UNIL | Université de Lausanne

Universität Zürich[UZH]

High throughput computing

Simon Scheidegger
simon.scheidegger@gmail.com
July 19th, 2018
Open Source Macroeconomics Laboratory – BFI/UChicago

High throughput computing/Cloud

Some definitions

- **Advanced Computing** – a very generic term for any kind of computing beyond mainstream consumer/business computing.

- Generally implies something about **scale**:
  → Fastest components
  → Aggregating lots of components

- Sometimes means 'just' **advanced software**, but usually implies **advanced hardware**.

- Example: Argonne National Lab systems, MIDWAY

  → support HPC, visualization, big data storage, high throughput computing, data analysis, etc.

More definitions (1)

Cyber - infrastructure:

The integration of potentially diverse computers, displays and visualizations, data, storage systems, instruments, sensors, etc. via software and networks (and policies/procedures?) to:

- **Provide comprehensive capabilities**
- **Provide aggregate capabilities** (sometimes)
- **Share resources** (sometimes)

More definitions (2)

Supercomputing, Parallel Computing, High Performance Computing (HPC):

→ **Supercomputing**: older term, originated when Cray and CDC supercomputers (1960-80s) were vastly more powerful than any other computers.

→ Still used for any kind of **tightly coupled, large-scale** parallel computing systems.

→ **Parallel computing: aggregating computers or processors to address problems by dividing up the work.**

→ **HPC: now, refers to parallel computing at large scale.**

More definitions (3)

- **Distributed Computing**: very general term, includes any kind of computing spanning more than one computer connected by a network.

- We all used **distributed** computer every day (connecting to the Internet for web,email, etc.).

- Examples: everything from email to web browsing to file sharing to B2B (business to business) applications to clusters to remote visualization to grids to clouds to....

More definitions (4)

- Grid Computing (see also next slide):

using distributed computing with standards to enable using multiple, often geographically distributed, computing resources, for such reasons as:

→ Resource sharing
→ Data Sharing
→ Work-flow applications
→ Aggregating multiple systems (coupled applications, “metacomputing,” high throughput computing, etc.)

Examples:
usually done for ‘science/research’ but also for business, with examples like “World Community Grid*”, etc.

https://www.worldcommunitygrid.org/discover.actio

Grid Computing Definition*

*https://en.wikipedia.org/wiki/Grid_computing

Grid computing is the most distributed form of parallel computing.

It makes use of computers communicating over the Internet to work on a given problem.

Because of the low bandwidth and extremely **high latency** available on the Internet, distributed computing typically deals only with **embarrassingly parallel** problems.

Many distributed computing applications have been created, of which **SETI@home, LHC@home and Folding@home** are the best-known examples.

Most grid computing applications use middle-ware, software that sits between the operating system and the application to **manage network resources** and standardize the software interface.

Often, distributed computing software makes use of "spare cycles", performing computations at times when a computer is idling.

Definition of cloud computing

Cloud computing enables users to consume a compute resource, such as a virtual machine (VMs), storage or an application, as a utility – just like electricity – rather than having to build and maintain computing infrastructures in house.

Cloud computing boosts several attractive benefits for end users. Three of the main benefits of cloud computing are:

→ Self-service provisioning: End users can spin up compute resources for almost any type of **workload on demand**. This eliminates the traditional need for IT administrators to provision and manage compute resources.

→ Elasticity: Users can scale up as computing needs increase and scale down again as demands decrease. This eliminates the need for massive investments in local infrastructure which may or may not remain active.

→ Pay per use: Compute resources are measured at a granular level, allowing users to pay only for the resources and workloads they use.

Examples: Amazon's EC2 compute service; Amazon S3 storage service; Apple's MobileMe (storage, file sharing, etc.); Flickr, Netflix, YouTube; etc.

A Microsoft Azure Center

High throughput computing (HTC)

- **High-throughput computing (HTC)** is a computer science term to describe the use of many computing resources over long periods of time to accomplish a computational task.

- Using distributed computing (potentially grid computing) to enable lots of jobs to be scheduled to available resources to complete as fast as possible.

- The term was popularized e.g. by Condor project (U Wisconsin).

- Examples: Condor, World Community Grid, LHC project, Open Science Grid, etc.

- Goal: to integrate multiple computing systems to enable large numbers of tasks to be schedule and completed as rapidly as possible. Resources: can be centrally managed servers (clusters, clouds) and/or distributed PCs.

The value of HTC

- Majority of computational science is performed on workstations/PCs
  - → HPC systems not needed for all tasks.
  - → Many tasks need to be repeated a lot.

- Running simulations to explore a parameter space
  - → Running simulations on different data sets.
  - → Analyzing experimental results that ongoing/repeated.

- HTC tools enable this, sometimes from the desktop (e.g. Condor)

- Think of Big Data analytics... (http://lhcathome.web.cern.ch/)

Example – HTCondor

https://research.cs.wisc.edu/htcondor/
http://chtc.cs.wisc.edu/

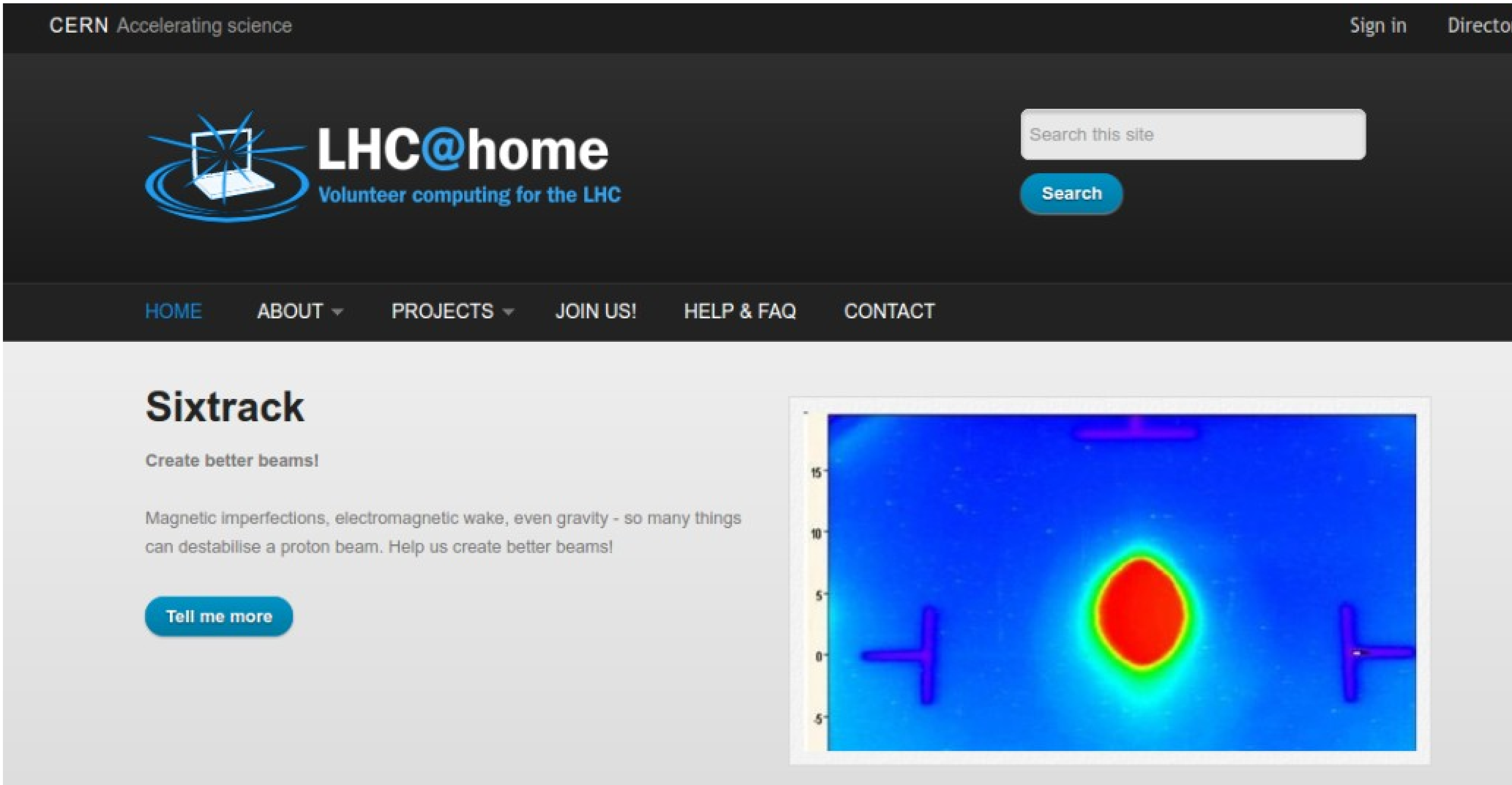
Many-Task Computing

- **Many-task computing is a new term, arguably the same as high throughput computing**: “the execution of independent, sequential jobs that can be individually scheduled on many different computing resources across multiple administrative boundaries” (Ian Foster - UChicago)

- Primary difference is the scale of the tasks is much shorter, increasing emphasis on scalability of the enabling infrastructure.

Example – LHC@home

http://lhcathome.web.cern.ch/



Some Compute Clouds

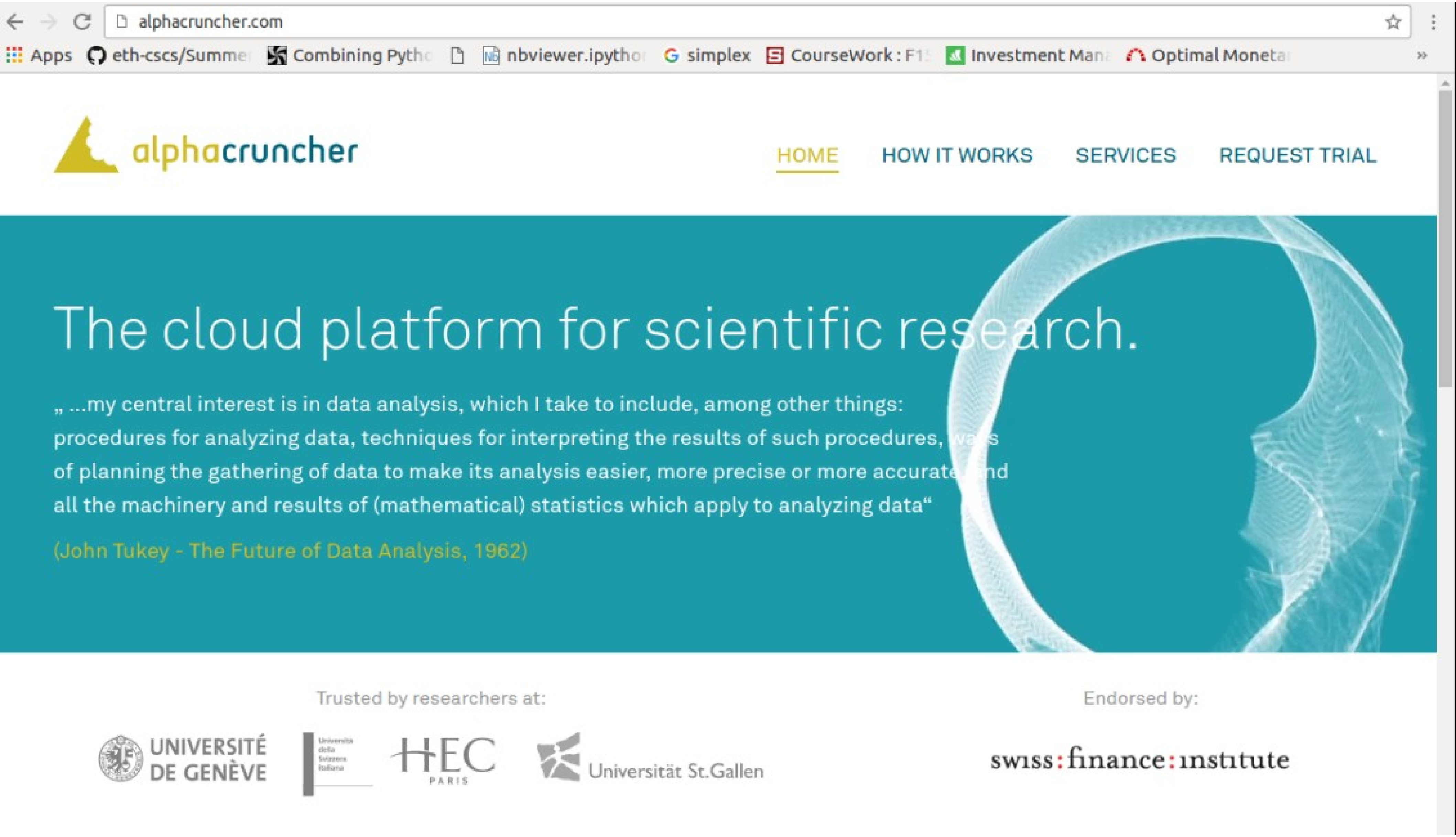
Science Cloud UZH
- https://www.s3it.uzh.ch/infrastructure/sciencecloud/

Virtual Workspaces
- http://workspace.globus.org/

Amazon Elastic Cloud (costs $, but very little)
- aws.amazon.com/ec2

Google Cloud
- https://cloud.google.com

Another example



Batch processing

Batch processing is the execution of a series of jobs in a program on a computer without manual intervention (non-interactive). Strictly speaking, it is a processing mode: the execution of a series of programs each on a set or "batch" of inputs, rather than a single input (which would instead be a custom job). However, this distinction has largely been lost, and the series of steps in a batch process are often called a "job" or "batch job" .

Batch processing has these benefits:
- It can shift the time of job processing to when the computing resources are less busy.
- It **avoids idling** the computing resources with minute-by-minute manual intervention and supervision.
- By keeping high overall rate of utilization, it amortizes the computer, especially an expensive one.
- It **allows the system to use different priorities** for interactive and non-interactive work.
- Rather than running one program multiple times to process one transaction each time, batch processes will run the program only once for many time, reducing system overhead.

Example: Slurm – Array jobs

- **Array jobs** can be used to create a **sequence of jobs** that share the same executable and resource requirements, but have different input files, to be submitted, controlled, and monitored as a single unit.

- The arguments **-a** or **--array** take an additional parameter that specify the **array indices**.

- Within the job you can read the environment variables **SLURM_ARRAY_JOB_ID**, which will be set to the first job ID of the array, and **SLURM_ARRAY_TASK_ID**, which will be set individually for each step.

- Within an array job, you can use %a and %A in addition to %j and %N to make the output file name specific to the job. %A will be replaced by the value of SLURM_ARRAY_JOB_ID and %a will be replaced by the value of SLURM_ARRAY_TASK_ID.

- Here is an example how an array job can looks like:

```
#!/bin/bash
#SBATCH -J Science1
#SBATCH --array 0-9
#SBATCH -o arraytest-%A_%a.out
#SBATCH -e arraytest-%A_%a.err
#SBATCH --ntasks=864
#SBATCH --mail-type=end
#SBATCH –mail-user=your.name@gmail.com
#SBATCH --time=08:00:00
echo "Hi, I am step $SLURM_ARRAY_TASK_ID in this array
job $SLURM_ARRAY_JOB_ID"
```

Array jobs on MIDWAY

Documentation: http://slurm.schedmd.com/job_array.html
Exmaple: https://www.rc.colorado.edu/support/examples-and-tutorials/array-jobs.html

- Ordinarily you have an input dataset, perhaps one file per intended job index. The example input dataset here is simply a set of randomly-generated, one-megabyte files.

- **sha1sum-array.sh** is a Slurm job that expects to run as an index of a Slurm job array. It writes the calculated SHA-1 hash to the --output file as well as to a dedicated hash data file.

```
#!/bin/bash

#SBATCH –job-name=ExampleJob_OSM
#SBATCH --time=5:00
#SBATCH --nodes=1
#SBATCH --output example-array-%a.out
#SBATCH --array=1-100%20

if [ -z "${SLURM_ARRAY_TASK_ID}" ]
then
    echo 1>&2 "Error: not running as a job array."
    exit 1
fi

echo "Array index: ${SLURM_ARRAY_TASK_ID}"

data_file="input-data-${SLURM_ARRAY_TASK_ID}"
sha1sum $data_file | tee "${data_file}.sha1"
```

```
for i  in $(seq 0 99)
do
  dd if=/dev/urandom of=input-data-${i} bs=1MB count=1
done
```

1. go to /OSE2019/day4/code_day4/array_jobs:
> cd OSE2019/day4/code_day4/array_jobs

2. Have a look at the code
> vi sha1sum_array.sh and generate_random_input_file.sh

Array jobs on MIDWAY (2)

```
sbatch sha1sum_array.sh
Submitted batch job 30982261
[simonsch@midway-login1 ~]$ squeue -u simonsch
          JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
      30982261_32    sandyb sha1sum- simonsch CF       0:01      1 midway078
      30982261_33    sandyb sha1sum- simonsch CF       0:01      1 midway078
      30982261_34    sandyb sha1sum- simonsch CF       0:01      1 midway249
      30982261_35    sandyb sha1sum- simonsch CF       0:01      1 midway249
      30982261_36    sandyb sha1sum- simonsch CF       0:01      1 midway249
      30982261_37    sandyb sha1sum- simonsch CF       0:01      1 midway249
      30982261_38    sandyb sha1sum- simonsch CF       0:01      1 midway249
      30982261_39    sandyb sha1sum- simonsch CF       0:01      1 midway452
      30982261_40    sandyb sha1sum- simonsch CF       0:01      1 midway452
      30982261_21    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_22    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_23    sandyb sha1sum- simonsch CF       0:03      1 midway077
      30982261_24    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_25    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_26    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_27    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_28    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_29    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_30    sandyb sha1sum- simonsch CF       0:03      1 midway078
      30982261_31    sandyb sha1sum- simonsch CF       0:03      1 midway078
30982261_[41-100%2    sandyb sha1sum- simonsch PD       0:00      1 (JobArrayTaskLimit)
```

Example: Slurm – chain jobs

- You can use chain jobs to **create dependencies between jobs**.

- This is often the case if a **job relies on the result of one or more preceding jobs**.

- Chain jobs can also be used if the runtime limit of the batch queues is not sufficient for your job. SLURM has an option **-d** or "**--dependency**" that allows to specify that a job is only allowed to start if another job finished.

- Here is an example how a chain job can looks like, the example submits 4 jobs (described in a job file) that will be executed on after each other with different CPU numbers:

```
#!/bin/bash
TASK_NUMBERS="1 2 4 8"
DEPENDENCY=""
JOB_FILE="myjob.slurm"
for TASKS in $TASK_NUMBERS ; do
    JOB_CMD="sbatch --ntasks=$TASKS"
    if [ -n "$DEPENDENCY" ] ; then
        JOB_CMD="$JOB_CMD --dependency
afterany:$DEPENDENCY"
    fi
    JOB_CMD="$JOB_CMD $JOB_FILE"
    echo -n "Running command: $JOB_CMD  "
    OUT=`$JOB_CMD`
    echo "Result: $OUT"
    DEPENDENCY=`echo $OUT | awk '{print $4}'`
done
```

More on Array jobs

https://rcc.uchicago.edu/docs/running-jobs/array/index.html

Questions?

1. Advice – http://lmgtfy.com/

http://lmgtfy.com/?q=cloud+computing