SKILLPILLS

Skill Pill: Terminal

Lecture 3: Sango

Valentin Churavy

Okinawa Institute of Science and Technology
v.churavy@gmail.com

March 10, 2017

OIST

Overview

SKILLPILLS

1. Bash scripts
2. Remote systems
   - SSH
   - Copying files a file with `scp`
3. Sango
   - First steps
   - Scheduler – SLURM

What are bash scripts?!

Bash scripts

Bash files are a collection/series of bash commands that are executed line by line. In essence bash files are scripts, and scripts are computer programs. Writing bash scripts is thus programming.

```
1  #!/bin/bash
2
3  # Simple hello world example.
4  echo "Hello world"
```

Excuting a bash script

bash script.sh or

chmod +x script.sh; ./script.sh

Variables

Variables in bash are untyped. A variable can be a number, a character or a string of character. There is also a limited forms of lists/arrays. Variables are often uppercase.

```
1  # Assignment (no space before or after =)!
2  a=30
3
4  # Referencing
5  echo $a
6
7  b=$a
8  echo $b
9
10 STRING="Hello world"
11 echo $STRING
```

Calling programs

Bash is a glue-language it is easy to call and interact with other programs and glue their outputs together.

```
1  # You can just use a program like you do in the shell
2  echo "Hello!" > output.txt
3
4  # We can also capture the output of a program in a variable
5  TEXT=$(cat output.txt)
6  echo ${TEXT}
```

Interpolation

```
1  echo "Your hostname is $(hostname)"
2  N=10
3
4  echo "N is equal to ${N}"
```

A simple for loop

```
1  for i in 1 2 3 4 5; do
2    echo $i
3  end
4
5  for i in `seq 1 10`; do
6    echo $i
7  done
8
9  ARRAY=("item1" "item2" "item3")
10 for elem in ${ARRAY[@]}; do
11   echo $elem
12 done
```

Everything else

Test

Bash has a lot more features to offer and I can't cover them all today. There are conditionals (if and else statements) $\cdots$

`http://www.tldp.org/LDP/Bash-Beginners-Guide/html/`

SSH – a simple yet powerful command

```
1    ssh [username@]server[:port] [cmd]
```

Useful options

`-X`	X11-forwarding in order to run graphical programs
`-D port`	Forward traffic from local port via ssh connection
`-t`	Execute pseudo-tty for shell like programs.

SSH keys

SSH keys are based on public-private encryption. One piece of information is secret and anoter is public. The secret bit is used to uniquely identify you!

```
# Generate new key
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

# Copy key to server (if you are on a mac this won't work :( )
ssh-copy-id [username@]server
```

An exploration of ~/.ssh/

```
1 ls ~/.ssh
2 config
3 id_rsa # KEEP THIS PRIVATE!
4 id_rsa.pub # This is your public key
5 known_hosts
```

~/.ssh/config

```
1  Host sango
2      Hostname sango.oist.jp
3      User valentin-churavy
4
5  Host login.oist
6      Hostname login.oist.jp
7      User valentin-churavy
```

SSH config

For hosts that we access often we can add entries in our `~/.ssh/config` file. Also see the manual file for ssh_config.

Copying file

scp

In order to copy a file to a server use:

```
1    scp localfile server:remotefile
```

and vice-versa:

```
1    scp server:remotefile localfile
```

To copy a file to your remote homefolder:

```
1    scp localfile server:.
```

scp takes many of the same arguments that cp takes!

Copying a file with rsync

rsync

rsync is a lot smarter then cp or scp and tries to minimise the amount of work necessary. It compares files that are already on at the destination and only copise files if they have changed!

```
1    rsync -a /source/path/ /destination/path/
```

It also recognizes remote servers in the scp format.

`-v` Verbose output

`-a` archive mode!

`--progress` show the progress we are making.

What is Sango?

Sango is our local supercomputer. It uses SLURM as a scheduler and is maintained by the Scientific Compute Section.

SCS

Information about Sango is available at `https://groups.oist.jp/scs`.
Open hours are: Every weekday 15:30 – 17:30 at B648.
Contact them through `it-help@oist.jp`

Activate account

Go to `https://groups.oist.jp/scs/registration-forms` to signup for an account. If you currently don't have an account you can use the test-system tombo for now!

Supercomputer etiqutte

- Be a good citizen!
- Don't be shy to ask question or to go to SCS if you have problems.
- Remember the Sango is a shared resource
- Don't run computational intense programs on the login nodes!
- Limit the resources you request.

Setup your user-directory

Check if you are already setup.

```
ls ~/.bashrc ~/.bash_profile

# if not execute
cp /etc/skel/.bashrc /etc/skel/.bash_profile ~/.
```

Now copy your ssh key to sango and tombo!

Accessing Sango from the outside world

After you have copied your ssh key to sango you should be able to access `login.oist.jp`

What is a scheduler

A scheduler allocates the resources of a supercomputer with the goal to maximise cluster utilisation.

- Time
- Memory
- CPUs: (Cores, Sockets, $\cdots$)
- Special resources:
  - GPUs
  - Network
  - $\cdots$

Allocating resources

```
1  salloc -n 1 -t 30 # one task for 30mins
```

`-p` Select the partition to run on.

`-n` Number of tasks

`-t` The time your job will need. (defaults to 8h)

`-c` Cpus per task.

`--mem-per-cpu` Memory required per cpu

```
1    srun --pty bash
```

`--x11` Forward the X11 server from the node.

`--pty` Connects your tty session with the node.

The `srun` command also takes alot of the same option as `salloc` and can be used independently. Formally `srun` is used to start a task withing an allocation.

Example: Running matlab

```
1 ssh -X sango
2 module load matlab
3 srun -c 4 --mem=8g --x11 --pty matlab
```

Finding information about our jobs

`sinfo -o "%P %C"` Cluster usage

`sango_usage` Cluster usage

`squeue -u ${USER}` Your queued jobs

`sstat -j <job-id>` Statistics about a running job

`scancel <job-id>` Cancel a specific job

`scancel -t PENDING -u ${USER}` Cancel all your pending jobs

```
#!/bin/bash

#SBATCH --job-name=test_ucsc
#SBATCH --partition=compute
#SBATCH --time=00:30:00
#SBATCH --mem-per-cpu=1G
#SBATCH --ntasks=1
#SBATCH --mail-user=%u@oist.jp
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --input=none
#SBATCH --output=job_%j.out
#SBATCH --error=job_%j.err

hostname
```

```
sbatch job_script.slurm
```

Other information

Slurm is very powerful and I recommend reading up on it.

Job arrays

Start jobs 16 jobs and limit to 4 jobs simultaneous.ys

```
1 sbatch --array=0-15%4
2 # inform yourself about SLURM_ARRAY_TASK_ID
```

Jobsteps

Within an allocation or a sbatch file you can use `srun` to start jobsteps that use a subset of the resources allocated.

The module system

A good bit of software is already installed on sango sue `module avail` to see what is available and `module load` to use the software. If you need special software talk to SCS!

Next SCS tutorial session

May 9th and 10th, from 13:15 in C700.