Job Wrapper scripts: Problems and Alternatives

Greg Thain
Center for High Throughput Computing
University of Wisconsin - Madison

Reliably running jobs (even on unreliable places)

Greg Thain
Center for High Throughput Computing
University of Wisconsin - Madison

Outline

A Very Short talk about "reliability"

Review of how HTCondor runs jobs

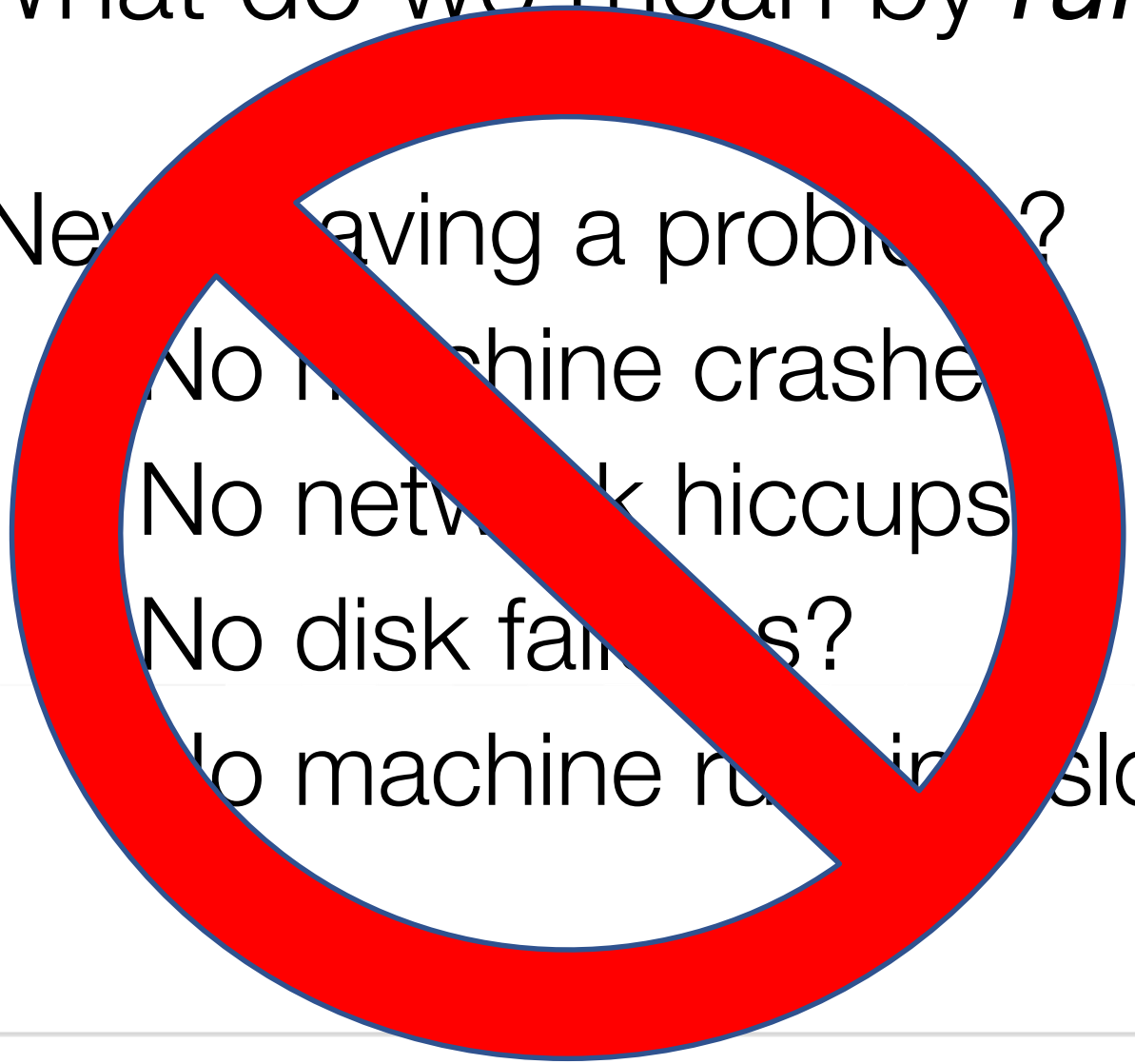
Problems with wrappers under HTCondor

Discuss solutions (some new, some old)

With the occasional simplification indicated by*

Reliabillity

What do we mean by *running jobs reliably*



Ne aving a probl ?

- No m chine crashe
- No netw k hiccups
- No disk fail s?
- lo machine ru in slower than it should?

Why do we accept the existence of failures?

A) We have to

B) We get more machines to use.

HTC should be happy to run on imperfect machines

e.g. gamer gpus

For us, *reliability* means

- We can ***detect*** errors
- We can ***classify*** errors
- We can ***report*** errors
- We can ***respond to*** errors
  - This one is beyond our scope of this talk

Review how HTCondor runs jobs

condor_starter runs the job on the EP*

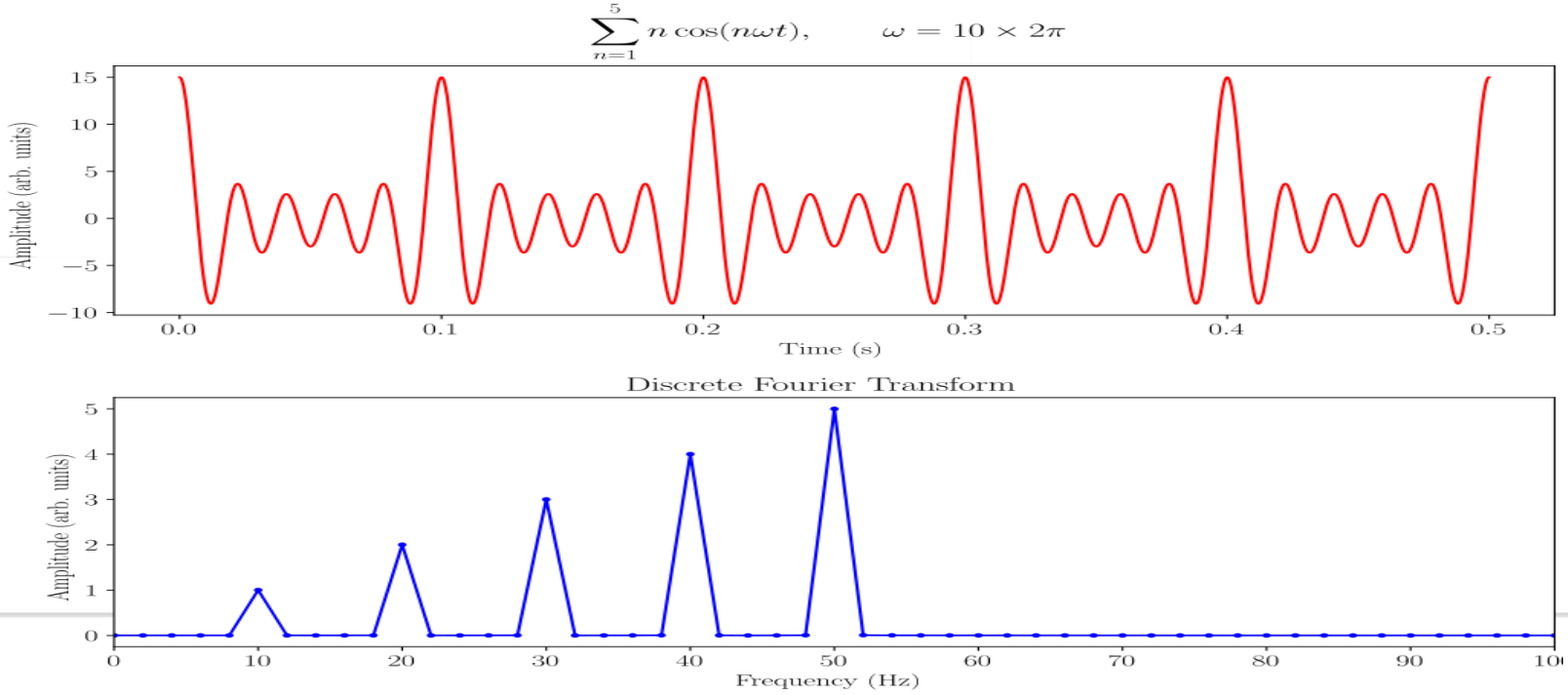
Makes a scratch directory, e.g. /var/lib/execute/dir_1234

Starts the program named in the executable = line in the submit file

How you think of your program…

Makes a scratch directory, e.g. /var/lib/execute/dir_1234

Starts the program named in the executable = line in the submit file



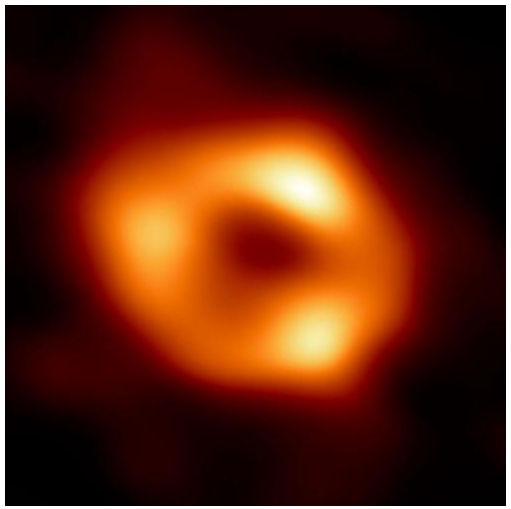
How you think of your program…

Makes a scratch directory, e.g. /var/lib/execute/dir_1234

Starts the program named in the executable = line in the submit file

How you think of your program…

Makes a scratch directory, e.g. /var/lib/execute/dir_1234



Starts the program named in the executable = line in the submit file

How HTCondor sees your program*…

Makes a scratch directory, e.g. /var/lib/execute/dir_1234

Starts the program named in the executable = line in the submit file

Just an opaque box

But not quite perfectly opaque..

HTCondor knows…

- Memory usage inside box
- CPU/GPU usage inside box
- Disk usage inside box
- Wall clock time

and….

- Exit code of job's main process (!)

(and some other stuff)

And sends this back to the AP!

Let's talk about exit codes…

- Unix exit code is eight bits a program returns to parent at exit
- ONLY WAY the program can communicate to the starter*
- By convention "zero" (0) means "good", all else "bad"

- But that's just a convention
- TRIVIA: What's the exit code for the "grep" program?

HTCondor does what about exit codes?

- Pure HTCondor doesn't do anything (just records them)
  - Should it – hold job with non-zero exit? Remove? Send email?
- DAGMan has more options – can resubmit
  - By default, dagman assumes non-zero exit is failure, and blocks DAG

- But HTCondor gives you knobs™

```
max_retries = 7
success_exit_code = 0
```

Wrapper scripts…

Ok, so what's all this about shell scripts?

- Exit code of a script is either
  - Argument to `exit` shell builtin function

OR

  - Exit code of last command the script ran

Typical shell wrapper for a job looks like

```
#!/bin/sh
some_setup
some_more_setup

the_actual_executable

some_cleanup
some_more_cleanup
```

Pop Quiz: what's the exit code?

```
#!/bin/sh
some_setup
some_more_setup

the_actual_executable

some_cleanup
some_more_cleanup
```

Pop Quiz: how can we fix this?

```
#!/bin/sh
some_setup
some_more_setup

the_actual_executable

some_cleanup
some_more_cleanup
```

Doesn't seem to hard to fix at first…

```
#!/bin/sh
some_setup
some_more_setup

the_actual_executable
saved_exit = $?
some_cleanup
some_more_cleanup
exit $saved_exit
```

But to fix everything is tedious, and error-prone

```
#!/bin/sh
some_setup
some_more_setup

the_actual_executable
saved_exit = $?
some_cleanup
some_more_cleanup
exit $saved_exit
```

What if there is an error here?

Is this *kind* of error the same as a job error? Do we want to respond the same way?

What's the bigger picture?

```
#!/bin/sh
some_setup
some_more_setup

the_actual_executable

some_cleanup
some_more_cleanup
```

Setup the environment

Cleanup the environment

Remember how HTCondor sees the job?

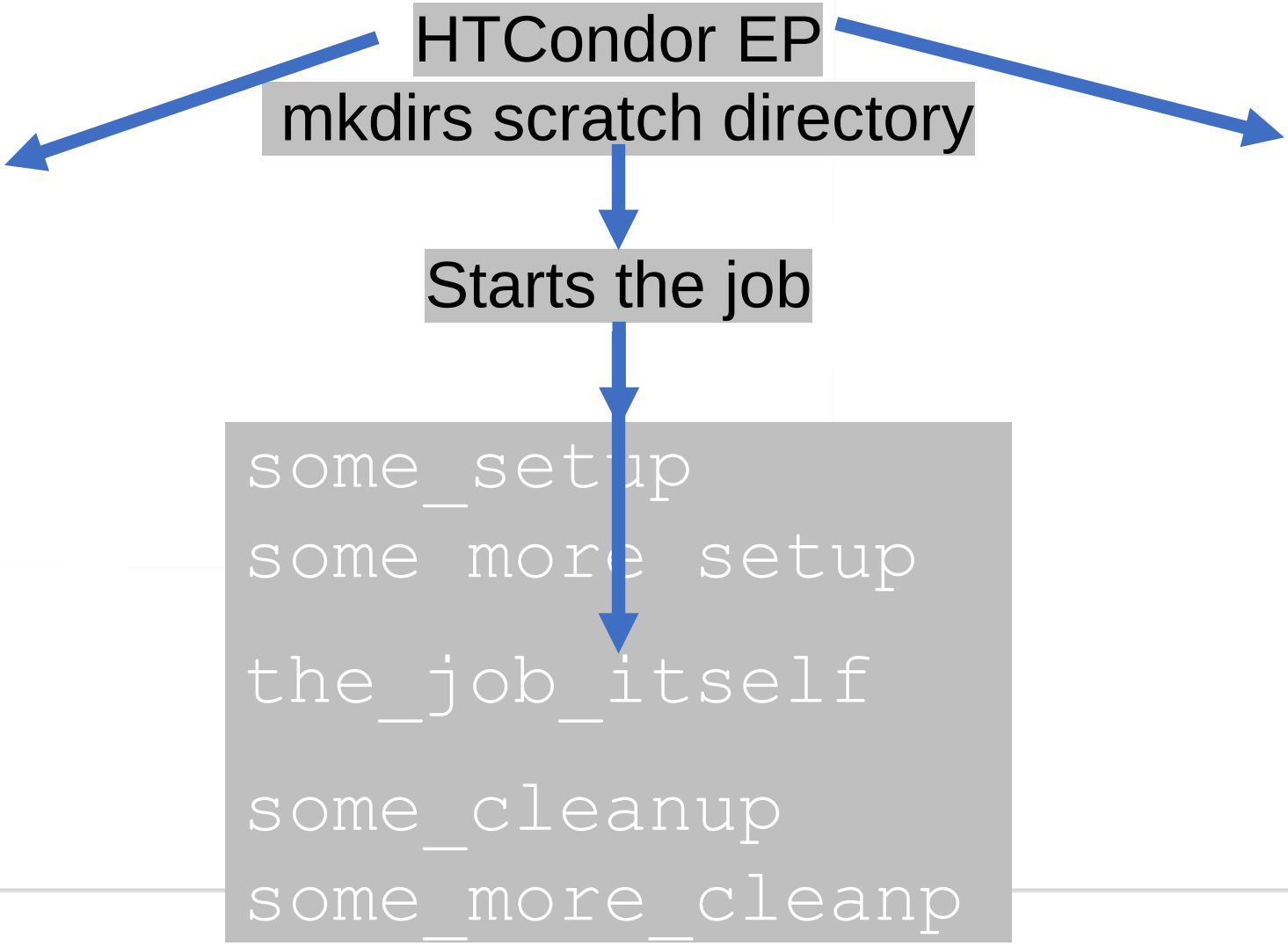
Fundamental Problem:
HTCondor can't differentiate a setup/cleanup problem
From a bona-fide job problem
(and we want to treat these differently)

That is to say…

- Wrappers *hide* activity from HTCondor
- Error codes are NOT sufficient!
  - And error codes belong to namespace
    - of the job – domain of the job
  - No Unix error for "failed to xfer sandbox"
  - Some Belong to the HTCondor domain

How to fix, and run more reliably

The proper fix



This is a common CS pattern

- Separating initialization / teardown from main work
  - Object Oriented Constructors/Destructors do this
  - Two Phase commit "Prepare Tran"
- And HTCondor knows all about errors in parts it manages
- So it can send them back home to the AP to make decisions

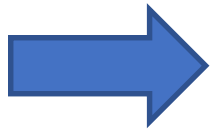
The means translating shell into submit

- Some work, worth it, not too hard.
- Read submit man page for all possibilities
- Some examples follow

Wrapper Env var

```
#!/bin/sh

export MYVAR=hello
..
```



Submit language

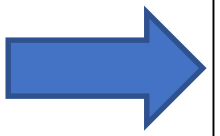
```
universe = vanilla


environment=MYVAR=hello
…
queue
```

Wrapper untar

```
#!/bin/sh


tar xzf a.tgz
..
rm -fr a/
```



Submit language

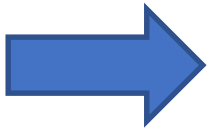
```
universe = vanilla


transfer_input_files = a/
…
queue
```

Wrapper wget

```
#!/bin/sh

wget http://..
..
rm -fr a/
```



Submit language

```
universe = vanilla

transfer_input_files = \
   http://...
…
queue
```

We asked local facilitators "why wrappers?"

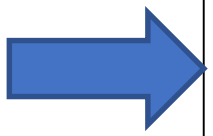
Four very common reasons

1. wget'ing file from 3rd party server
    1. Or putting output files there
2. Setting an environment variable to point to scratch dir
3. Full description of inputs and outputs before and after job run
4. wget + untar

Wrapper wget

```
#!/bin/sh

wget http://...
..
```



Submit language

```
universe = vanilla


Transfer_input_files = \
    http://example.com/foo/bar

queue
```

Wrapper ls -R

```
#!/bin/sh

/bin/ls -CFR

my_real_job

/bin/ls -CFR
```

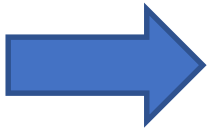
Submit language

```
universe = vanilla
manifest = true
manifest_dir = some_directory
…
queue
```

Wrapper wget + untar

```
#!/bin/sh

wget http://..
tar xzf a.tar
..
rm -fr a/
```



Submit language

```
universe = vanilla

transfer_inp                - \
    unta                .../foo.tar.gz
…
```

Doesn't exist (yet)

Quiz time:

- Which is faster –

HTCondor explicit file transfer ?

Or

Shared Filesystem?

What an expert will answer:

"It depends"

But what is a deeper answer?

- What's the error domain of a shared file i/o error
- Can the job report an shared file i/o error back to HTCondor?
  - NO!
- Reliability can give better time-to-finish than something higher pref

Summary

- The more we tell HTCondor to do, the better outcomes
- A bit of work to translate familiar shell to submit, but worth it
  - But not all or nothing
- What are we missing that you still need these kinds of wrappers?

Thank you and questions

Thank you – Questions?

This work is supported by the NSF under Cooperative Agreement OAC-2030508. Any options, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.