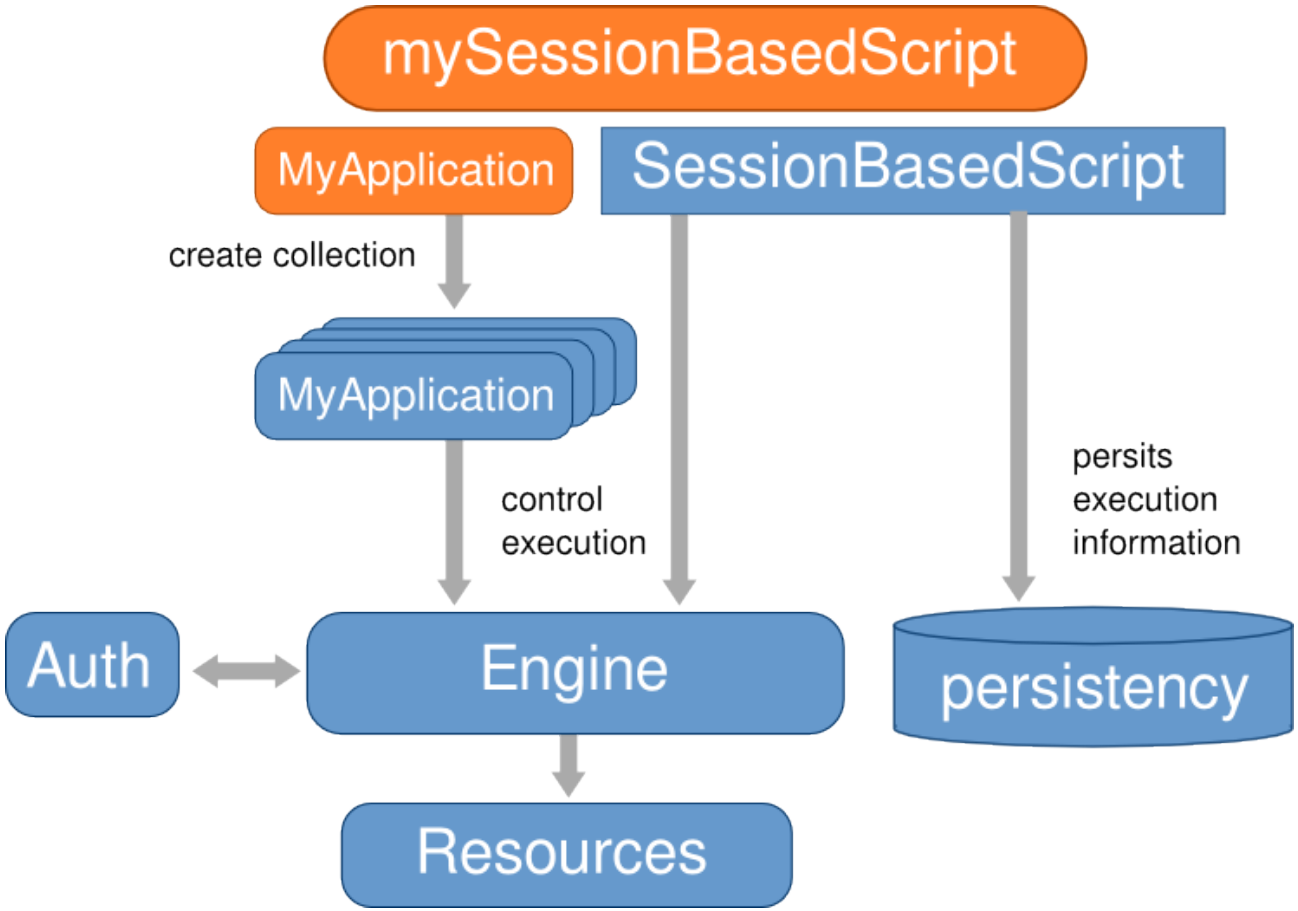
How is GC3Pie different? (II)

GC3Pie can run applications in parallel, or sequentially, or any combination of the two, and do arbitrary processing of data in the middle.

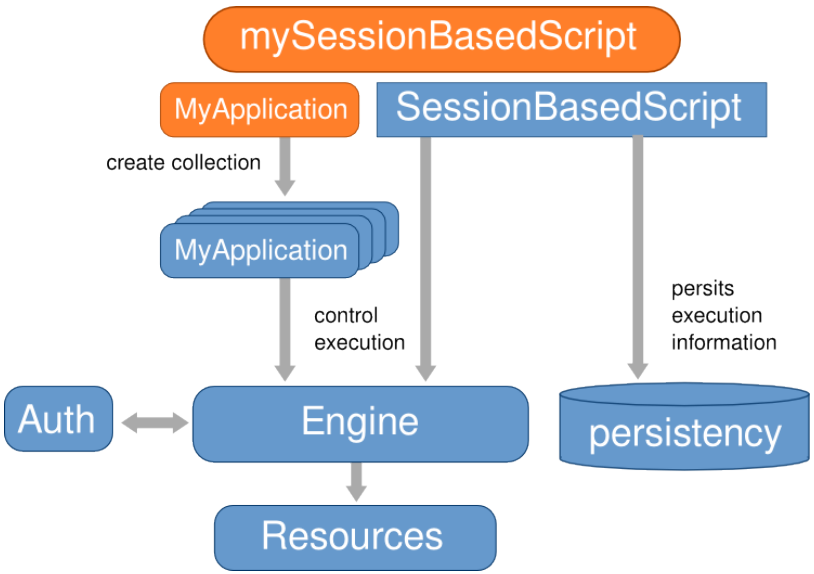
Think of workflows, except you can write them in the Python programming language.

Which means, you can create them dynamically at runtime, adapting the schema to your problem.

GC3Pie Execution model



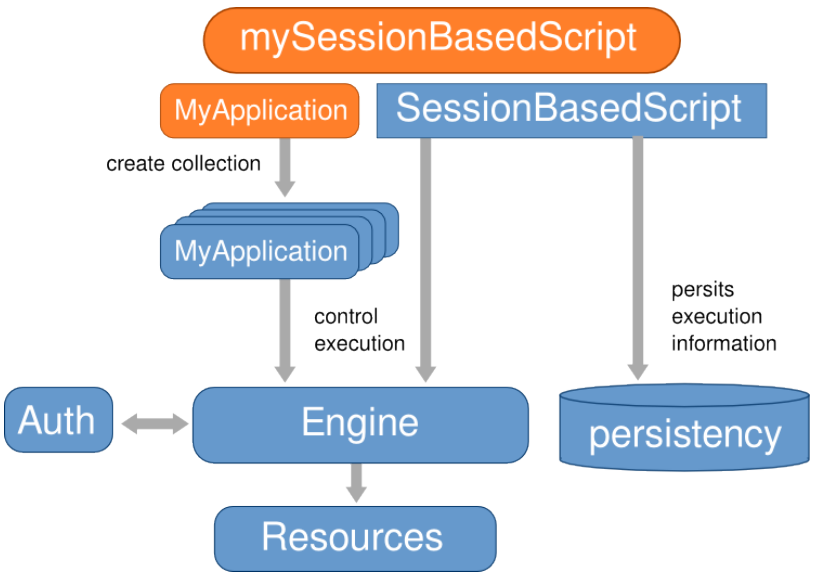
GC3Pie Execution model



An application is a subclass of the `gc3libs.Application` class.

Applications can be grouped in collections

GC3Pie Execution model



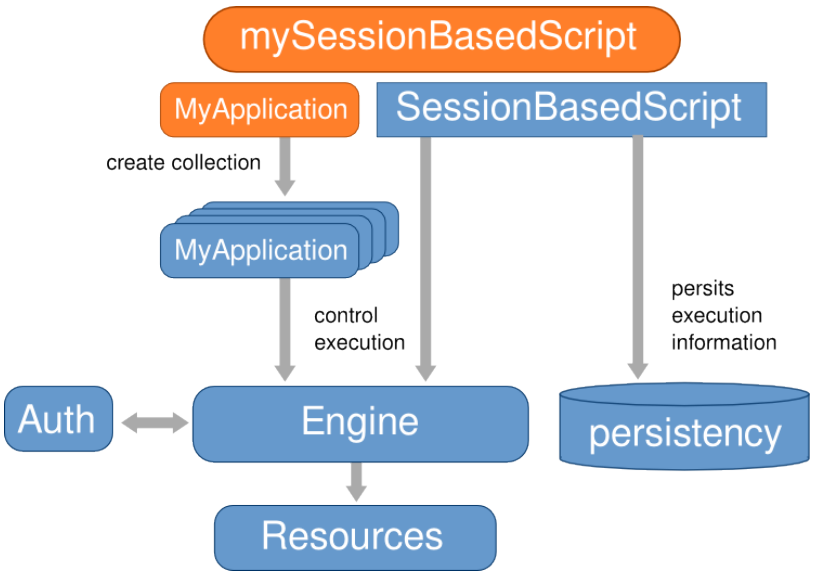
Execution of collections is delegated to an `Engine`.

GC3Pie Execution model

Resource
is a computational endpoint where the execution of the *Application* will take place.

GC3Pie can execute processes on a large variety of computational resources: cloud-based VMs, batch-queueing clusters, computational Grids, and -of course- any Linux/UNIX host where you can SSH into.

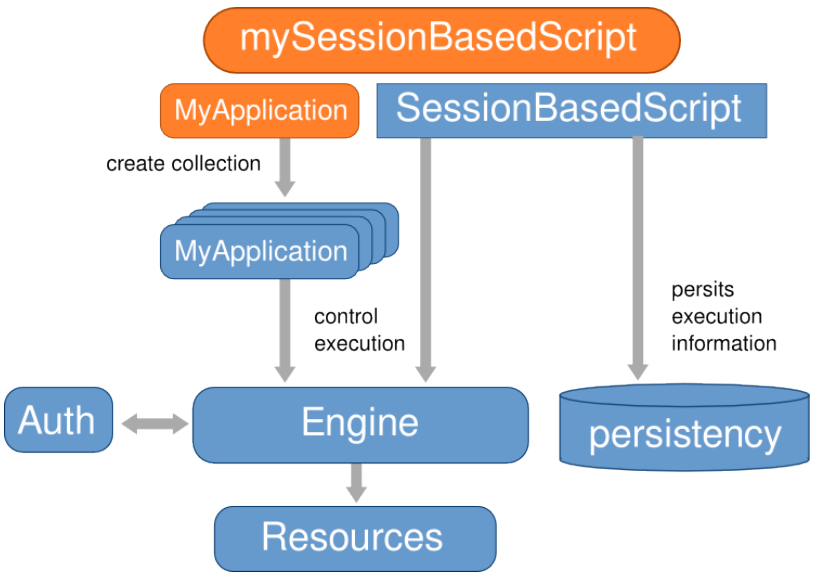
GC3Pie Execution model



Each resource has its own access mechanism and thus requires its own authentication.

Execution Engine handles the access to computational resources transparently.

GC3Pie Execution model



A convenient SessionBasedScript class contains already most of the control logic for instructing the execution engine

`SessionBasedScript` takes also care of persisting execution information

Job dependency management

An `Engine` manages all jobs concurrently. What if there are inter-application dependencies?

GC3Pie provides Task composition support (workflow), created programmatically from Python code.

Which means, no graphical editor. But also means you can create workflows on-the-fly as your computation proceeds.

References

- Website: https://code.google.com/p/gc3pie/
- Documentation: http://gc3pie.readthedocs.org/en/latest/

Thank you for your attention!