Putting It All Together: Optimizing Workflows

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

Open Science Grid

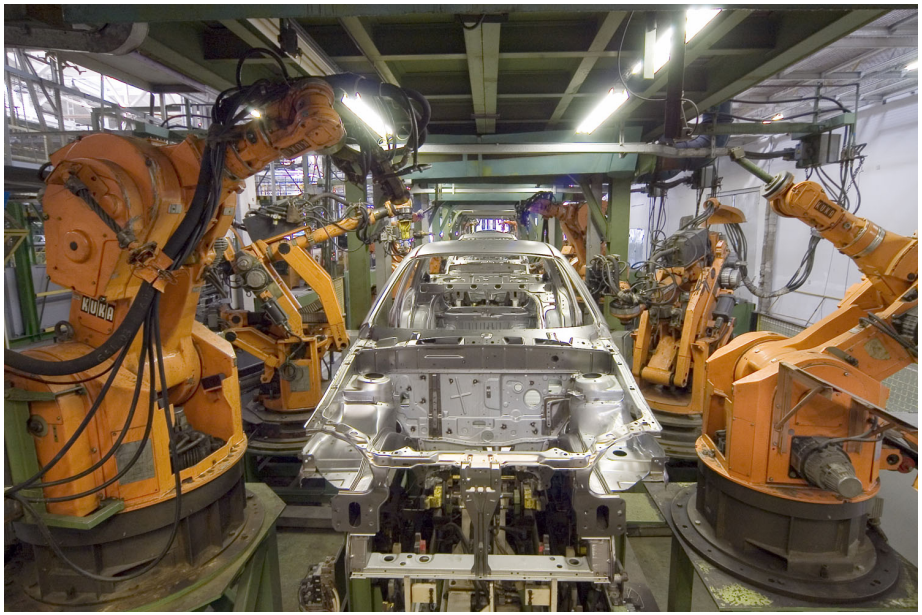
Key HTC Tactics

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**

Make it easier to manage all your jobs

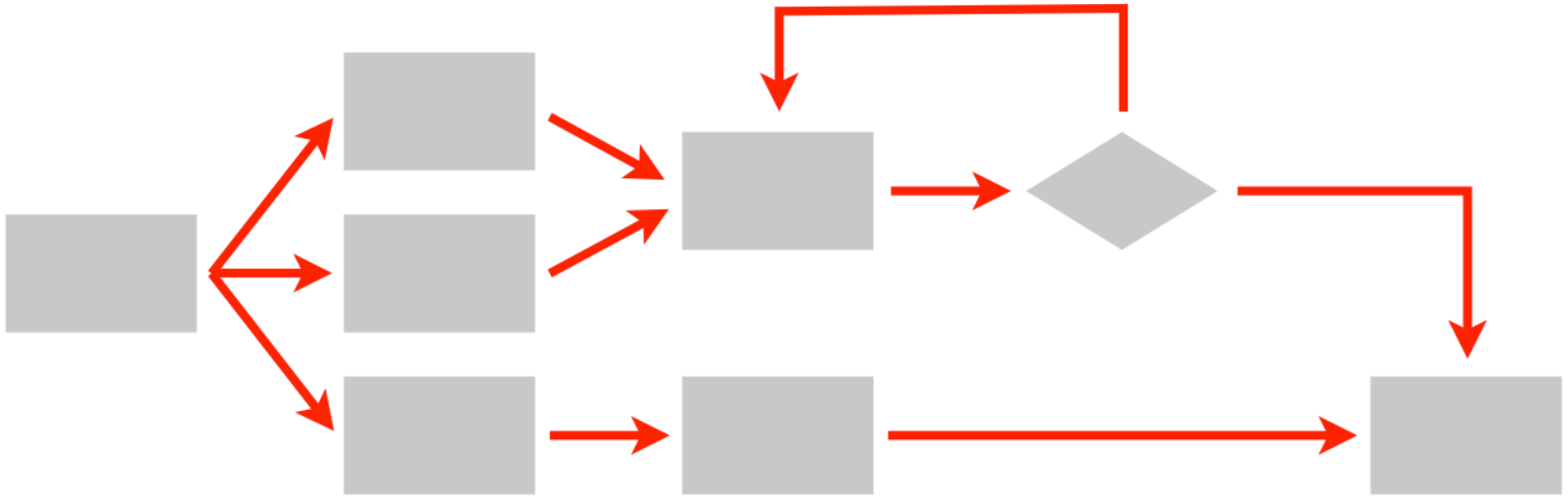
What to Automate?

- 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

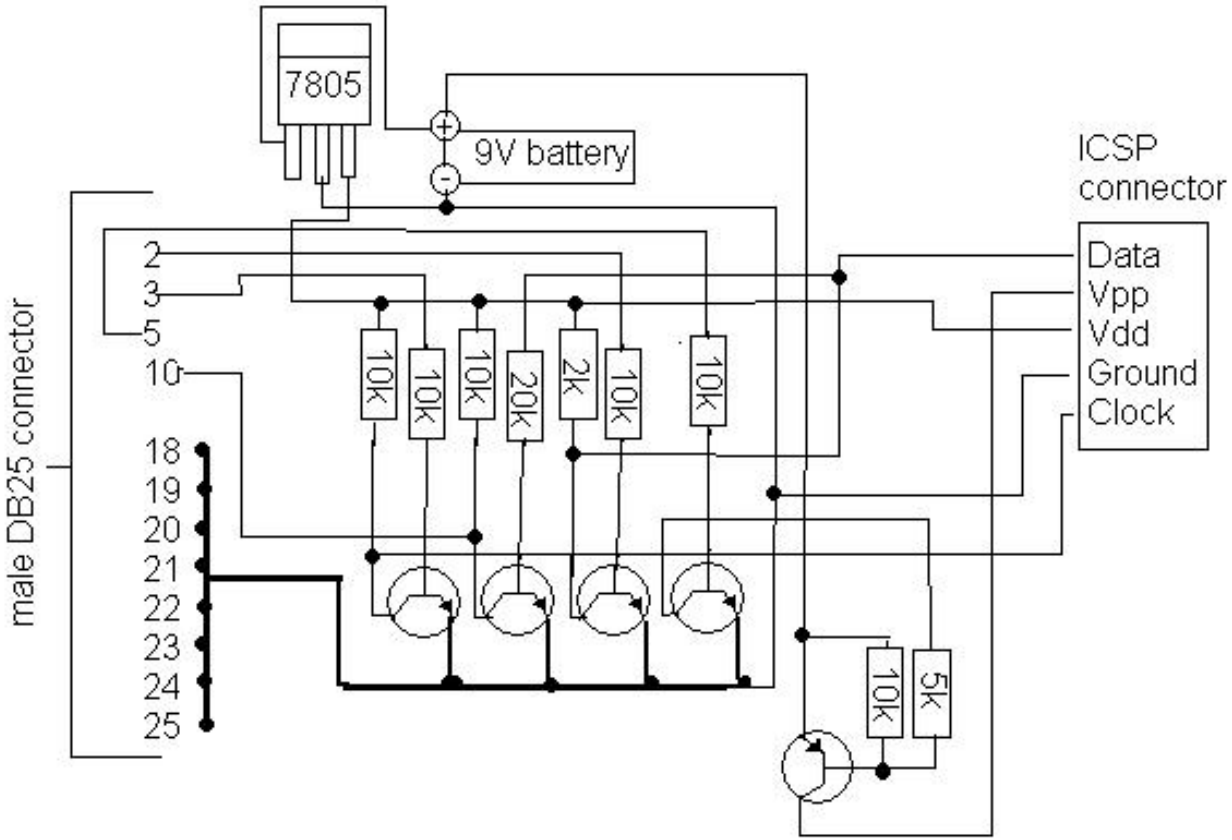
- non-computing "workflows" are all around you, especially in science
  - instrument setup
  - experimental procedures and protocols

- when planned/documented, workflows help with:
  - organizing and managing processes
  - saving time with **automation**
  - objectivity, reliability, and reproducibility

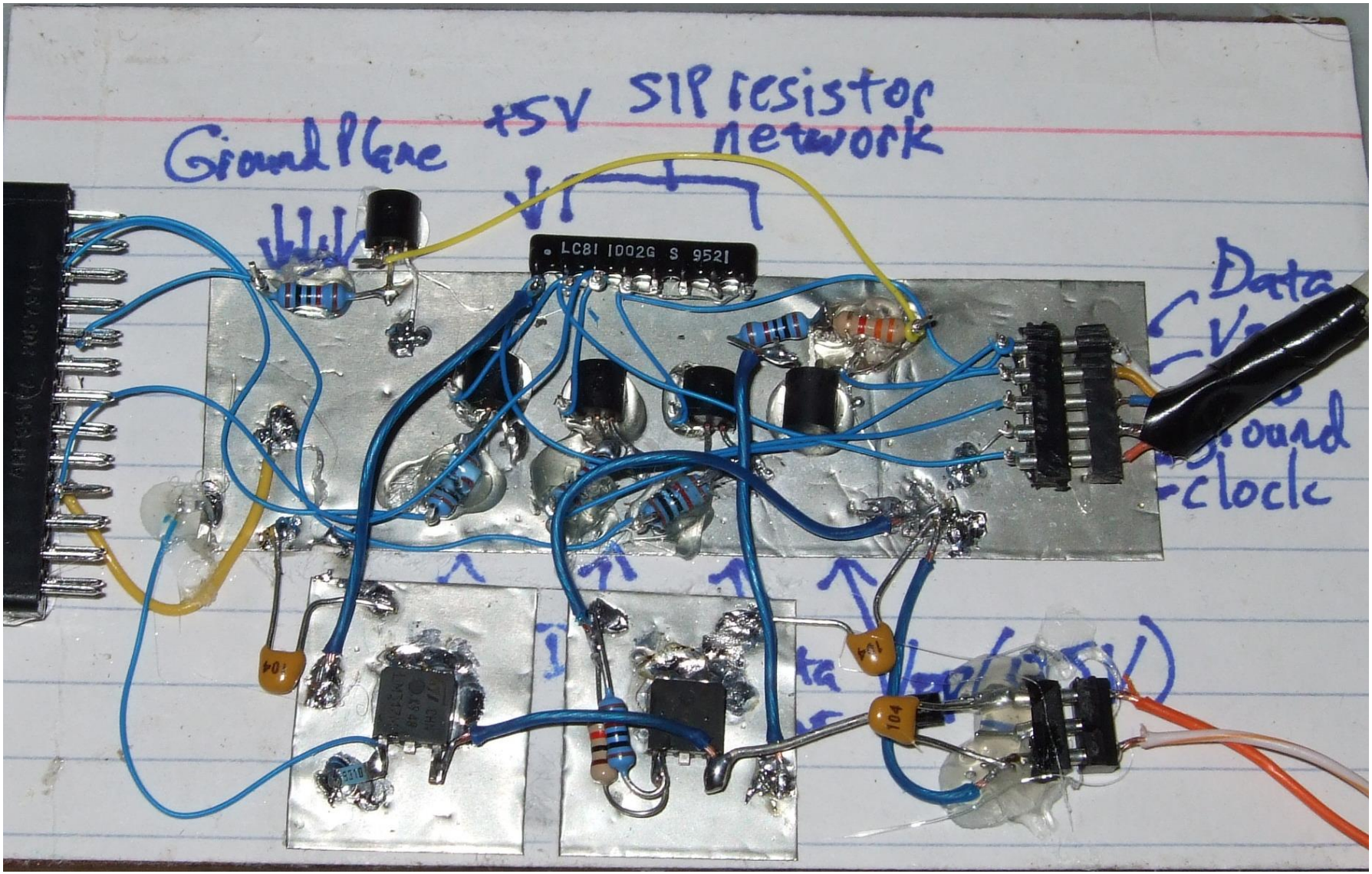
    (THE TENETS OF GOOD SCIENCE!)

Getting the most out of workflows

From schematics…



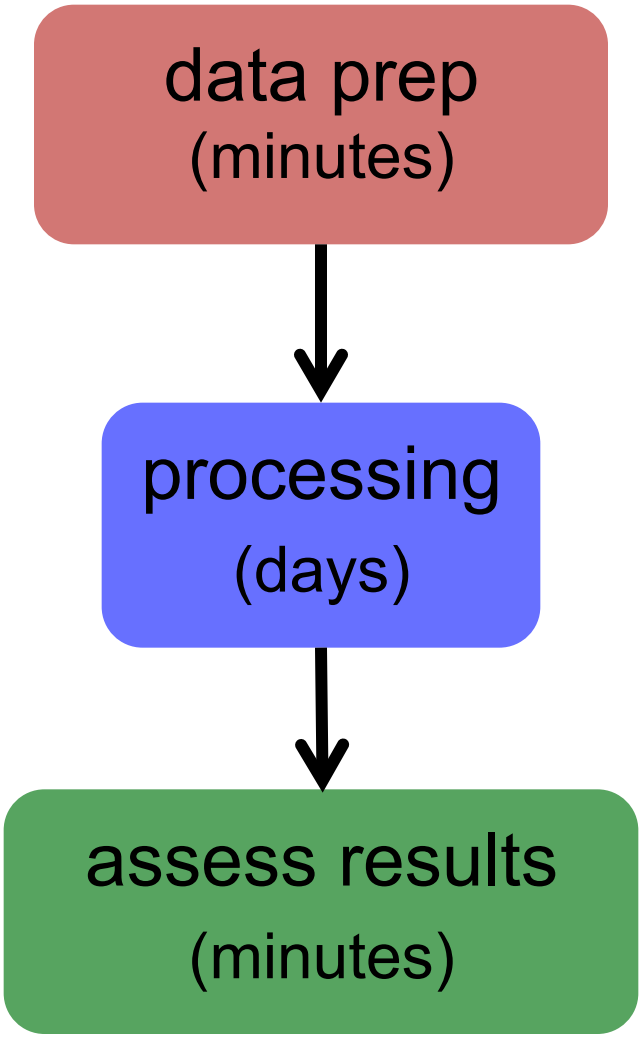
… to the real world



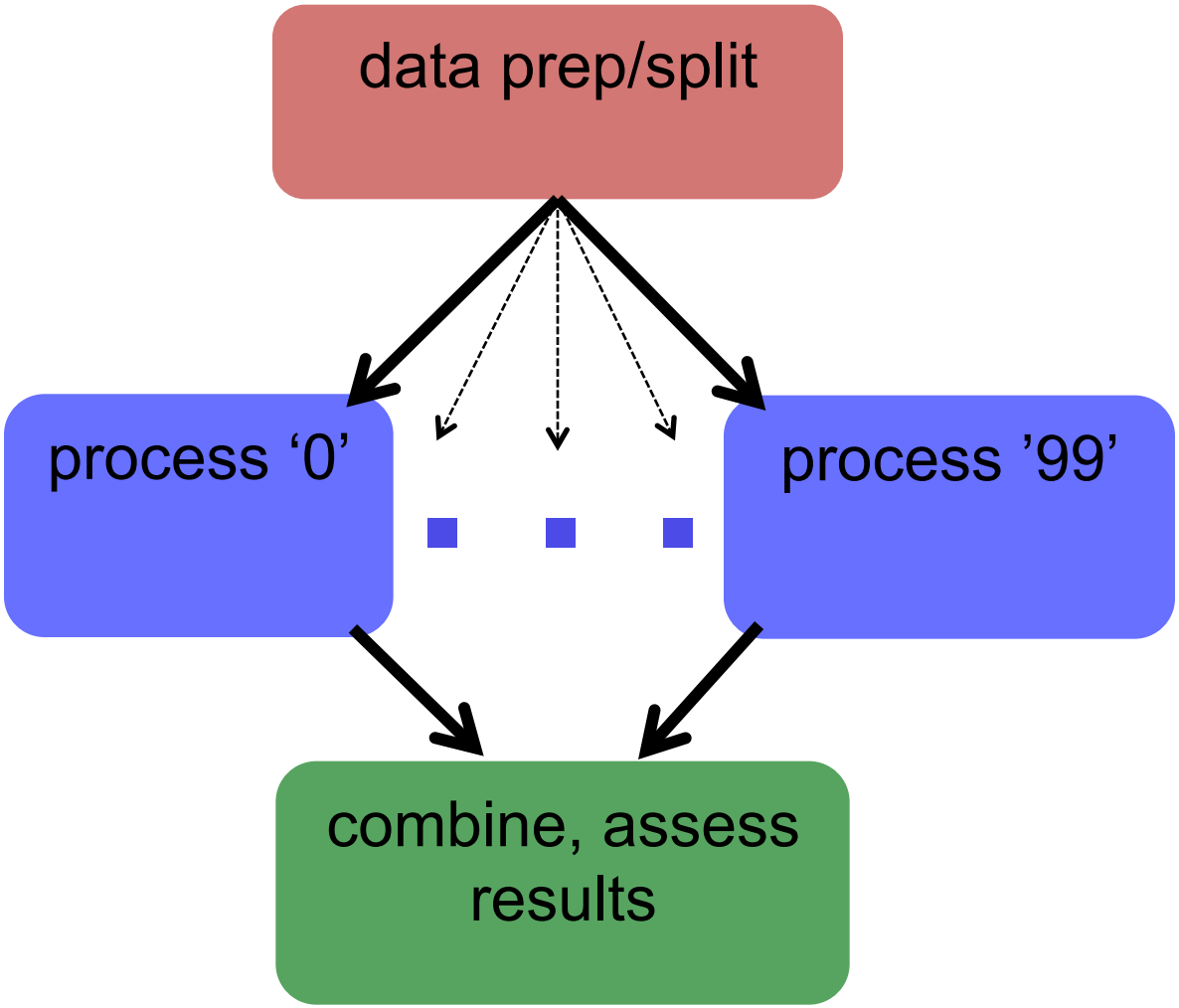
Optimizing a Workflow

1. **Draw out the workflow, identify pieces**
2. Modular development: test and optimize each piece
   - divide or consolidate ‘pieces’
   - determine resource requirements
   - identify steps to be automated or checked
3. Put the pieces together gradually
4. Bonus features
   - Error proofing
   - Additional automation

Workflow Drawing, v.1



Workflow Drawing, v.2 (w/ HTC)



Open Science Grid

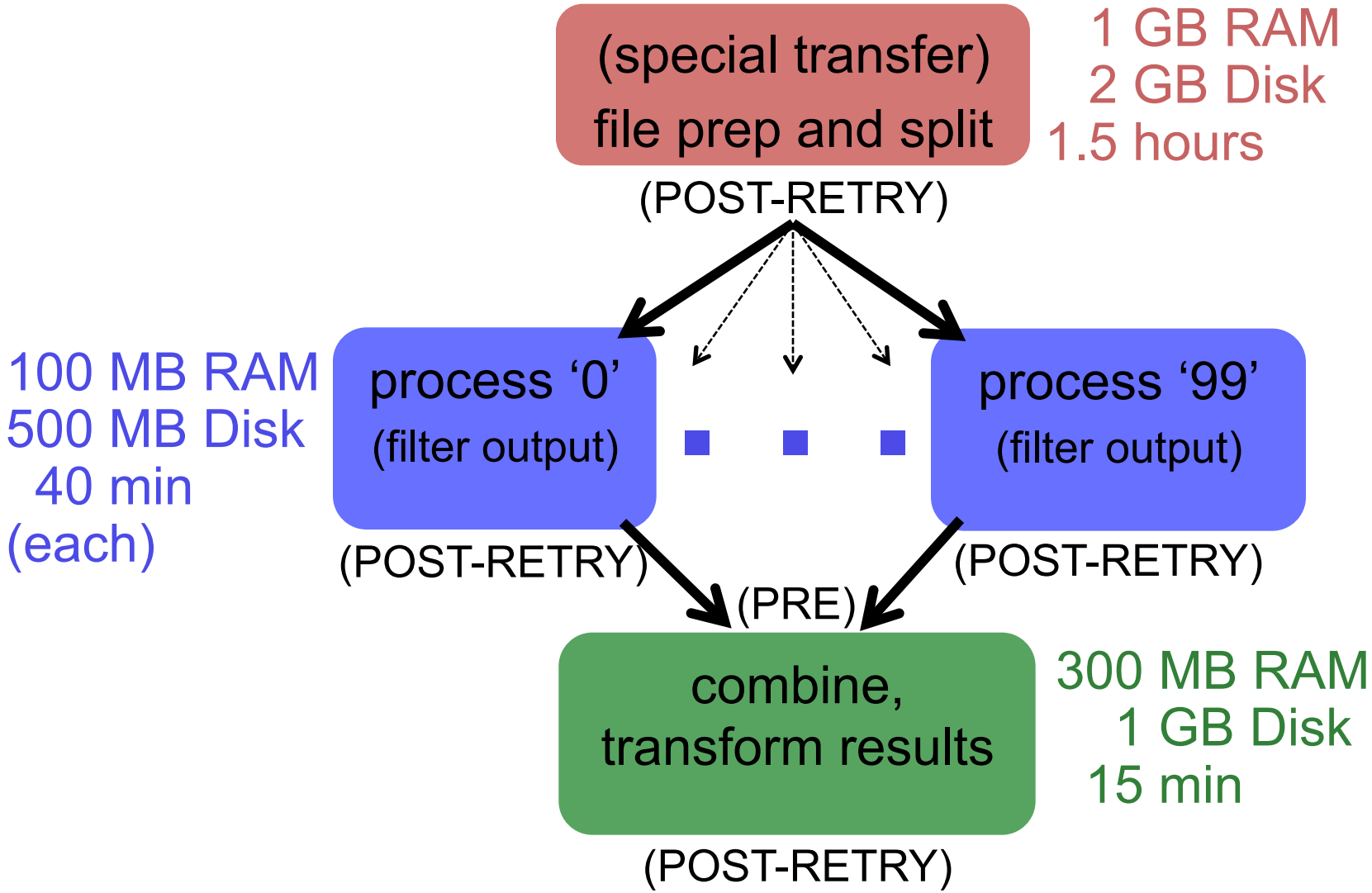
Workflow Pieces

- What are the main steps of the previous workflow?
  - Splitting the data - probably a script?
  - Jobs to process the data - need submit files, scripts, software, etc.
  - Combining the data - probably another script

Optimizing a Workflow

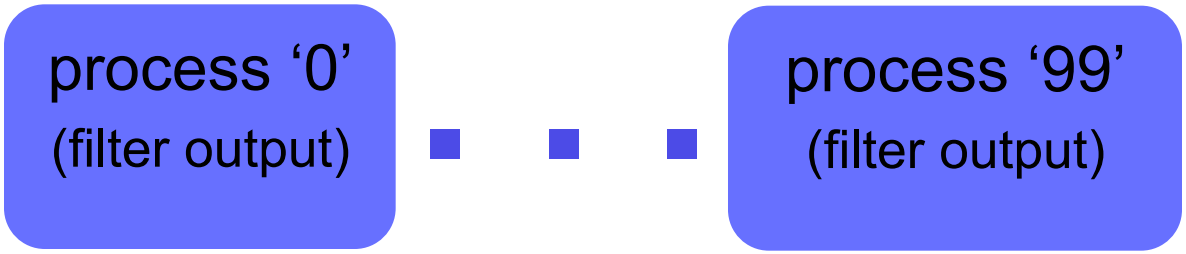
1. Draw out the workflow, identify pieces
2. **Modular development: test and optimize *each piece***
   - divide or consolidate ‘pieces’
   - determine resource requirements
   - identify steps to be automated or checked
3. Put the pieces together gradually
4. Bonus features
   - Error proofing
   - Additional automation

To Get Here …

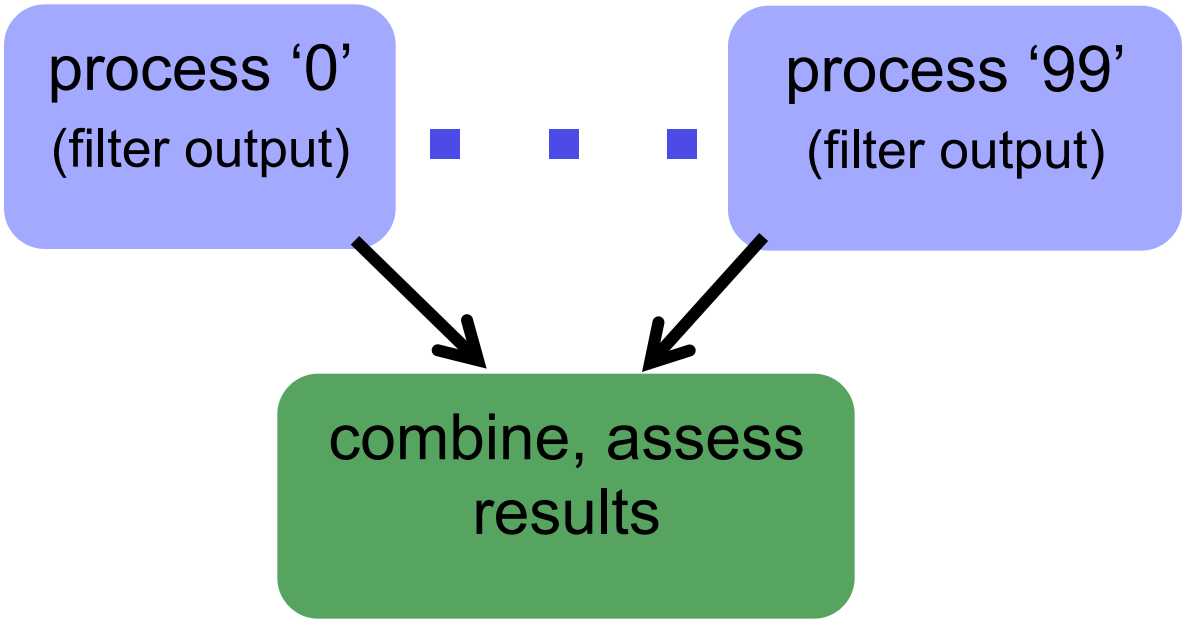

(special transfer)
file prep and split
1 GB RAM
2 GB Disk
1.5 hours
(POST-RETRY)
100 MB RAM
500 MB Disk
40 min
(each)
process ‘0’
(filter output)
process ‘99’
(filter output)
(POST-RETRY)
(POST-RETRY)
(PRE)
combine,
transform results
300 MB RAM
1 GB Disk
15 min
(POST-RETRY)

Start Here

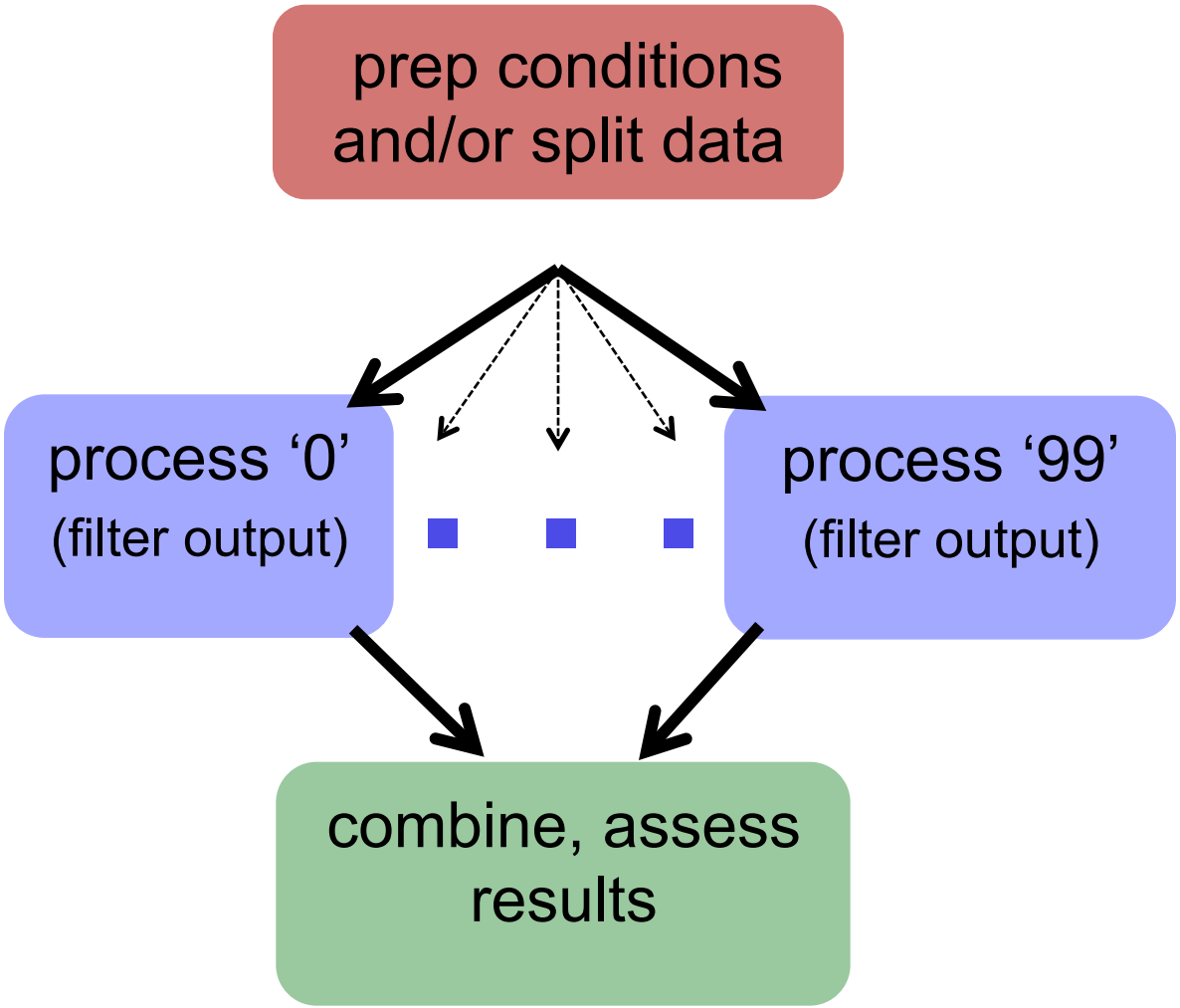
Start with **one** piece of the workflow and apply the testing/optimization ideas from the previous presentation (one job, small test, scale test)



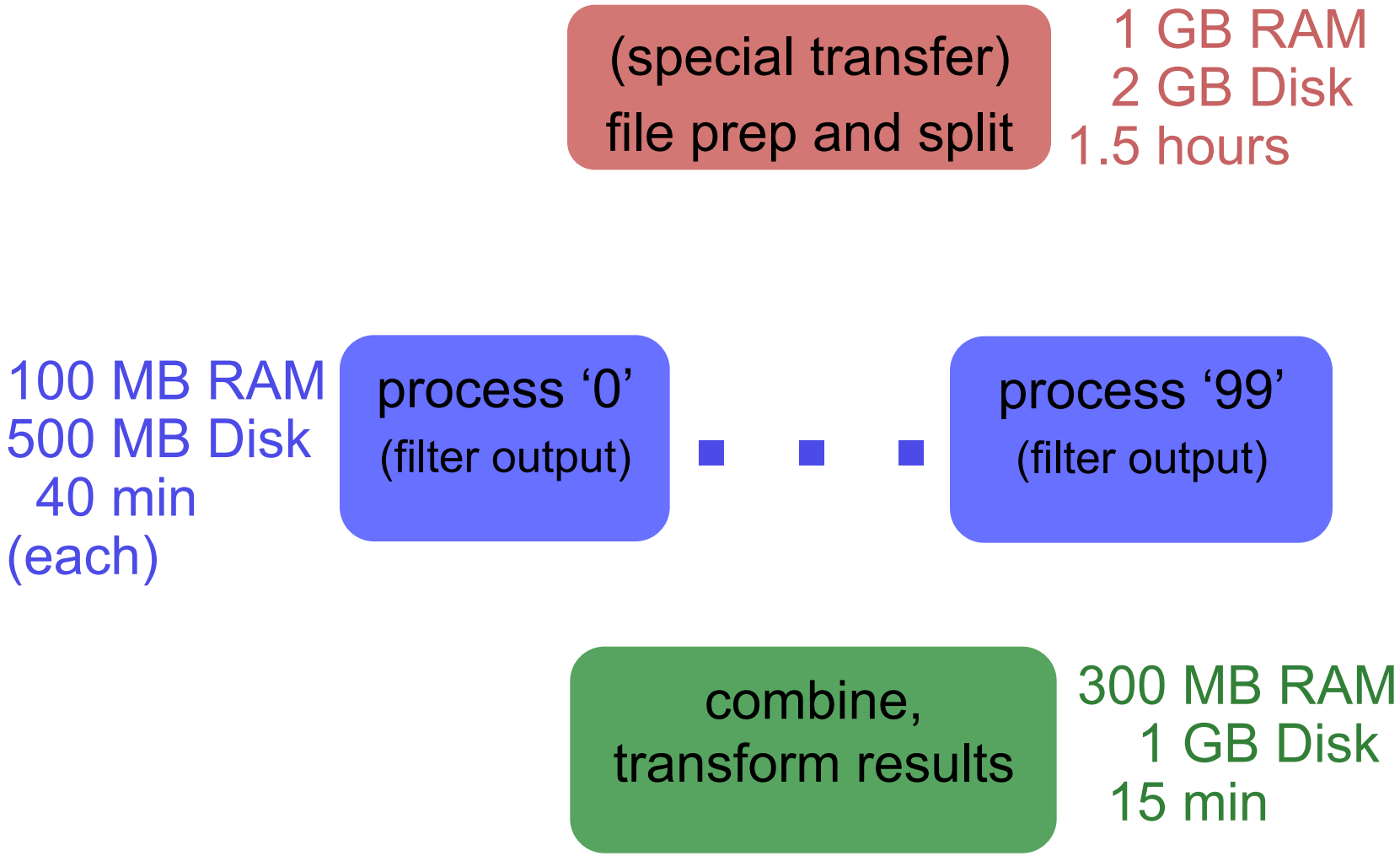
Test Another Step



And Another Step



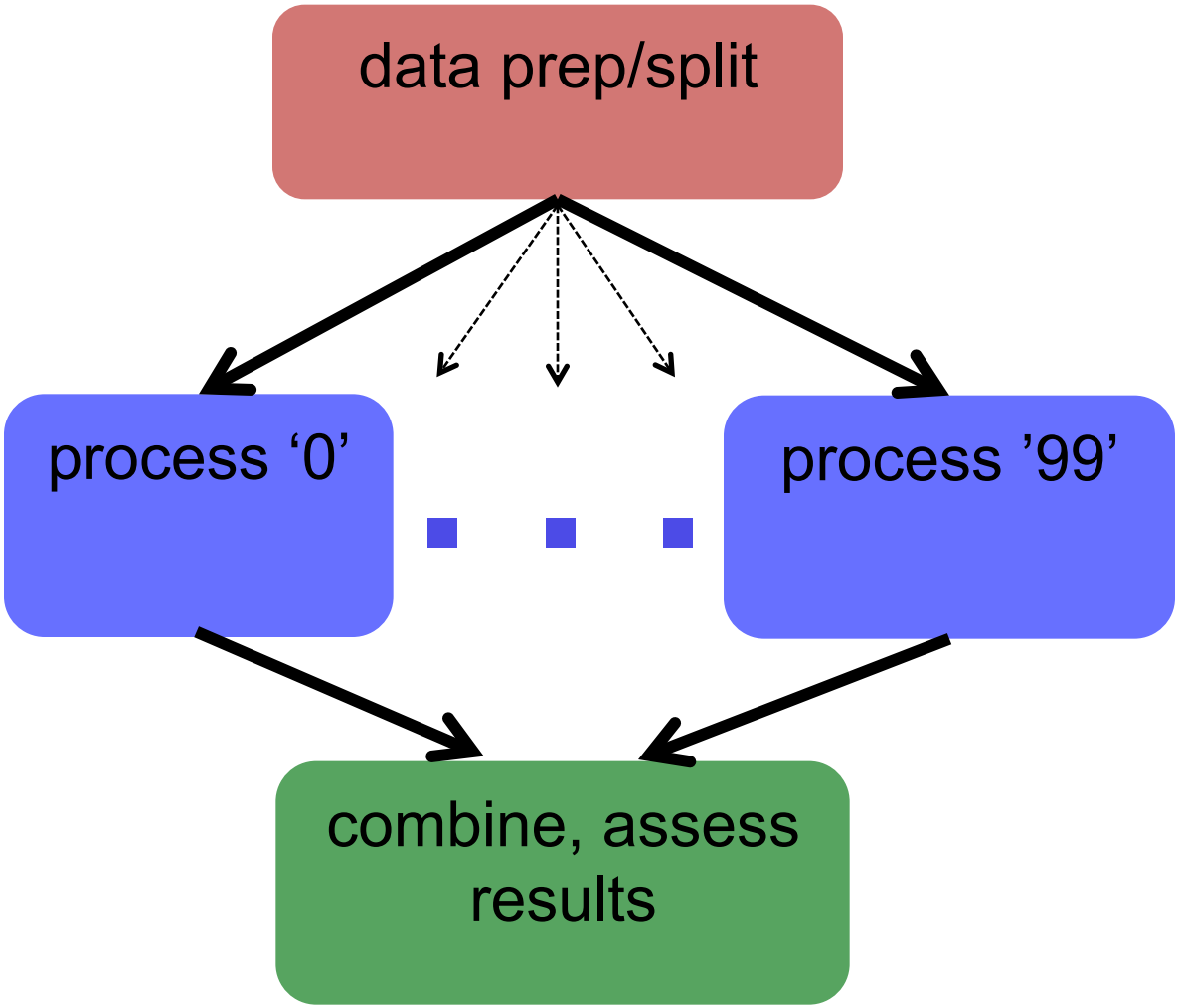
End Up with This



Optimizing a Workflow

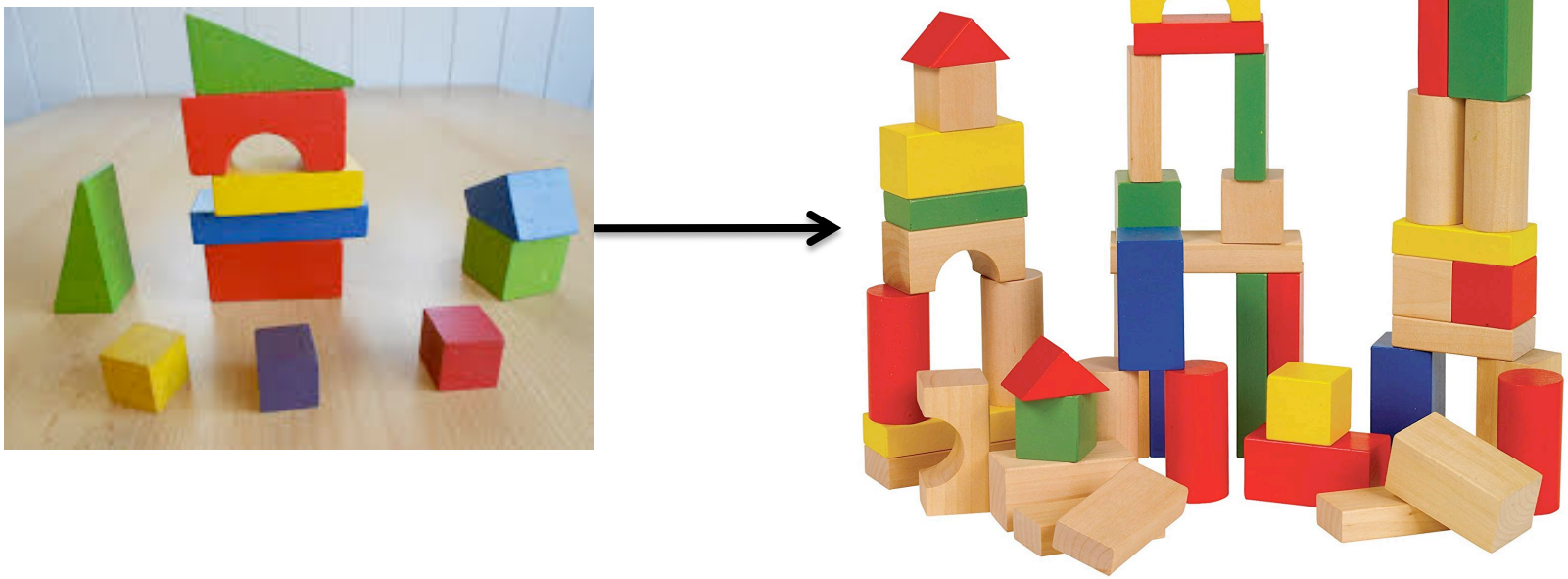
1. Draw out the workflow, identify pieces
2. Modular development: test and optimize each piece
   - divide or consolidate ‘pieces’
   - determine resource requirements
   - identify steps to be automated or checked
3. **Put the pieces together gradually**
4. Bonus features
   - Error proofing
   - Additional automation

DAGs Automate Workflows

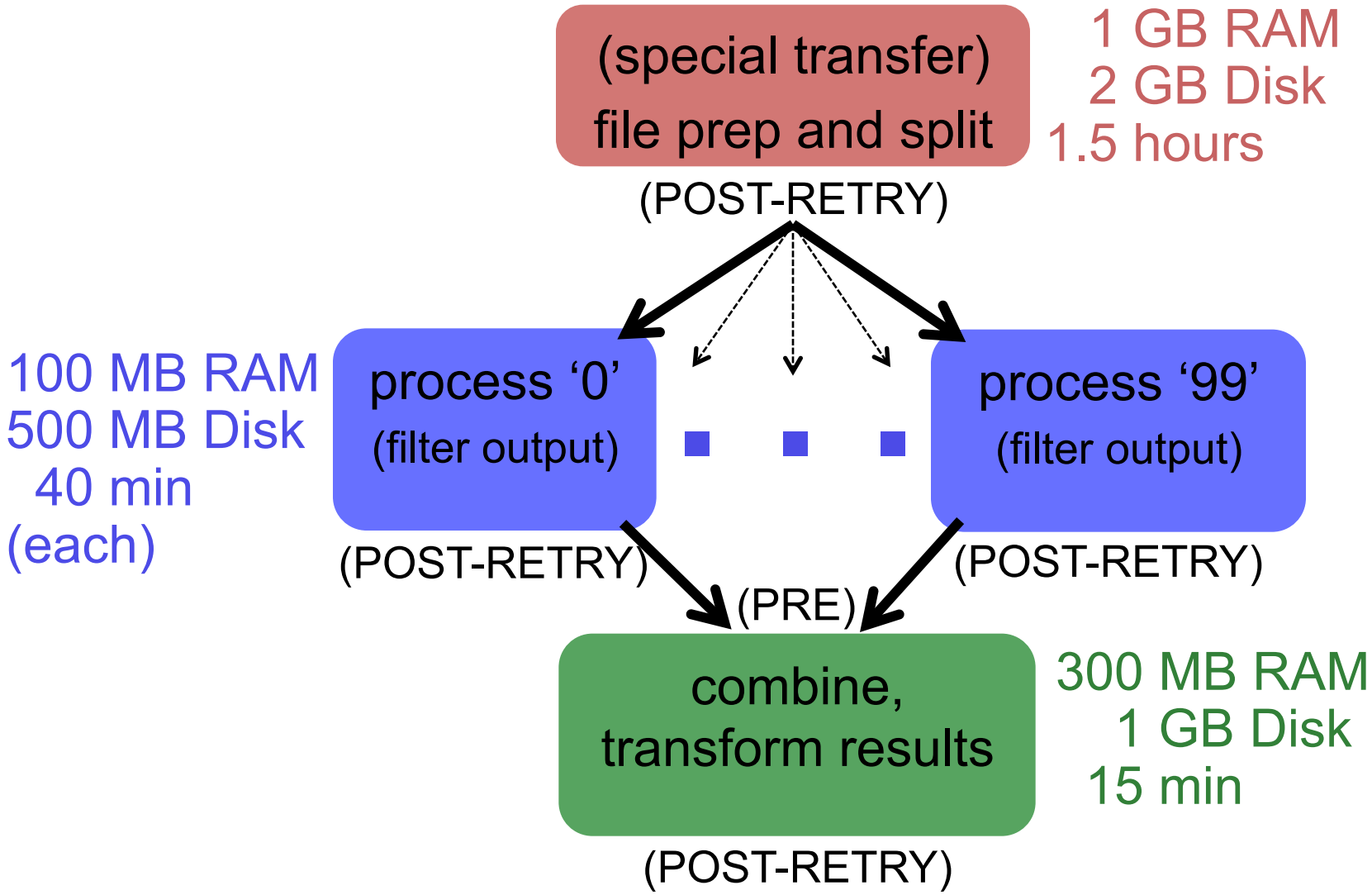


Scaling Workflows

- Same principles as before -- run a test DAG, with a small amount of data/jobs, before running the full thing.



In Full Detail



Solutions for Large Workflows

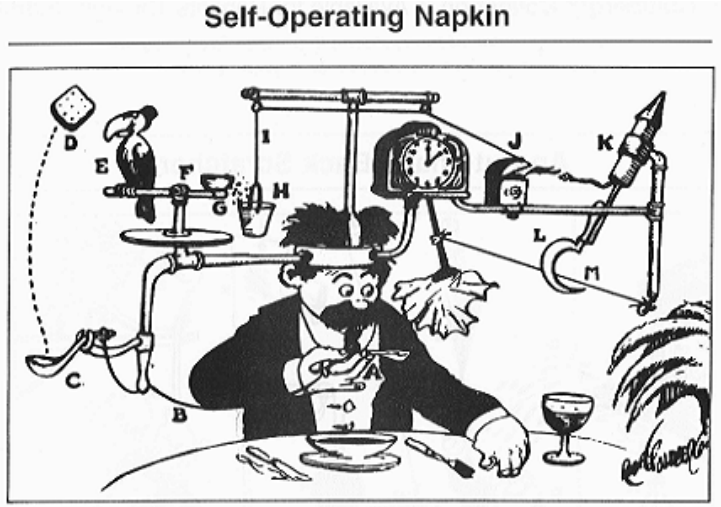
- Use a DAG to throttle the number of idle or queued jobs ("max-idle" and/or "DAGMAN CONFIG")
  - new HTCondor options to do this in a submit file as well
- Add more resiliency measures
  - "RETRY" (works per-submit file)
  - "SCRIPT POST" (use $RETURN, check output)
- Use SPLICE, VAR, and DIR for modularity/organization

Optimizing a Workflow

1. Draw out the workflow, identify pieces
2. Modular development: test and optimize each piece
   - divide or consolidate ‘pieces’
   - determine resource requirements
   - identify steps to be automated or checked
3. Put the pieces together gradually
4. **Bonus features**
   - Error proofing
   - Additional automation

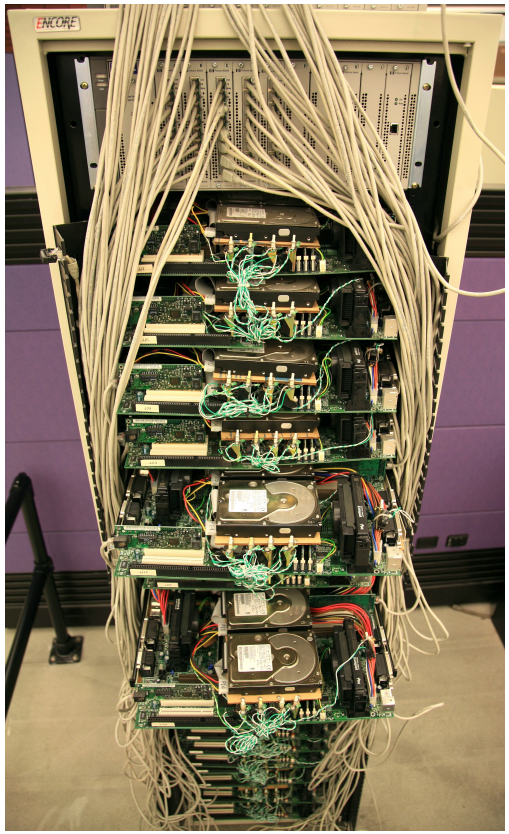
Robust Workflows

- Your DAG runs at scale! Now what?
  - Need to make it run *everywhere, everytime*
  - Need to make it run *unattended*
  - Need to make it run *when someone else tries*



Make It Run Everywhere

- What does an OSG machine have?
  - Prepare for very little
- Bring as much as possible with you, including:
  - executable
  - likely, more of the "environment"



Make It Work Everytime

- What could possibly go wrong?
  - Eviction
  - Non-existent dependencies
  - File corruption
  - Performance surprises
    - Network
    - Disk
    - …
  - *Maybe* even a bug in your code

Performance Surprises

One bad node can ruin your whole day

- **"Black Hole" machines**
  - Depending on the error, email OSG!
- ***REALLY* slow machines**
  - use periodic_hold / periodic_release

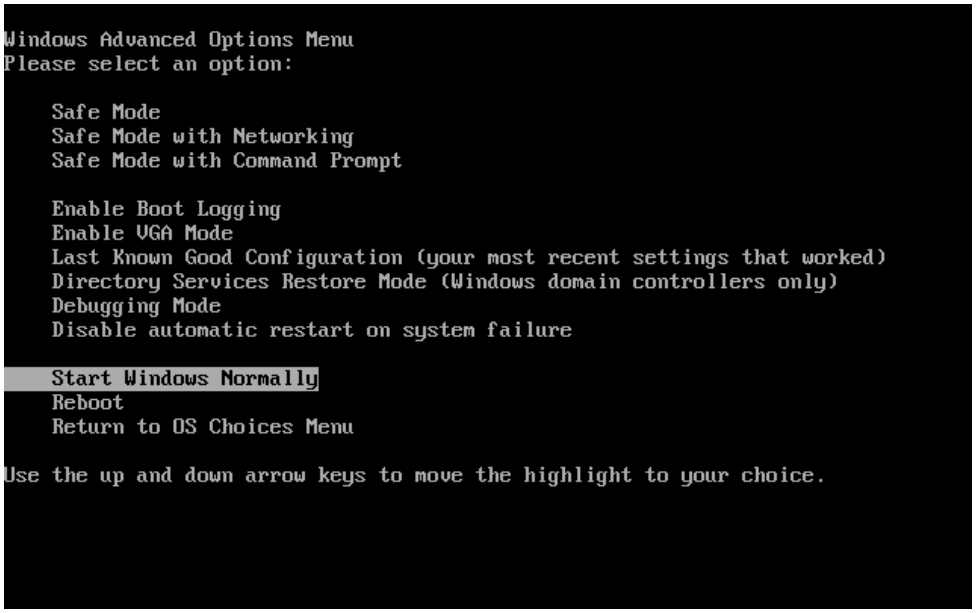
Error Checks Are Essential

If you don't check, it will happen…

- Check expected file existence, and repeat with a finite loop or number of retries
  - better yet, check *rough* file size too
- Advanced:
  - RETRY for *specific* error codes from wrapper
  - "periodic_release" for specific hold reasons

Handling Failures

- Understand something about failure
- Use DAG “RETRY”, when useful
- Let the rescue dag continue…



Make It Run(-able) for Someone Else

- Automation is a step towards making your research reproducible by someone else
  - Work hard to make this happen.
  - It's *their* throughput, too.
- Can benefit those who want to do similar work

Make It Work Unattended

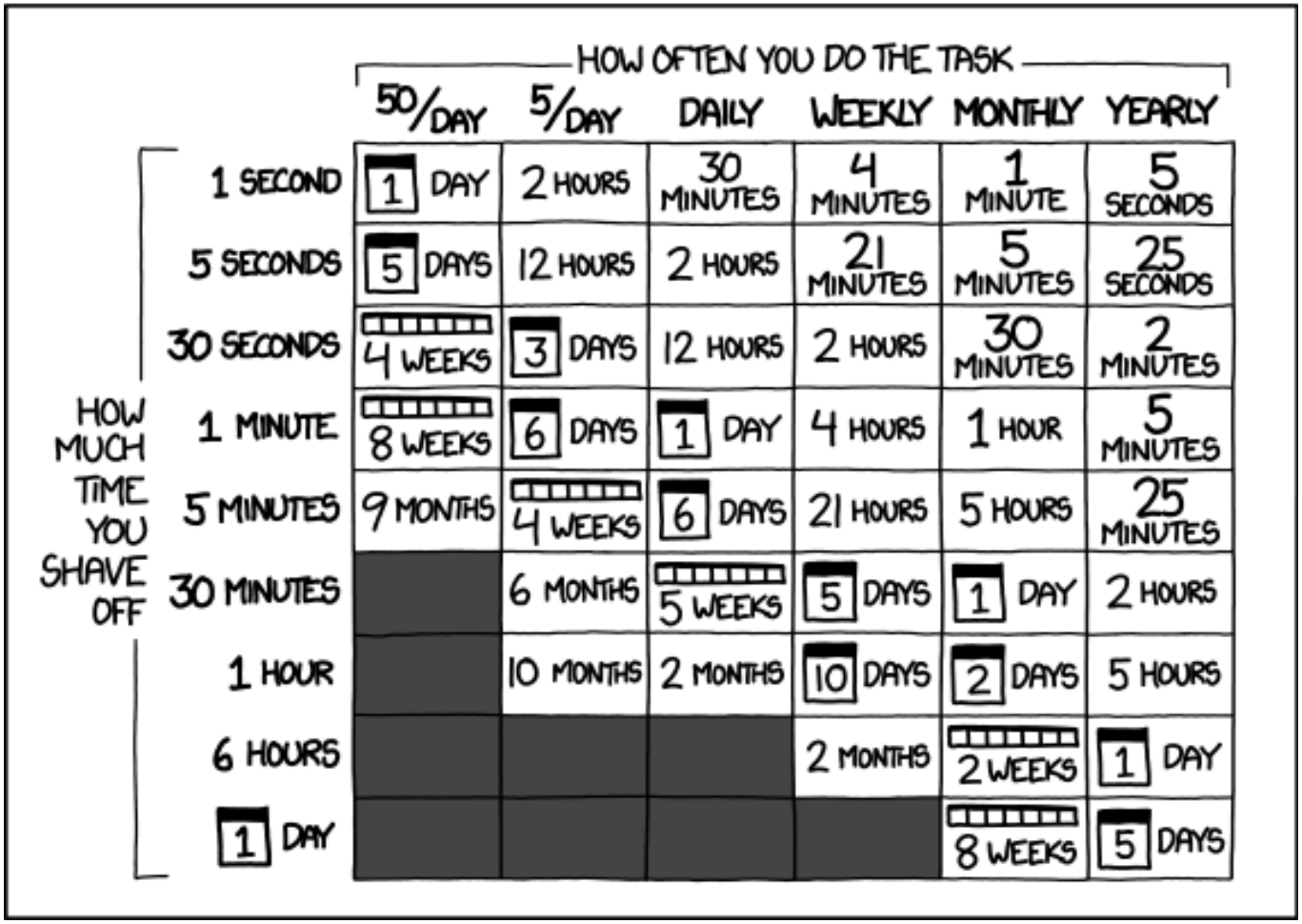
- Remember the ultimate goal:

  Automation! Time savings!

- Potential things to automate:
  - Data collection
  - Data preparation and staging
  - Submission (condor cron)
  - Analysis and verification
  - LaTeX and paper submission ☺

PARTING THOUGHTS

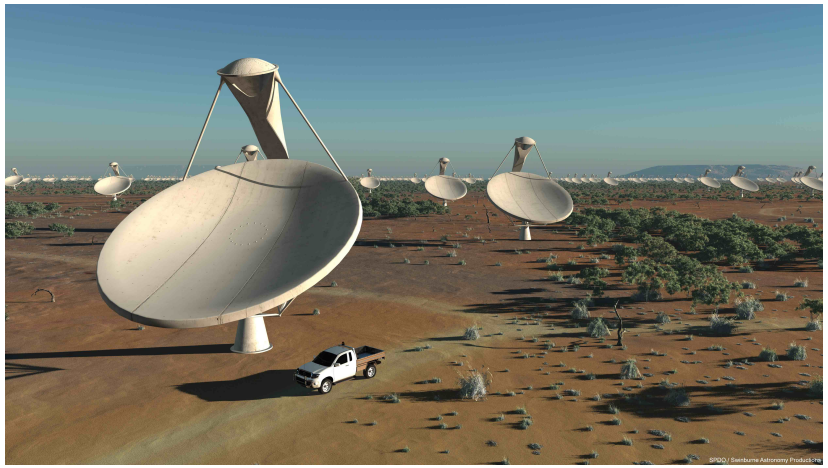
Automating workflows can save you time...



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!!



Open Science Grid

Getting Research Done

- End goal: getting the research done
- Hopefully you now have the tools to get the most out of:
  - **Computing**: which approach and set of resources suit your problem?
  - **High Throughput computing**: optimize throughput, use portable data and software
  - **Workflows**: build, test and scale

Questions?

- Now: Exercises 2.1 (2.2 Bonus)
- Next:
  - Lunch
  - Discovery Tour + Group Photo
  - HTC Showcase!