Managing task-parallel computations with jobber

Thomas Orgis, Hinnerk Stüben

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

ZKI-Arbeitskreis Supercomputing PC^2 Paderborn
28. April 2022

`https://thomas.orgis.org/jobber`

teaser

- university compute cluster with very diverse user base
- many inexperienced users with serial or at most node-local parallel computations
- enabling them to fill cluster node allocations sensibly
- prerequisite: learning to phrase the set of tasks as shell script lines as universally applicable skill
- no need for details on the batch system, re-use personal setup on different sites

teaser example

- medical parameter study with statistics over generated data sets of certain size (one crucial parameter) and combinations of analysis methods
- researcher knows R and the science at hand, but not much else regarding computer use
- a bit of help to pass parameters into the existing R script from shell wrapper

teaser example

- medical parameter study with statistics over generated data sets of certain size (one crucial parameter) and combinations of analysis methods
- researcher knows R and the science at hand, but not much else regarding computer use
- a bit of help to pass parameters into the existing R script from shell wrapper

The task that seemed impossible until end-of-month deadline even with reduced problem size was easily finished through dozens of loosely coordinated jobs filling gaps in the cluster.

teaser example

- medical parameter study with statistics over generated data sets of certain size (one crucial parameter) and combinations of analysis methods
- researcher knows R and the science at hand, but not much else regarding computer use
- a bit of help to pass parameters into the existing R script from shell wrapper

The task that seemed impossible until end-of-month deadline even with reduced problem size was easily finished through dozens of loosely coordinated jobs filling gaps in the cluster.

(Researcher was rather happy about that and did not hesitate to try the larger data set sizes after all.)

teaser batch script

```
#!/bin/bash
#SBATCH --job-name=someRstuff
#SBATCH --partition=std
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --time=06:00:00
#SBATCH --export=NONE
. /sw/batch/init.sh

module switch env env/2020Q3-gcc-openmpi
module load R/4.0.2
module load jobber/r1254

# Simplest case. Could run multiple jobs of differing runtime
# sequentially or in parallel here (with or without srun).
jobber task.list 1

if jobber task.list more; then
  sbatch $0
fi
```

teaser job chain control

- stop execution of job chain:
  `jobber task.list stop`
- enable executon again:
  `jobber task.list start`
- make things more parallel:
  `for n in $(seq 1 $N); do sbatch batch.sh; done`
- check progress:
  `jobber task.list done failed todo`

but first ...

Why was this inevitable?

once upon a time, a physics student

- Data analysis for student lab experiments via office spreadsheet software?

once upon a time, a physics student

- Data analysis for student lab experiments via office spreadsheet software?
- Write Perl scripts for common computations on measurements in text files (Text::NumericData on CPAN) and for automated plotting using scriptable tools.

taking the routine out of routine

- Working in an experimental physics group where routine measurements were routinely analysed in the same ways using recurring mouse click patterns in glorified office spreadsheet software?

taking the routine out of routine

- Working in an experimental physics group where routine measurements were routinely analysed in the same ways using recurring mouse click patterns in glorified office spreadsheet software?
- Locate the scripting mechanism in that software, but in the end …

taking the routine out of routine

- Working in an experimental physics group where routine measurements were routinely analysed in the same ways using recurring mouse click patterns in glorified office spreadsheet software?
- Locate the scripting mechanism in that software, but in the end ...
- ... write more of the text data scripts and configurations for analysis pipes with a GUI for the colleagues to do the routine analyses on a batch of measurements.

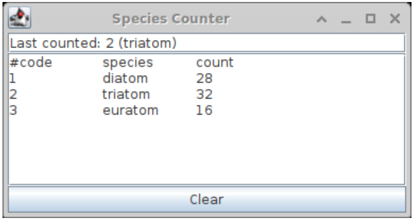
counting with your fingers

- Working alongside biologists who look at dead things through microscopes, counting different types of dead things, manually writing down counts in spreadsheets software?

counting with your fingers

- Working alongside biologists who look at dead things through microscopes, counting different types of dead things, manually writing down counts in spreadsheets software?
- Write a trivial Java GUI that counts key presses for a massive productivity boost

counting with your fingers



```
Last counted: 2 (triatom)
#code       species     count
1           diatom      28
2           triatom     32
3           euratom     16
```

(a useful Java program, 2009 vintage)

thin films and thick benches

- Working on thin-film solar cell simulation using a big and expensive GUI around command line tools (Sentaurus TCAD Workbench), trying to organize lots of parameter studies with interdependencies?

thin films and thick benches

- Working on thin-film solar cell simulation using a big and expensive GUI around command line tools (Sentaurus TCAD Workbench), trying to organize lots of parameter studies with interdependencies?
- Write shell-based replacement using Makefiles and a file hierarchy to handle dependencies of simulations.

thin films and thick benches

- Working on thin-film solar cell simulation using a big and expensive GUI around command line tools (Sentaurus TCAD Workbench), trying to organize lots of parameter studies with interdependencies?
- Write shell-based replacement using Makefiles and a file hierarchy to handle dependencies of simulations.
  (**Bonus:** Does not crash all the time!)

a pattern

Organize things differently for fun and profit!

solving my own problems

- **simple big problem:** high-resolution simulation run that can be split into consecutive pieces on the time axis

solving my own problems

- **simple big problem:** high-resolution simulation run that can be split into consecutive pieces on the time axis
- parallelize with OpenMP and/or MPI, just one chain of jobs or a single big one if HPC site permits it

solving my own problems

- **simple big problem:** high-resolution simulation run that can be split into consecutive pieces on the time axis
- parallelize with OpenMP and/or MPI, just one chain of jobs or a single big one if HPC site permits it (we rather don't;-)

solving my numerous problems

- **lots** of **little problems** amounting to a big one: a parameter study with varying resolutions and dynamics, a pile of model runs with varying runtimes

solving my numerous problems

- **lots** of **little problems** amounting to a big one: a parameter study with varying resolutions and dynamics, a pile of model runs with varying runtimes
- lots of trial and error with job parameters, re-running failed computations

something out there

- Keeping track in spreadsheets? No!

something out there

- Keeping track in spreadsheets? No!
- Domain-specific, fat, hungy & fragile GUI? No!

something out there

- Keeping track in spreadsheets? No!
- Domain-specific, fat, hungy & fragile GUI? No!
- Got something that nicely works as a simple shell tool? Well ...

not ...

- **array jobs:** specific to batch system and less flexible

not ...

- **array jobs:** specific to batch system and less flexible (but you can use `jobber joblist exec:$id` in array jobs just fine)

not ...

- **array jobs:** specific to batch system and less flexible (but you can use `jobber joblist exec:$id` in array jobs just fine)
- just **srun** for parallelization: specific to batch system and not keeping track

not …

- **array jobs:** specific to batch system and less flexible
  (but you can use `jobber joblist exec:$id` in array jobs just fine)
- just **srun** for parallelization: specific to batch system and not keeping track
  (but maybe `srun jobber task.list all` for distribution of taks across multiple job nodes/cores instead of
  `jobber --parallel=$cores task.list all`)

not ...

- **array jobs:** specific to batch system and less flexible (but you can use `jobber joblist exec:$id` in array jobs just fine)
- just **srun** for parallelization: specific to batch system and not keeping track (but maybe `srun jobber task.list all` for distribution of taks across multiple job nodes/cores instead of `jobber --parallel=$cores task.list all`)
- **GNU parallel**: not keeping track and wanting clever rules to construct task commands

not …

- **array jobs:** specific to batch system and less flexible
  (but you can use `jobber joblist exec:$id` in array jobs just fine)
- just **srun** for parallelization: specific to batch system and not keeping track
  (but maybe `srun jobber task.list all` for distribution of taks across multiple job nodes/cores instead of
  `jobber --parallel=$cores task.list all`)
- **GNU parallel**: not keeping track and wanting clever rules to construct task commands
  (manual rather explicit in that it is a really complex tool to master)

not ...

- **array jobs:** specific to batch system and less flexible (but you can use `jobber joblist exec:$id` in array jobs just fine)
- just **srun** for parallelization: specific to batch system and not keeping track (but maybe `srun jobber task.list all` for distribution of taks across multiple job nodes/cores instead of `jobber --parallel=$cores task.list all`)
- **GNU parallel**: not keeping track and wanting clever rules to construct task commands (manual rather explicit in that it is a really complex tool to master)
- **moreutils parallel:** simpler design, but also not keeping track or coordinating instances

not ...

- **array jobs:** specific to batch system and less flexible (but you can use `jobber joblist exec:$id` in array jobs just fine)
- just **srun** for parallelization: specific to batch system and not keeping track (but maybe `srun jobber task.list all` for distribution of taks across multiple job nodes/cores instead of `jobber --parallel=$cores task.list all`)
- **GNU parallel**: not keeping track and wanting clever rules to construct task commands (manual rather explicit in that it is a really complex tool to master)
- **moreutils parallel:** simpler design, but also not keeping track or coordinating instances
- **xargs:** similar, good for ad-hoc parallelism

jobber features

- organize a list of tasks, posed as **lines of shell script**
- **syncronize** access while picking tasks
- record and **manage state** of each task, to know what to re-run
- **pack** multiple small/short tasks to **better fill job time slots and allocated compute resources**
- work on the same tasks in **batch system** or just a random **personal computer**, possibly going back and forth
- easy **job chains** via **generic** batch script

generic batch script

```
#!/bin/bash
#SBATCH --job-name=manytasks
# we got 16-core nodes, not shared
#SBATCH --nodes=1
#SBATCH --time=12:00:00

# cluster-specific init stuff
# maybe load some modules

# Pack  many single-processor-jobs into our time slot,
# until time runs out.
jobber --parallel=16 --time=$((11*3600)) task.list all

# Continue work in next batch job.
if jobber task.list more; then
  sbatch $0
fi
```

(pretty generic)

one way (for HPC people)

- master-worker program using OpenMP on a node
- some number of OMP threads
- `#pragma omp critical` section around a function that determines the next piece of work to do, if any
- actual work with custom code, or maybe using `system()`

another way (for HPC people)

- proper master-worker program using MPI
- central process distributing tasks
- `MPI_Recv()` and `MPI_Send()` on both sides
- worker processes getting bits of work from master
- maybe centralized I/O

another way (for HPC people)

- proper master-worker program using MPI
- central process distributing tasks
- `MPI_Recv()` and `MPI_Send()` on both sides
- worker processes getting bits of work from master
- maybe centralized I/O

Who writes real programs these days?

jobber implementation

- **no central server process** for management (serverless;-)
- safe operations on a single database **file with POSIX rename semantics**, also on network file systems
- **flush and sync**, see Stewart Smith: „Eat My Data - How everybody gets file I/O wrong“ (`https://www.slideshare.net/nan1nan1/eat-my-data`, `https://www.youtube.com/watch?v=LMe7hf2G1po`)
- behaves **like a database** with server backend, but still just **plain files** for data and control
- limitation: transaction rate low, but no issue for sensible tasks needing **more than a few seconds** each

one thing left to do

```
#!/bin.sh
LANG=C; set -ex

test -e jobber ||
wget https://thomas.orgis.org/jobber/jobber

test -e jobber.sig ||
wget https://thomas.orgis.org/jobber/jobber.sig

#gpg --search-key thomas@orgis.org
gpg --verify jobber.sig

chmod +x jobber
mkdir -p man1
./jobber -h=-100 | pod2man > man1/jobber.1
./jobber -h=-100 | pod2text > jobber.txt
```