A Quick Manual for Automatic SLURM Helper(ASH)

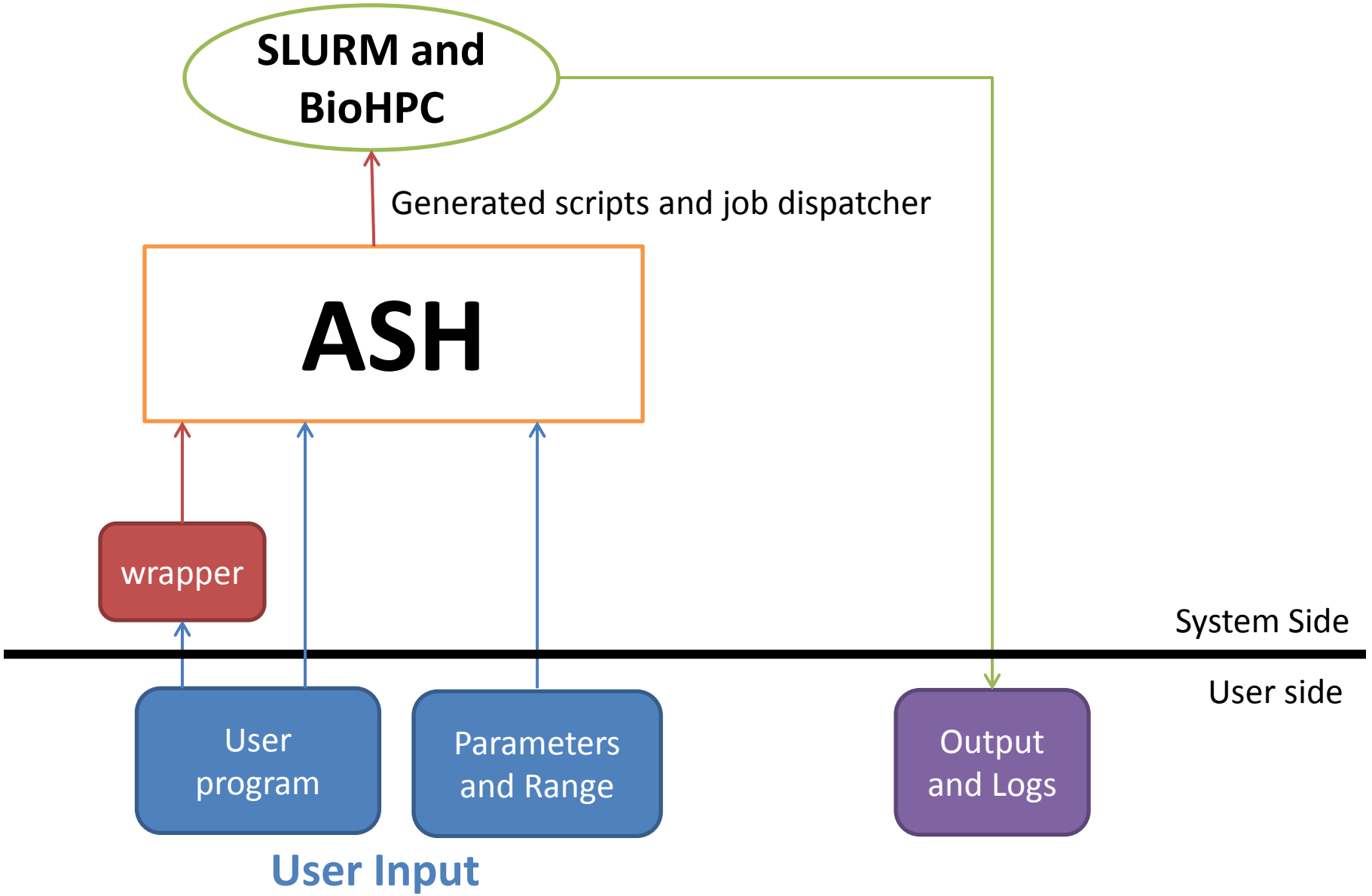
Shenghua Wan

04/21/2014

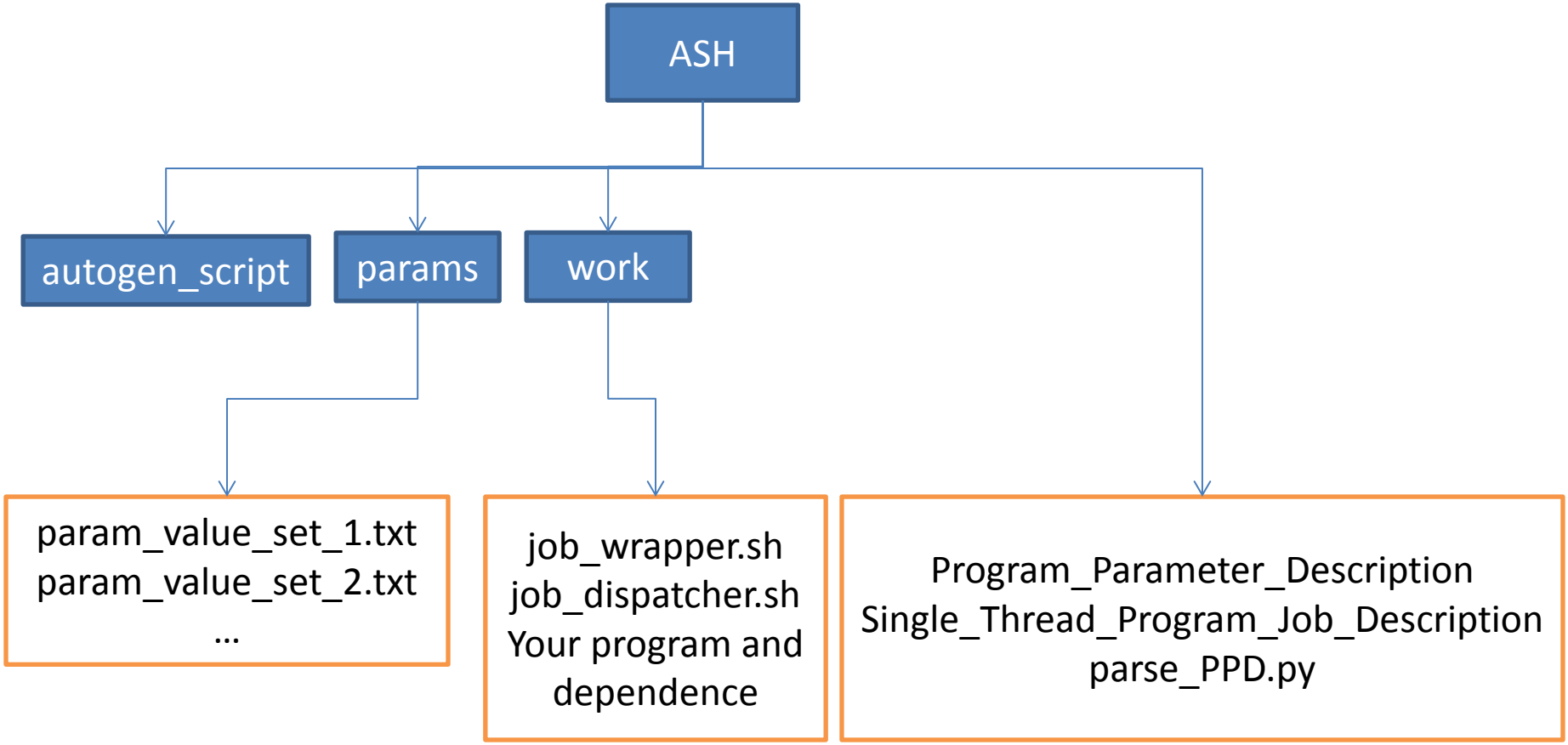
Introduction

- Automatic SLURM Helper (ASH) is a framework that helps you submit a large number structured jobs, e.g. evaluating combinations of parameters, to a cluster managed by SLURM. It will choose a partition of the cluster and monitor the available nodes in that partition. Whenever there exists some nodes available, it will feed your jobs to that partition. After giving a description of your job to the configuration of ASH, you can run all the jobs in one click.

Overview


SLURM and BioHPC
Generated scripts and job dispatcher
ASH
wrapper
System Side
User side
User program
Parameters and Range
Output and Logs
User Input

Overview-Folder Hierarachy



How It Works-Summary

1. **Put your program and dependent files in subfolder “work”**
2. **Give the program name and full path to it.**
3. Indicate the number and the range of the each positional parameters.
4. Indicate the number of processes or threads that you want to run on each node.
5. Run script generator.
6. Run script dispatcher.

How It Works-Give Program Name

- **First, give the program name and full path.**
- General format: ASH assumes your program in the following format

  full-path-to-prog param_1 param_2 … param_N

where full-path-to-prog is your program name with full path given, i.e. starting with ‘/’.

- Other formats, e.g. MATLAB command-line call, can be converted to this one by a provided wrapper script discussed later.

How It Works-General Format

- General format: ASH assumes your program in the following format

full-path-to-prog param_1 param_2 … param_N

- If your program interface is in this form, you do not need job_wrapper.sh. Just edit file Program_Parameter_Description.

Edit file: Program_Parameter_Description

Sample of Program_Parameter_Description

##Description
This template works for the following pattern.
myprog P_1 P_2 ... P_N
##where
"myprog" is the name of your program, and
P_K (K is in an integer set [1,N]) is a K-th parameter given to the program, and
the possible values are given in param_value_set_K.txt.
comments begins with '#' and will be ignored.
#0. program path + name
/home2/swan/code-swan/slurm-helper/work/myprog
or if you need a wrapper: /home2/azaritsky/HTC_scripts/slurm-helper-v2/work/job_wrapper.sh
#1. number of parameters
2
#2. parameter value set, e.g. param1.txt contains the possible values for first parameter (the default folder is ./params)
param_value_set_1.txt
param_value_set_2.txt

How It Works-Other Formats

- If the invocation of your program is in formats other than

  full-path-to-prog param_1 param_2 … param_N

e.g. matlab -nodisplay -nodesktop -nosplash -singleCompThread -r "runQLCHOneAtATime(job_id, task_id);"

- You will need job_wrapper.sh to convert the interface to general format.
- As usual you also need to edit file Program_Parameter_Description

  Edit file: Program_Parameter_Description and job_wrapper.sh

Sample of Program_Parameter_Description for other formats

#0. program path + name
/home2/azaritsky/HTC_scripts/slurm-helper-v2/work/job_wrapper.sh
#1. number of parameters
2
#2. parameter value set, e.g. param1.txt contains the possible values for first parameter (the default folder is ./params)
param_value_set_1.txt
param_value_set_2.txt

Edit file: Program_Parameter_Description and job_wrapper.sh

Sample of job_wrapper.sh

```
#!/bin/bash

module add matlab/2013a

cd /home2/azaritsky/HTC_scripts/slurm-helper-v2/work

#interpret the arguments according to the application interfaces
job_id=$1
task_id=$2

matlab -nodisplay -nodesktop -nosplash -singleCompThread -r
"runQLCHOneAtATime($job_id,$task_id);"
```

Edit file: Program_Parameter_Description and job_wrapper.sh

How It Works-Summary

1. Give the program name and full path to it.
2. **Indicate the number and the range of the each positional parameters.**
3. Indicate the number of processes or threads that you want to run on each node.
4. Run script generator.
5. Run script dispatcher.

How It Works-Indicate Parameters

- **Second, indicate the number and the range of the positional parameters.**

e.g. your program's input is in the form of

full-path-to-prog p1 p2

i.e. there are two parameters, and the range of p1 is a discrete integer set {1,2,3} and the range of p2 is a set of strings {DNA, RNA}.

In other words, the combinations you want to try is (1, DNA), (2, DNA), (3, DNA), (1, RNA), (2, RNA), (3, RNA)

Edit file: Program_Job_Description

How It Works-Indicate Parameters cont.

- According to previous page, you want try the combination of two parameters whose range sets are {1,2,3} and {DNA, RNA}.
- **Prepare two files in subfolder "params":**

  param_value_set_1.txt

  param_value_set_2.txt

And put one value in a separate line.

In **param_value_set_1.txt**
1
2
3

In **param_value_set_2.txt**
DNA
RNA

How It Works-Summary

1. Give the program name and full path to it.
2. Indicate the number and the range of the each positional parameters.
3. **Indicate the number of processes or threads that you want to run on each node.**
4. Run script generator.
5. Run script dispatcher.

How It Works-Pack Multiple Processes on A Node

- Indicate the number of processes running on each node.
- This is a step to lift the efficiency of BioHPC system.

e.g. run 20 processes simultaneously

In **Single_Thread_Program_Job_Description**

20

Edit file: Single_Thread_Program_Job_Description

How It Works-Summary

1. Give the program name and full path to it.
2. Indicate the number and the range of the each positional parameters.
3. Indicate the number of processes or threads that you want to run on each node.
4. **Run script generator.**
5. Run script dispatcher.

How It Works-Script Generator

- Run python parse_PPD.py
- The script generator will read the description files and generate scripts in a subfolder called “autogen_script”.

Program_Parameter_Description

Single_Thread_Program_Job_Description

How It Works-Summary

1. Give the program name and full path to it.
2. Indicate the number and the range of the each positional parameters.
3. Indicate the number of processes or threads that you want to run on each node.
4. Run script generator.
5. **Run script dispatcher.**

How It Works-Script Dispatcher

- cd work
- python job_dispatcher.py TIME INDEX QUEUE
- Time is number of seconds to check the node availability when the queue is full.

(Use 300 for default. A tiny value will put heavy load on the submission node, e.g. Nucleus005)

- INDEX is the starting job index for current run. It is used for interruption recovery. Use 0 for default and when interruption happens, consult the log for an index to pick up what is left.
- Queue is the partition to submit jobs to. Default is 128GB. Currently this framework is only allowed to run on this partition to ensure availability of BioHPC.
- Please consult the BioHPC administration if you want to use another partition.

How It Works-Summary

1. Give the program name and full path to it.
2. Indicate the number and the range of the each positional parameters.
3. Indicate the number of processes or threads that you want to run on each node.
4. Run script generator.
5. Run script dispatcher.

Misc

- If you want to kill all the jobs submitted by yourself, run

```
squeue | grep USERID | awk '{print $1}' | xargs scancel
```

Replace USERID to your BioHPC ID.

This is useful during the debugging when you want to stop and cancel all the submitted jobs.

Pitfalls: Other running jobs that is not submitted by ASH will also be cancelled.