Scientific Workflow Management System

Ka-Ming Tam, Mark Jarrell

Department of Physics and Astronomy, Louisiana State University

Center for Computation and Technology, Louisiana State University

Consortium for Innovation in Manufacturing & Materials

Background

The advance of computational tools and numerical methods has greatly enhanced the complexity of computational tasks. Nowadays, it is unusual to have a computational task which can be completed by a single function call. Different scales, length or time, involve different kinds of approximations and thus different numerical schemes. A complete picture of the system can only be revealed by considering the interdependence among different scales. Reading and feeding data into modules corresponding to different scales can turn quickly into a complicated task. In additional, simulations corresponding to real systems often requires the input of experimental data or a large data set from ab-initio calculations. For some simulation, it may even require a constant input of real time data from different locations. Moreover, the final data set generated can often be large, while the information contained in it can be very minimal. Data analysis designed to pull relevant information from a large data pool is critical for scientific discovery. Scientific Workflow management system is designed to streamline these processes.

Workflow management system

The Pegasus workflow management system contains three main layers. The user interface layer which interprets the workflow language to the workflow as depicted in a directed acyclic graph. The work flow generation and execution plan layer which generates the dependences of modules and data. The user service layer which provides the statistics and the status of the running workflow.

The type of workflow supported by the Pegasus can always be depicted as a direct acyclic graph. Each computing module is represented by the vertex of the graph, and the dependences among modules are represented by the directed edge which link the vertices together. The graph has a tree structure, only acyclic graph without loop is supported. As a user, the first step of utilizing the Pegasus is to develop a workflow plan independent of the execution environment of the simulation.

Multifractal analysis of surface roughness

$$d = -\lim_{r \to 0} \frac{\log(N(r))}{\log r}$$

N and r can be treated as the "mass" and length of a geometrical object. This formula describes the dimension of all usual objects, for example a uniform square has dimension 2, and a uniform cube has a dimension 3. If the object does not have an integer dimension, it is said to be fractal.

Many systems involve stochastic or random process cannot be described simply in terms of a single exponent, instead a function of exponents (dimensions) are required to characterize it. Interesting system under this class include the surface growth by stochastic deposition, quantum wavefunctions of disorder systems, and turbulent flows. The multifractal exponents can be defined as

$$D_q = \lim_{r \to 0} \frac{1}{q-1} \frac{\log \sum_{i=1}^{N(r)} p_i^q}{\log r}$$

The geometrical object is put into a grid of many boxes, p is the "mass" of a box with length r. We will not discuss further the detailed meaning of this quantity in this poster, instead we focus on the procedure for computing it from the raw data from the many realizations of a random system.

For this example, we are given with a data file which contains many realizations of two dimensional data. The goal is to calculate the averaged multifractal spectrum from the data set. This example serves as a template for a typical workflow in which multiple data format transformations, and data analyses via packages are involved. One can generalize it to adapt for other data formats and analyses required for the other projects.

Read in and transform data

```
#This is the input file
webpage = File("densityd5.txt")

# the split job that splits the webpage into smaller chunks
split = Job("split")
split.addArguments("-l","256","-a","5","-d",webpage,"part.")
split.uses(webpage, link=Link.INPUT)
# associate the label with the job. all jobs with same label
# are run with PMC when doing job clustering
split.addProfile( Profile("pegasus","label","p1"))
dax.addJob(split)
```

Read in and then split the input file

The data file, which contains data from each realization, is read in, and then split into separated files in preparation for analysis. If it were a hdf5 file, this can be done by calling hdf5 library. Here a standard linux tool "split" is used.

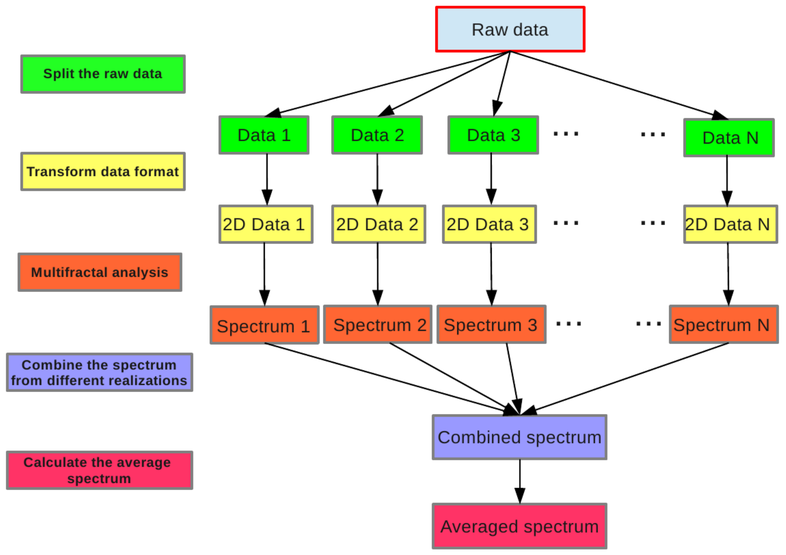
Spawn jobs

```
n = 0
for c in range(0,1000):
    n = n + 1

    d =  str(c).zfill(5)
    part = File("part.%s" % d)
    split.uses(part, link=Link.OUTPUT, transfer=True, register=False)
```

Looping over realizations, here we assume there are 1000 of them

We assume there are 1000 realizations, and each of them will be fed into the analyzing package separately. The workflow is spawned from 1 truck to 1000 branches.



Direct acyclic graph for the workflow

Transform data format

```
fin = File("fin.txt.%s" % d)

f1 = Job("f1")
f1.addProfile( Profile("pegasus","label","p1"))
f1.addArguments(",part)
f1.setStdout(fin)
f1.uses(part, link=Link.INPUT)
f1.uses(fin, link=Link.OUTPUT, transfer=True, register=False)
dax.addJob(f1)

#adding dependency
dax.depends(f1, split)
```

Adding the job program f1 which transform the data from 1d to 2d grid

Adding dependence job f1 is added to job split

We need to transform the data from a 1D grid to a 2D grid for the multifractal analysis package, here the function f1 is registered in the workflow to do this transformation.

A1 A2 A3 A4 A5 A6 A7 A8 A9 → A1 A2 A3 / A4 A5 A6 / A7 A8 A9

Call package for data analysis

```
f2 = Job("f2")
f2.addProfile( Profile("pegasus","label","p1"))
f2.addArguments(fin,dat2,"2","16","4","D")
f2.setStdout(dat3)
f2.uses(fin, link=Link.INPUT)
f2.uses(dat2, link=Link.INPUT)
f2.uses(dat3, link=Link.OUTPUT, transfer=True, register=False)
dax.addJob(f2)
files.append(dat3)
#adding dependency
dax.depends(f2, f1)
```

Call the package for calculating the multifractal spectrum for each realization

This step feeds the data from each realization separately into the package which performs a multifractal analysis. One can replace the analysis package to do other analyses.

Collect data from different package calls

```
cat = Job("cat")
cat.addArguments(*files)
for f in files:
    cat.uses(f, link=Link.INPUT)
output = File("binaries.txt")
cat.setStdout(output)
cat.uses(output, link=Link.OUTPUT, transfer=True, register=False)
dax.addJob(cat)
```

Collecting all the output files into a single file

This step collects the output data from the above multifractal analysis for each realization. A standard linux tool "cat" is used here. Again, if the data were is in hdf5 format this can be done by calling hdf5 library instead.

Feed the collected data into a package for averaging

```
ave = File("ave.dat")
f3 = Job("f3")
f3.addProfile( Profile("pegasus","label","p1"))
f3.addArguments(",output)
f3.setStdout(ave)
f3.uses(output, link=Link.INPUT)
f3.uses(ave, link=Link.OUTPUT, transfer=True, register=False)
dax.addJob(f3)
```

Finally, calculate the averaged multifractal spectrum

The last step of this calculation involves another pipeline workflow which feeds the above file which contain data from all realizations into a function for averaging. One can replace this by other function if desired.

Conclusions and future works

We demonstrate the application of the Pegasus workflow management system by an example. This example calculates the averaged multifractal spectrum of a large pool of realizations from two-dimensional surface simulations. This example contains most of the steps appeared in typical scientific workflow which can be represented in term of a direct acyclic graph. These included pipelining workflow, split workflow, and merge workflow. These three workflow elements can be considered as the building blocks. More complicated workflow based on these three workflow building blocks can be built easily.

The workflow management system allows us to organize complicated computational tasks which involve multiple calling of different packages and data format transformations.

Acknowledgements

Supported by the National Science Foundation through cooperative agreement OIA-1541079 and the Louisiana Board of Regents.

References

E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, Pegasus: a Workflow Management System for Science Automation, Future Generation Computer Systems, vol. 46, pp. 17-35, 2015.