Consortium des Equipements
de Calcul Intensif
en Fédération Wallonie-Bruxelles

Introduction to Scientific Workflow Management

david.waroquiers@uclouvain.be
damien.francois@uclouvain.be
November 2017

You follow workflows...

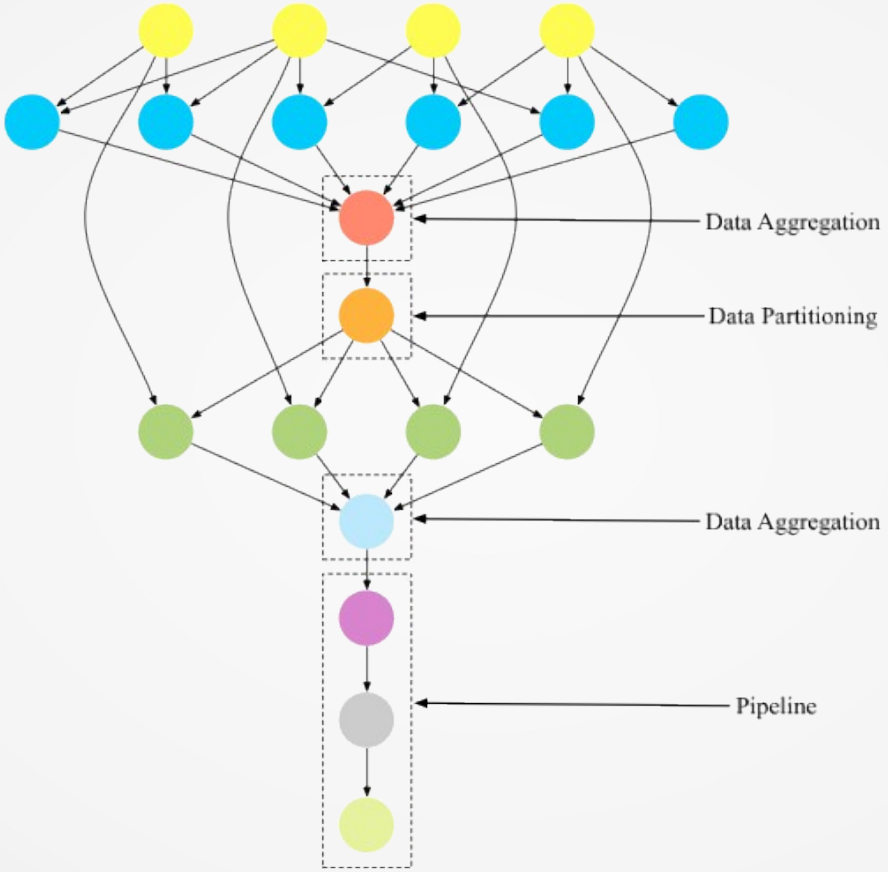
"A workflow is a precise description of a scientific procedure — a multi-step process to coordinate multiple tasks, acting like a sophisticated script"

P. Romano, "Automation of in-silico data analysis processes through workflow management systems," Brief Bioinform, vol. 9, no. 1, pp. 57–68, Jan. 2008

Yours can be simple..

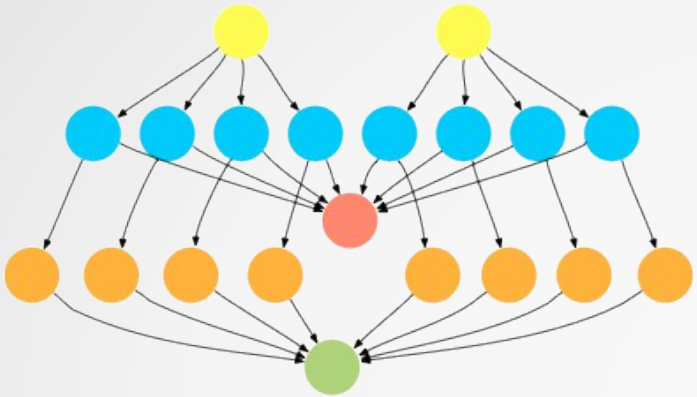
Adapted from http://dx.doi.org/10.1109/WORKS.2008.4723958

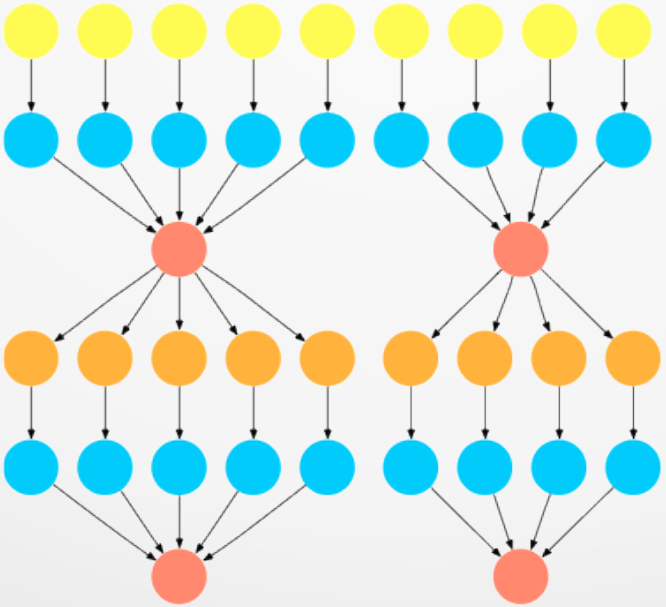
.. or complex

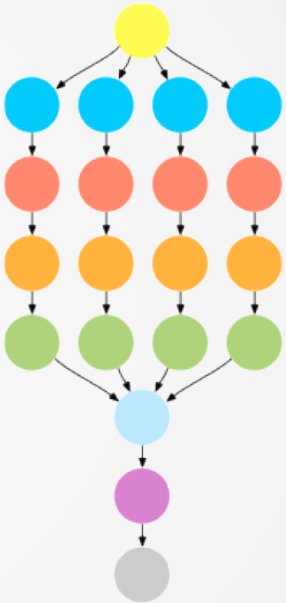


http://dx.doi.org/10.1109/WORKS.2008.4723958

.. or complex

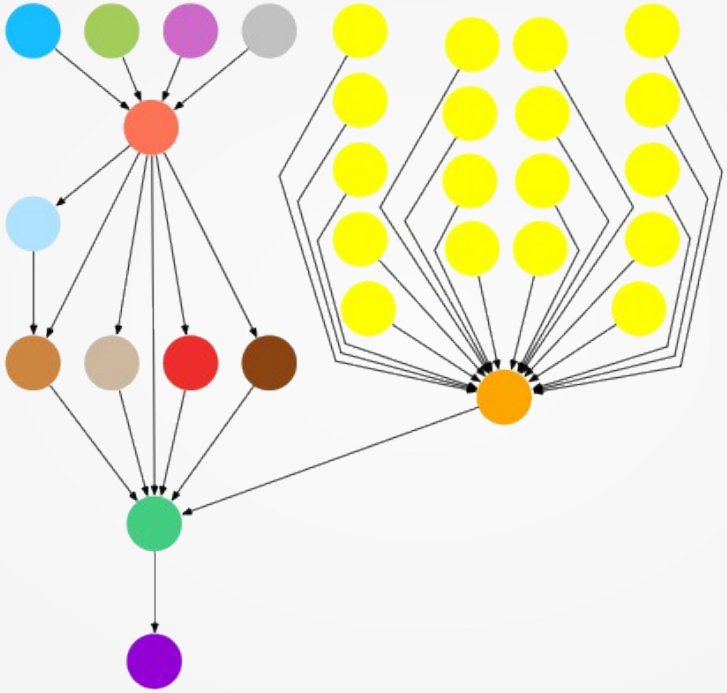






http://dx.doi.org/10.1109/WORKS.2008.4723958

.. or even bizarre



http://dx.doi.org/10.1109/WORKS.2008.4723958

By hand?

- Error prone
- Cumbersome
- Does not scale

- How do you share?
- How do you explain?
- How do you recall provenance?

Workflow systems

Platforms that offer

(1) invocation of the service applications and handling the heterogeneity of data types and interfaces on multiple computing platforms;
(2) monitoring and recovery from failures;
(3) optimization of memory, storage, and execution, including concurrency and parallelization;
(4) data handling: mapping, referencing, movement, streaming, and staging;
(5) logging of processes and data provenance tracking; and
(6) security and monitoring of access policies.

C. Globe & D. Roure. in The fourth Paradigm: Data-Intensive Scientific Discover, pp 137-145, Microsoft Research, 2009

Using a workflow system?

- Error prone
- Cumbersome
- Does not scale

No as it is automated !

- How do you share?
- How do you explain?
- How do you recall provenance?

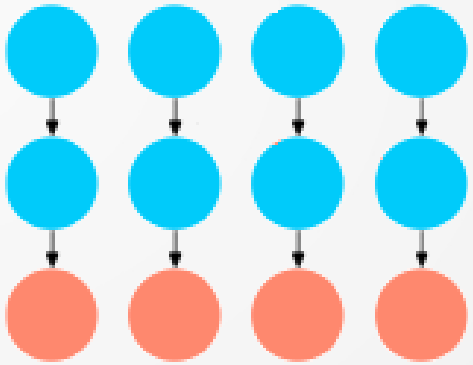
All in one file !

GNU make

- Remember we used GNU make as a parallel computing tool?
- We can use it as a simple workflow management too as well

```
# Sample Makefile to process each file with
# lower.sh then upper.sh
#
all: d1.res d2.res d3.res d4.res

# Build intermediary files
%.tmp: %.txt
    ./lower.sh $< $@


# Build final result
%.res: %.tmp
    ./upper.sh $< $@
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Makefile" 14L, 219C written
```

=



GNU make

- Remember we used GNU make as a parallel computing tool?
- We can use it as a simple workflow management too as well

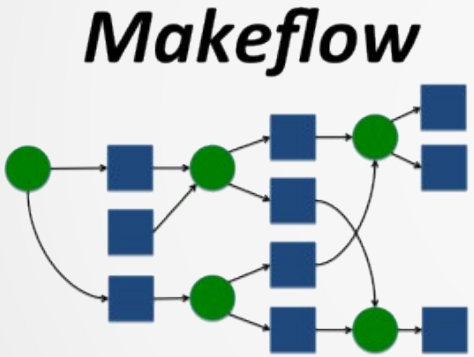
```
# Sample Makefile to process each file with
# lower.sh then upper.sh
#
all: d1.res d2.res d3.res d4.res

# Build intermediary files
%.tmp: %.txt
    ./lower.sh $< $@


# Build final result
%.res: %.tmp
    ./upper.sh $< $@
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"Makefile" 14L, 219C written
```

Nice little trick in Make v4: put

```
SHELL=srun
.SHELLFLAGS= <slurm options> bash -c
```

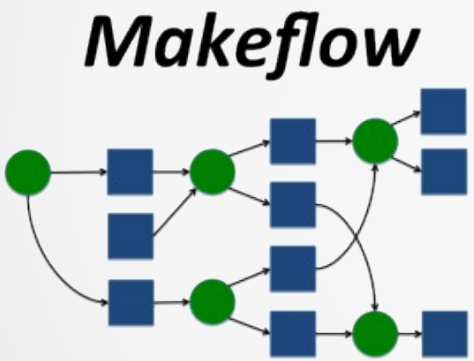
at the beginning of you Makefile
to run all commands through Slurm

Makeflow = Make + Workflow



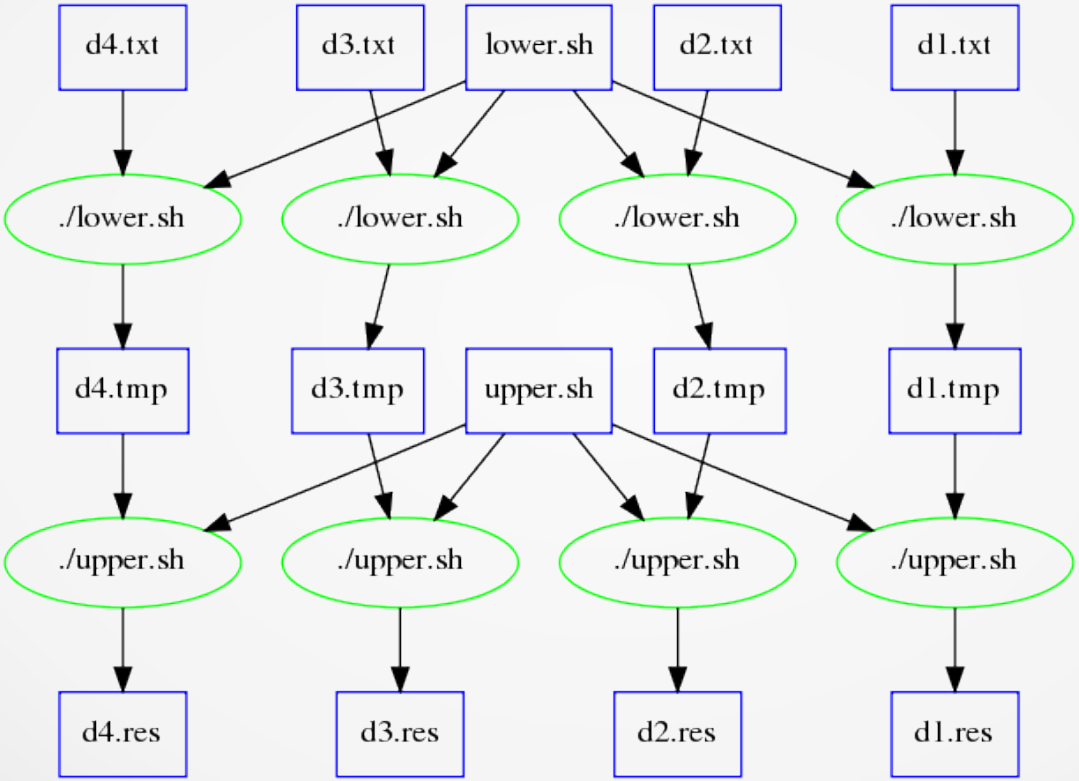

```
# Sample Makeflow to process each file with
# lower.sh then upper.sh
#
d1.tmp: d1.txt lower.sh
    ./lower.sh d1.txt d1.tmp

d2.tmp: d2.txt lower.sh
    ./lower.sh d2.txt d2.tmp

d3.tmp: d3.txt lower.sh
    ./lower.sh d3.txt d3.tmp

d4.tmp: d4.txt lower.sh
    ./lower.sh d4.txt d4.tmp

d1.res: d1.tmp upper.sh
    ./upper.sh d1.tmp d1.res

d2.res: d2.tmp upper.sh
    ./upper.sh d2.tmp d2.res

d3.res: d3.tmp upper.sh
    ./upper.sh d3.tmp d3.res

d4.res: d4.tmp upper.sh
    ./upper.sh d4.tmp d4.res
```

Makeflow = Make + Workflow




```
dfr@hmem00:~/parcomp $ makeflow -T slurm Makefile.expanded
parsing Makefile.expanded...
checking Makefile.expanded for consistency...
Makefile.expanded has 8 rules.
recovering from log file Makefile.expanded.makeflowlog...
starting workflow....
submitting job: ./lower.sh d4.txt d4.tmp
submitted job 643315
submitting job: ./lower.sh d3.txt d3.tmp
submitted job 643316
submitting job: ./lower.sh d2.txt d2.tmp
submitted job 643317
submitting job: ./lower.sh d1.txt d1.tmp
submitted job 643318
job 643315 completed
submitting job: ./upper.sh d4.tmp d4.res
submitted job 643319
job 643316 completed
submitting job: ./upper.sh d3.tmp d3.res
submitted job 643320
job 643317 completed
submitting job: ./upper.sh d2.tmp d2.res
submitted job 643321
job 643318 completed
submitting job: ./upper.sh d1.tmp d1.res
submitted job 643322
job 643319 completed
job 643320 completed
job 643321 completed
job 643322 completed
nothing left to do.
```

Dependency graph



makeflow_viz -D dot Makefile.expanded > example.dot && dot -Tpng < example.dot > res.png

Passing Slurm options

Batch Job Refinement

When executing jobs, Makeflow simply uses the default settings in your batch system. If you need to pass additional options, use the `BATCH_OPTIONS` variable or the `-B` option to Makeflow.

When executing jobs, Makeflow simply uses the default settings in your batch system. If you need to pass additional options, use the `BATCH_OPTIONS` variable or the `-B` option to Makeflow.

When using Condor, this string will be added to each submit file. For example, if you want to add `Requirements` and `Rank` lines to your Condor submit files, add this to your Makeflow:

```
BATCH_OPTIONS = Requirements = (Memory>1024)
```

When using SGE, the string will be added to the qsub options. For example, to specify that jobs should be submitted to the `devel` queue:

```
BATCH_OPTIONS = -q devel
```


http://ccl.cse.nd.edu/software/manuals/makeflow.html

Other options

dfr@hmem00 — bash

```
 -f,--summary-log=<file>         Write summary of workflow to this file upon success or failure.
 -j,--max-local=<#>              Max number of local jobs to run at once.    (default is # of cores)
 -J,--max-remote=<#>             Max number of remote jobs to run at once.
                                                  (default 1000 for -Twq, 100 otherwise.)
 -l,--makeflow-log=<logfile>     Use this file for the makeflow log.         (default is X.makeflowlog)
 -L,--batch-log=<logfile>        Use this file for the batch system log.     (default is X.<type>log)
 -m,--email=<email>              Send summary of workflow to this email address upon success or failure.
 -o,--debug-file=<file>          Send debugging to this file. (can also be :stderr, :stdout, :syslog, or :journal)
 -R,--retry                      Automatically retry failed batch jobs up to 100 times.
 -r,--retry-count=<n>            Automatically retry failed batch jobs up to n times.
    --wait-for-files-upto=<n>    Wait for output files to be created upto n seconds (e.g., to deal with NFS semantics).
 -S,--submission-timeout=<#>     Time to retry failed batch job submission.  (default is 3600s)
 -X,--change-directory           Change directory: chdir to enable executing the Makefile in other directory.
 -z,--zero-length-error          Force failure on zero-length output files

*Monitor Options:

 -M,--monitor=<dir>              Enable the resource monitor, and write the monitor logs to <dir>.
    --monitor-limits=<file>      Use <file> as value-pairs for resource limits.
    --monitor-interval=<#>       Set monitor interval to <#> seconds.        (default is 1 second)
    --monitor-with-time-series   Enable monitor time series.              (default is disabled)
    --monitor-with-opened-files Enable monitoring of openened files.        (default is disabled)
    --monitor-log-fmt=<fmt>      Format for monitor logs.                    (default resource-rule-%06.6d)


~
~
~
~
~
~
~
                                                                                    26,0-1        All
```

Makeflow

- Syntax similar to make
- Able to submit jobs to Slurm
- Simple but useful

- No file input/output management
- Does not scale to hundreds of jobs

Enters

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe."
- Abraham Lincoln