Getting the Most out of HTC with Workflows

Friday

Lauren Michael <lmichael@wisc.edu>
Research Computing Facilitator
University of Wisconsin - Madison

Open Science Grid

Why are we here?

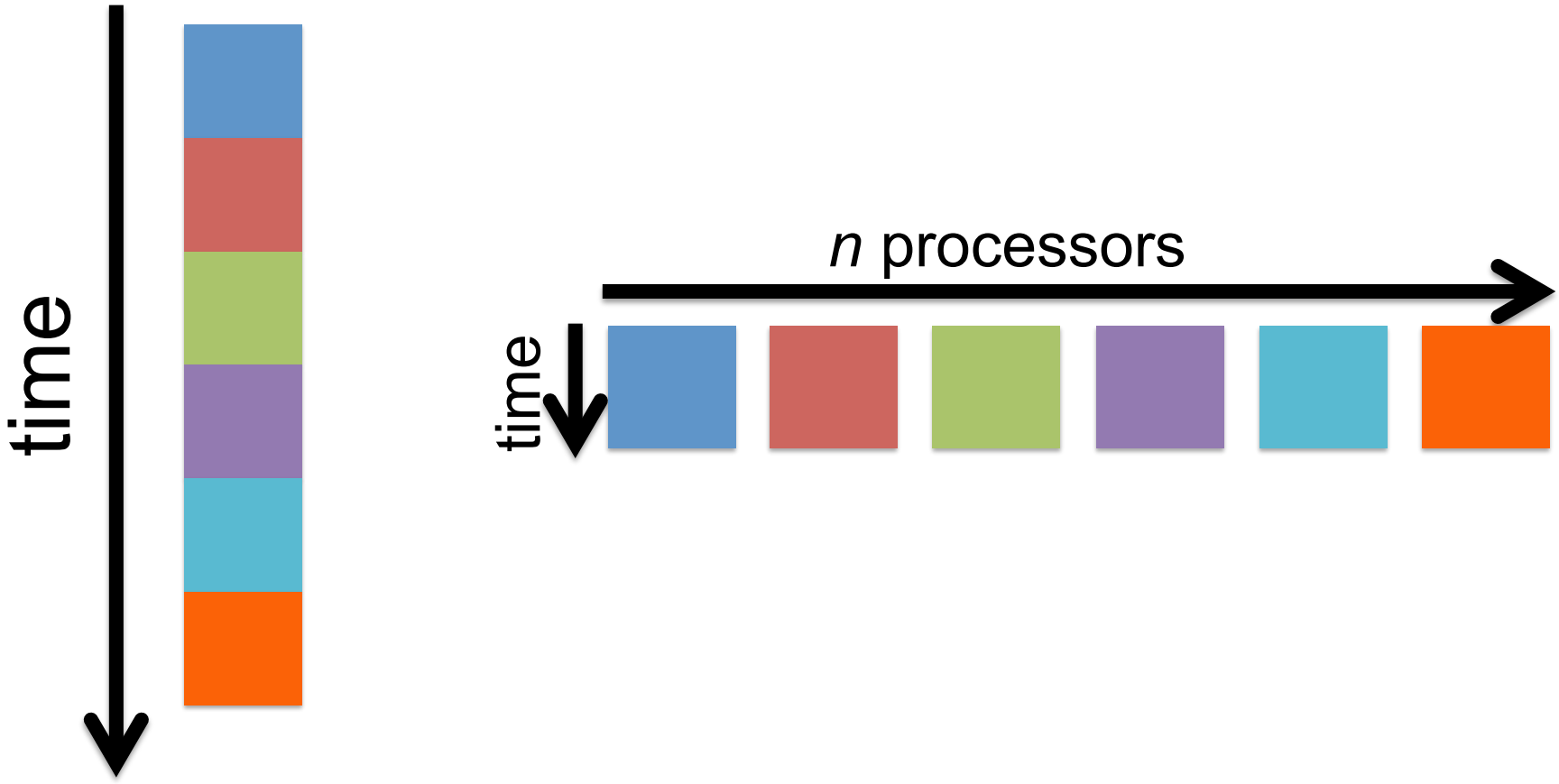
Why are we here?

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)

HTC Payoffs

For science with *MANY* independent calculations…



Open Science Grid

Focus on Throughput

What is *throughput* in computing?

- time from *submission* to *overall completion*

What is *High Throughput* Computing?

- many 'smaller' independent tasks
- optimizing time-to-completion
  - including automation of HTC and non-HTC steps within an overall "*workflow*"

Open Science Grid

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

Open Science Grid

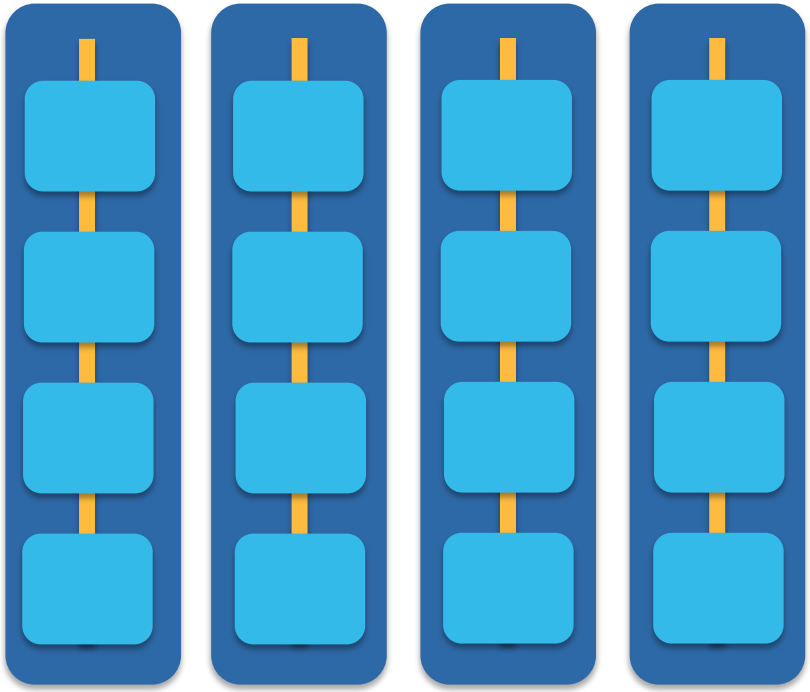
Typical HTC Problems

- 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* or programs/functions that use multiple CPUs on the same server

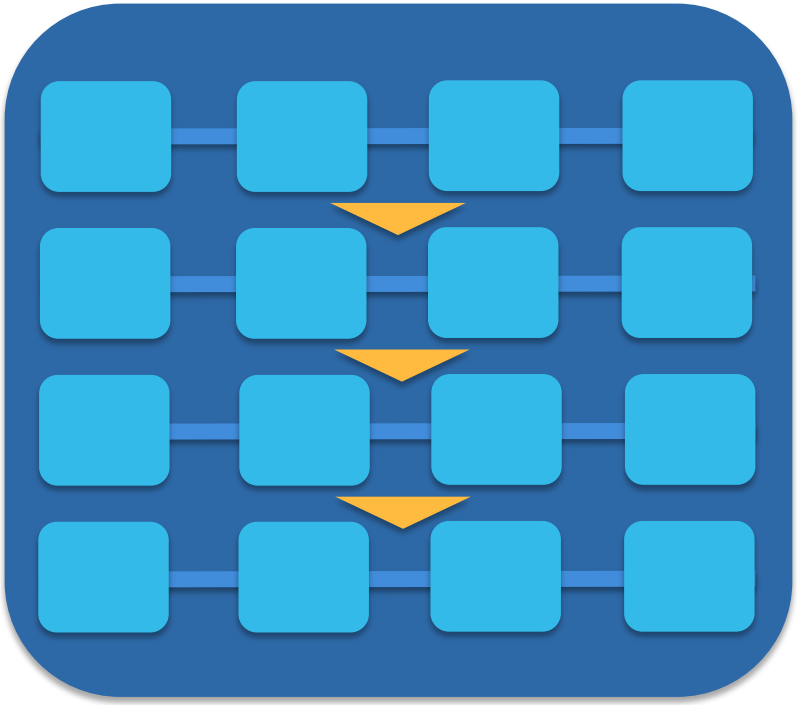
Ultimately: Can you break it up?

Parallelization and Throughput

high-throughput

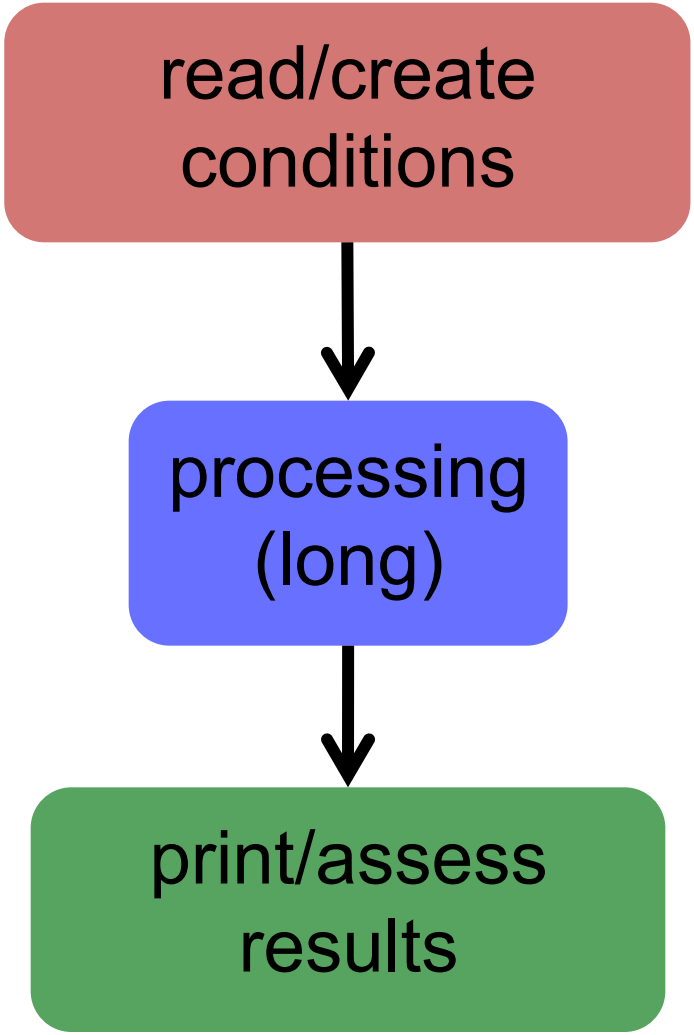


high-performance (e.g.MPI)



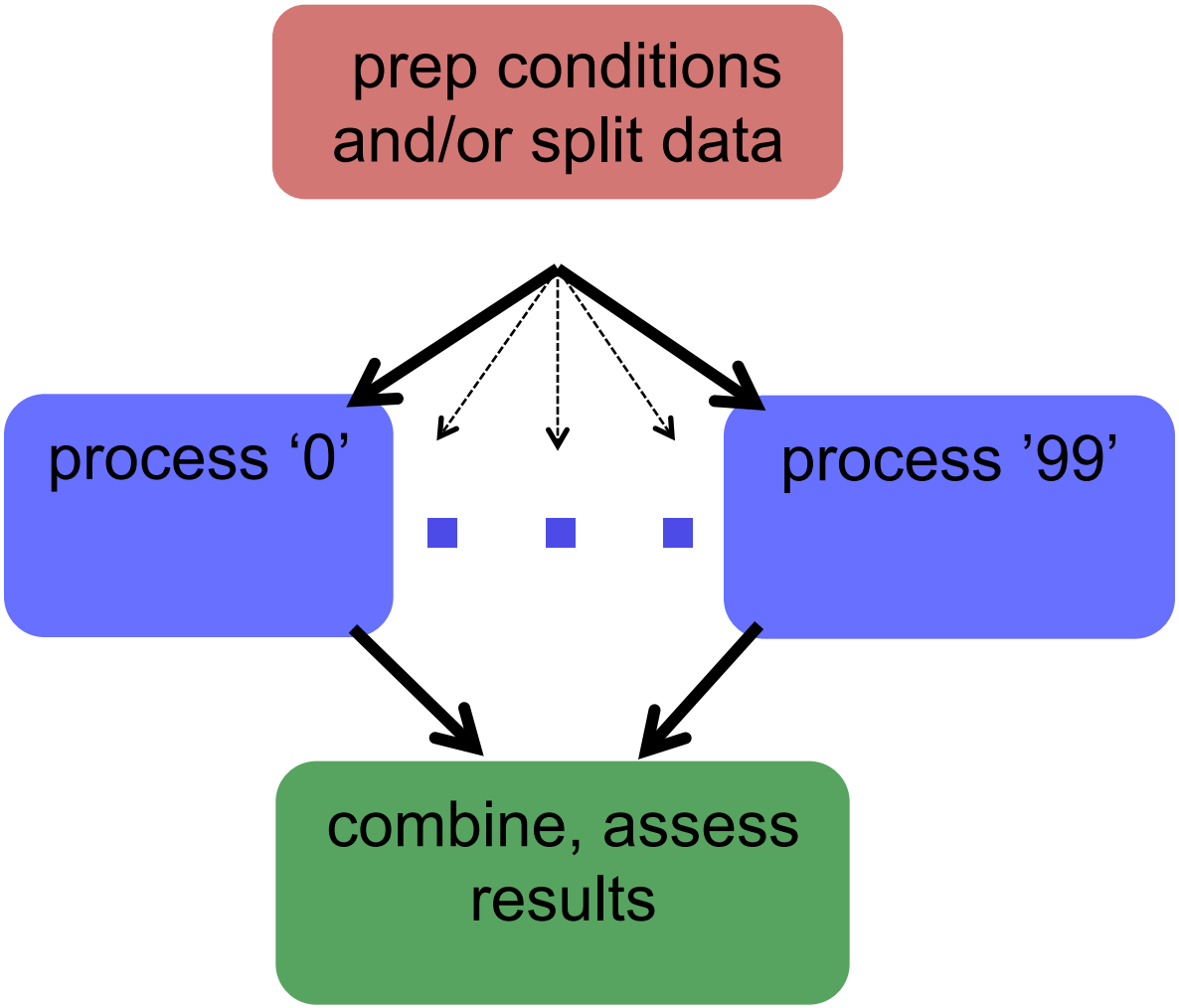
Open Science Grid

Many programs



Open Science Grid

with HTC!



Open Science Grid

Key HTC Tactics

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

Open Science Grid

Key HTC Tactics

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

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

```
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
           Cpus                 :                1         1
           Disk (KB)            :   122358   125000  13869733
           Memory (MB)          :       30      100       100
```

Open Science Grid

Key HTC Tactics

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

Open Science Grid

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

Open Science Grid

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

Open Science Grid

Key HTC Tactics

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

Open Science Grid

Bring *What* with You?

- Software (that was Wednesday)
- Parameters and random numbers
  - create a single, standard executable, responsive to:
    - arguments
    - input files (better)
  - generate and record ahead of time
    - reproducibility!
    - perhaps in an earlier DAG job
- What else?

Open Science Grid

Wrapper Scripts are Essential

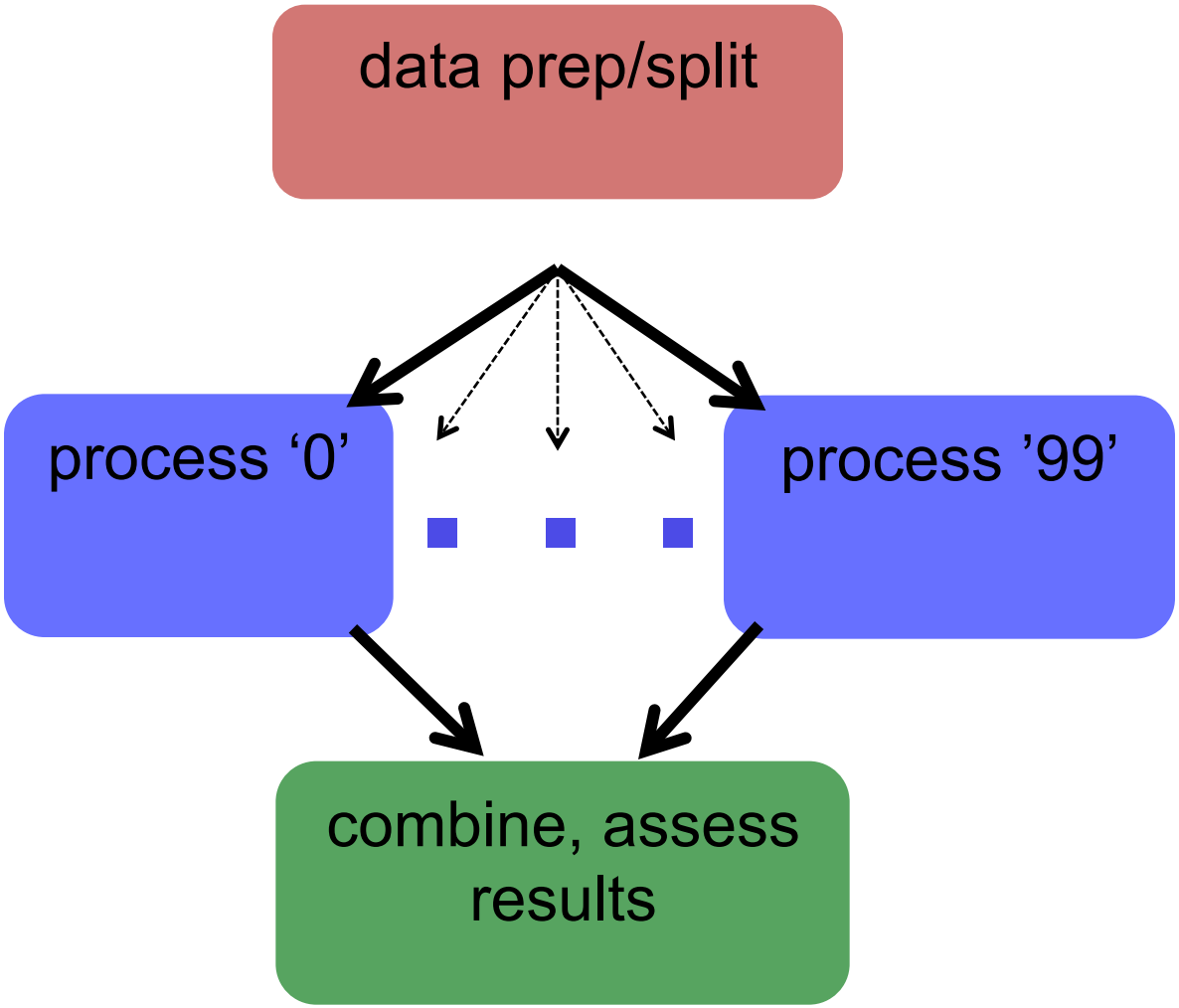
- Before task execution (bring it with you!)
  - transfer/prepare files and directories
  - setup/configure software environment and other dependencies
- Task execution
  - prepare complex commands and arguments
  - batch together many ‘small’ tasks
- After task execution
  - filter/combine/compress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

Open Science Grid

Key HTC Tactics

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

DAGs Automate Workflows!



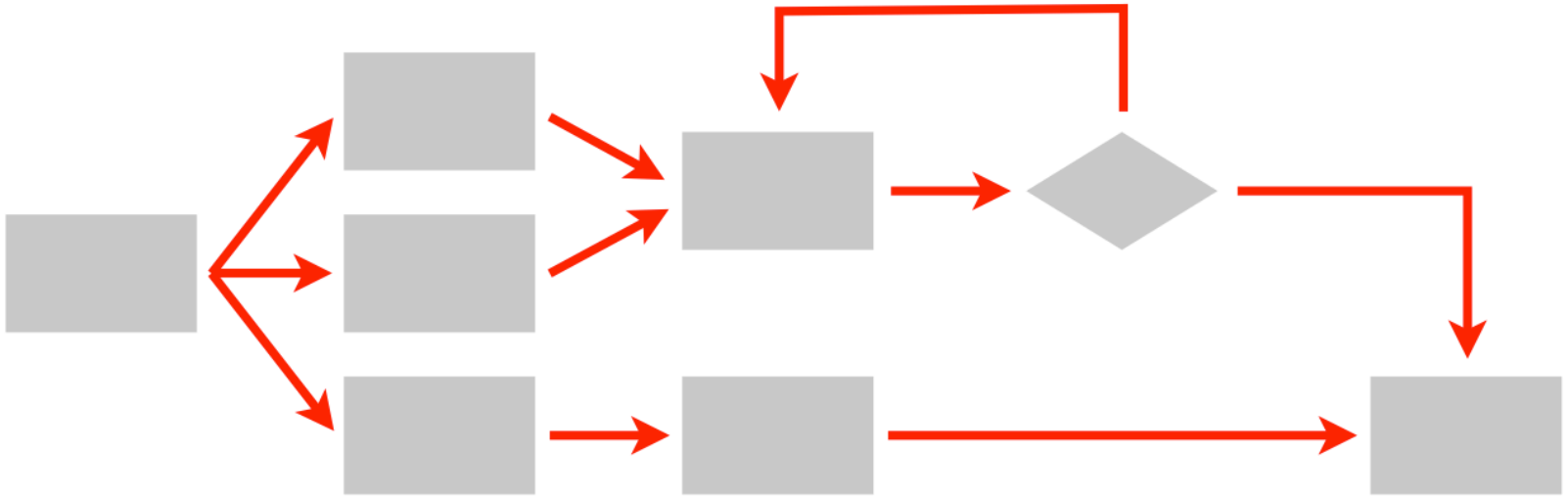
Workflows *Should* Make ~~Life~~ Science Easier

- non-computing "workflows" are all around you … especially in science
  - instrument setup
  - experimental procedures

- when planned/documented, workflows help with:
  - organizing and managing processes
  - saving time with **automation**
  - objectivity, reliability, and reproducibility (THE TENENTS OF GOOD SCIENCE!)

Workflows are like Computing Algorithms

- Steps
- Connections
- (Metadata)

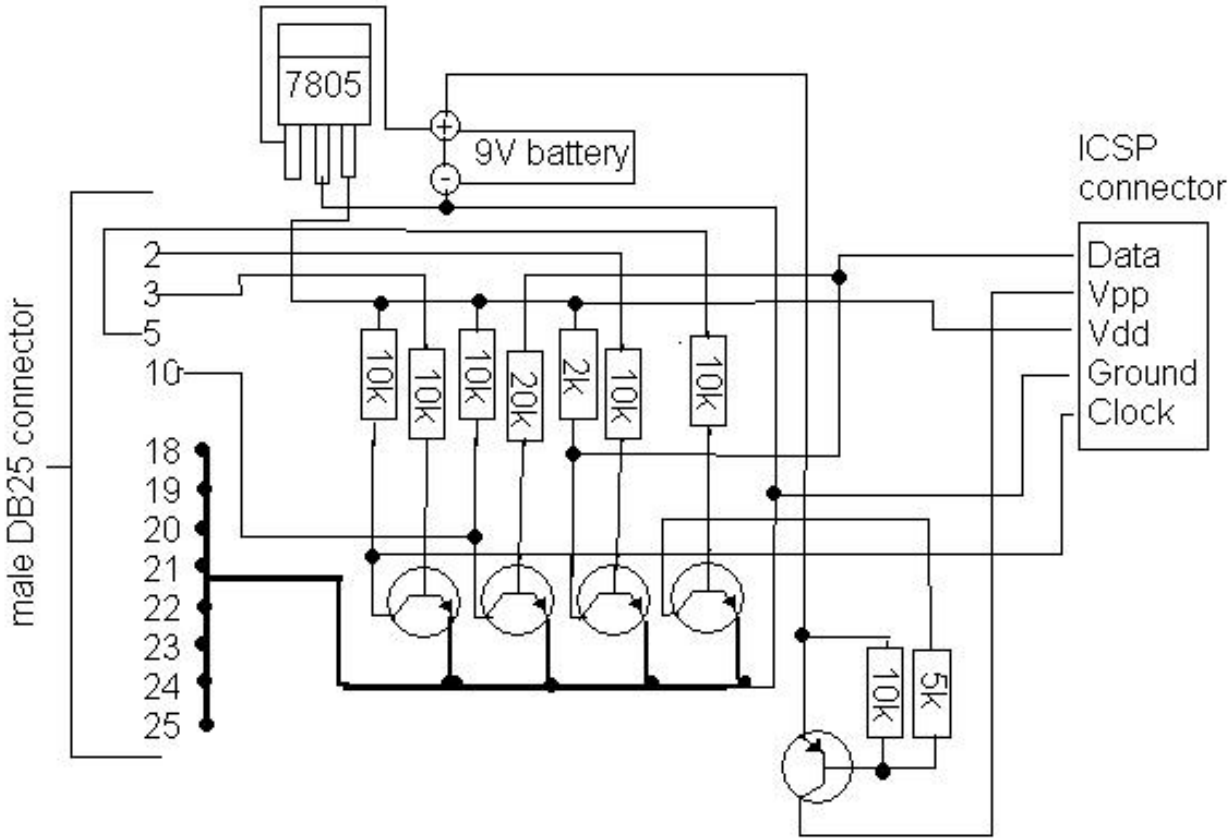


Building a Good Workflow

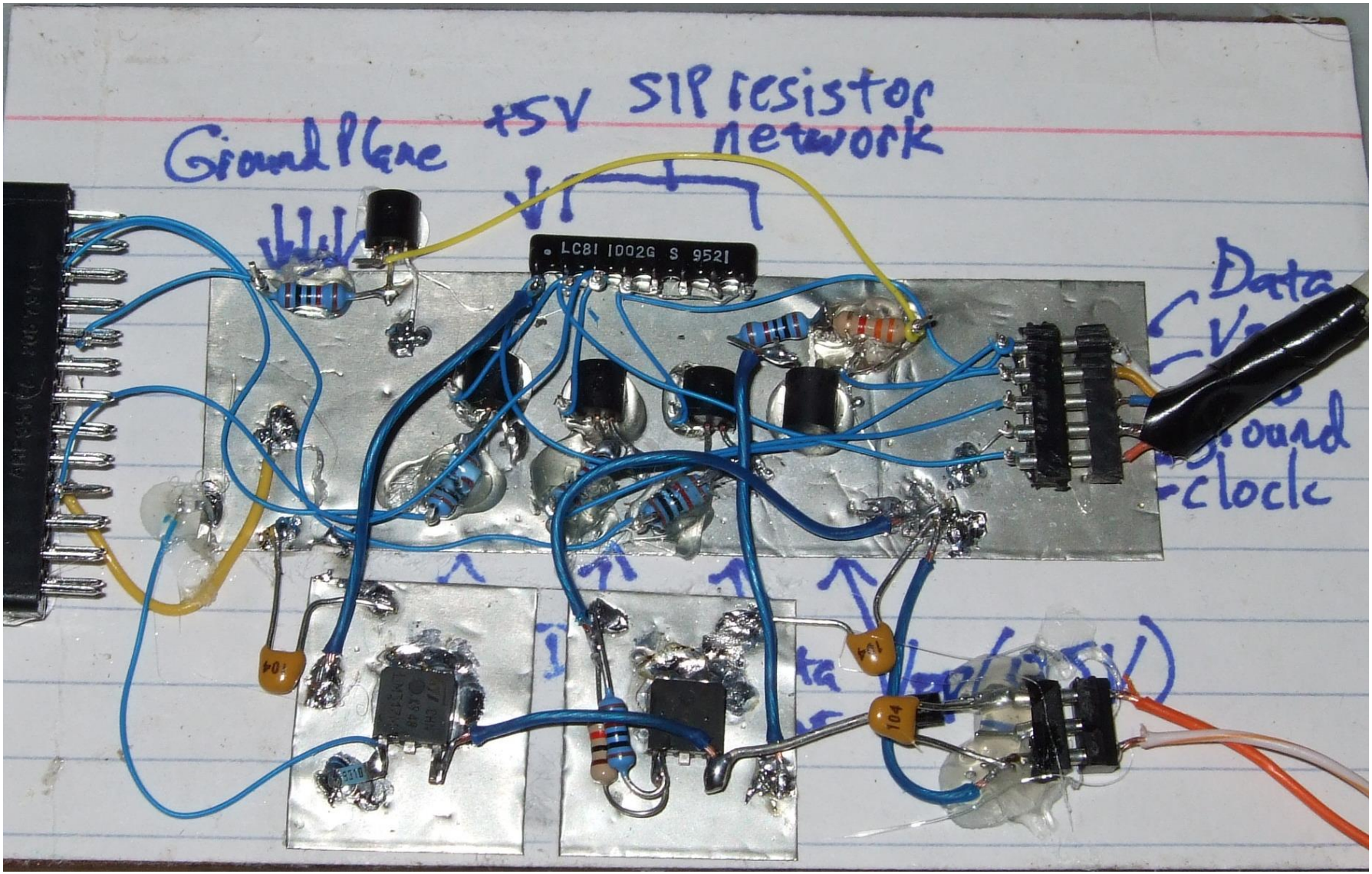
1. Draw out the *general* workflow
2. Define details (test 'pieces' with HTCondor jobs)
   - divide or consolidate 'pieces'
   - determine resource requirements
   - identify steps to be automated or checked
3. Build it modularly; test and optimize
4. Scale-up gradually
5. What more can you automate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

(And remember to document!)

From schematics…



… to the real world



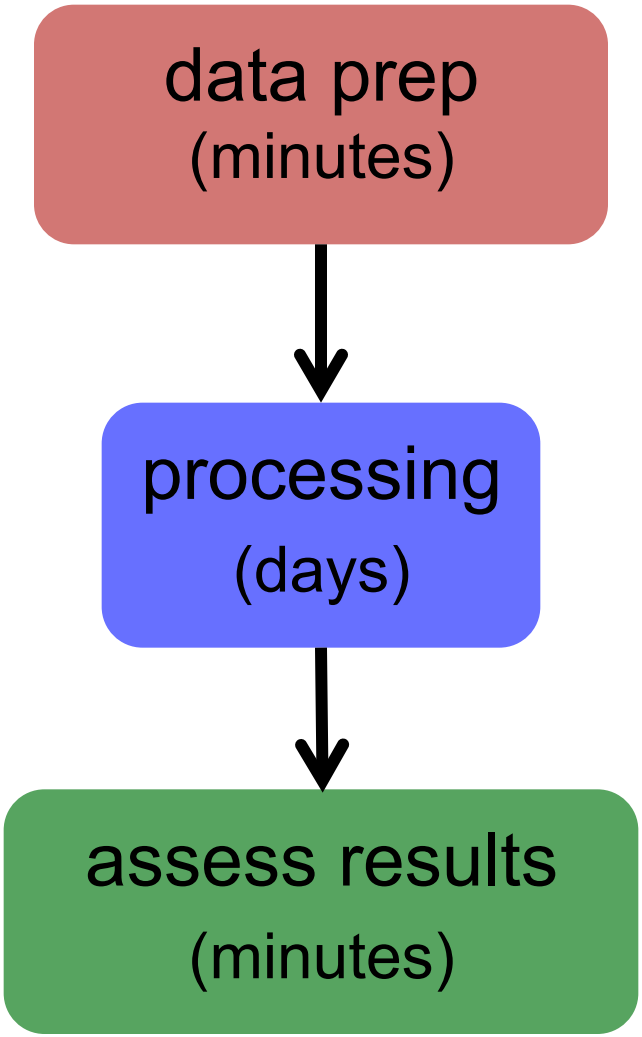
Building a Good Workflow

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. What more can you automate or error-check?

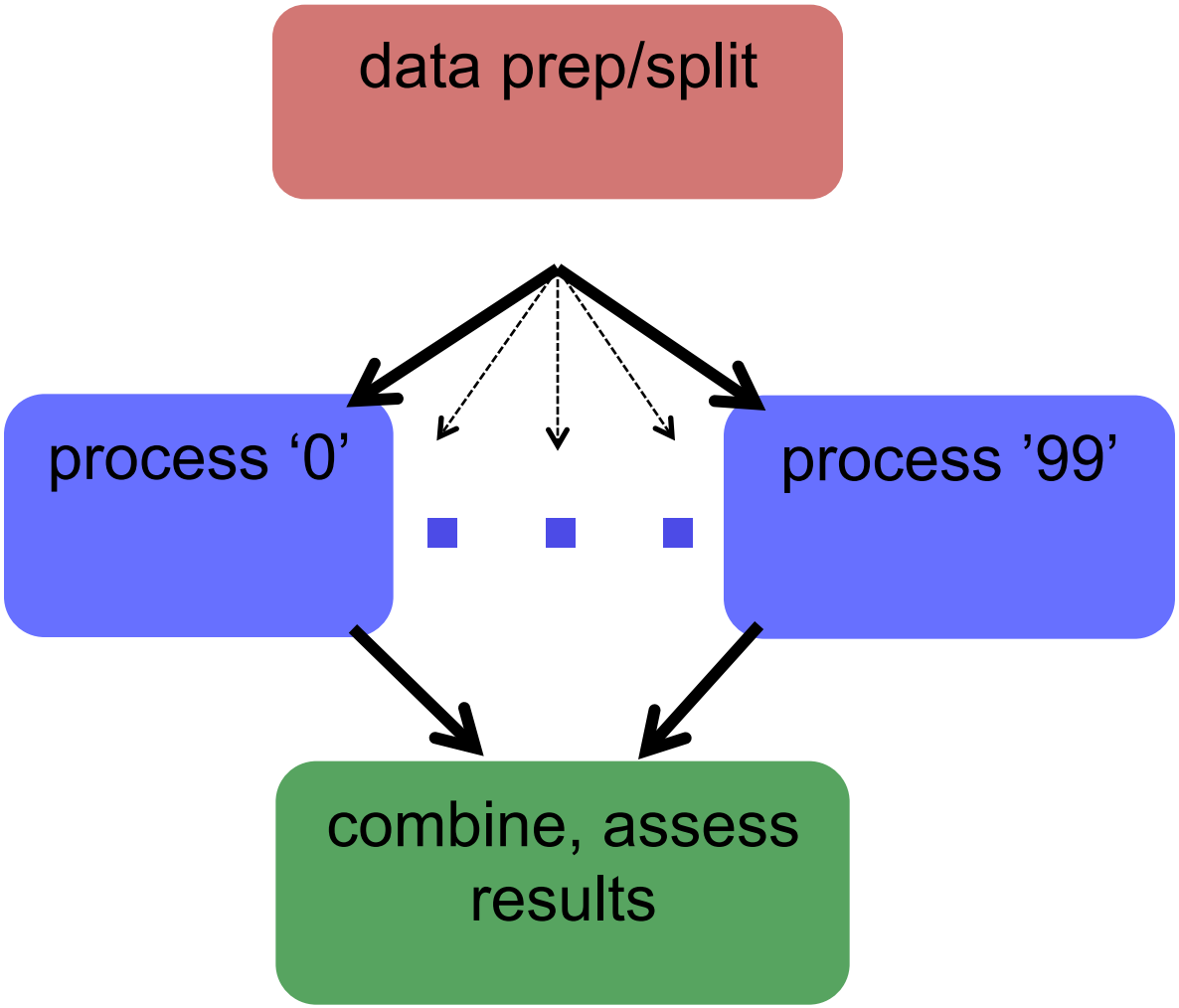
(And remember to document!)

Open Science Grid

Start with This



Parallelize with HTC Splitting



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. What more can you automate or error-check?

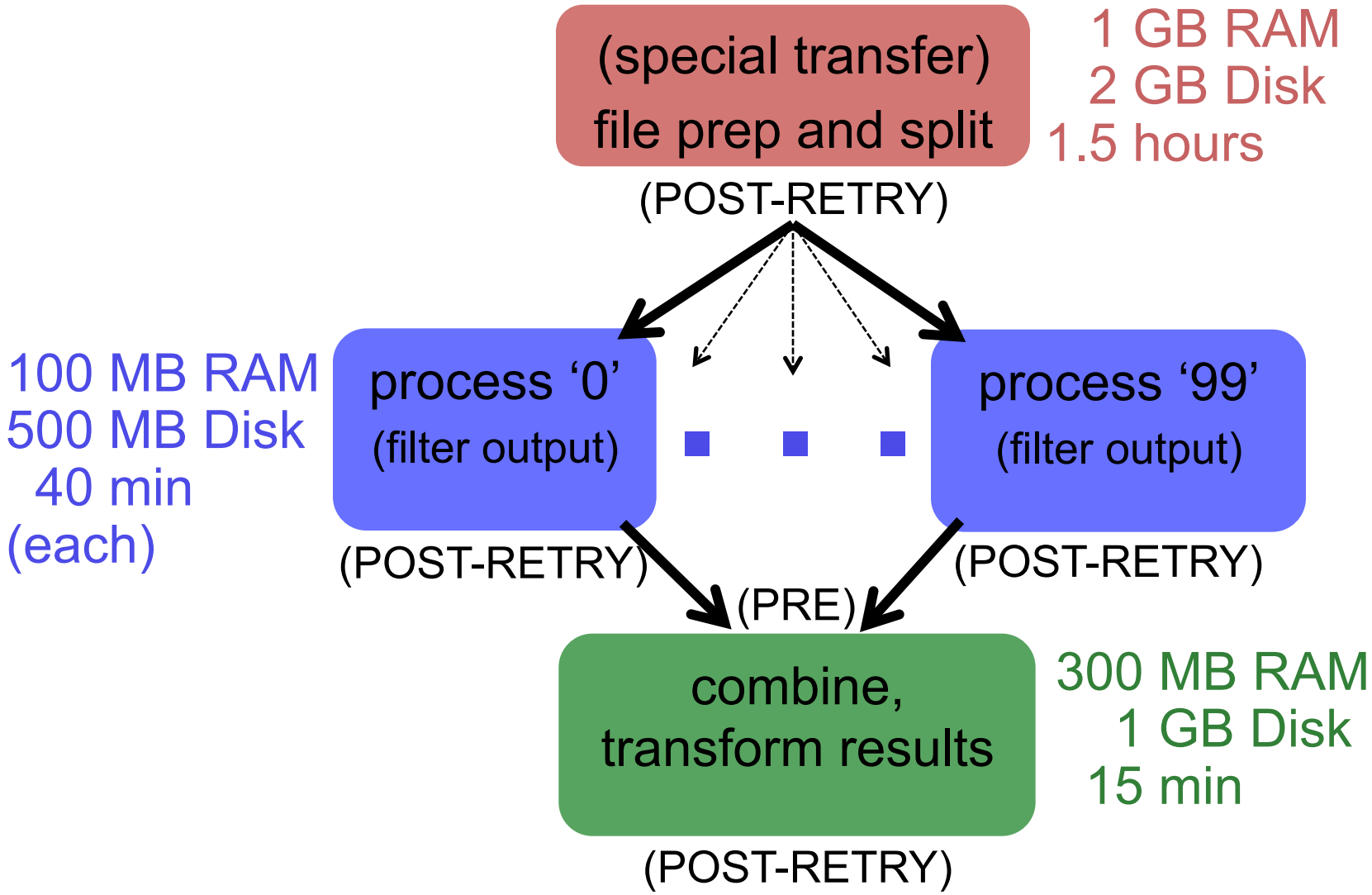
(And remember to document!)

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



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. **What more can you automate or error-check?**

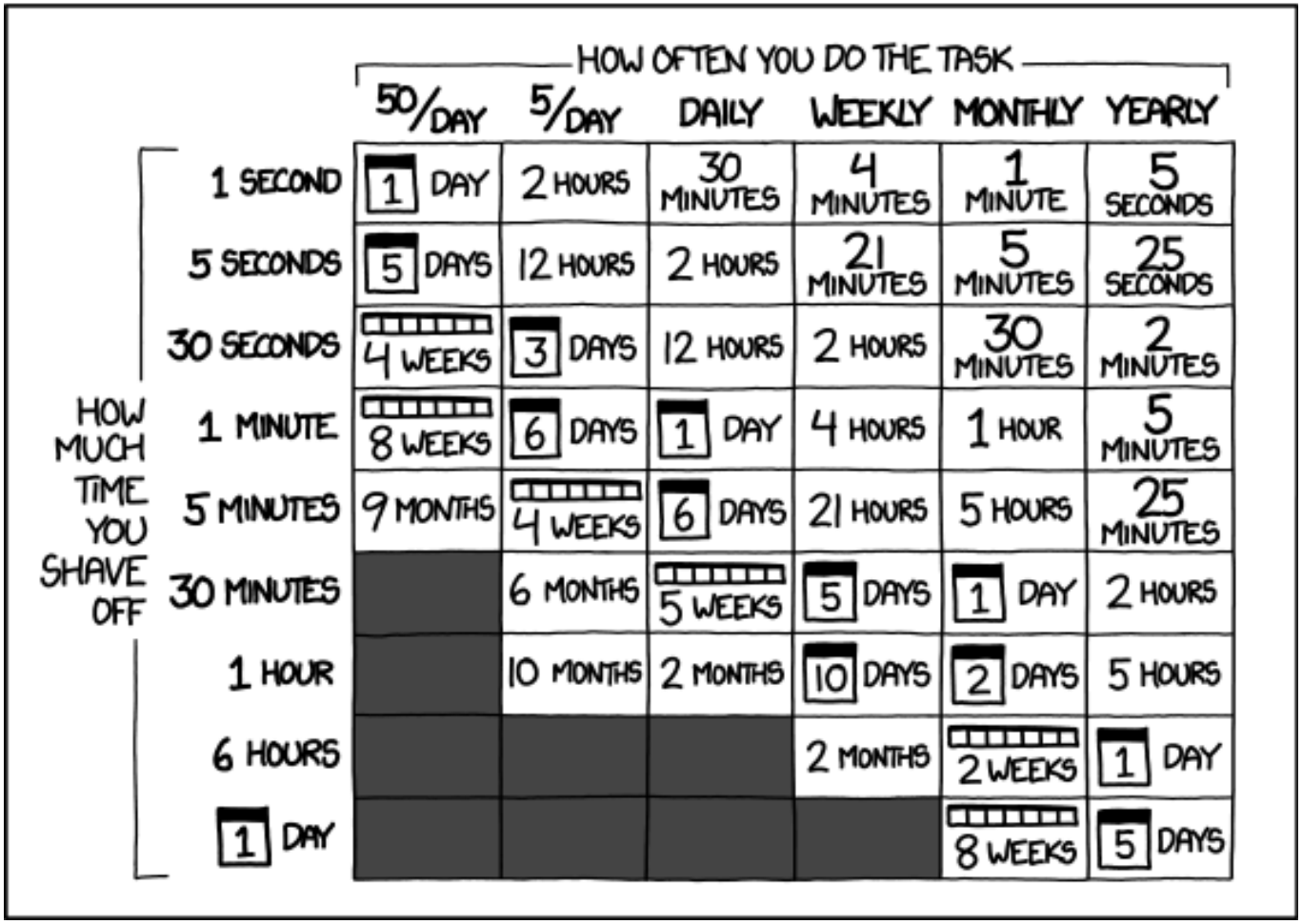
(And remember to document!)

Open Science Grid

Key HTC Tactics

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

Automation Trade-offs!

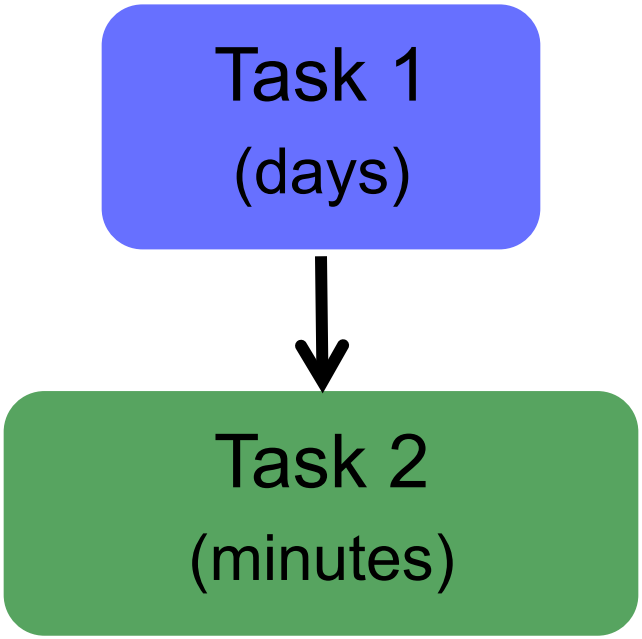


http://xkcd.com/1205/

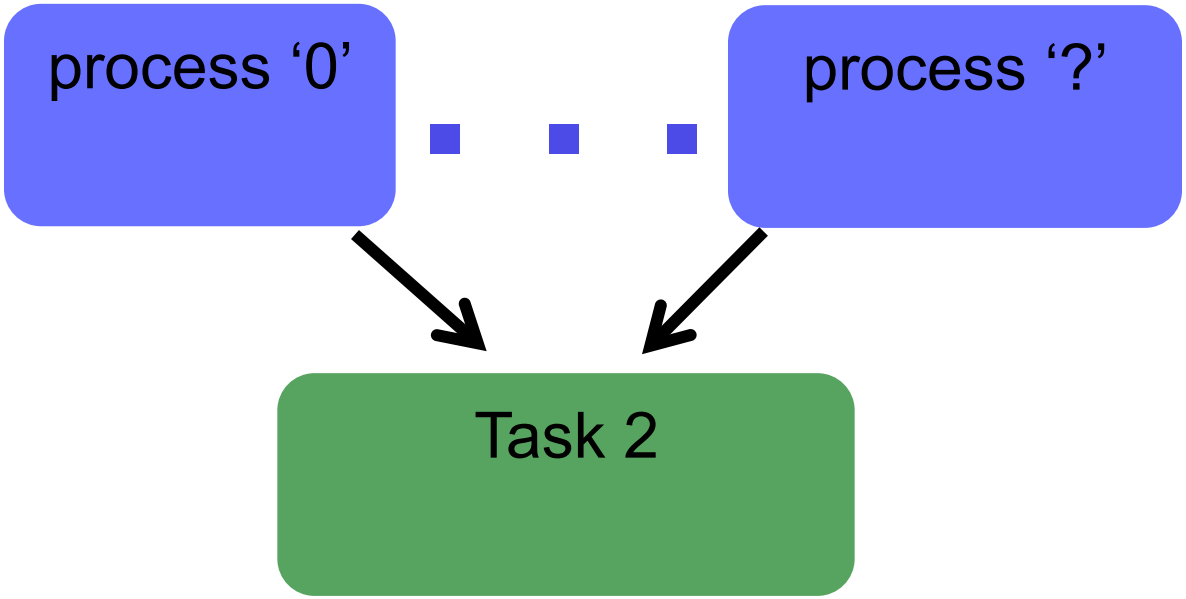
… but there are even more benefits of automating workflows!!

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

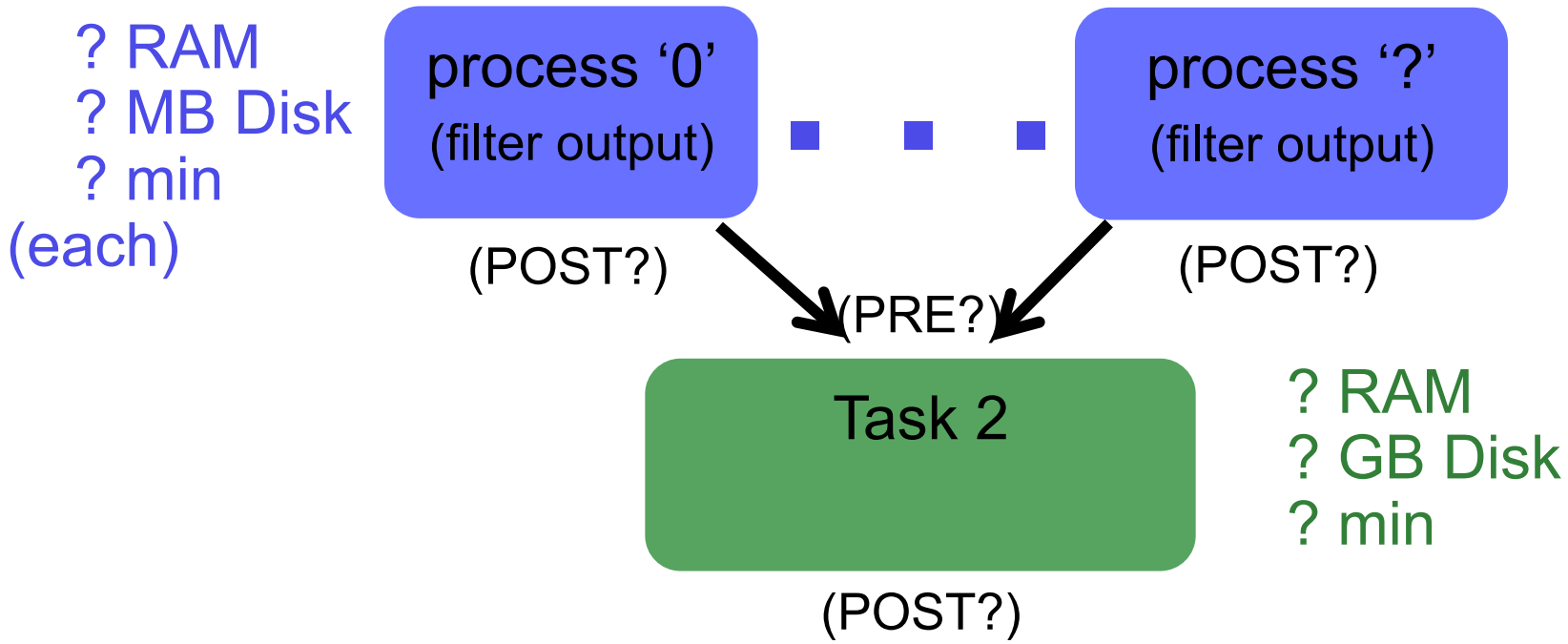
Exercise 1



Exercise 1



Exercise 2


? RAM
? MB Disk
? min
(each)
process '0'
(filter output)
process '?'
(filter output)
(POST?)
(POST?)
(PRE?)
Task 2
? RAM
? GB Disk
? min
(POST?)

Open Science Grid

Questions?

- Feel free to contact me:
  - lmichael@wisc.edu
- Now: “Joe’s Workflow” Exercise 1.1,1.2
  - In groups of 2-3
- Later:
  - Lecture: From Workflow to Production
  - Exercises 1.3, 1.4