

Getting the Most out of HTC with Workflows

Friday

Christina Koch ckoch5@wisc.edu
Research Computing Facilitator
University of Wisconsin - Madison

Why are we here?

Why are we here?

To do SCIENCE!!!

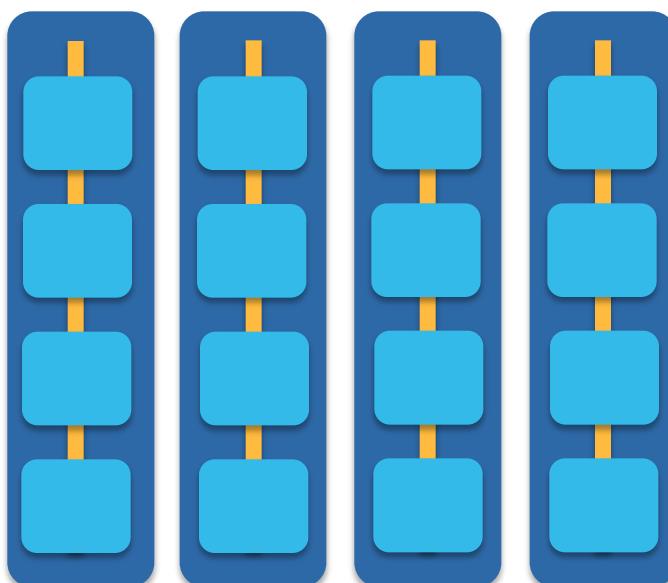
- A lot of science is best-done with computing – sometimes, LOTS of computing
- Science needs to be reproducible
- And, we'd really like science to happen FAST(er)



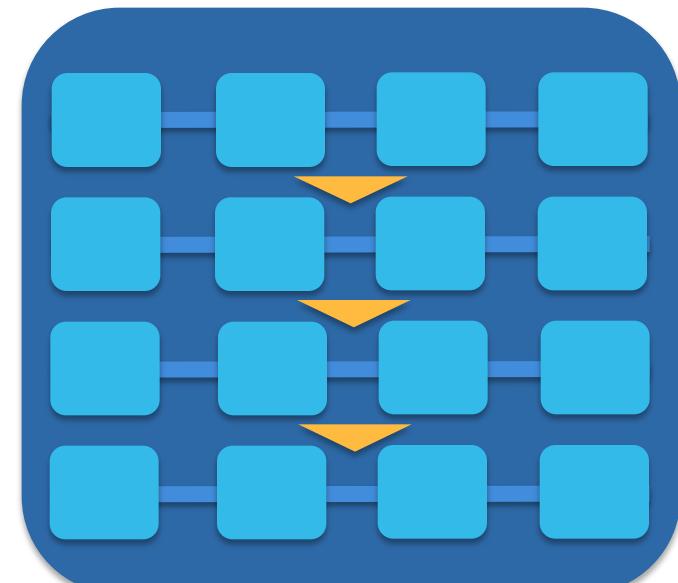
GETTING THE MOST OUT OF COMPUTING (FOR SCIENCE)

Computing types

- At the beginning of the week, we talked about two different approaches for tackling large compute tasks...



high-throughput



high-performance (e.g. MPI)

Two Strategies

High Throughput

Focus: Workflows with many *small, largely independent* compute tasks

Optimize: *throughput*, or time from *submission* to *overall completion*

High Performance

- Focus: Workflows with *large, highly coupled* tasks
- Optimize: software, communication between processes

Making Good Choices

- How do you choose the best approach?
- Guiding question:

Is your problem “HTC-able”?



Typical HTC Problems

- batches of similar program runs (>10)
- “loops” over independent tasks
- others you might not think of ...
 - programs/functions that
 - process files that are already separate
 - process columns or rows, separately
 - iterate over a parameter space
 - *a lot* of programs/functions that use multiple CPUs on the same server

Ultimately: Can you break it up?

What is not HTC?

- fewer numbers of jobs
- jobs individually requiring significant resources
 - RAM, Data/Disk, # CPUs, time
(though, “significant” depends on the HTC compute system you use)
- restrictive licensing

The Real World

- However, it's not just about finding the right computing approach to your problem.
- These approaches will be **most** effective if they're running on appropriate compute systems.



The Real World

- Not all compute systems are created equal.

- Two questions to ask:

What resources are available to me?

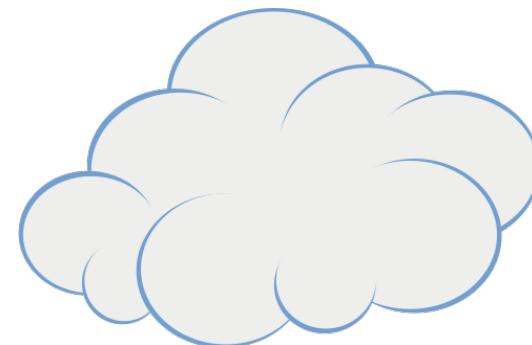
Which one is the best match for the kind of computing I want to do?

Campus Resources

- Start with your local campus compute system
- Some considerations:
 - Who has access? Are there allocations?
 - What kind of system? What is it optimized for?
- An HPC cluster may not handle lots of jobs well, in the same way that an HTC system has limited multicore capabilities - be aware of how a system matches/doesn't match your computation strategy.
- Ask questions! Be a good citizen!
- If local resources are limited, explore other options.

Beyond your campus

- Open Science Grid!
 - This afternoon, Tim will talk about ways to access OSG after the school is over



- Other grids
 - European Grid Infrastructure
 - Other national and regional grids
 - Commercial cloud systems

The payoff

- HTC is, beyond everything, scalable
 - If you can run 10 jobs, you can run 10,000, maybe even 10 million
- Worth pursuing the right kind of resources (if you can) for the right kind of problem.





Open Science Grid

GETTING THE MOST OUT OF HTC

Key HTC Tactics

- 1. Increase Overall Throughput**
- 2. Don't Overuse Resources!**
- 3. Bring Dependencies With You**
- 4. Scale Gradually, Testing Generously**
- 5. Automate As Many Steps As Possible**

Throughput, revisited

- In HTC, we optimize *throughput*: time from submission to overall completion



Instead of making individual jobs as fast as possible...



...optimize how long it takes for *all* jobs to finish

- We do this by breaking large processes into smaller pieces

Breaking up is hard to do...

- Ideally into parallel (separate) jobs
 - reduced job requirements = more matches
 - not always easy or possible
- Strategies
 - break HTC-able steps out of a single program
 - break up loops
 - break up input
- Self-checkpointing if jobs are too long

Batching (Merging) is easy

- A single job can
 - execute multiple independent tasks
 - execute multiple short, sequential steps
 - avoid transfer of intermediate files
- Use scripts!
 - need adequate error reporting for each “step”
 - easily handle multiple commands and arguments

Key HTC Tactics

1. Increase Overall Throughput
- 2. Utilize Resources Efficiently!**
3. Bring Dependencies With You
4. Scale Gradually, Testing Generously
5. Automate As Many Steps As Possible

Know and Optimize Job Use of Resources!

- **CPUs** (“1” is best for matching; essential for OSG)
 - restrict, if necessary/possible
 - software that uses all available CPUs is BAD!
- **CPU Time**

> ~5 min, < ~1 day; **Ideal: 1-2 hours**
- **RAM** (not always easily modified)
- **Disk** per-job (execute) and in-total (submit)
- **Network Bandwidth**
 - minimize transfer: filter/trim/delete, compress

The job log shows all

001 (2576205.000.000) 06/07 11:57:57 Job executing on host:
<128.104.101.248:9618>

005 (2576205.000.000) 06/07 14:12:55 Job terminated.

(1) Normal termination (return value 0)

 Usr 0 00:00:00, Sys 0 00:00:00 - Run Remote Usage

 Usr 0 00:00:00, Sys 0 00:00:00 - Run Local Usage

 Usr 0 00:00:00, Sys 0 00:00:00 - Total Remote Usage

 Usr 0 00:00:00, Sys 0 00:00:00 - Total Local Usage

5 - Run Bytes Sent By Job

104857640 - Run Bytes Received By Job

5 - Total Bytes Sent By Job

104857640 - Total Bytes Received By Job

Partitionable Resources : Usage Request Allocated

	Usage	Request	Allocated
Cpus	:	1	1

Disk (KB)	: 122358	125000	13869733
-----------	----------	--------	----------

Memory (MB)	: 30	100	100
-------------	------	-----	-----

Key HTC Tactics

1. Increase Overall Throughput
2. Don't Overuse Resources!
- 3. Bring Dependencies With You**
4. Scale Gradually, Testing Generously
5. Automate As Many Steps As Possible

Bring *What* with You?

- Software (covered Wednesday)
- Data and other input files
 - Parameters and random numbers: generate and record ahead of time (for reproducibility!)
- What else?



Wrapper Scripts are Essential

- Before task execution (bring it with you!)
 - transfer/prepare files and directories
 - setup/configure software environment and other dependencies
- Task execution
 - prepare complex commands and arguments
 - batch together many ‘small’ tasks
- After task execution
 - filter/combine/compress files and directories
 - check for and report on errors

Key HTC Tactics

1. Increase Overall Throughput
2. Don't Overuse Resources!
3. Bring Dependencies With You
- 4. Scale Gradually, Testing Generously**
5. Automate As Many Steps As Possible

Testing, testing, testing!

- Will be a major focus of our exercises today.
- Allows you to optimize resource use (see HTC tactic #2)
- Just because it worked for 10 jobs, doesn't mean it will work for 10,000 jobs (scaling issues)
 - Data transfer (in and out)
 - Discover site-specific problems

Key HTC Tactics

1. Increase Overall Throughput
2. Don't Overuse Resources!
3. Bring Dependencies With You
4. Scale Gradually, Testing Generously
- 5. Automate As Many Steps As Possible**

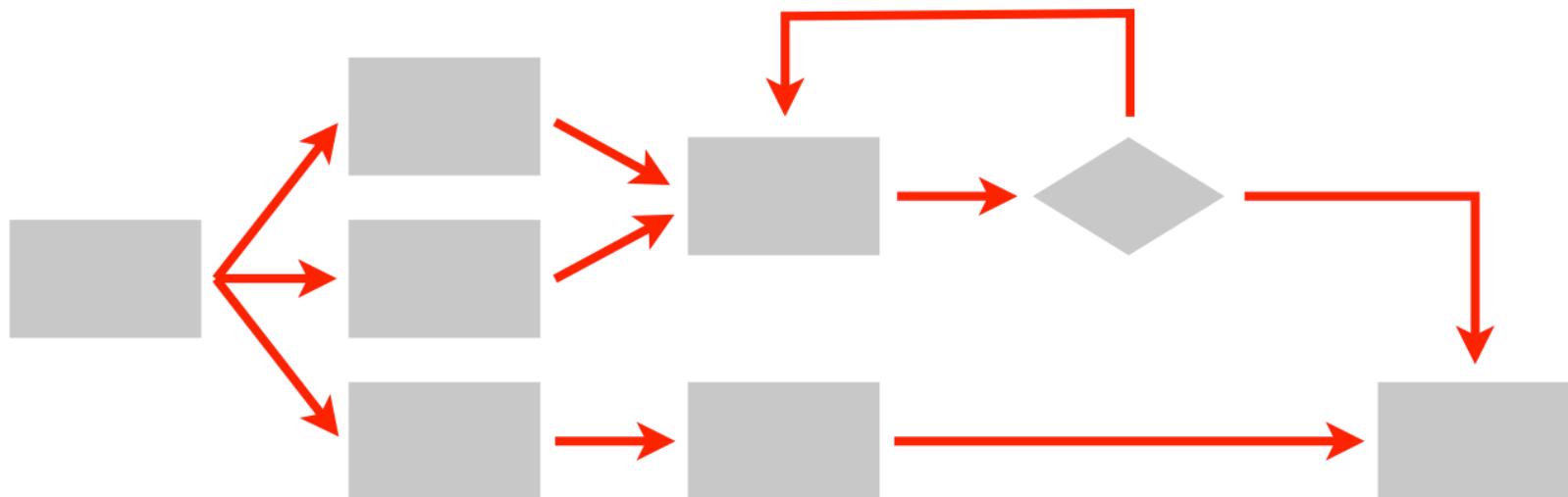
What to Automate?

- Submitting many jobs (using HTCondor)
- Writing submit files using scripts
- Running a series of jobs, or workflow



What is a workflow?

- A series of ordered steps
 - Steps
 - Connections
 - (Metadata)

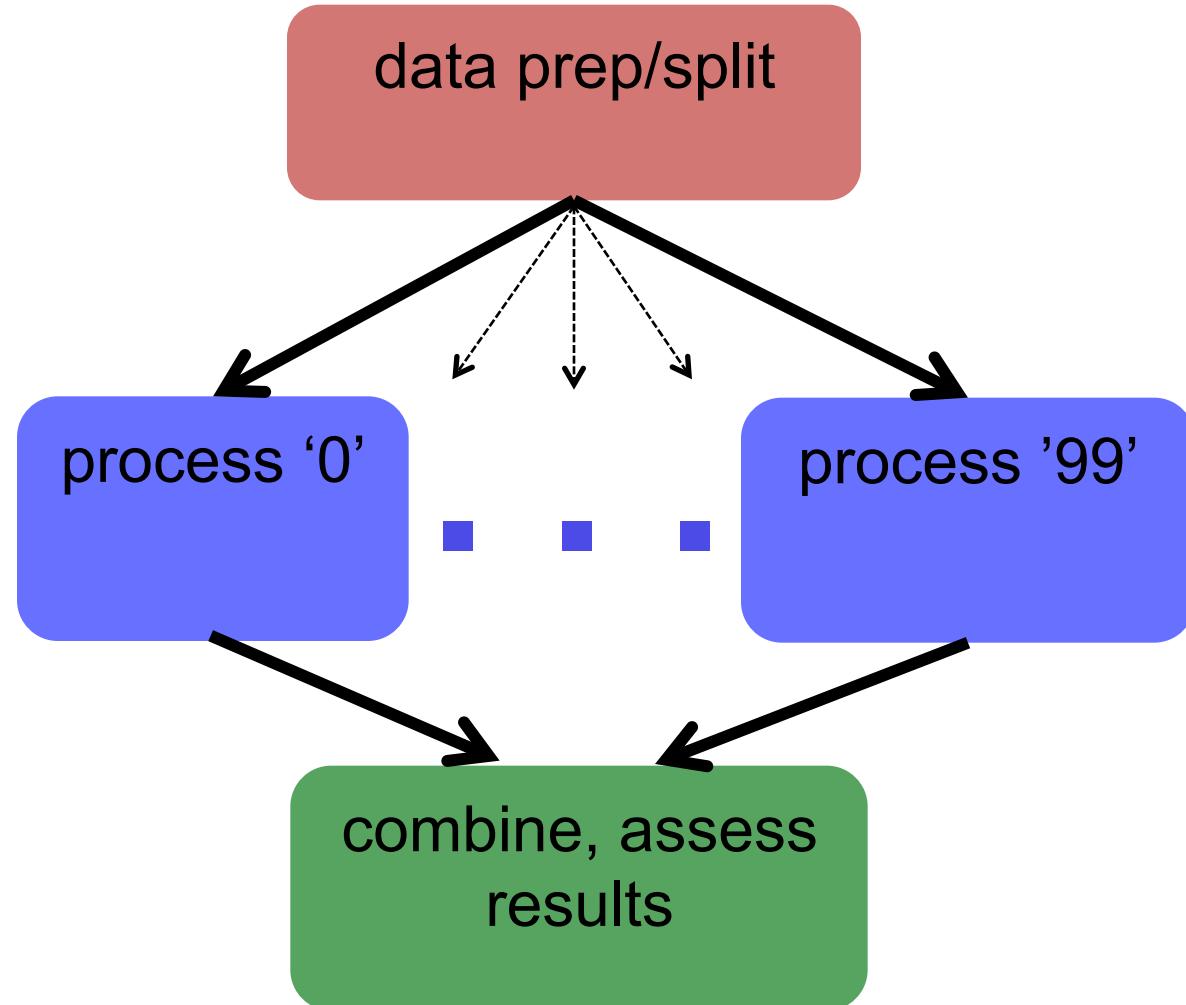


We ❤️ workflows

- non-computing “workflows” are all around you ... especially in science
 - instrument setup
 - experimental procedures and protocols
- when planned/documented, workflows help with:
 - organizing and managing processes
 - saving time with **automation**
 - objectivity, reliability, and reproducibility
(THE TENETS OF GOOD SCIENCE!)



DAGs Automate Workflows!





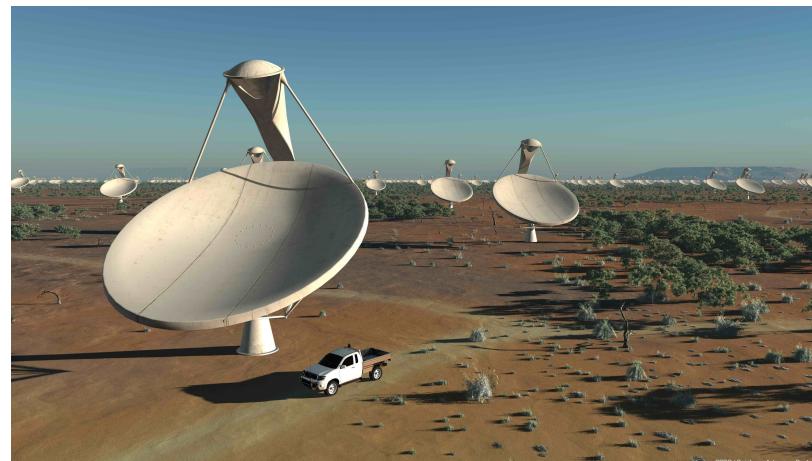
Automating workflows can save you time...

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND		1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS		5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS		4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
HOW MUCH TIME YOU SHAVE OFF	1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	2 WEEKS	1 DAY
1 DAY						8 WEEKS	5 DAYS

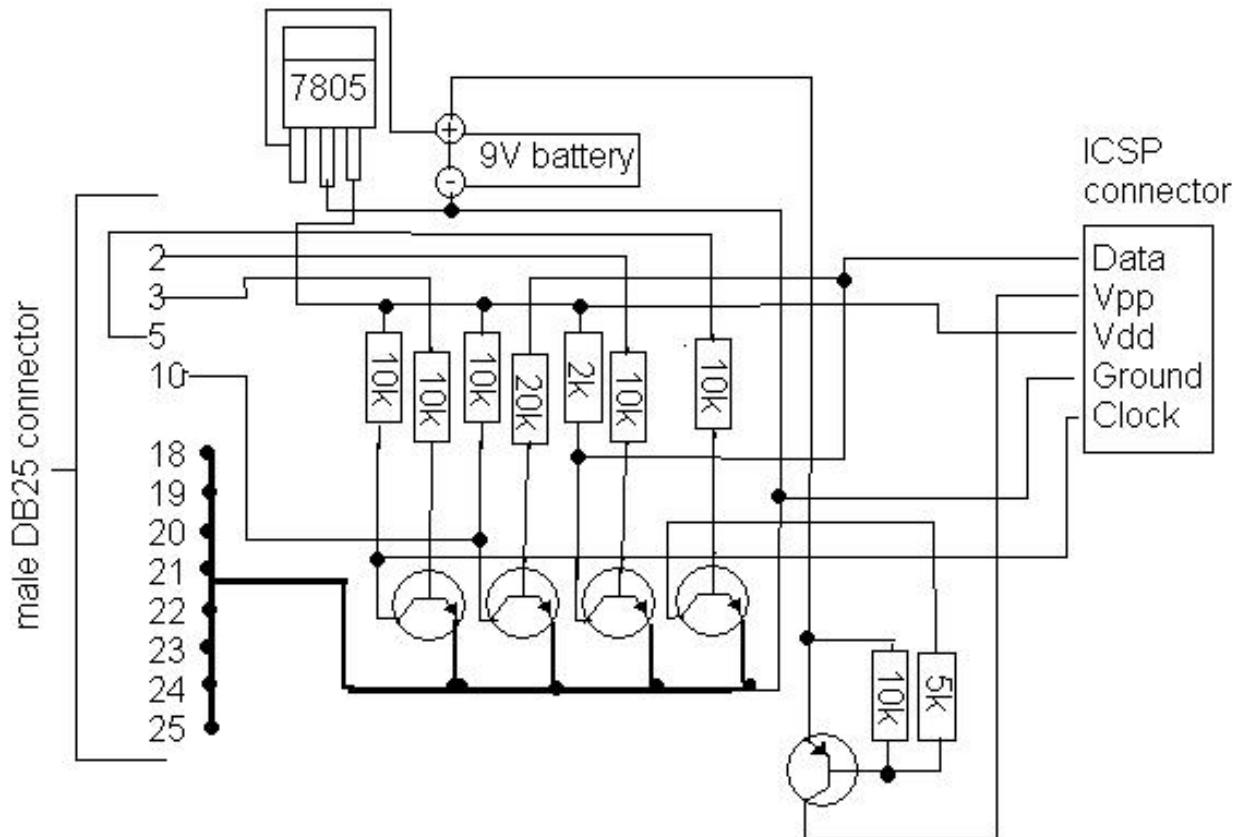
... but there are even more benefits of automating workflows

- Reproducibility
- Building knowledge and experience
- New ability to imagine greater scale, functionality, possibilities, and better SCIENCE!!



GETTING THE MOST OUT OF WORKFLOWS, PART 1

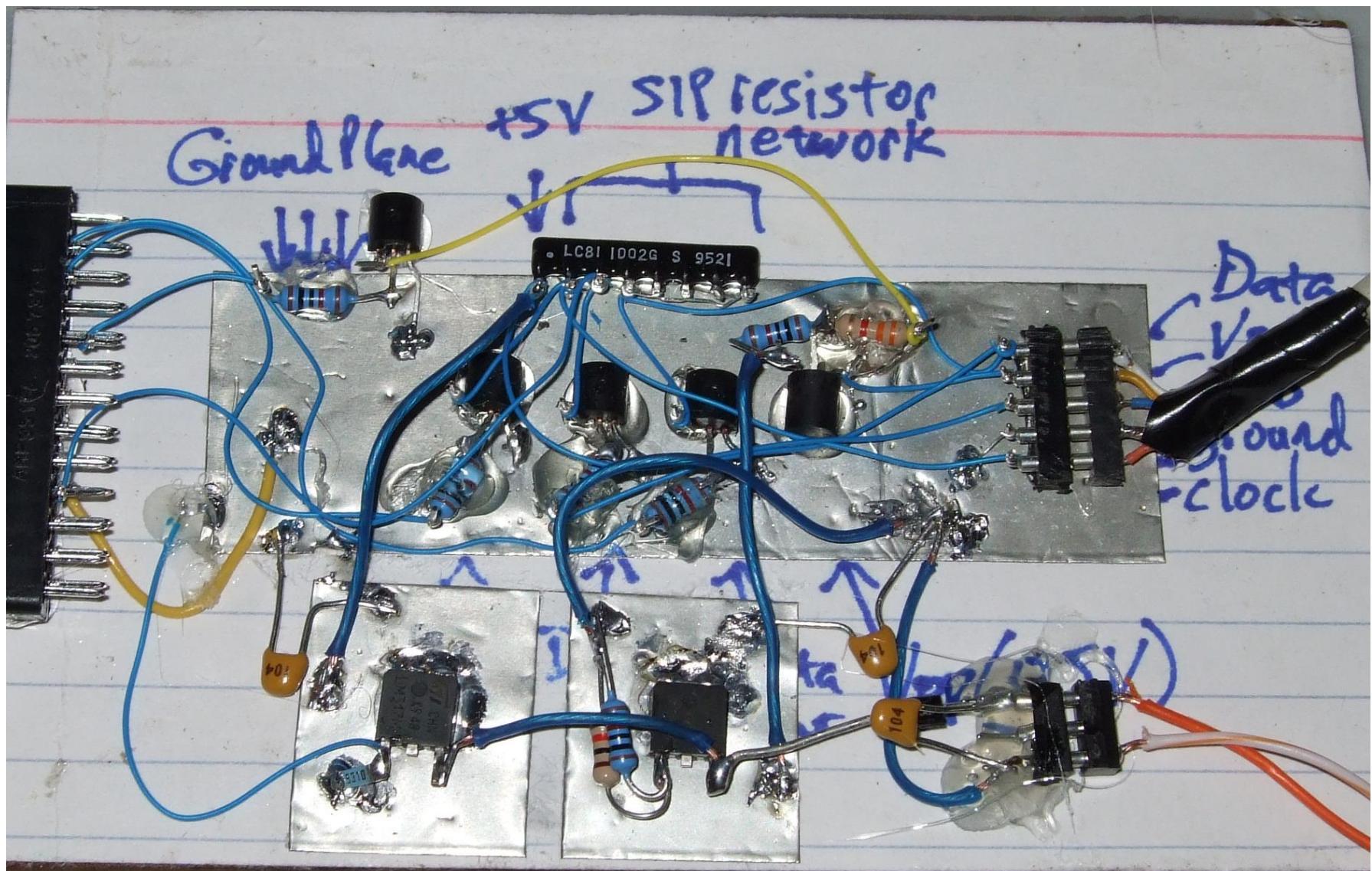
From schematics...





Open Science Grid

... to the real world

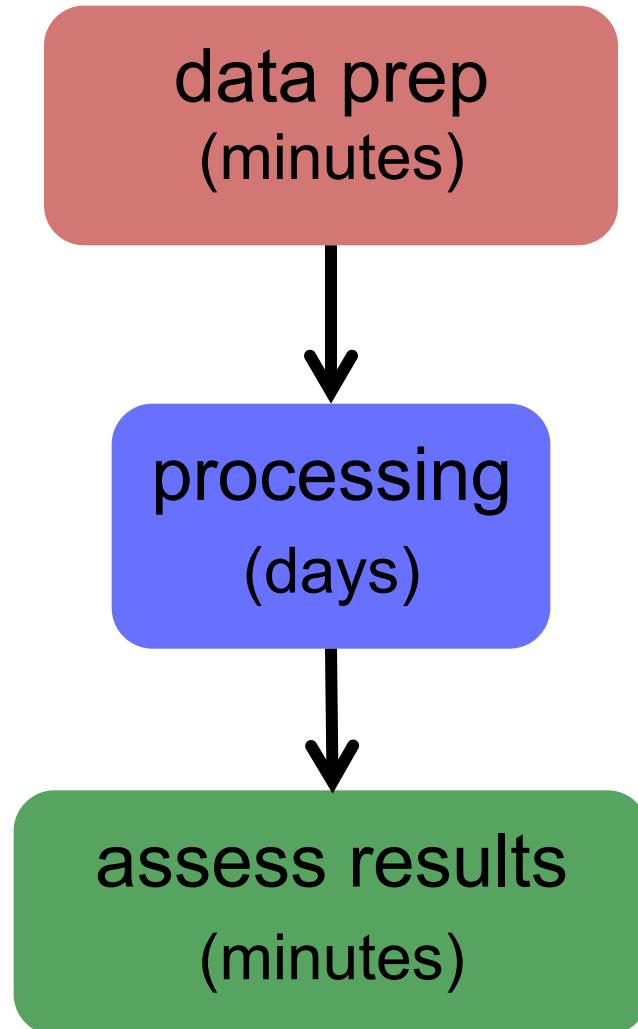


Building a Good Workflow

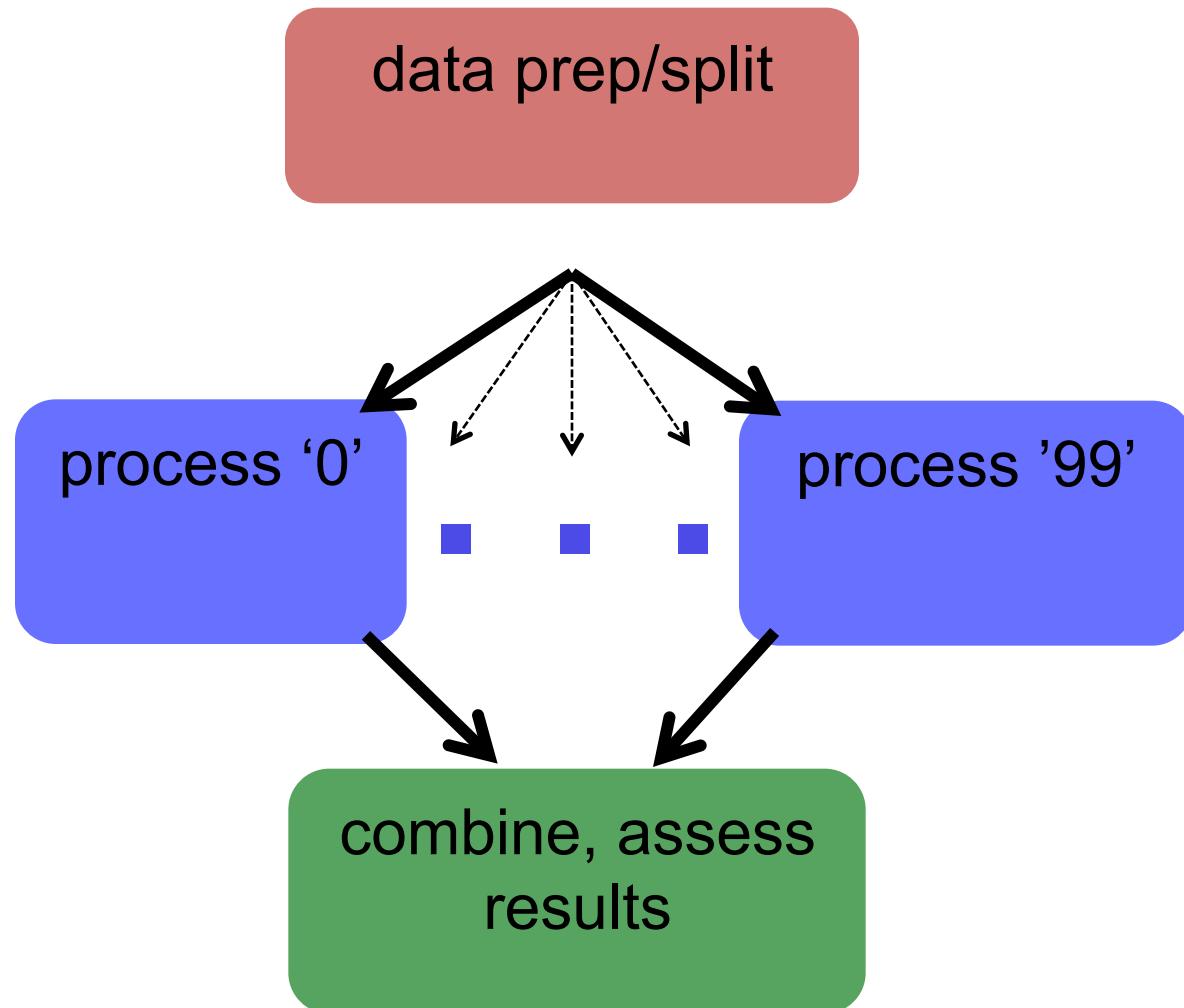
1. Draw out the *general workflow*
2. Define details (test ‘pieces’ with HTCondor jobs)
 - divide or consolidate ‘pieces’
 - determine resource requirements
 - identify steps to be automated or checked
3. Build it modularly; test and optimize
4. Scale-up gradually
5. Make it work consistently
6. What more can you automate or error-check?

(And remember to document!)

Workflow, version 1



Workflow, version 2 (HTC)



Building a Good Workflow

1. Draw out the *general* workflow
2. **Define details (test ‘pieces’ with HTCondor jobs)**
 - **divide or consolidate ‘pieces’**
 - **determine resource requirements**
 - **identify steps to be automated or checked**
3. Build it modularly; test and optimize
4. Scale-up gradually
5. Make it work consistently
6. What more can you automate or error-check?

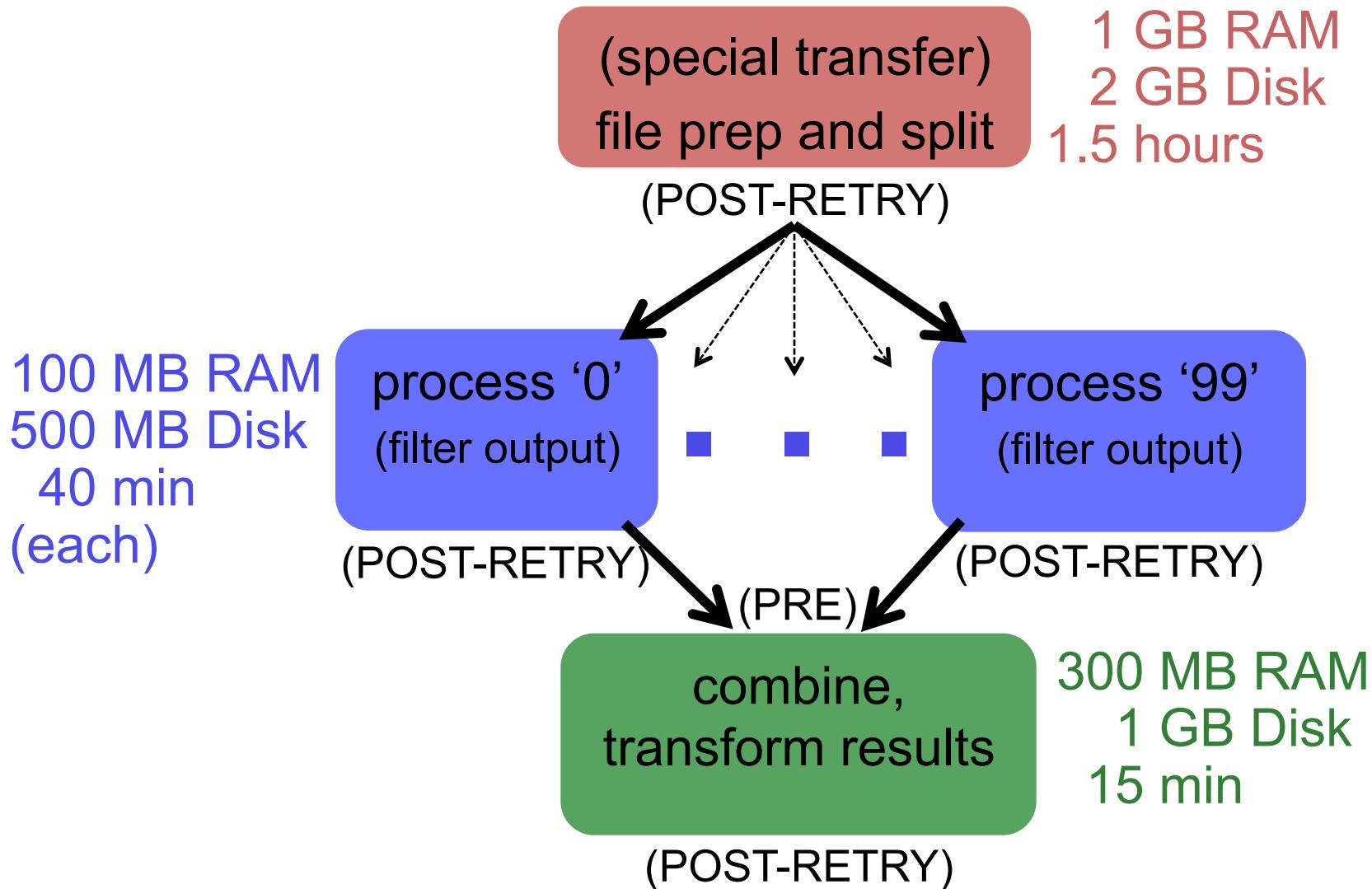
(And remember to document!)

Determine Resource Usage

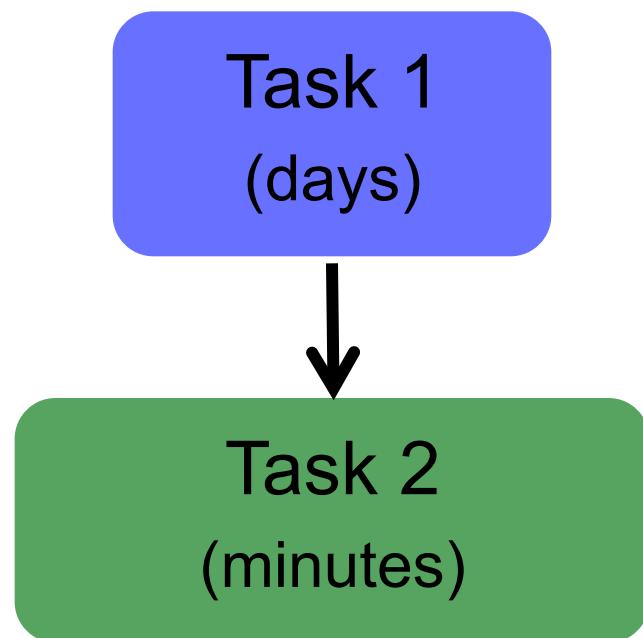
- Run locally first
- Then get one job running remotely
 - (on execute machine, not submit machine)!
 - get the logistics correct! (HTCondor submission, file and software setup, etc.)
- Once working, run a couple of times
 - If big variance in resource needs, should you take the...

Average? Median? Worst case?

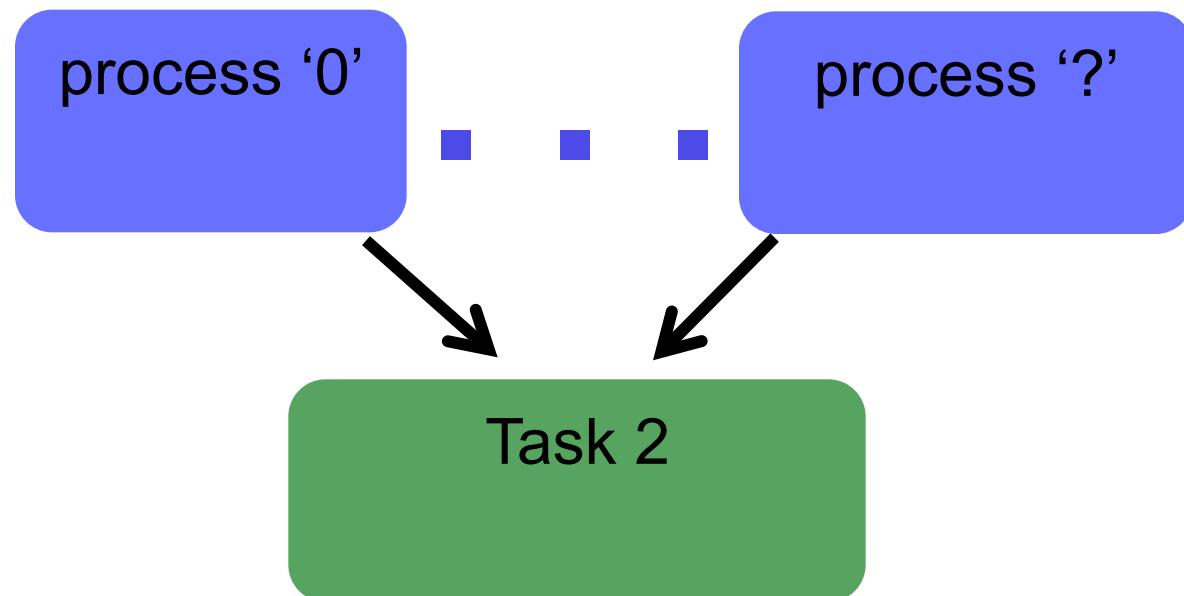
End Up with This



Exercise 1.1

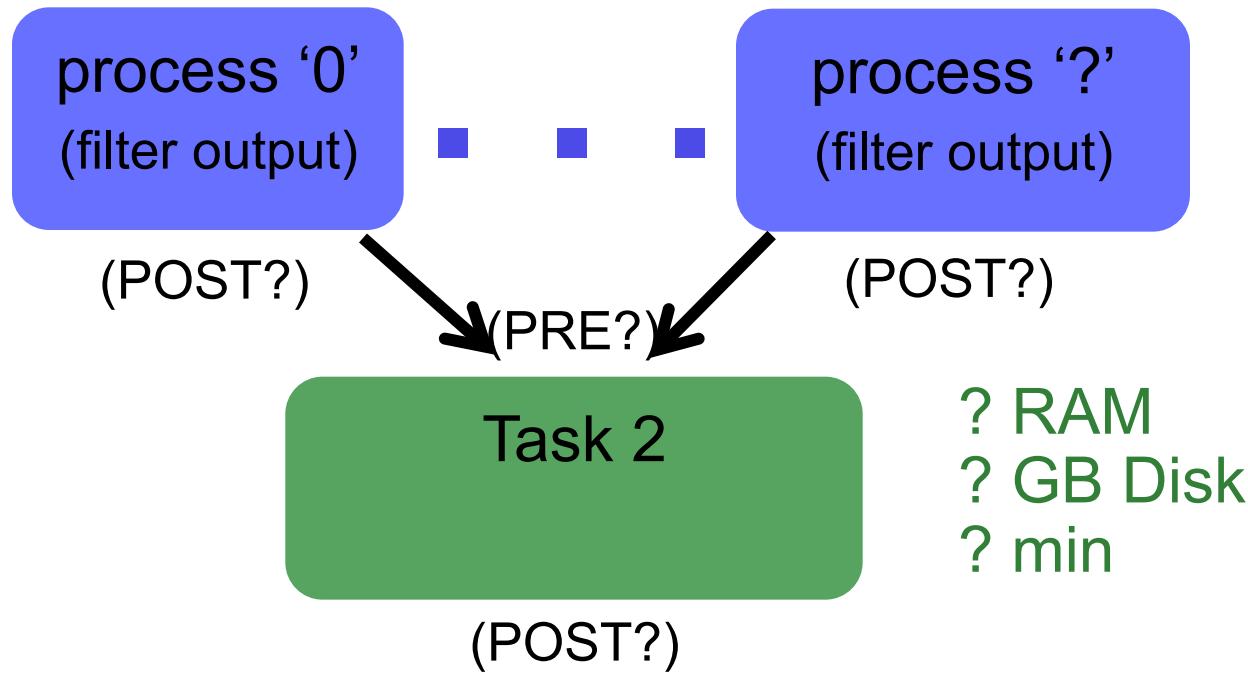


Exercise 1.1



Exercise 1.2

? RAM
? MB Disk
? min
(each)



? RAM
? GB Disk
? min

Questions?

- Now: “Joe’s Workflow” Exercise 1.1,1.2
 - In groups of 2-3
 - Read carefully!
- Later:
 - Lecture: From Workflow to Production
 - Exercises 2.1, 2.2