Distribution of simulation jobs on cluster of machines

(Sridharan Rajagopalan, sr443@uw.edu)

I want to bring to your kind attention that testing this code requires access to the Baker labarotory clusters. I will be more than willing to go over the demonstratio of the code over coffee. I surely owe one for your all of your kind help throughout the course

Summary of research questions and results

1. Is it possible to write code (from CSE 140 course) where I could automate the process of distributing computations in different machines?
   yes, it is possible
2. Can I make it general in a way that I could use for any kind of job distribution?
   may be it is possible, but I am currently not there yet
3. I learned how to (effectively) communicate between different machines, count the free nodes available on each machine and decide on the best way to launch jobs
4. Is it possible to do system calls and command-line execution on different machines?
   yes, it is

Motivation and background

In Baker laboratory,dept of biochemistry, we do simulations of proteins on a regular basis. This involves submitting simulation jobs on machines (we call it digs) we have. Currently, the way I do is I log into each machine (ssh) and then submit the jobs manually by pressing enter key!

There are packages such as Parallel (GNU) that is written in perl and could be used for this purpose, I wanted to write my own code that I could have better grip of and spit out the error if an exception occurs.

Dataset:

The protein dataset is available at www.pdb.org. Since we use this regularly, proteins from this database are available at one common place in Baker lab clusters, accessible to people from Baker Lab.my script involves looking up at this home-built database and do desired calculations on the set of proteins provided as a text input.

Methodology (algorithm or analysis).

I have defined this problem in various functions, where individual functions could be useful at later stages of development. I initially aimed at creating the list of proteins and the corresponding commandline (each protein will have a unique commandline) in generator (yielding instead of returning), so that I do not take too much memory. However, I had difficulty in resuming the generator at the last point (ie it does not pick up from where it is left, but starts all over again). so I went back to returning the list of proteins and iterating through them.

If you execute it as follows,

```
./Job_Distribution.py -help
```

it gives instructions on what input is necessary and what is optional.

The correct way is
./Job_Distribution.py -file pdblist2 -cstfile Orgpho_S_H_DE_oxya_FP1_match.cst -node 10

The algorithm involves generating the list of input proteins on which simulation has to be done. for each of the protein, check if the data for that protein is available in our home database, if yes, then prepare the commandline for that protien. If the protein does not exist in our database, then open a file called “Notfound.txt” and write to that file. At the end of the script, the file contains the names of proteins on which simulation has not been done (because they are not present in home database)

One nice feature I put in is to not to overload the machines with computations. I mean if I have 20 nodes available for running simulations, I should not overload with more than 20 jobs. If I have more than 20 jobs, I will submit 20 of them and check periodically (right now I kept 10 min wait time) for nodes that get free and submit the remaining jobs iteratively. In this way I do not have to worry about overloading or not optimally using the available nodes.

I have one routine which if called, kill all the jobs in all the machine under my username. This function is important because sometimes the simulation goes wrong for various reasons and will start consuming lot of memory. In a such case, I should kill the jobs.

Results.
I tested on different scenarios such as follows;

1. if number of jobs > nodes_available, I should wait and submit appropriately

2. if number of jobs < nodes_available, all jobs will be submitted once.

3. If the optional argument -nodes (in the commandline) is given, that would overwrite the number of nodes to use (although the total nodes are much higher.)

4. I imported some of the functions into another script and tested its efficiency. I also tried to stop the jobs by calling a specific funciton called ‘killalldigs’ that would kill all the jobs

Reproducing your results.
The results are quite reproducible in my hand. After a few more tests, I will hand them over to my colleagues so that they could let me know if they come up with some bugs

Colloboration:
No one did

Reflection:
This project is very useful in my everyday work. I now can submit my 1000 of simulations jobs by just pressing once the return key!

This code is could be used in other clusters such as Hyak with slight modifications. In that way, I am glad that I have it written it more general