EURO
BELGIUM

A short

Introduction to Workflows

in a HPC context.

damien.francois@uclouvain.be – Calcul Intensif et Stockage de Masse

Q: What do you do when to want to run computations on a cluster?

A: you write a submission script and submit it to the scheduler (e.g. Slurm)

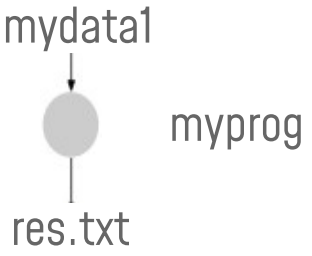
```
#!/bin/bash
# Submission script for demonstrating
# slurm usage.

# Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
# Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00

# Operations
echo "Job start at $(date)"
# Job steps
srun ~/bin/myprog < mydata1

echo "Job end at $(date)"

~
                          19,0-1        All
```



```
#!/bin/bash
# Submission script for demonstrating
# slurm usage.

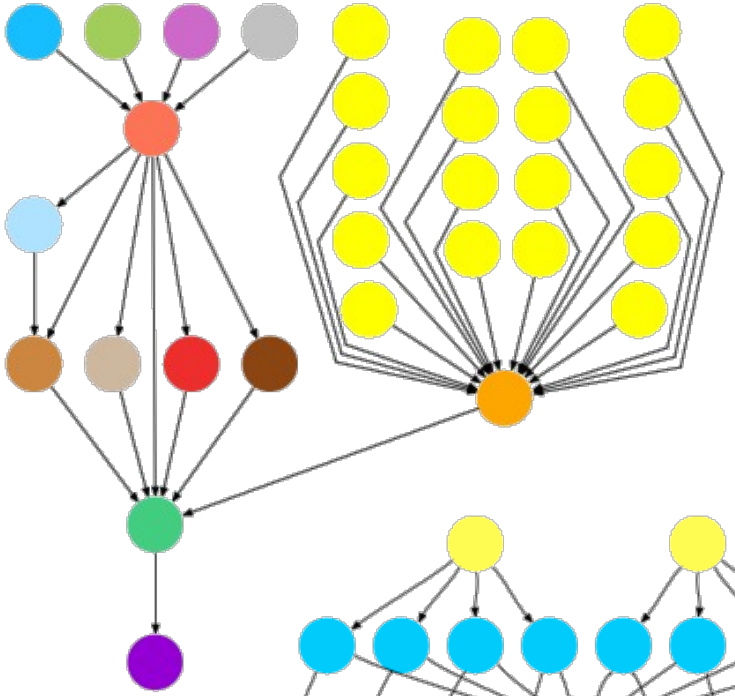
# Job parameters
#SBATCH --job-name=demo
#SBATCH --output=res.txt
# Needed resources
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=2000
#SBATCH --time=1:00:00

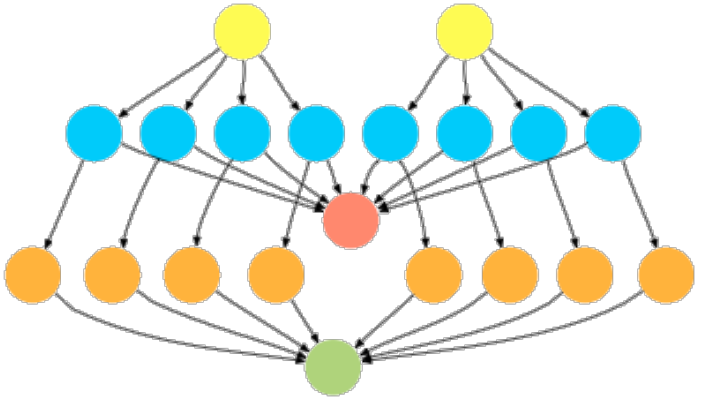
# Operations
echo "Job start at $(date)"
# Job steps
srun ~/bin/myprog < mydata1

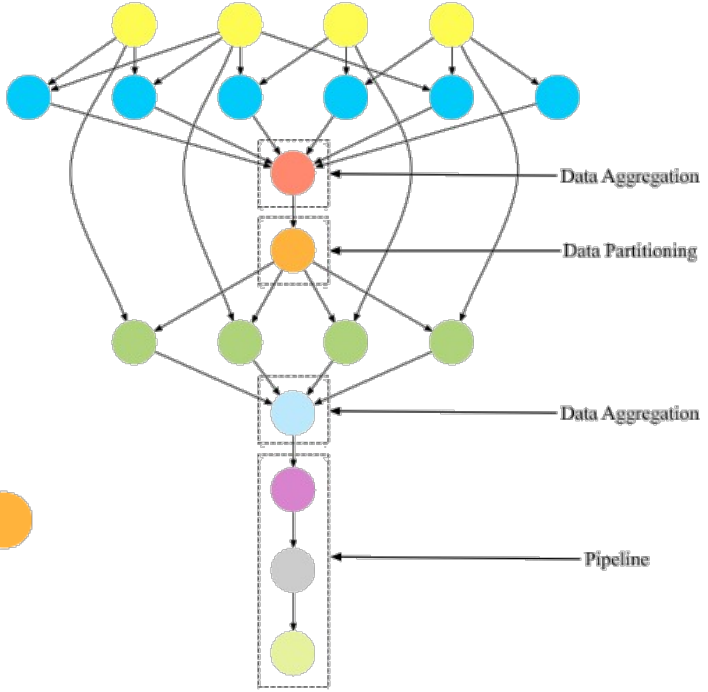
echo "Job end at $(date)"
~
                              19,0-1        All
```

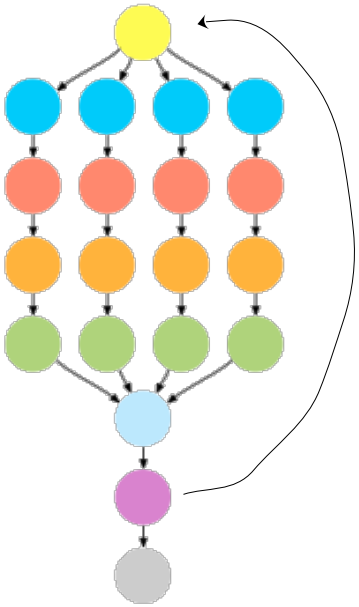


Q: What if …









A: you need to think “workflows”

Workflows

- exist in business, healthcare, administration, science, etc.

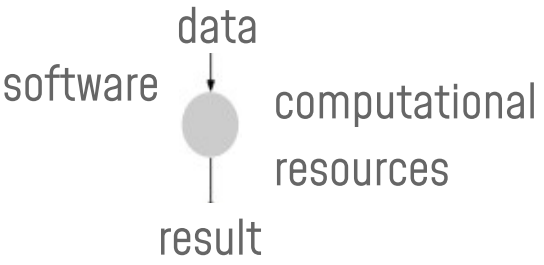
> "*A workflow is a precise description of a scientific procedure – a multi-step process to coordinate multiple tasks, acting like a sophisticated script*"

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

- also exist in IT operations, machine learning, Internet of Things, etc.

Workflows

- a list of tasks or operations (the “work”)
- a set of dependencies between tasks (the “flow”)
- but also
  - a set of data sources
  - computational resources
  - scientific software



Workflows by hand...

- error prone
- cumbersome
- complex to share
- difficult to track provenance



Workflow management systems

can do some (or all) of the following:

- Compute dependencies and organise work
- Submit jobs to the scheduler
- Generate job descriptions (templating, sweeping, etc.)
- Install scientific software
- Monitor jobs and recover from failures, fault detection, “smart” reruns
- Handle data: mapping, referencing, movement, streaming, and staging
- Log processes and data provenance tracking
- Enable sharing of data, results, workflows, with security and monitoring of access policies.
- Provide performance analysis and prediction

An example you already know of...



2.2 A Simple Makefile

Here is a straightforward makefile that describes the way an executable file called `edit` depends on eight object files which, in turn, depend on eight C source and three header files.

In this example, all the C files include `defs.h`, but only those defining editing commands include `command.h`, and only low level files that change the editor buffer include `buffer.h`.

```
edit : main.o kbd.o command.o display.o \
       insert.o search.o files.o utils.o
        cc -o edit main.o kbd.o command.o display.o \
                   insert.o search.o files.o utils.o

main.o : main.c defs.h
        cc -c main.c
kbd.o : kbd.c defs.h command.h
        cc -c kbd.c
command.o : command.c defs.h command.h
        cc -c command.c
display.o : display.c defs.h buffer.h
```

https://www.gnu.org/software/make/manual/html_node/Simple-Makefile.html

https://github.com/mitjafelicijan/makext

GNU Make

can do some (or all) of the following:

- Compute dependencies and organise work
- ~~Submit jobs to the scheduler~~
- ~~Generate job descriptions (templating, sweeping, etc.)~~
- ~~Install scientific software~~
- ~~Monitor jobs and recover from failures, fault detection,~~ "smart" reruns
- ~~Data handling: mapping, referencing, movement, streaming, and staging~~
- ~~Log processes and data provenance tracking~~
- ~~Enable sharing of data, results, workflows, with security and monitoring of access policies.~~
- ~~Provide performance analysis and prediction~~

Actually, with a little trick...



GNU-Make version **4** was recently released. This new version comes with a number of improvements like **GNU Guile integration**, **Loadable objects** (see http://plindenbaum.blogspot.fr/2014/08/a-gnu-make-plug-in-for-illumina-fastqs.html). It also allows to specify the default shell to be invoked (see http://plindenbaum.blogspot.fr/2014/01/parallelizing-rstats-using-make.html)

> http://www.gnu.org/software/make/manual/make.html : The program used as the shell is taken from the variable **SHELL**. If this variable is not set in your makefile, the program /bin/sh is used as the shell. The argument(s) passed to the shell are taken from the variable **.SHELLFLAGS**. The default value of **.SHELLFLAGS** is -c normally, or -ec in POSIX-conforming mode.

So, if you want to parallelize GNU-Make with **SLURM** you can wrap the shell into **srun** using **SHELL** and **.SHELLFLAGS**. Here is an example, creating and concatenating 100 files containing the hostname:

```
ifdef SLURM_JOB_ID
SHELL=srun
.SHELLFLAGS= -N1 -n1  bash -c
endif
```

http://plindenbaum.blogspot.com/2014/09/parallelizing-gnu-make-4-in-slurm.html

GNU Make

can do some (or all) of the following:



- Compute dependencies and organise work
- Submit jobs to the scheduler
- ~~Generate job descriptions (templating, sweeping, etc.)~~
- ~~Install scientific software~~
- ~~Monitor jobs and recover from failures, fault detection,~~ "smart" reruns
- ~~Data handling: mapping, referencing, movement, streaming, and staging~~
- ~~Log processes and data provenance tracking~~
- ~~Enable sharing of data, results, workflows, with security and monitoring of access policies.~~
- ~~Provide performance analysis and prediction~~

Workflow management systems

- ~~error prone~~ safe
- ~~cumbersome~~ convenient
- ~~complex~~ easy to share
- ~~difficult~~ simple to track provenance

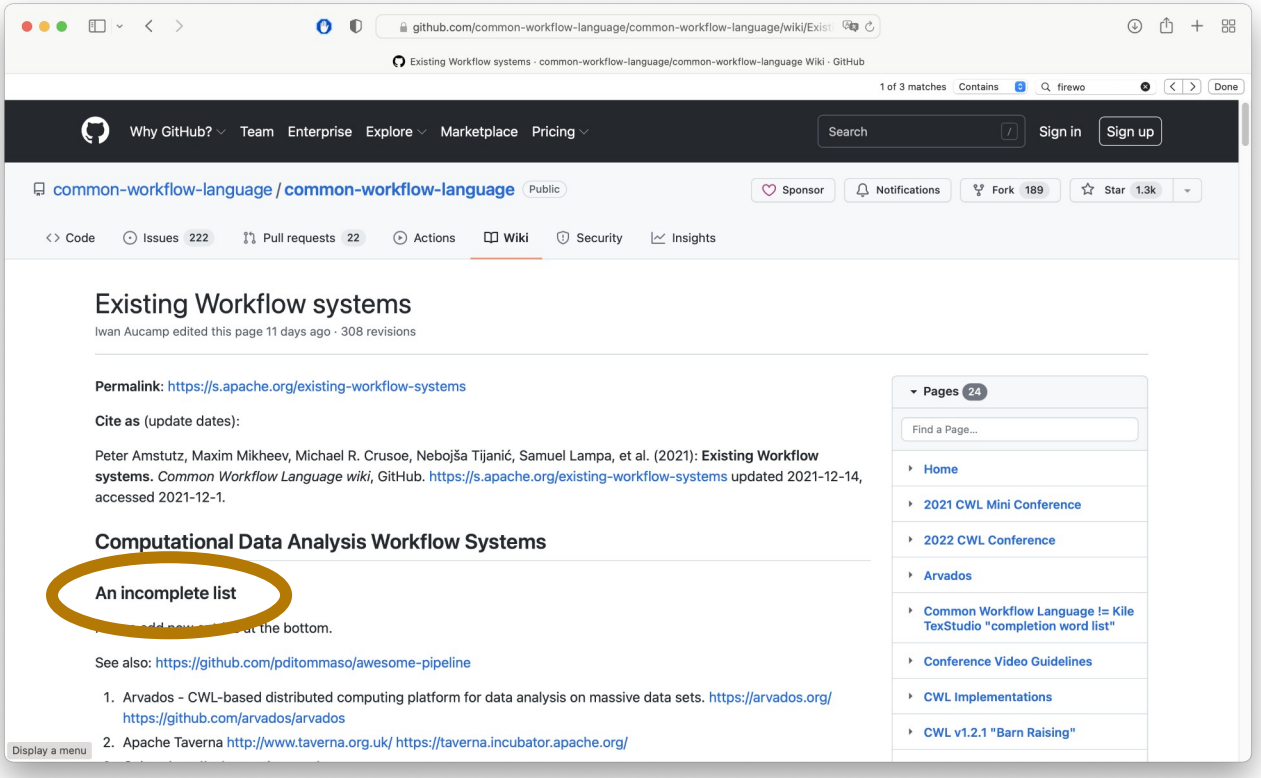


> *"The main goals of scientific workflows, then, are (i) to save "human cycles" by enabling scientists to focus on domain-specific (science) aspects of their work, rather than dealing with complex data management and software issues; and (ii) to save machine cycles by optimizing workflow execution on available resources."*

G. Scherp, W. Hasselbring Towards a model-driven transformation framework for scientific workflows 2010, Procedia Computer Science 1(1):1519-1526

Workflow management systems

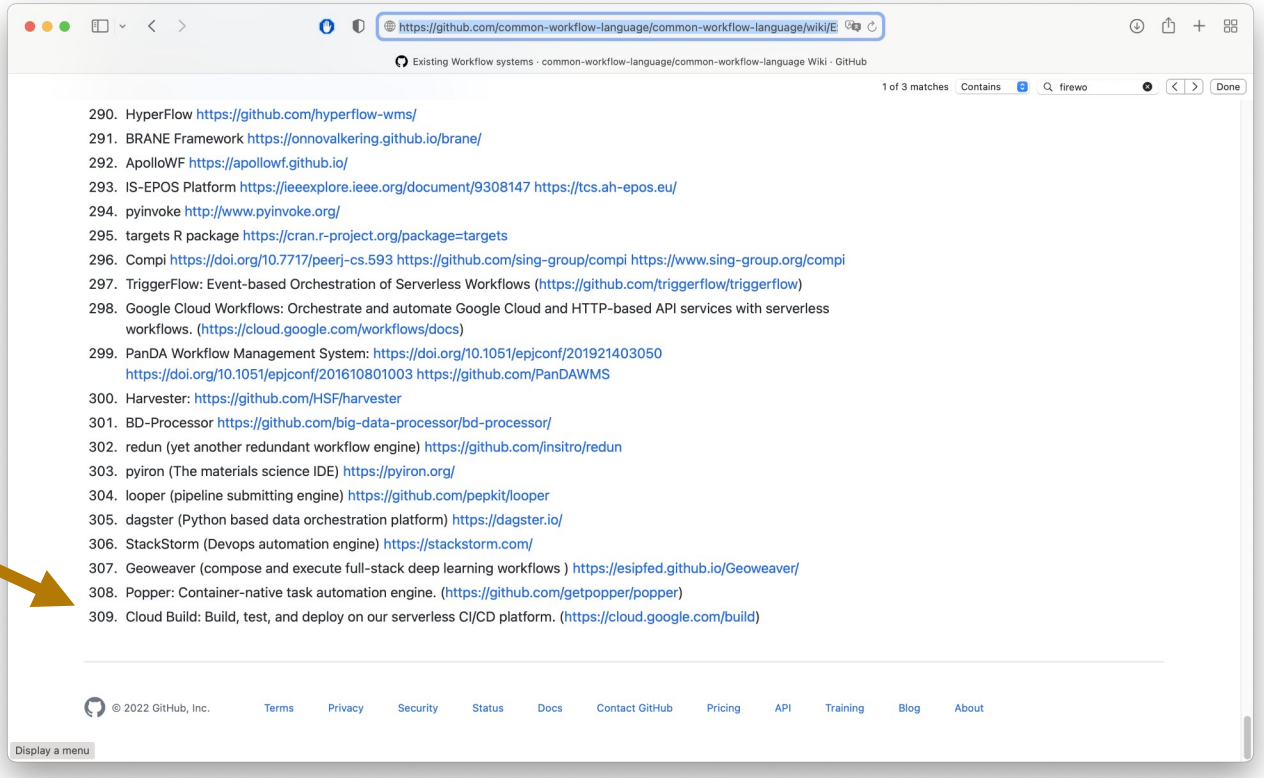
An “incomplete” list …



https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems

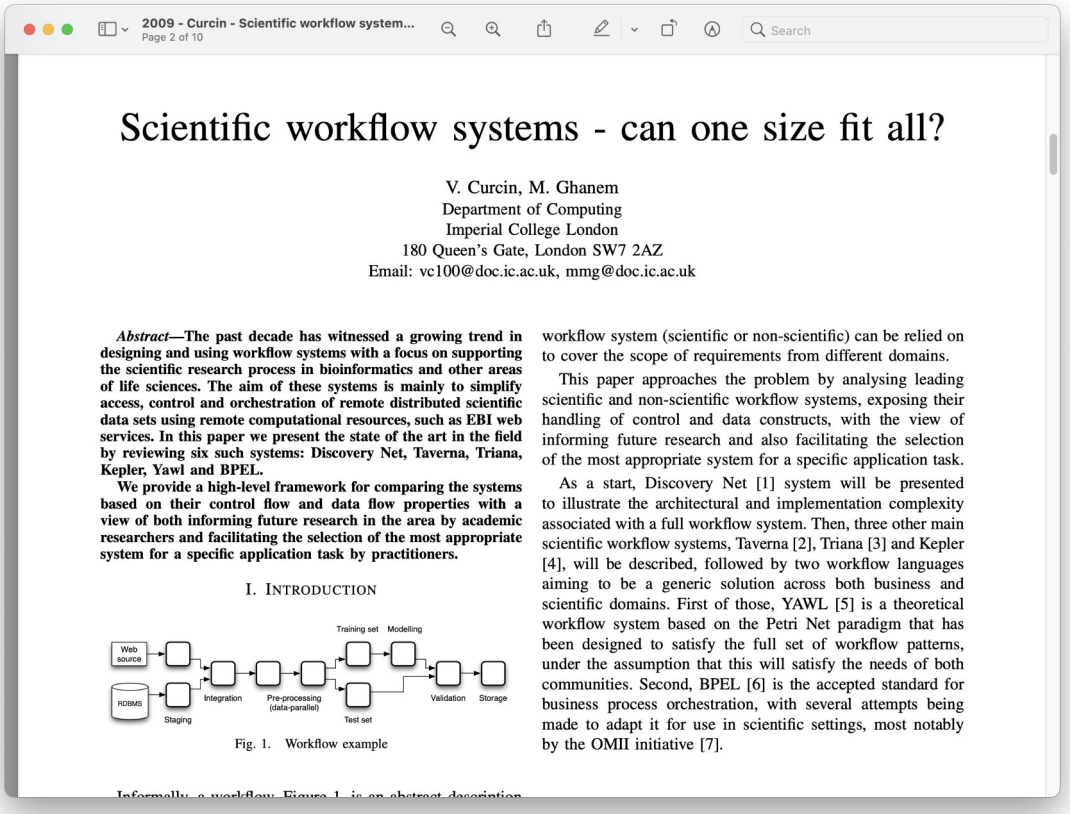
Workflow management systems

… of 309 entries (!?)



https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems

Why so many?



Scientific workflow systems - can one size fit all?

V. Curcin, M. Ghanem
Department of Computing
Imperial College London
180 Queen's Gate, London SW7 2AZ
Email: vc100@doc.ic.ac.uk, mmg@doc.ic.ac.uk

***Abstract*—The past decade has witnessed a growing trend in designing and using workflow systems with a focus on supporting the scientific research process in bioinformatics and other areas of life sciences. The aim of these systems is mainly to simplify access, control and orchestration of remote distributed scientific data sets using remote computational resources, such as EBI web services. In this paper we present the state of the art in the field by reviewing six such systems: Discovery Net, Taverna, Triana, Kepler, Yawl and BPEL.**

We provide a high-level framework for comparing the systems based on their control flow and data flow properties with a view of both informing future research in the area by academic researchers and facilitating the selection of the most appropriate system for a specific application task by practitioners.

I. Introduction

Fig. 1. Workflow example

Informally, a workflow, Figure 1, is an abstract description

workflow system (scientific or non-scientific) can be relied on to cover the scope of requirements from different domains.

This paper approaches the problem by analysing leading scientific and non-scientific workflow systems, exposing their handling of control and data constructs, with the view of informing future research and also facilitating the selection of the most appropriate system for a specific application task.

As a start, Discovery Net [1] system will be presented to illustrate the architectural and implementation complexity associated with a full workflow system. Then, three other main scientific workflow systems, Taverna [2], Triana [3] and Kepler [4], will be described, followed by two workflow languages aiming to be a generic solution across both business and scientific domains. First of those, YAWL [5] is a theoretical workflow system based on the Petri Net paradigm that has been designed to satisfy the full set of workflow patterns, under the assumption that this will satisfy the needs of both communities. Second, BPEL [6] is the accepted standard for business process orchestration, with several attempts being made to adapt it for use in scientific settings, most notably by the OMII initiative [7].

Curcin, Vasa & Ghanem, Moustafa. (2009). Scientific workflow systems - Can one size fit all?. Cairo Int Biomed Eng Conf. 2008. 1 - 9. 10.1109/CIBEC.2008.4786077.

Why so many?

2009 - Curcin - Scientific workflow system...
Page 2 of 10

Scientific workflow systems - can one size fit all?

V. Curcin, M. Ghanem
Department of Computing
Imperial College London
180 Queen's Gate, London SW7 2AZ
Email: vc100@doc.ic.ac.uk, mmg@doc.ic.ac.uk

***Abstract*—The past decade has witnessed a growing trend in designing and using workflow systems with a focus on supporting the scientific research process in bioinformatics and other areas of life sciences. The aim of these systems is mainly to simplify access, control and orchestration of remote distributed scientific data sets using remote computational resources, such as EBI web services. In this paper we present the state of the art in the field by reviewing six such systems: Discovery Net, Taverna, Triana, Kepler, Yawl and BPEL.**

We provide a high-level framework for comparing the systems based on their control flow and data flow properties with a view of both informing future research in the area by academic researchers and facilitating the selection of the most appropriate system for a specific application task by practitioners.

I. Introduction

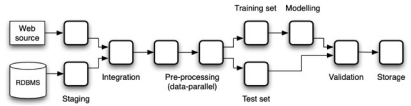


Fig. 1. Workflow example

workflow system (scientific or non-scientific) ca
to cover the scope of requirements from differe

This paper approaches the problem by ana
scientific and non-scientific workflow systems,
handling of control and data constructs, with
informing future research and also facilitating
of the most appropriate system for a specific ap

As a start, Discovery Net [1] system will
to illustrate the architectural and implementati
associated with a full workflow system. Then, th
scientific workflow systems, Taverna [2], Triana
[4], will be described, followed by two workf
aiming to be a generic solution across both
scientific domains. First of those, YAWL [5] i
workflow system based on the Petri Net para
been designed to satisfy the full set of work
under the assumption that this will satisfy the
communities. Second, BPEL [6] is the accepte
business process orchestration, with several a
made to adapt it for use in scientific settings,
by the OMII initiative [7].

Informally, a workflow, Figure 1, is an abstract description

en.wikipedia.org/wi

Betteridge's law of headlines - Wikipedia

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipe

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal

Betteridge's law of headlines

From Wikipedia, the free encyclopedia

Betteridge's law of headlines is an adage that states: "Any headline that ends in a question mark can be answered by the word *no*." It is named after Ian Betteridge, a British technology journalist who wrote about it in 2009, although the principle is much older.[1][2] It is based on the assumption that if the publishers were confident that the answer was *yes*, they would have presented it as an assertion; by presenting it as a question, they are not accountable for whether it is correct or not. The adage does not apply to questions that are more open-ended than strict yes–no questions.[3]

Curcin, Vasa & Ghanem, Moustafa. (2009). Scientific workflow systems - Can one size fit all?. Cairo Int Biomed Eng Conf. 2008. 1 - 9. 10.1109/CIBEC.2008.4786077.

In this session

we will give you an overview of tools

- Relevant to the HPC environments (not cloud, K8s, Hadoop, etc.)
- Standalone (not language-specific libraries)
- With a simple DSL (no XML or other convoluted language)
- General purpose (not reserved to 'omics' for instance)
- Mature, active community, easy to install

Simple types of workflows

wide

workflows



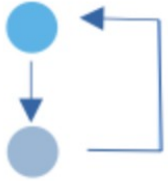
deep

workflows



cyclic

workflows



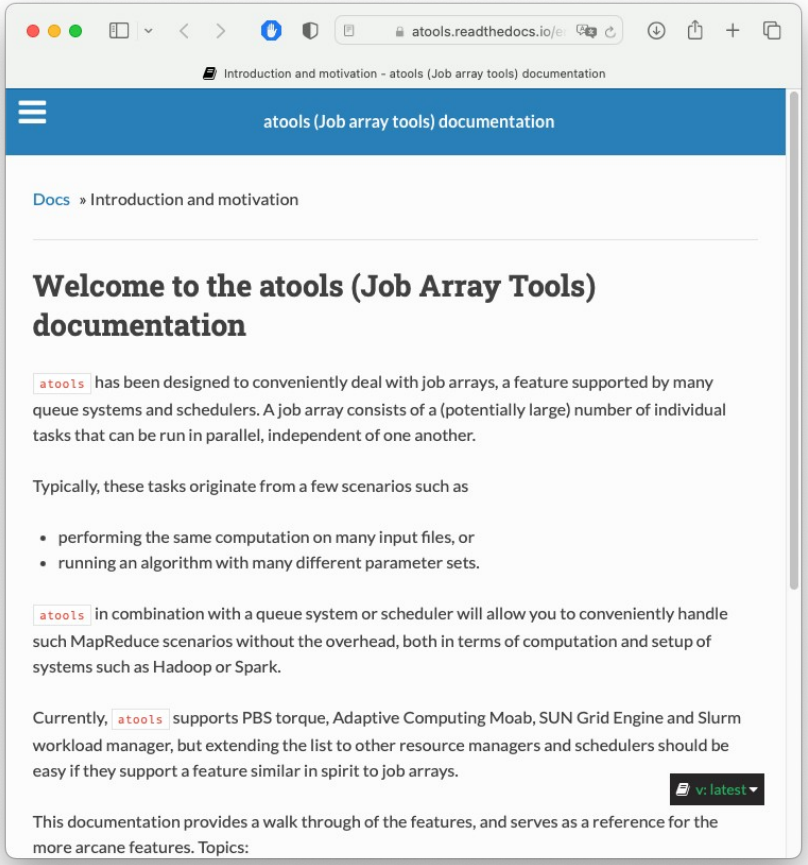
atools

```
alpha,beta
0.5,-1.3
0.5,-1.5
0.5,-1.9
0.6,-1.3
...
```

```
#!/bin/bash
...
alog --state start
source <(aenv --data data.csv)
Rscript bootstrap.R $alpha $beta
alog  --state end  --exit $?
```

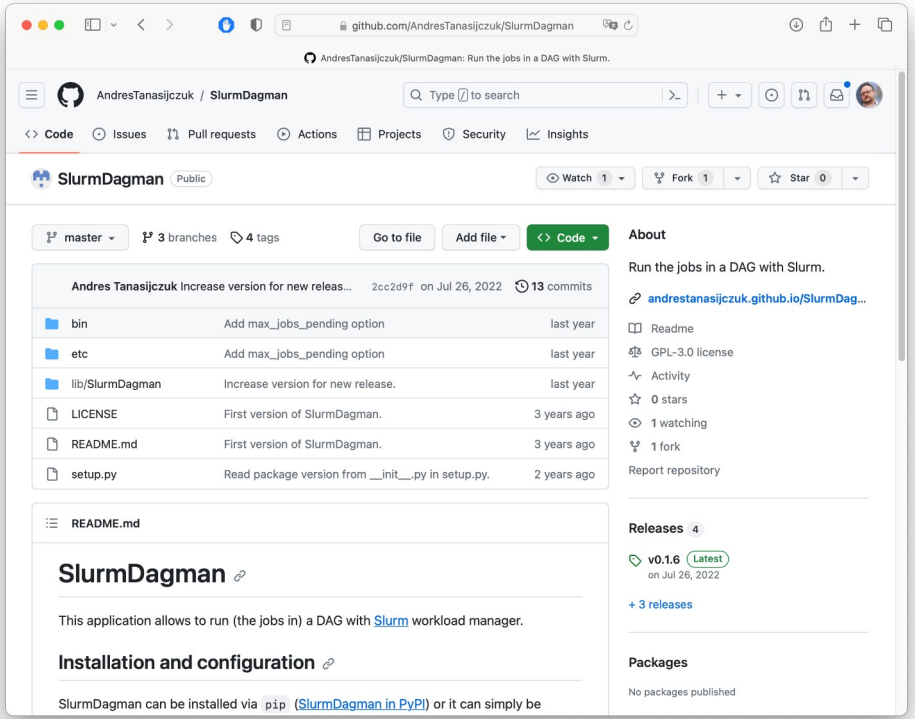
```
1 started by r1i1n3 at 2016-09-02 11:47:45
2 started by r1i1n3 at 2016-09-02 11:47:45
3 started by r1i1n3 at 2016-09-02 11:47:46
2 failed by r1i1n3 at 2016-09-02 11:47:46: 1
3 completed by r1i1n3 at 2016-09-02 11:47:47
```

```
$ arange  --data data.csv  --log bootstrap.pbs.log10493`
```



https://atools.readthedocs.io/en/latest/

SlurmDagman



DAG file syntax

The syntax of the DAG file is similar to the one used in HTCondor DAGMan. The following four constructs are recognized by SlurmDagman in a DAG file:

```
JOB dag-node slurm-submission-file
VARS dag-node var-name=var-value [...]
RETRY dag-node max-retries
PARENT parent-dag-node[,...] CHILD child-dag-node[,...]
```

https://github.com/AndresTanasijczuk/SlurmDagman

Makeflow




```
capitol.anim.gif: capitol.jpg capitol.90.jpg capitol.180.jpg capitol.270.jpg capitol.360.
    LOCAL $(CONVERT) -delay 10 -loop 0 capitol.jpg capitol.90.jpg capitol.180.jpg capitol

capitol.90.jpg: capitol.jpg
    $(CONVERT) -swirl 90 capitol.jpg capitol.90.jpg

capitol.180.jpg: capitol.jpg
    $(CONVERT) -swirl 180 capitol.jpg capitol.180.jpg

capitol.270.jpg: capitol.jpg
    $(CONVERT) -swirl 270 capitol.jpg capitol.270.jpg

capitol.360.jpg: capitol.jpg
    $(CONVERT) -swirl 360 capitol.jpg capitol.360.jpg

capitol.jpg:
    LOCAL $(CURL) -o capitol.jpg $(URL)
```

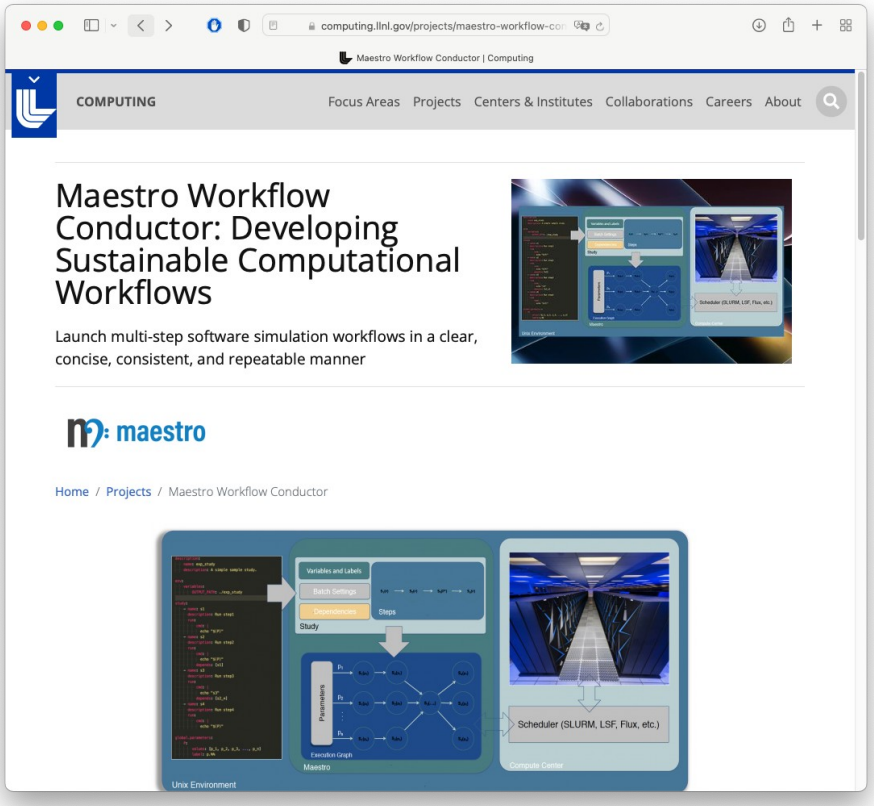
Use the `-T slurm` option to submit jobs to the SLURM batch system.

This will add the values for cores and memory. These values will be added onto `sbatch` in this format:

```
-N 1 -n 1 -c ${CORES} --mem=${MEMORY}M --time=${WALL_TIME_in_minutes}
```

https://cctools.readthedocs.io/en/latest/makeflow/

Maestro




```yaml
description:
    name: Build archive
    description: A simple archive building study

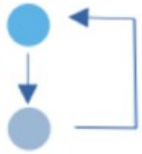
batch:
    type:  slurm
    queue: debug
    host: localhost
    bank: ceci

study:
    […]

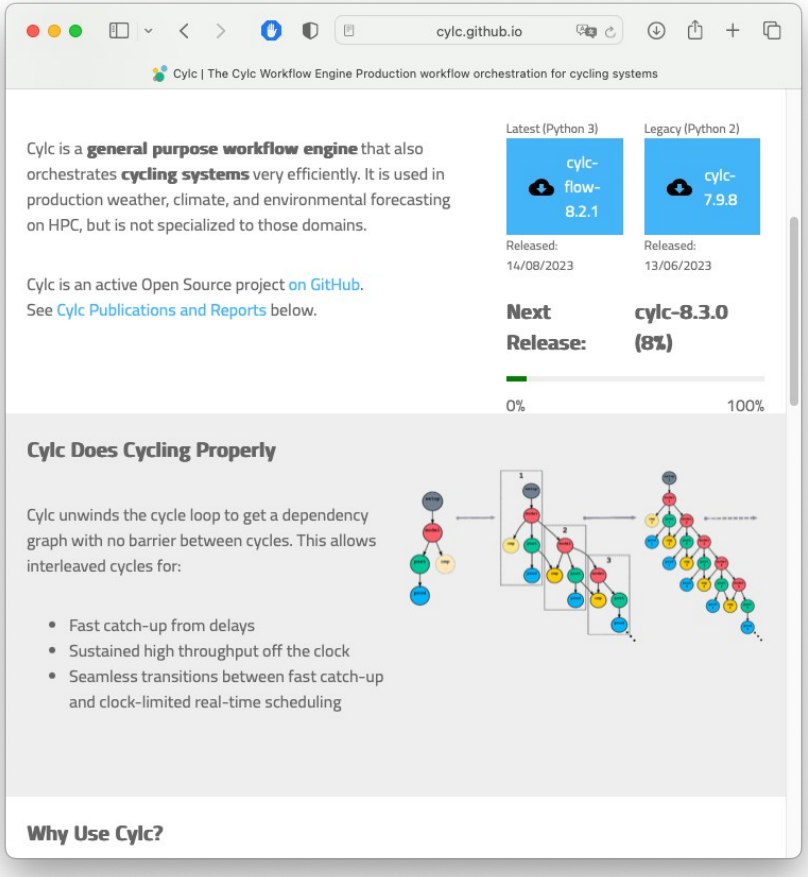
    - name: compress
      description: compress the archive with gzip
      run:
         cmd: |
            ml gzip
            gzip -k $(compressoption) ../../build/archive.tar
         depends: [build]
         nodes: 1
         procs: 2
         walltime: “10:00”

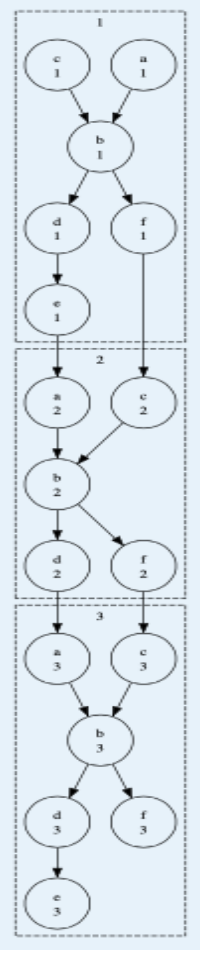
global.parameters:
    compressoption:
        values: ['--fast', '--best']
        label: COPT.%%
```

https://github.com/LLNL/maestrowf



Cycl





```
[scheduler]
    allow implicit tasks = True
[scheduling]
    cycling mode = integer
    initial cycle point = 1
    [[graph]]
        P1 = """
            a & c => b => d & f
            f[-P1] => c  # (1)
        """
        P2 = """
            d => e
            d[-P1] => a  # (2)
        """
        2/P2 = """
            e[-P1] => a  # (3)
        """
```

https://cylc.github.io

Micro-scheduling systems

to organise a large number of tasks in a small number of jobs

HyperQueue

https://github.com/It4innovations/hyperqueue

Macro-scheduling systems

to organise jobs across multiple clusters



https://github.com/ecmwf/troika

Full-featured systems

that offer a web interface, data management, etc.
a selection

Name	Website	Domain
Fireworks	https://materialsproject.github.io/fireworks/	Material science
Nextflow	https://nextflow.io	Bioinformatics
Merlin	https://merlin.readthedocs.io/	Machine learning
DAGMan	https://htcondor.readthedocs.io/	Particle physics
Pegasus	https://pegasus.isi.edu/	Agnostic

How about letting Claude/ChatGPT do it?

Agents are good at writing scripts.

But you must stay in charge.

You can't control what you don't master

My recommendation: Write the first few submission scripts by yourself, then, when you are confident you understand what is going on, let the agent do the work.

The old-school version of this was to copy the submission script from a colleague. Faster in the short term, more problems in the long term.

Further reading

Deelman, Ewa & Gannon, Dennis & Shields, Matthew & Taylor, Ian. (2009). **Workflows and e-Science: An overview of workflow system features and capabilities**. *Future Generation Computer Systems*. 25. 524-540. 10.1016/j.future.2008.06.012.

Liu, Ji & Pacitti, Esther & Valduriez, Patrick & Mattoso, Marta. (2015). **A Survey of Data-Intensive Scientific Workflow Management**. *Journal of Grid Computing*. 13. 10.1007/s10723-015-9329-8.

Badia, Rosa M. & Ayguade, E. & Labarta, Jesús. (2017). **Workflows for science: a challenge when facing the convergence of HPC and Big Data**. *Supercomputing Frontiers and Innovations*. 4. 27-47. 10.14529/jsfi170102.

Ferreira da Silva, Rafael & Filgueira, Rosa & Pietri, Ilia & Jiang, Ming & Sakellariou, Rizos & Deelman, Ewa. (2017). **A characterization of workflow management systems for extreme-scale applications.** *Future Generation Computer Systems*. 75. 10.1016/j.future.2017.02.026.

Deelman, Ewa & Peterka, Tom & Altintas, Ilkay & Carothers, Christopher & Dam, Kerstin & Moreland, Kenneth & Parashar, Manish & Ramakrishnan, Lavanya & Taufer, Michela & Vetter, Jeffrey. (2017). **The future of scientific workflows.** *The International Journal of High Performance Computing Applications*. 32. 109434201770489. 10.1177/1094342017704893.