MORGRIDGE
INSTITUTE FOR RESEARCH

Moving your job's data

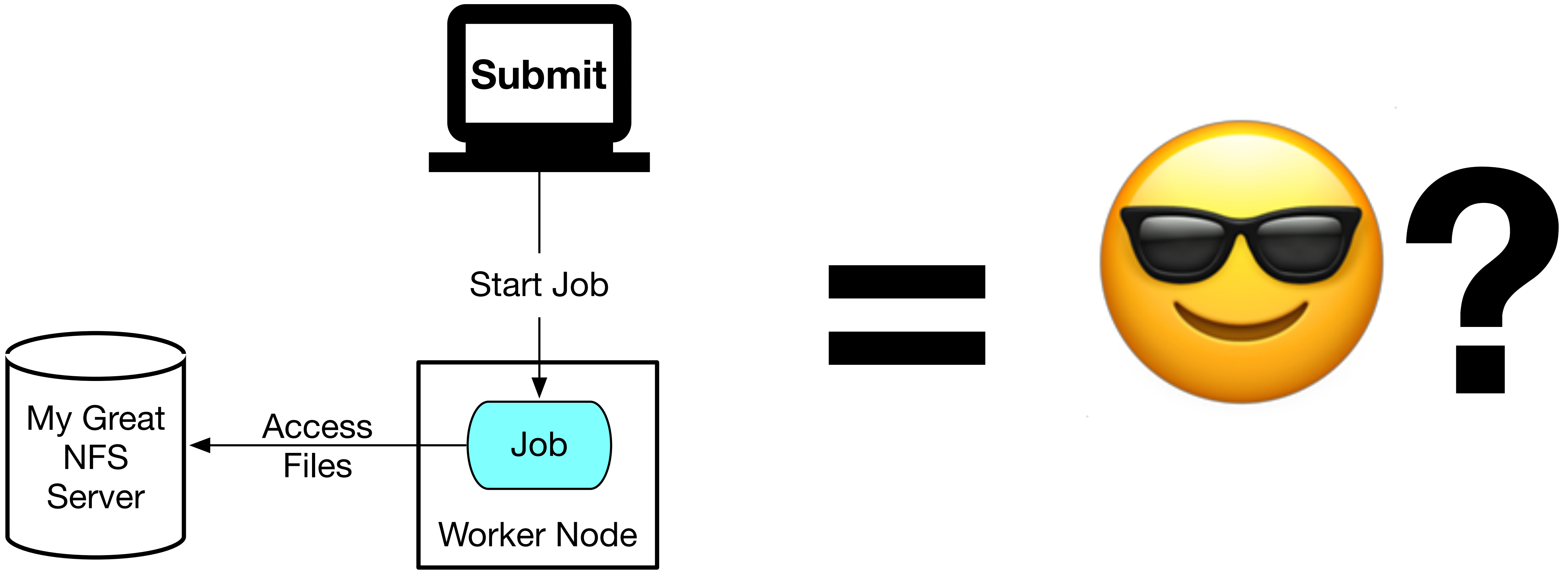
Or, more appropriately,
"moving your job**s**' data"

Brian Bockelman
HTCondor Week 2019

HTCondor

Why “move” my data?

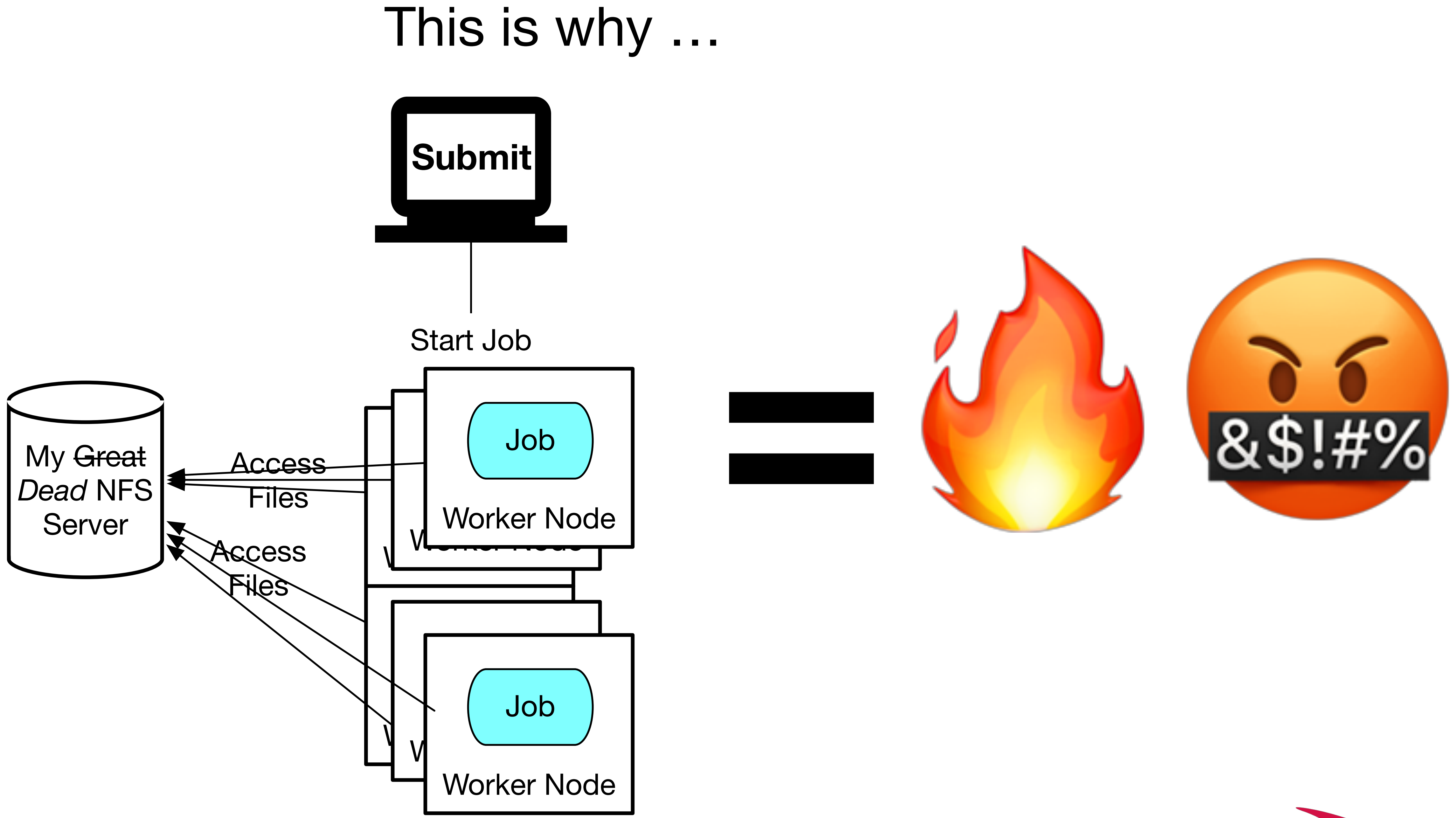
Why is this even a talk?



MORGRIDGE INSTITUTE FOR RESEARCH

HTCondor

Why “move” my data?



MORGRIDGE INSTITUTE FOR RESEARCH

HTCondor

It’s all about management…

- In order for HTCondor to deliver throughput, **it must manage resources.**
    - I/O resources are perhaps the most important ones to manage!
    - HTCondor can limit I/O activity based on concurrency level or I/O level.
- But first, you must tell us what you need…

Step 1: HTCondor File Transfer

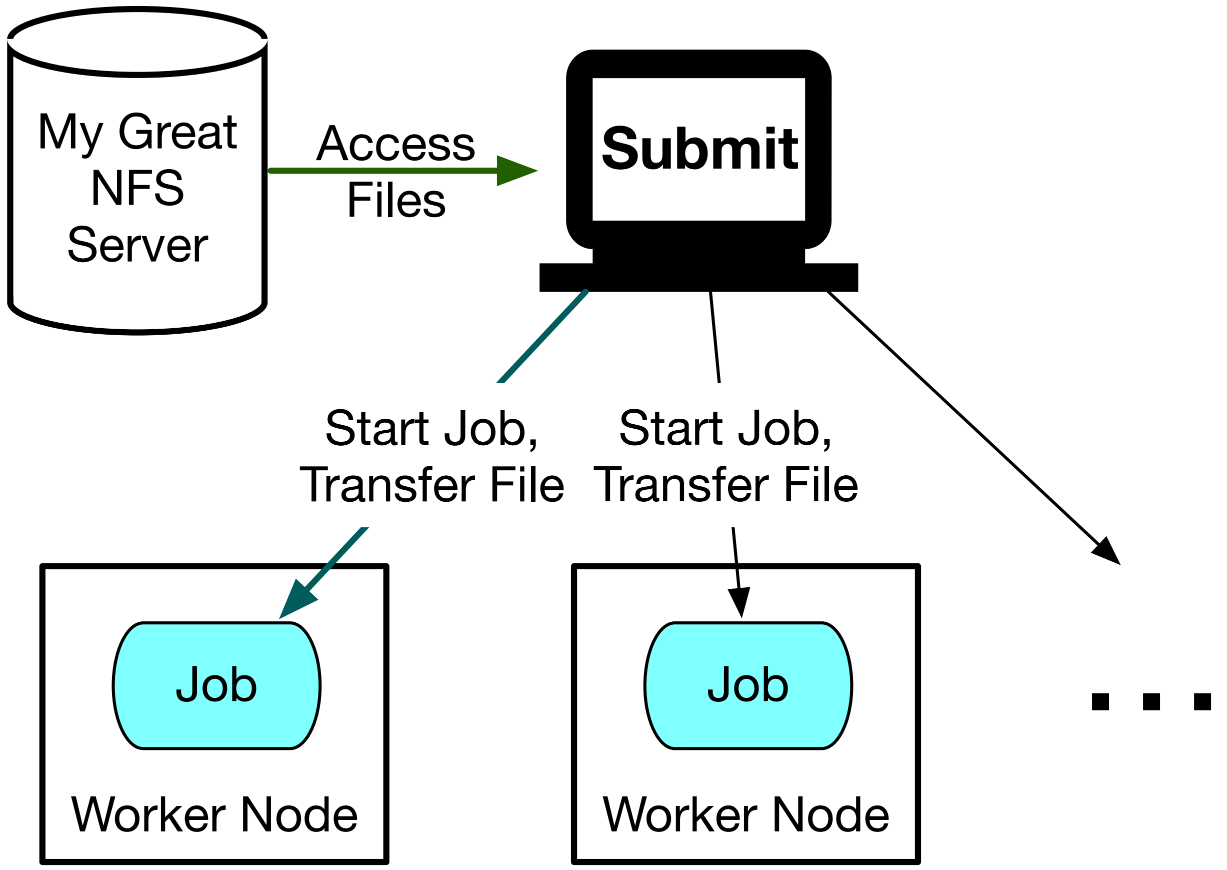
- By declaring your jobs' inputs and outputs to HTCondor, you:
  - Allow HTCondor to *manage* the movement of files.
  - Allow HTCondor to prepare your job environment. HTCondor *knows* to not even start your job if the input is unavailable.
  - Have the ability to make your job portable to other infrastructures.
- In the simplest - and most common - case, HTCondor will also perform the file transfer.

```
universe = vanilla
executable = science.exe
arguments = $(Process)
transfer_input_file = \
              input.txt
output = science.out
error = science.err
log = science.log
queue
```

MORGRIDGE INSTITUTE FOR RESEARCH

HTCondor

Step 1: HTCondor File Transfer



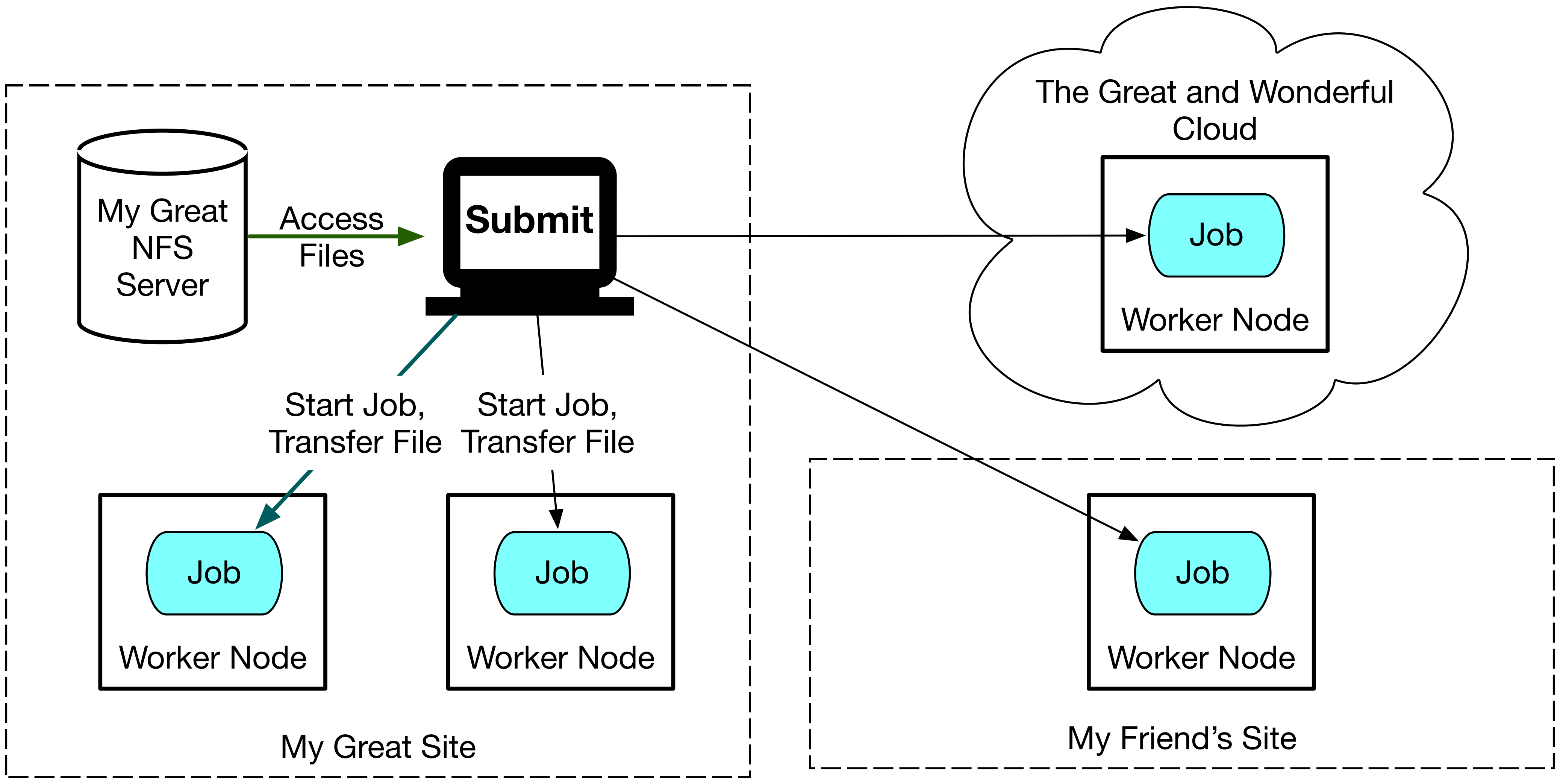
```
universe = vanilla
executable = science.exe
arguments = $(Process)
transfer_input_file = \
              input.txt
output = science.out
error = science.err
log = science.log
queue
```

MORGRIDGE INSTITUTE FOR RESEARCH

HTCondor

If HTCondor knows the job I/O requirements…

… it can take you places!



Ever tried exporting NFS offsite? 🤬

Arcane...

For users

- If you use the `-spool` option, HTCondor will make a copy of your input files to a private directory. This allows you to make changes locally while your jobs are running.
- The `stream_output` submit file command will cause HTCondor to stream output back to the submit host while the job is running. Useful - but use sparingly (consider `condor_tail` or `condor_ssh_to_job` as well).
- `max_transfer_output_mb` allows you to put a maximum cap on the data you transfer back; a useful sanity check if your job produced 100GB when you expected 100KB.
- `encrypt_input_files` allows you to force some files to be encrypted in flight - even if HTCondor would not otherwise do this.
- The `transfer_output_remaps` command allows you to provide arbitrary mappings from files in the job execute directory

For admins

- `MAX_CONCURRENT_UPLOADS` / `MAX_CONCURRENT_DOWNLOADS` provide an absolute limit on the number of files being transferred at a time
- `FILE_TRANSFER_DISK_LOAD_THROTTLE` will further lower the number of concurrent file transfers based on the I/O load measured on the submit host's storage.
- `MAX_TRANSFER_OUTPUT_MB` sets the schedd-wide default for maximum data transfers per jobs (users can override).
- `MAX_TRANSFER_QUEUE_AGE` is the maximum time, in seconds, that a transfer is allowed to proceed before it is killed.

MORGRIDGE INSTITUTE FOR RESEARCH

HTCondor

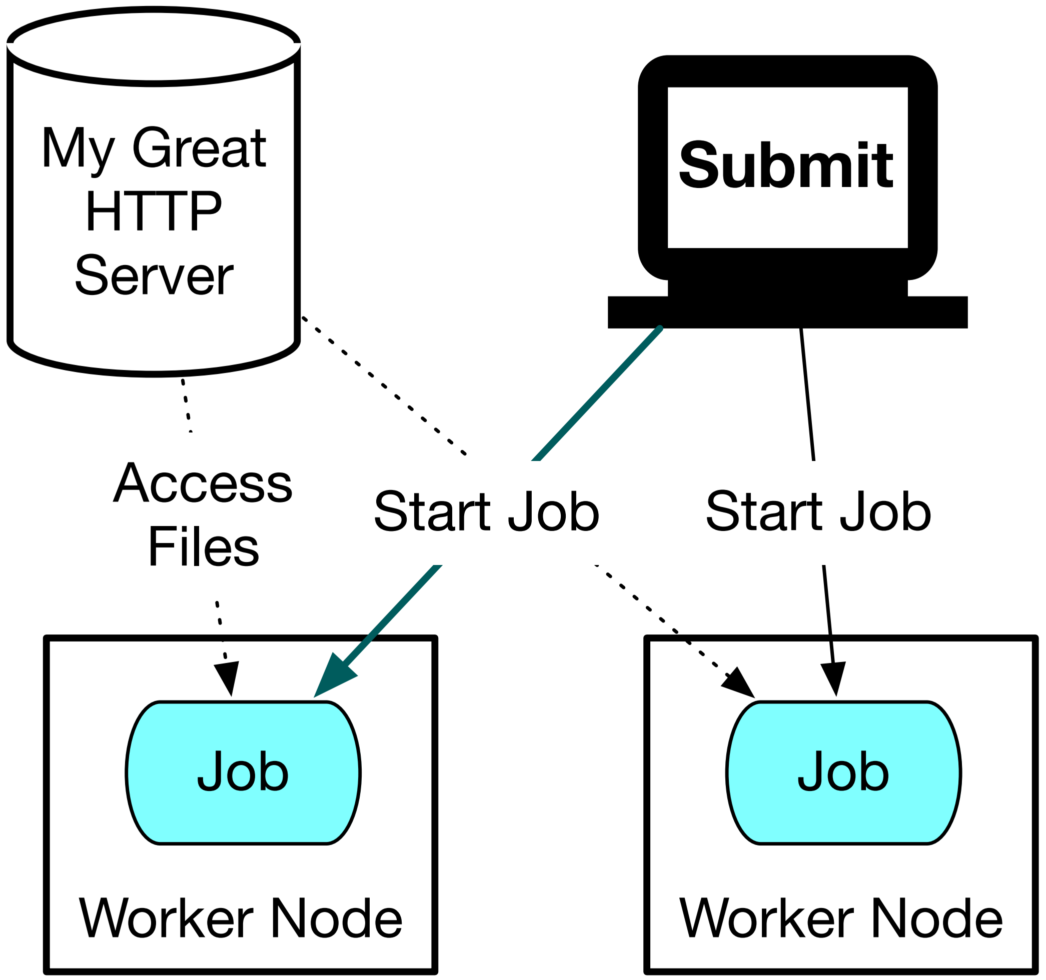
URL-based Transfer

- A typical first response for new sysadmins when they see HTCondor file transfer is “gross, how do you scale this?”
  - The answer is often simple: a 25Gbps connection and a SSD is far cheaper than your time configuring new things!
- However, there are legitimate use cases when “upgrade your hardware” is not an option!
  - User jobs need to access data not on the submit host.
  - Take advantage of specific features of other protocols (e.g., HTTP caching)
  - Enormous scale - single workflows need more than 25Gbps sustained.

What now? URL file transfers!

URL-based Transfer

- The **`transfer_input_files`** command *also* accepts URLs.
    - If given a URL, HTCondor will attempt to download the corresponding resource prior to starting the job.
    - Allows one to pull in files from somewhere besides the submit host.
    - If it's unable to download all the URL inputs, it won't attempt to start the job.
- Anything that the venerable curl library can handle (HTTPS, FTP) is usable in HTCondor by default.
    - Is your favorite protocol not supported by curl?
    - **Great news**: additional protocols can be added by writing a plugin to HTCondor. Can be as simple as a bash script.

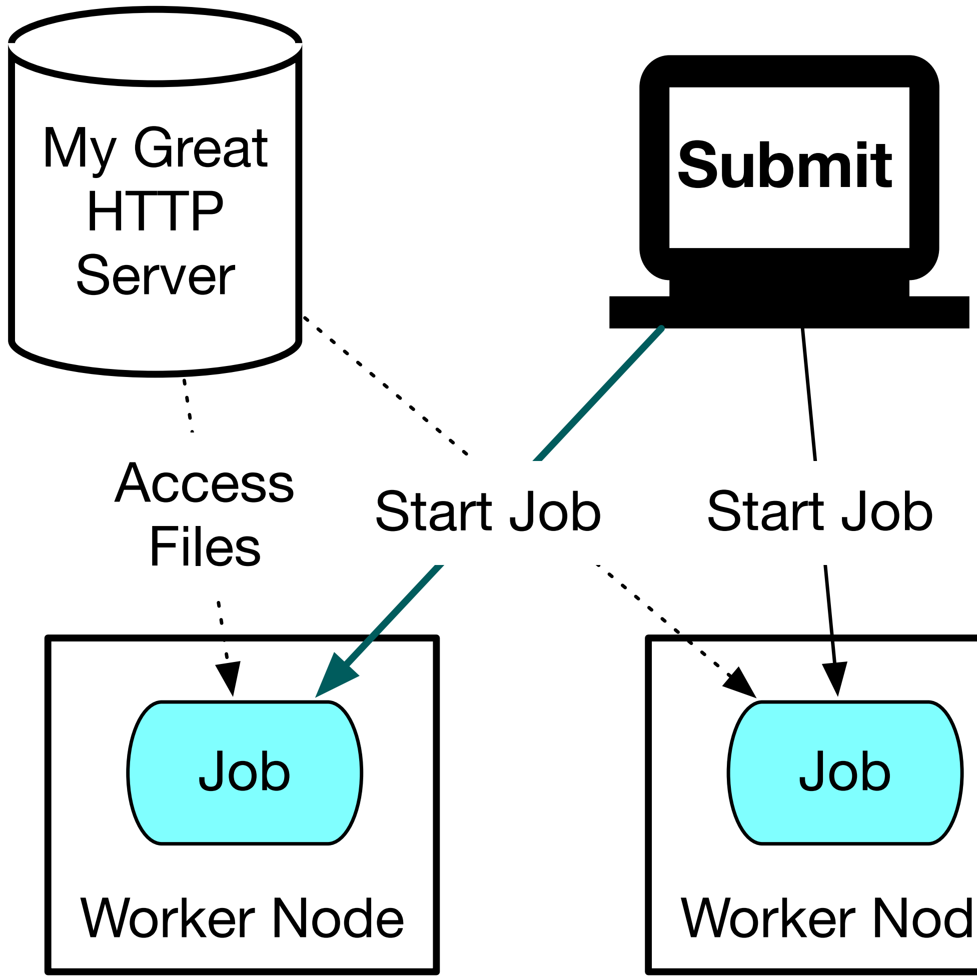


I can do that!

- Wait, why not call **`curl`** inside my job? I can do that!
  - **Miron has a lot of questions**: Are you sure you call **`curl`** correctly? Did you pass the right headers to make caching work? Did you discover the right proxy? Did you set timeouts appropriately? Did you fine-tune your retry policy?
  - When the transfer fails, is this reflected correctly in the job status?
  - If HTCondor doesn't know about it, HTCondor can't schedule it!
- Same as with normal file transfers, HTCondor can do the hard work and (difficult) management if it is told what URLs are needed.

URL-based Transfer

```
universe = vanilla
executable = science.exe
arguments = $(Process)
transfer_input_file = \
 https://example.co/input
output = science.out
error = science.err
log = science.log
queue
```



Off into the (near) future!

- A significant number of improvements have been done for URL-based transfers in the last year:
  - **Error messages *greatly* improved**: URL-based transfers can now provide sane, human-readable error messages when they fail (instead of just an exit code). Available in 8.8 series.
  - URLs for output: Individual **output files can be URLs**, allowing stdout to be sent to the submit host and large output data sent elsewhere. Available in 8.9.1.
  - If you use HTCondor to manage credentials, these **tokens can be used directly** by URL-based transfers. Available in 8.9.1.
  - File transfers are now sorted by the submit host and URLs are transferred last. This means you can ensure some inputs (such as your S3 credentials!) are at the worker node before URL transfers are invoked. Available in 8.9.1.
  - Have an interesting use case? Jobs can now supply their own file transfer plugins — great for development! Available in 8.9.2.

Off Into the (Farther) Future

- **Checksum / manifest files**: We will be checksumming files on-the-fly (catching integrity error) and providing a “manifest” - a record of what the correct file contents were when they were read.
    - *Allows data reuse!* If we know the correct contents, then we can decide whether a file needs to be re-transferred between subsequent jobs.
- **<Your favorite protocol here>**: What types of URLs would you like to see in HTCondor? S3? Google Drive? Dropbox? Xrootd?
    - Strongest input so far this week has been S3 support.

MORGRIDGE INSTITUTE FOR RESEARCH

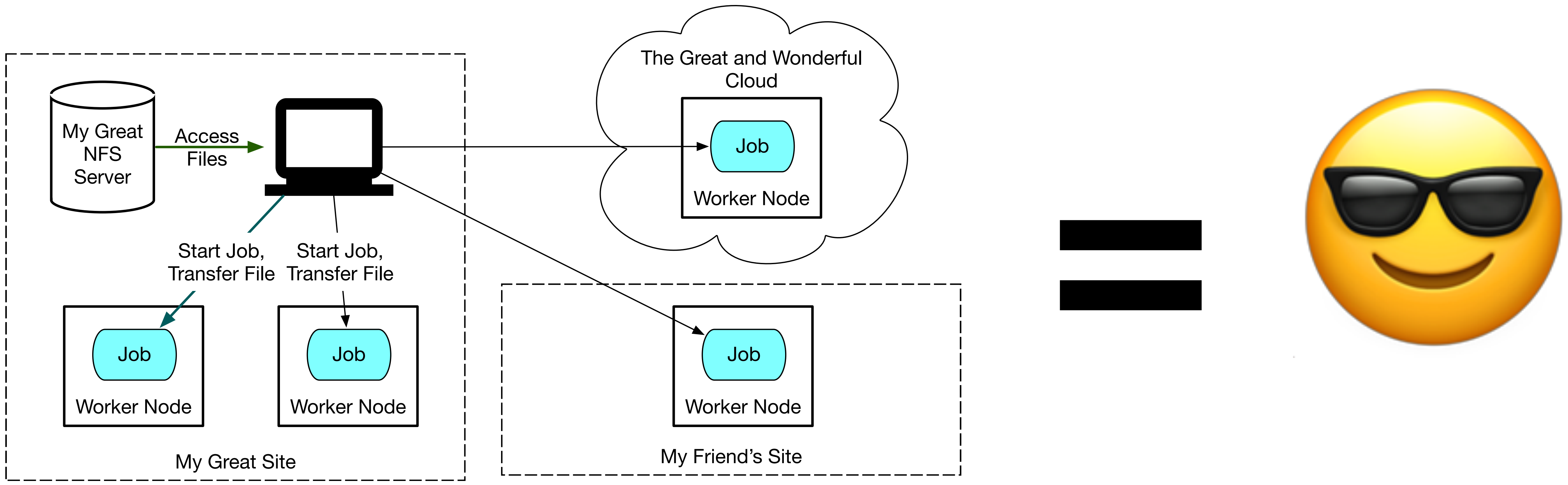
HTCondor

Know Your Limits

- When to *not* use HTCondor file transfer? HTCondor file transfer is based on moving input to the worker node prior to job startup. This may not be appropriate for your job:
  - It may be computationally expensive to determine what inputs are needed.
  - The job may read a small portion (kilobytes) of a very large file (terabytes) that is prohibitively large to move completely.
- There's ways to transform both of the above to HTCondor-file-transfer-friendly workflows, but it takes work!
- The best thing to do is to understand all the options and trade-offs, not do a one-size-fits-all solution.

Questions?

Have an interesting use case? We'd love to hear more!



My Great NFS Server
Access Files
Start Job, Transfer File
Start Job, Transfer File
Job
Worker Node
Job
Worker Node
My Great Site
The Great and Wonderful Cloud
Job
Worker Node
Job
Worker Node
My Friend's Site
=