Once more, with feeling! A monitoring feedback loop for HTC jobs with unknown requirements

Ben Tovar
University of Notre Dame

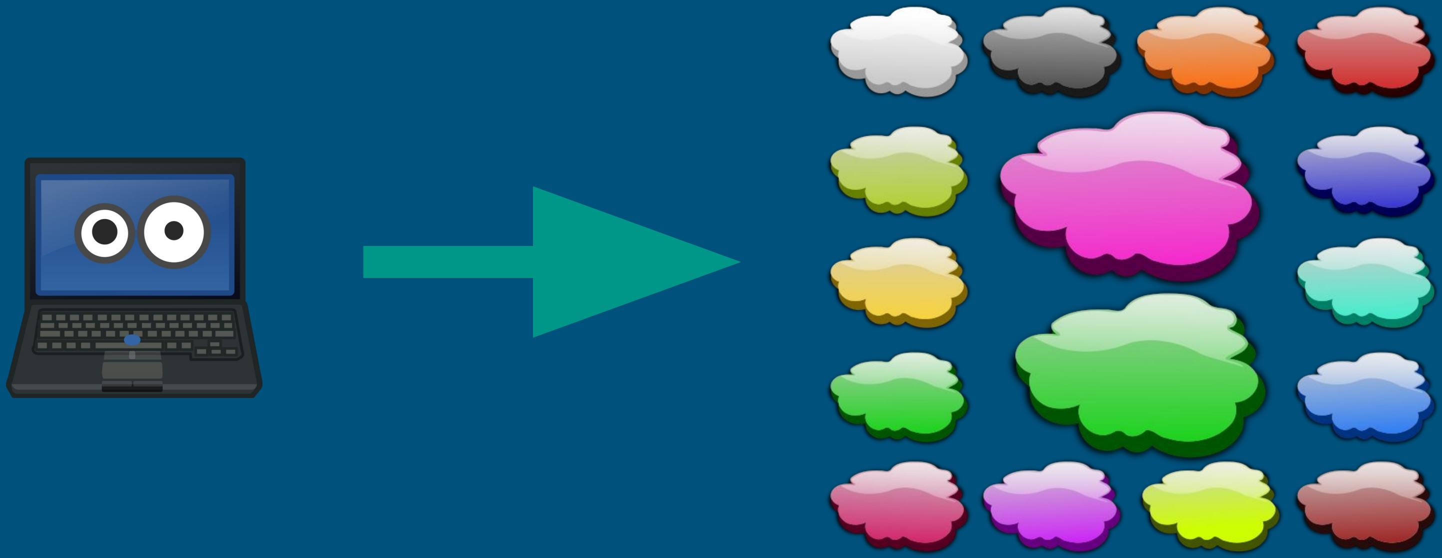
btovar@nd.edu

CCTools

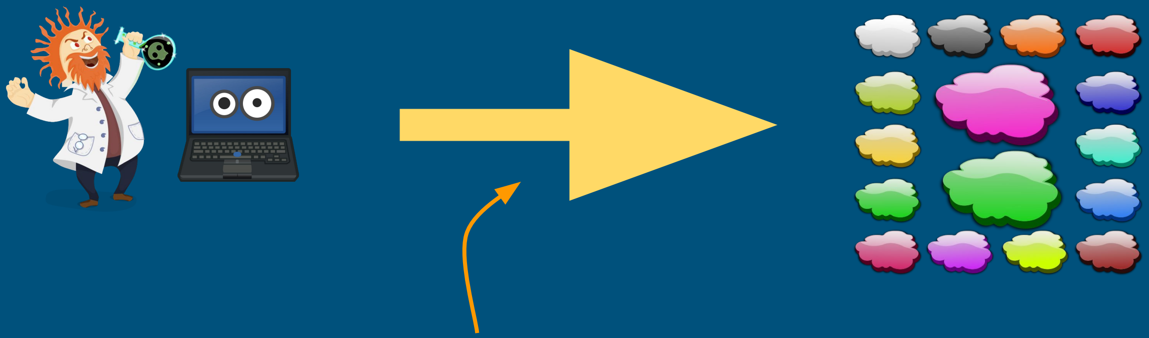
UNIVERSITY OF
NOTRE DAME

Where we are

Scientist says:
"This demo task runs on my laptop, but I need much more for the real application. It would be great if we can run O(25K) tasks like this on this cloud/grid/cluster I have heard so much about."



Who we are

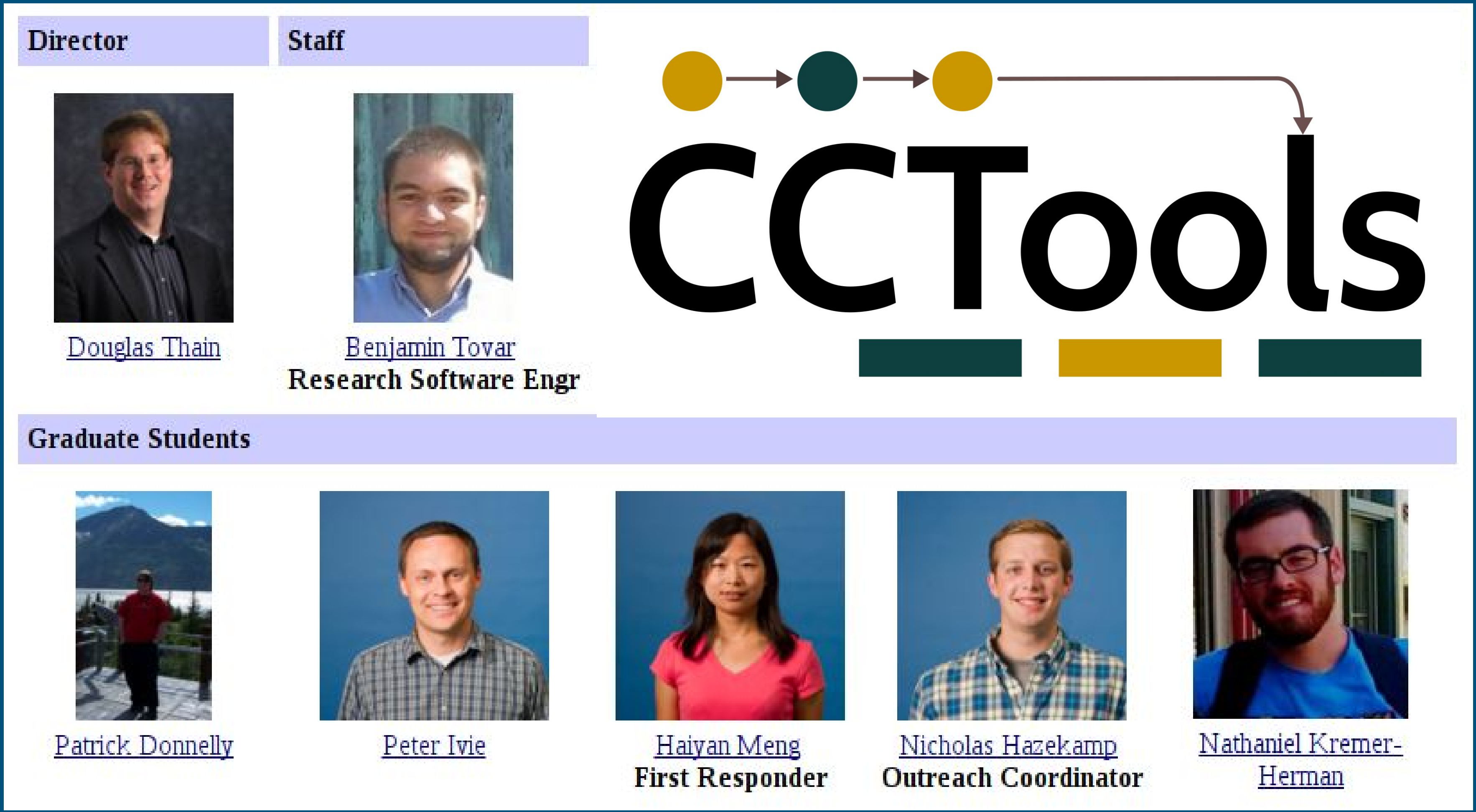


The Cooperative Computing Lab

Computer Science and Engineering

University of Notre Dame

Cooperative Computing Lab



Director

Douglas Thain

Staff

Benjamin Tovar
Research Software Engr

CCTools

Graduate Students

Patrick Donnelly

Peter Ivie

Haiyan Meng
First Responder

Nicholas Hazekamp
Outreach Coordinator

Nathaniel Kremer-Herman

Not shown, grad students: Tim Shaffer , Chao Zheng

CCL Objectives

- Harness all the resources that are available: desktops, clusters, clouds, and grids.
- Make it easy to scale up from one desktop to national scale infrastructure.
- Provide familiar interfaces that make it easy to connect existing apps together.
- Allow portability across operating systems, storage systems, middleware…
- Make simple things easy, and complex things possible.
- No special privileges required.

CCTools

- Open source, GNU General Public License.
- Compiles in 1-2 minutes, installs in $HOME.
- Runs on Linux, Solaris, MacOS, Cygwin, FreeBSD, …
- Interoperates with many distributed computing systems.
  - Condor, SGE, Torque, Globus, iRODS, Hadoop…
- Components:
  - Makeflow – A portable workflow manager.
  - Work Queue – A lightweight distributed execution system.
  - All-Pairs / Wavefront / SAND – Specialized execution engines.
  - Parrot – A personal user-level virtual file system.
  - Chirp – A user-level distributed filesystem.

Long-tail of science

Individual researchers and small laboratories that:

Need to curate, manage, and analyse large amounts of data.

May not know how to access computational resources available to them.

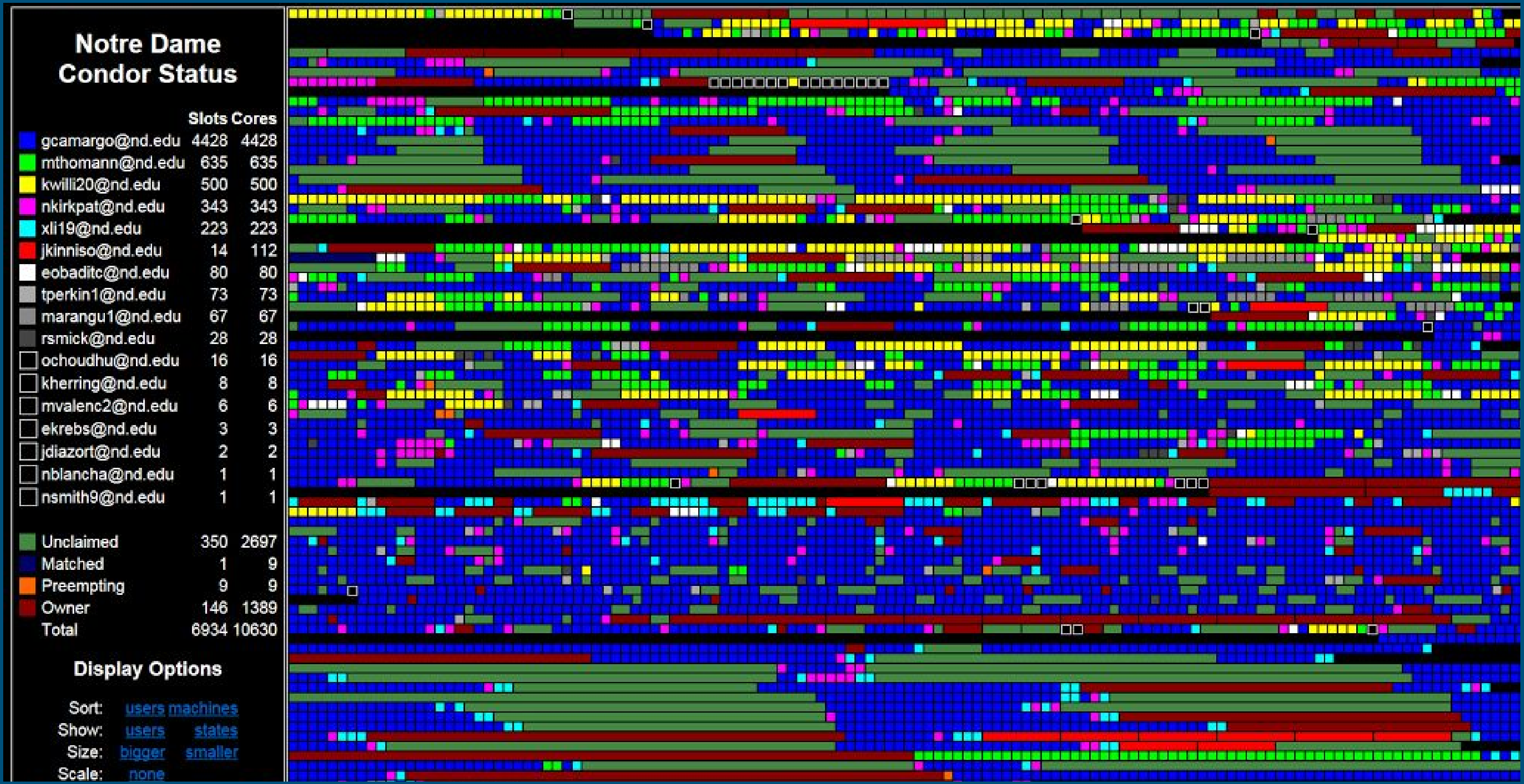
May not have immediate access to the required resources.

(i.e., they know their discipline, but they do not have an HTC expert in their team)

Notre Dame's happy opportunistic situation

- ~25k cores at Notre Dame's Center for Research Computing (CRC)
- They belong to different individual PIs, but they are available through **condor** when not used by their owners.

condor.cse.nd.edu



Notre Dame Condor Status

		Slots	Cores
🟦	gcamargo@nd.edu	4428	4428
🟩	mthomann@nd.edu	635	635
🟨	kwilli20@nd.edu	500	500
🟪	nkirkpat@nd.edu	343	343
🟦	xli19@nd.edu	223	223
🟥	jkinniso@nd.edu	14	112
⬜	eobaditc@nd.edu	80	80
▒	tperkin1@nd.edu	73	73
▒	marangu1@nd.edu	67	67
■	rsmick@nd.edu	28	28
□	ochoudhu@nd.edu	16	16
□	kherring@nd.edu	8	8
□	mvalenc2@nd.edu	6	6
□	ekrebs@nd.edu	3	3
□	jdiazort@nd.edu	2	2
□	nblancha@nd.edu	1	1
□	nsmith9@nd.edu	1	1
🟩	Unclaimed	350	2697
🟦	Matched	1	9
🟧	Preempting	9	9
🟫	Owner	146	1389
	Total	6934	10630

Display Options

Sort: users machines

Show: users states

Size: bigger smaller

Scale: none

Dialogue with our target users

Dialogue with our target users

- How much memory does it use?

- Eh....

Dialogue with our target users

- How much memory does it use?

- Eh....

- What about disk usage?

- Well...

Dialogue with our target users

- How much memory does it use?

- Eh....

- What about disk usage?

- Well...

- A ballpark figure?

- Mmm... It runs in my laptop...

Dialogue with our target users

- How much memory does it use?

- Eh....

- What about disk usage?

- Well...

- A ballpark figure?

- Mmm... It runs in my laptop...

- Surely you have a list of all the files used?

- ...

Dialogue with our target users

- How much memory does it use?

- Eh....

- What about disk usage?

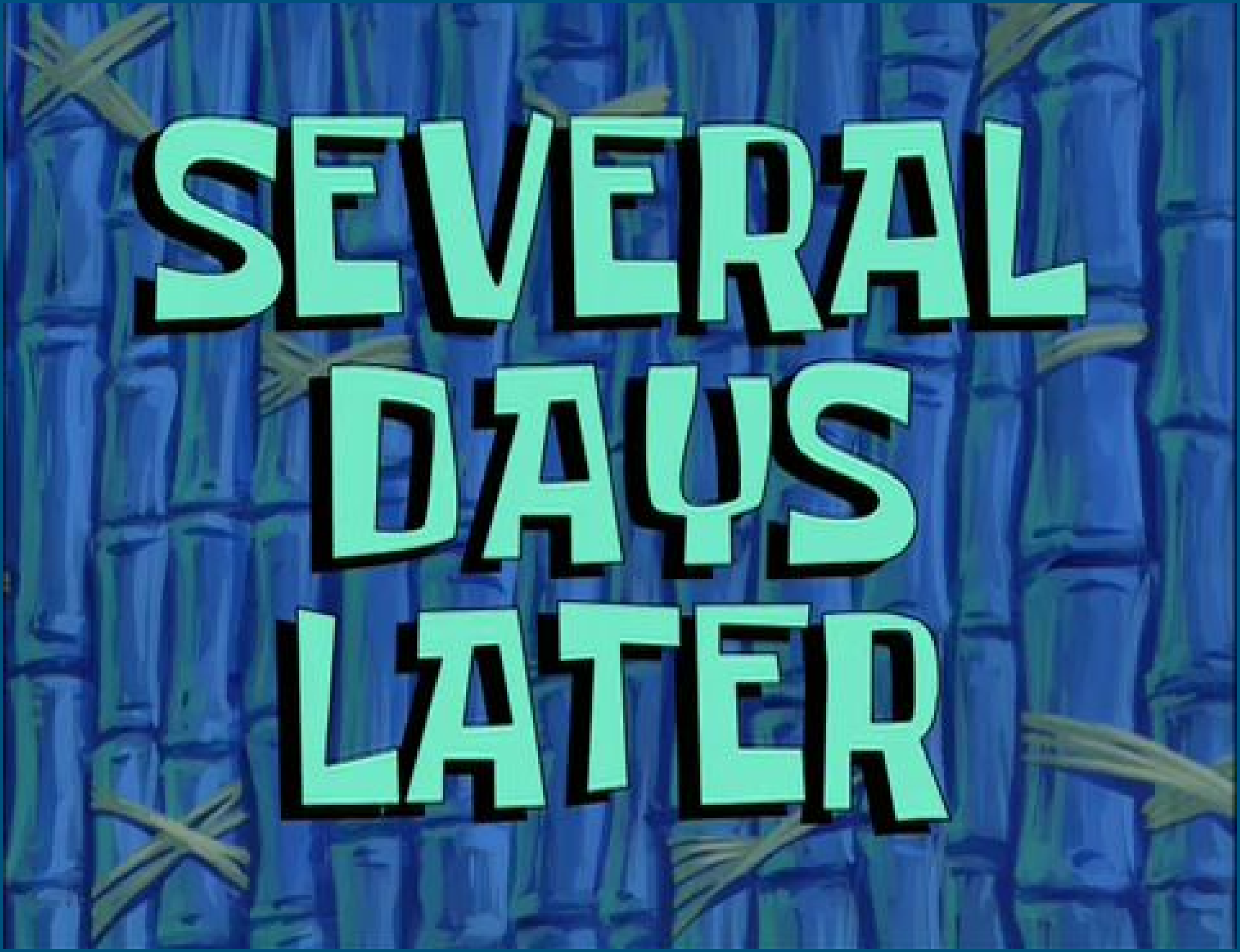
- Well...

- A ballpark figure?

- Mmm... It runs in my laptop...

- Surely you have a list of all the files used?

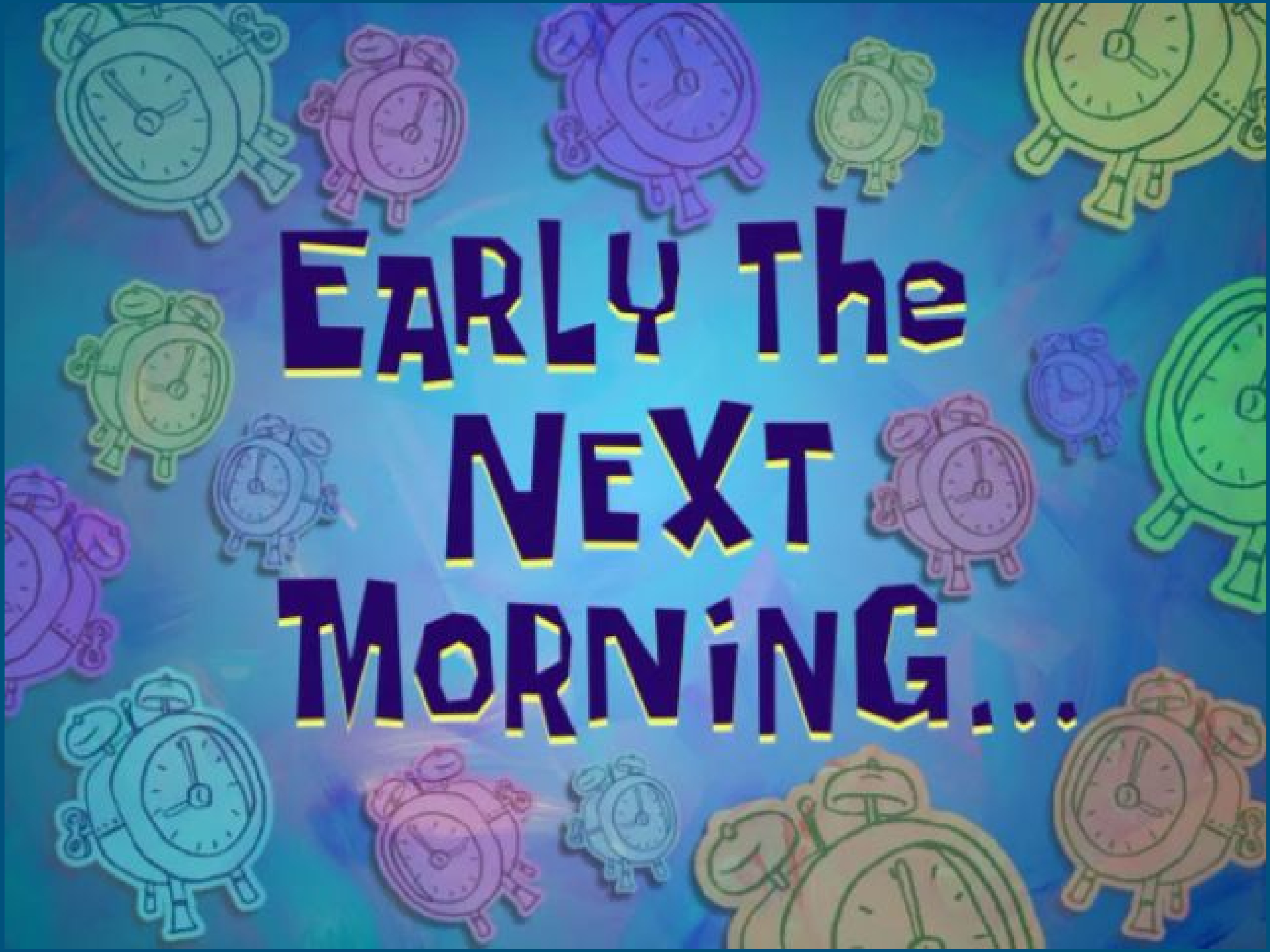
- ...


SEVERAL
DAYS
LATER

Dialog with our target users

- Ok, I think we got the condor info right... SuperSequencer3000 seems to be working on the remotes nodes now.

- Yaaaay! I'll run our workflow shortly!


EARLY THE
NEXT
MORNING...

Dialog with our target users

- It does not work anymore? Did you change anything?

- No!

Dialog with our target users

- It does not work anymore? Did you change anything?
- No!

- Your jobs are running out of disk. Nothing changed?
- No!

Dialog with our target users

- It does not work anymore? Did you change anything?

- No!

- Your jobs are running out of disk. Nothing changed?

- No!

- Wait, that parameter looks different from last time.

Dialog with our target users

- It does not work anymore? Did you change anything?

- No!

- Your jobs are running out of disk. Nothing changed?

- No!

- Wait, that parameter looks different from last time.

- Oh, that! Yes, we did change that...

Dialog with our target users

- **It does not work anymore? Did you change anything?**
- No!

- **Your jobs are running out of disk. Nothing changed?**
- No!

- **Wait, that parameter looks different from last time.**
- Oh, that! Yes, we did change that...

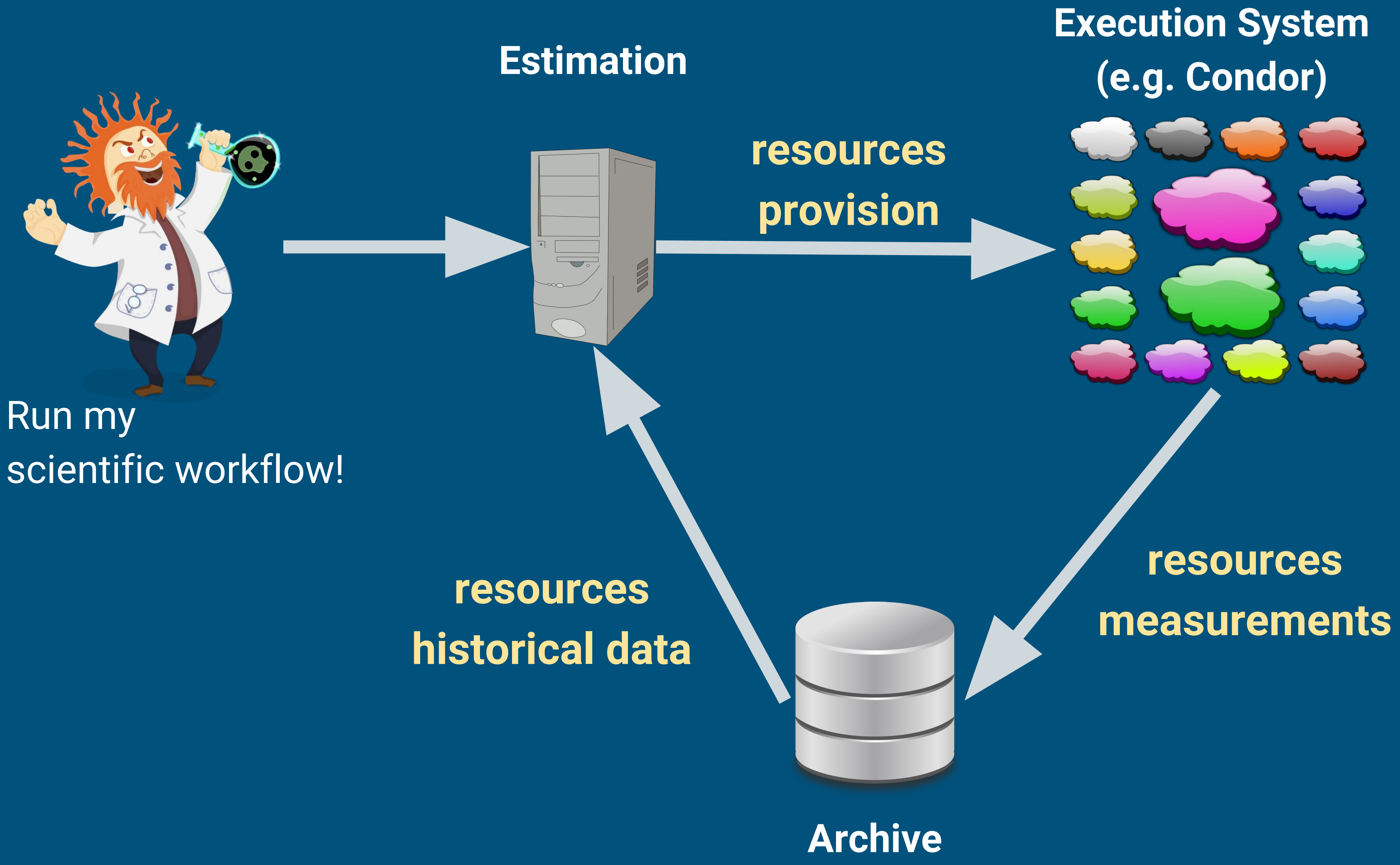
Dialog with our target users

- It does not work anymore? Did you change anything?
- No!

- Your jobs are running out of disk. Nothing changed?
- No!

- Wait, that parameter looks different from last time.
- Oh, that! Yes, we did change that...
...but we need to change that parameter often for our research...

Where we want to be



Estimation

Execution System
(e.g. Condor)

Run my
scientific workflow!

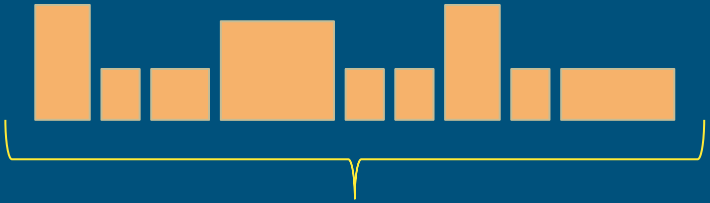
resources
provision

resources
measurements

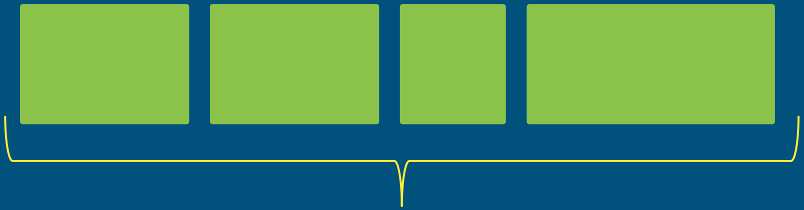
resources
historical data

Archive

Tasks with Unknown Resource Requirements

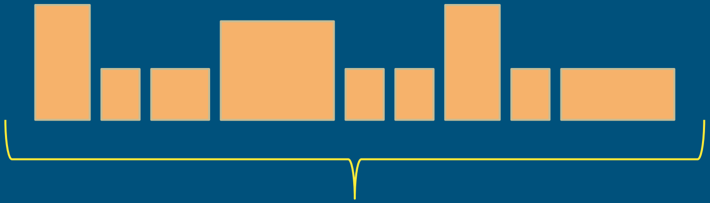


Tasks which size
(e.g., cores, memory, and disk)
is not known until runtime.

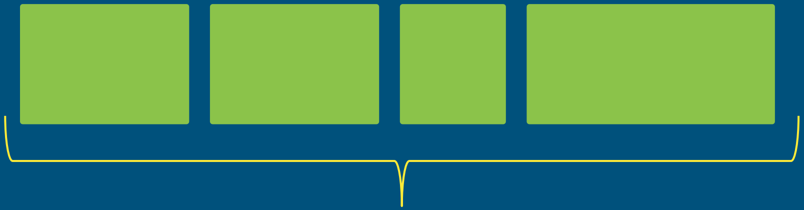


Available condor slots

Tasks with Unknown Resource Requirements

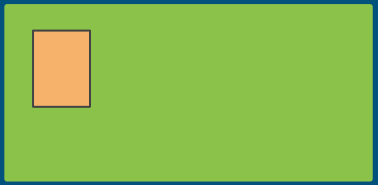


Tasks which size
(e.g., cores, memory, and disk)
is not known until runtime.

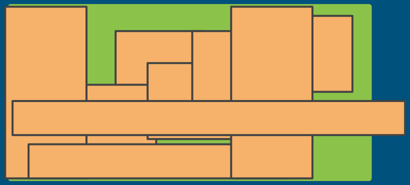


Available condor slots

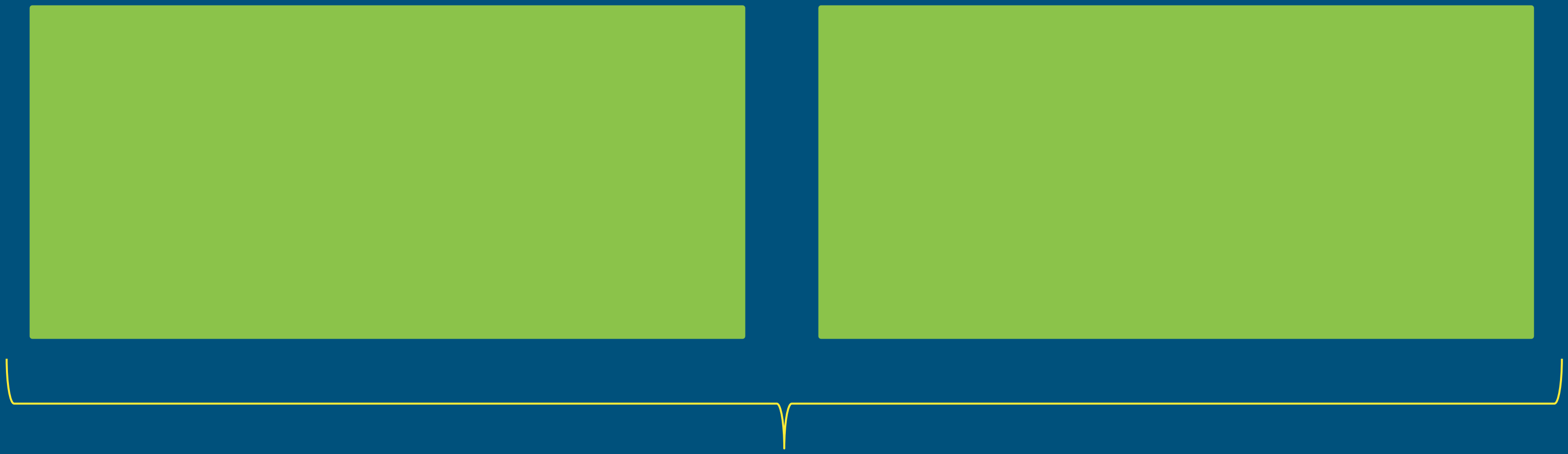
One task per slot:
Wasted resources, reduced throughput.



Many tasks per slot (e.g. with pilot job):
Resource contention/exhaustion, reduce throughput

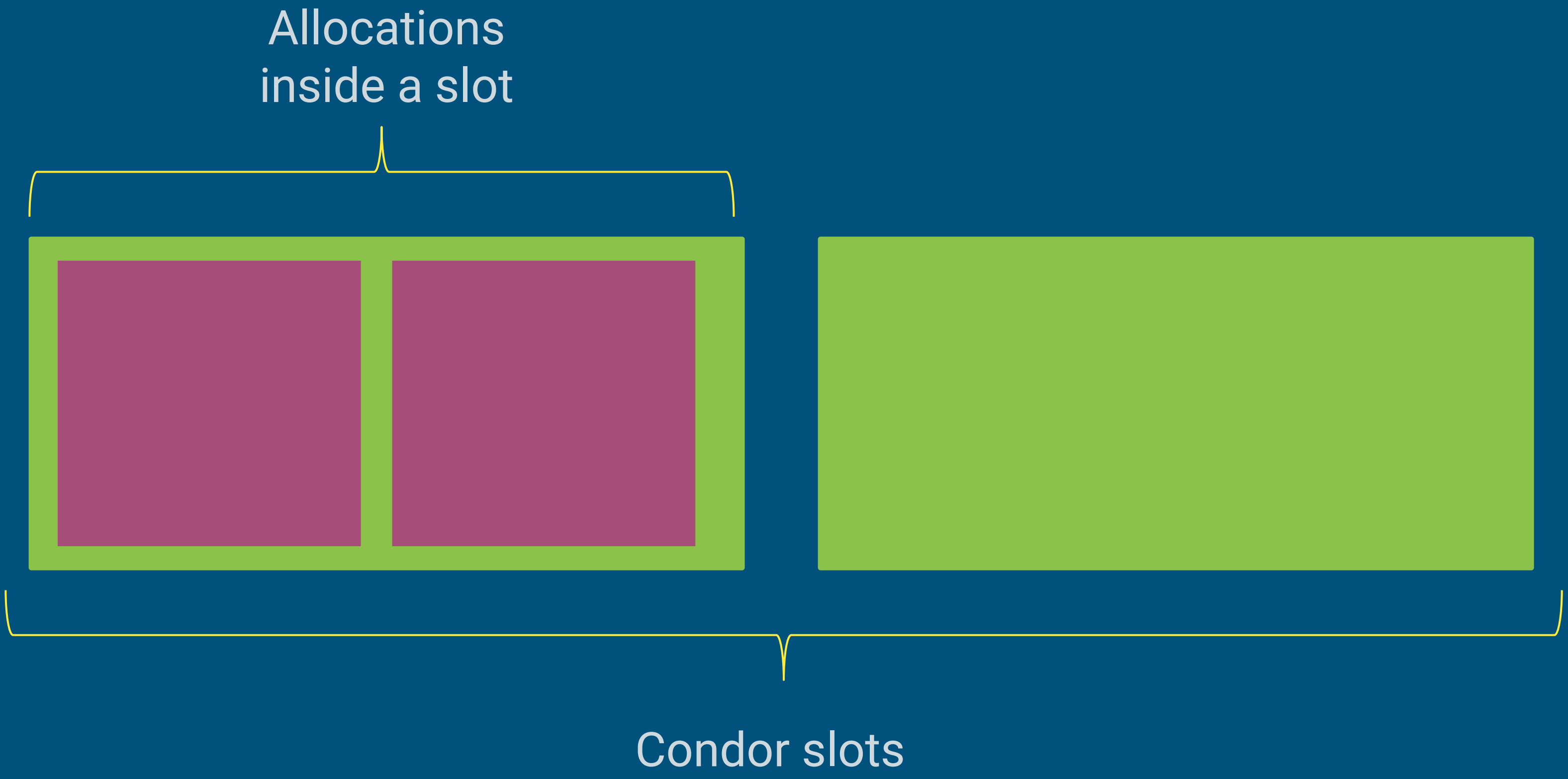


Task-in-the-Box



Condor slots

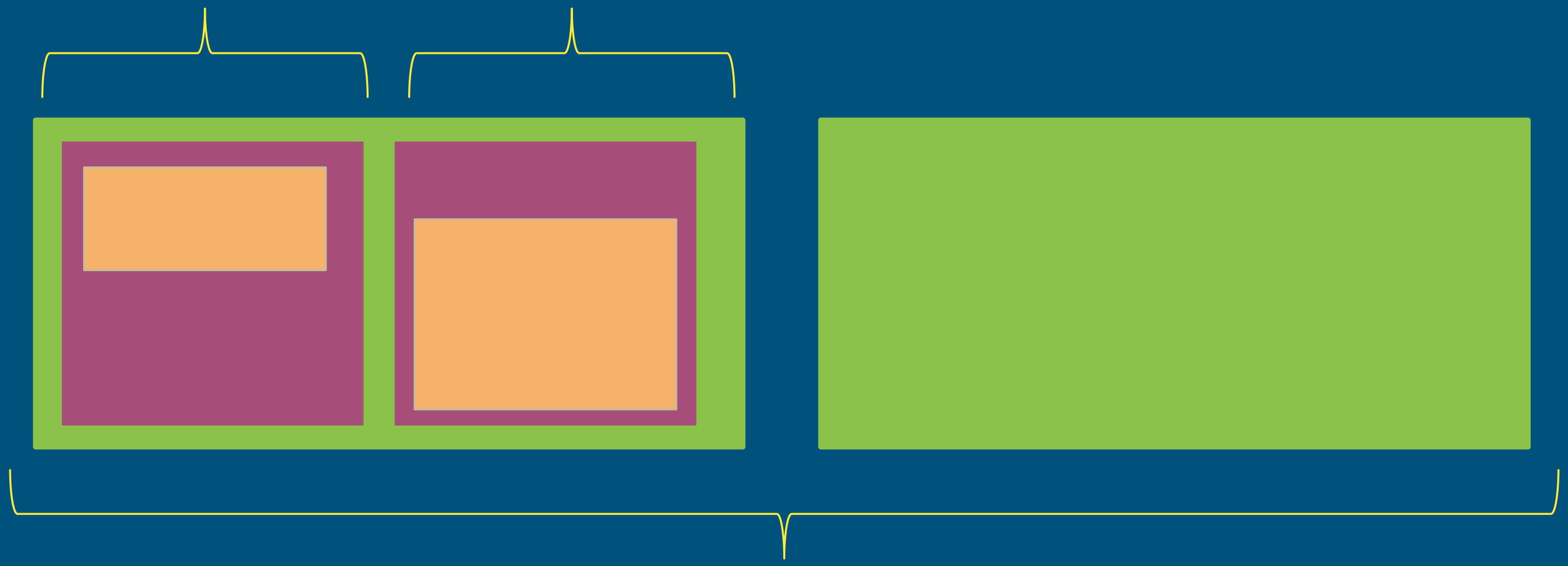
Task-in-the-Box



Allocations inside a slot

Condor slots

Task-in-the-Box

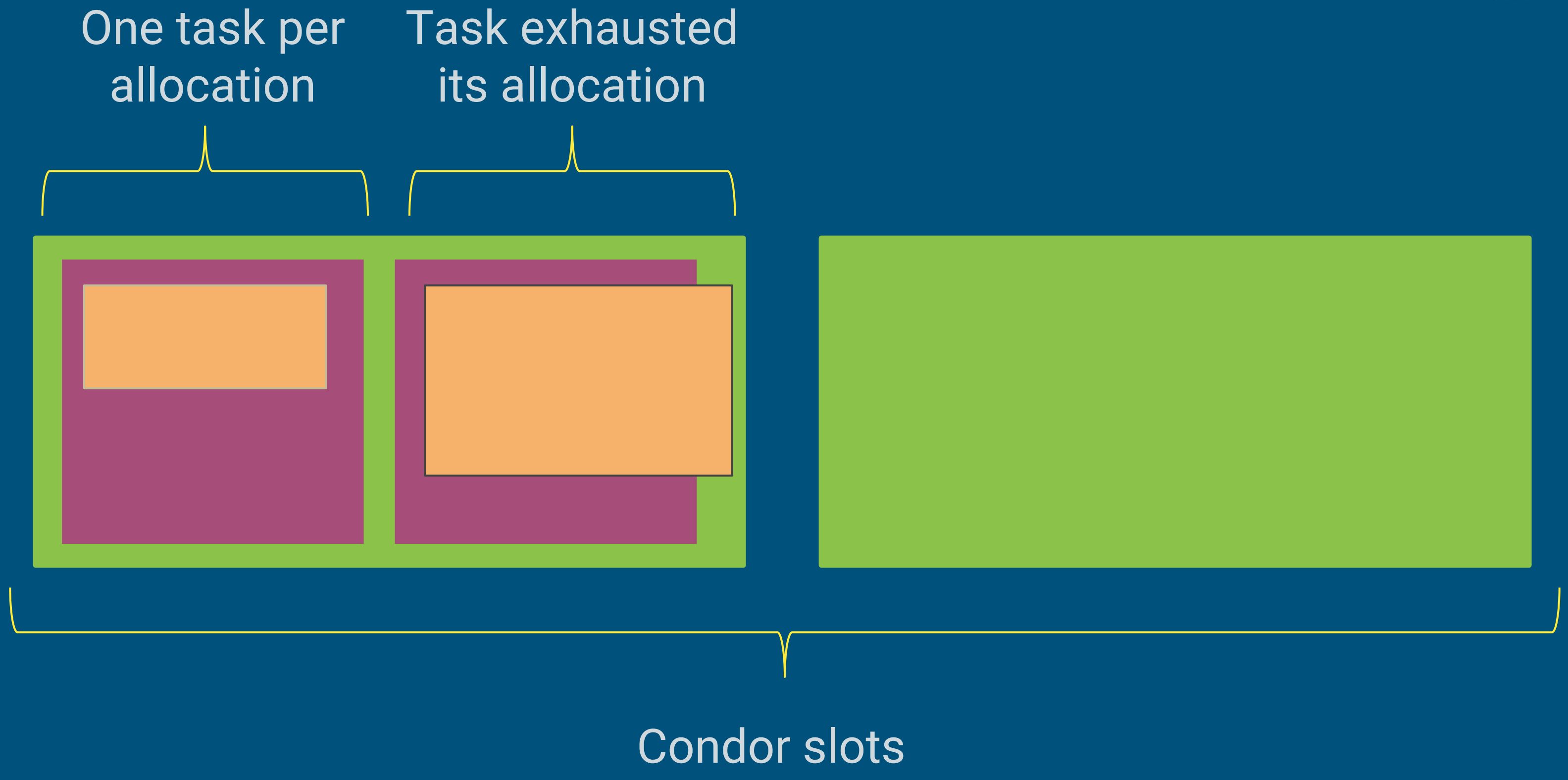


One task per allocation

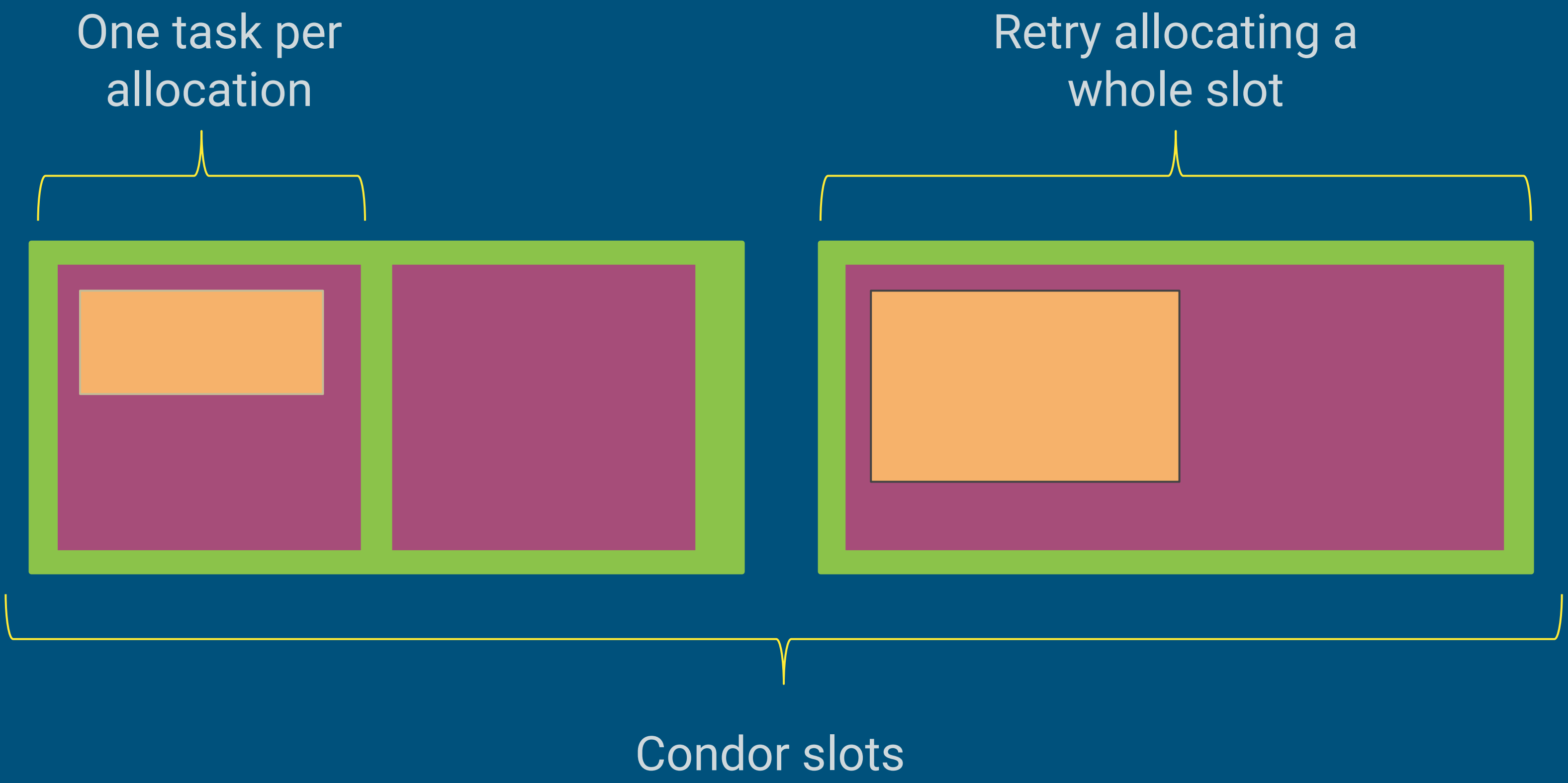
One task per allocation

Condor slots

Task-in-the-Box



Task-in-the-Box



Main Challenges

What is a good allocation size?

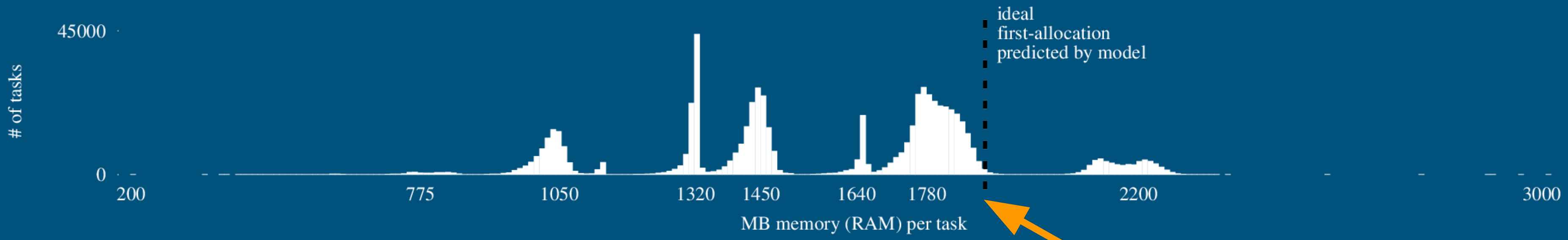
How do we measure the tasks?

How do we enforce the allocations?

One-guess policy result (guess once, then use max seen)

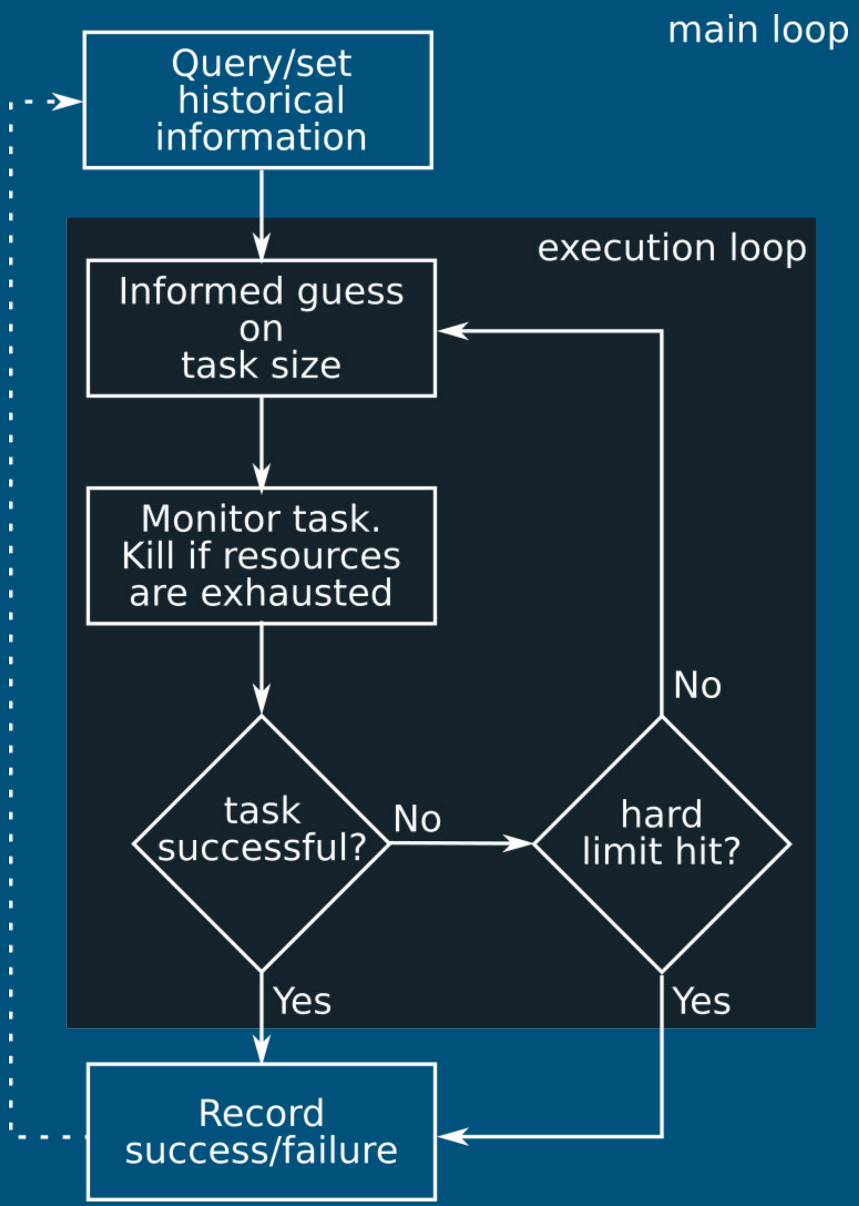
Real result from a production High-Energy Physics CMS analysis (Lobster NDCMS)

Histogram Peak Memory vs Number of Tasks
O(700K) tasks that ran in O(26K) cores managed by WorkQueue/Condor.



First-allocation that maximizes expected throughput
(increase of %40 w.r.t. no task is retried)

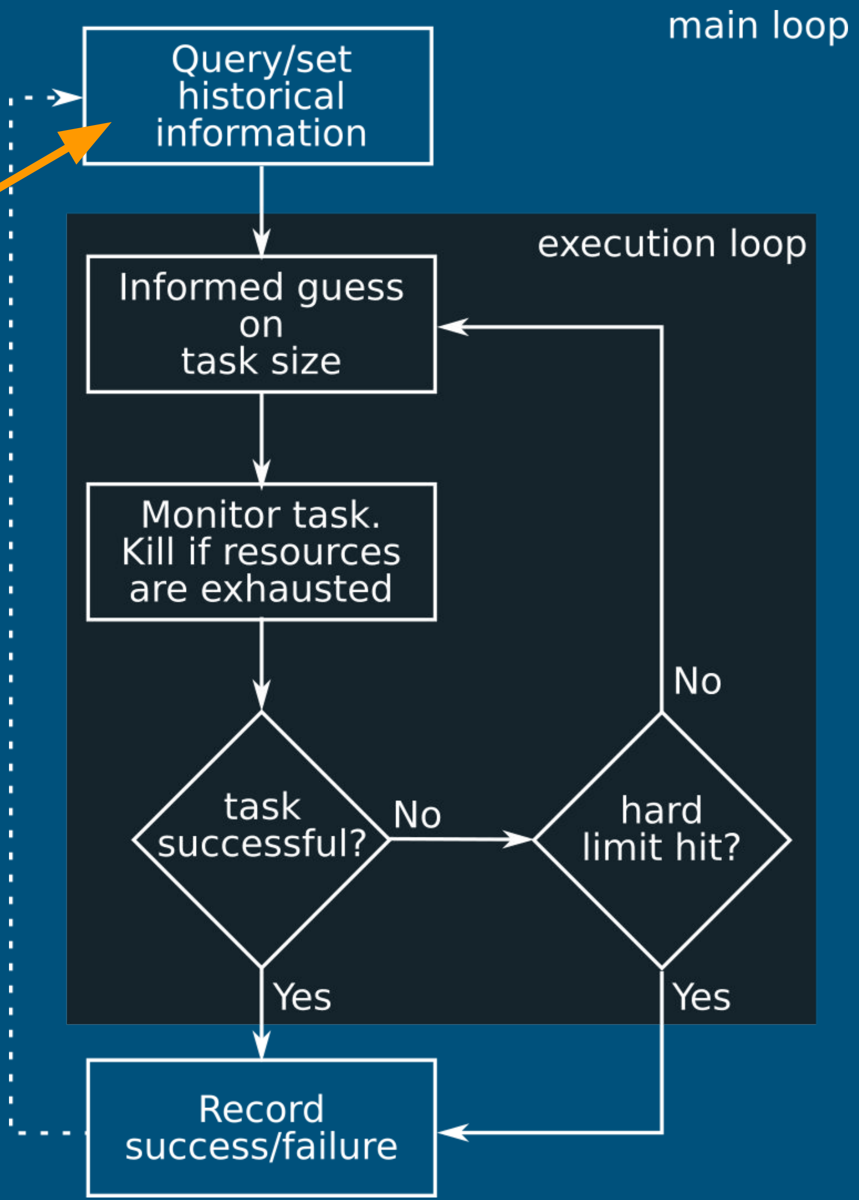
And around it goes...


main loop
Query/set historical information
execution loop
Informed guess on task size
Monitor task. Kill if resources are exhausted
task successful?
No
hard limit hit?
No
Yes
Yes
Record success/failure

And around it goes...

What do we know?

Historical data?
Probability distribution?
Perfect information?
Empirical distribution?



main loop

Query/set historical information

execution loop

Informed guess on task size

Monitor task. Kill if resources are exhausted

task successful?

No

hard limit hit?

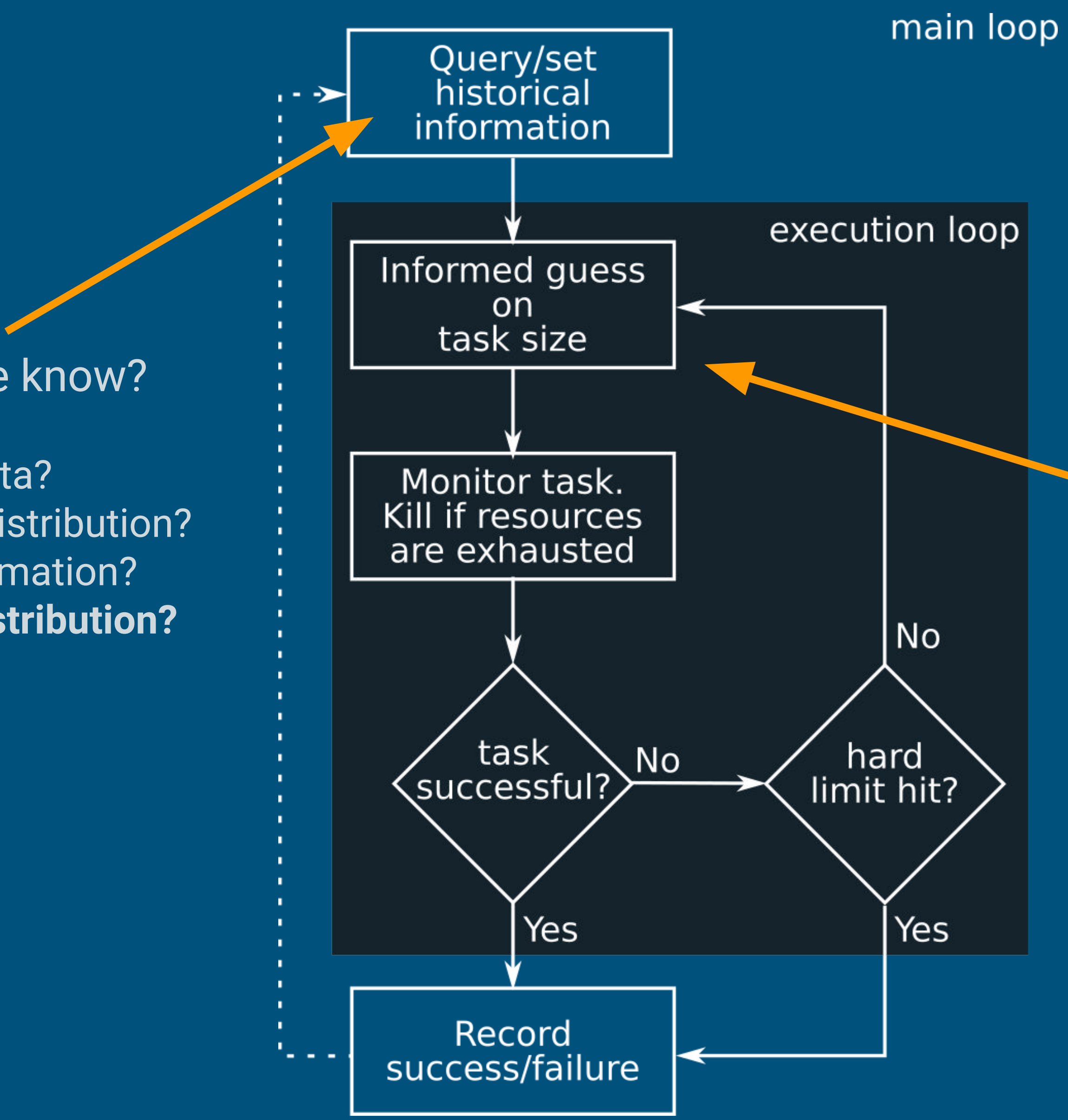
No

Yes

Yes

Record success/failure

And around it goes...


main loop
Query/set historical information
execution loop
Informed guess on task size
Monitor task. Kill if resources are exhausted
task successful?
No
hard limit hit?
No
Yes
Yes
Record success/failure

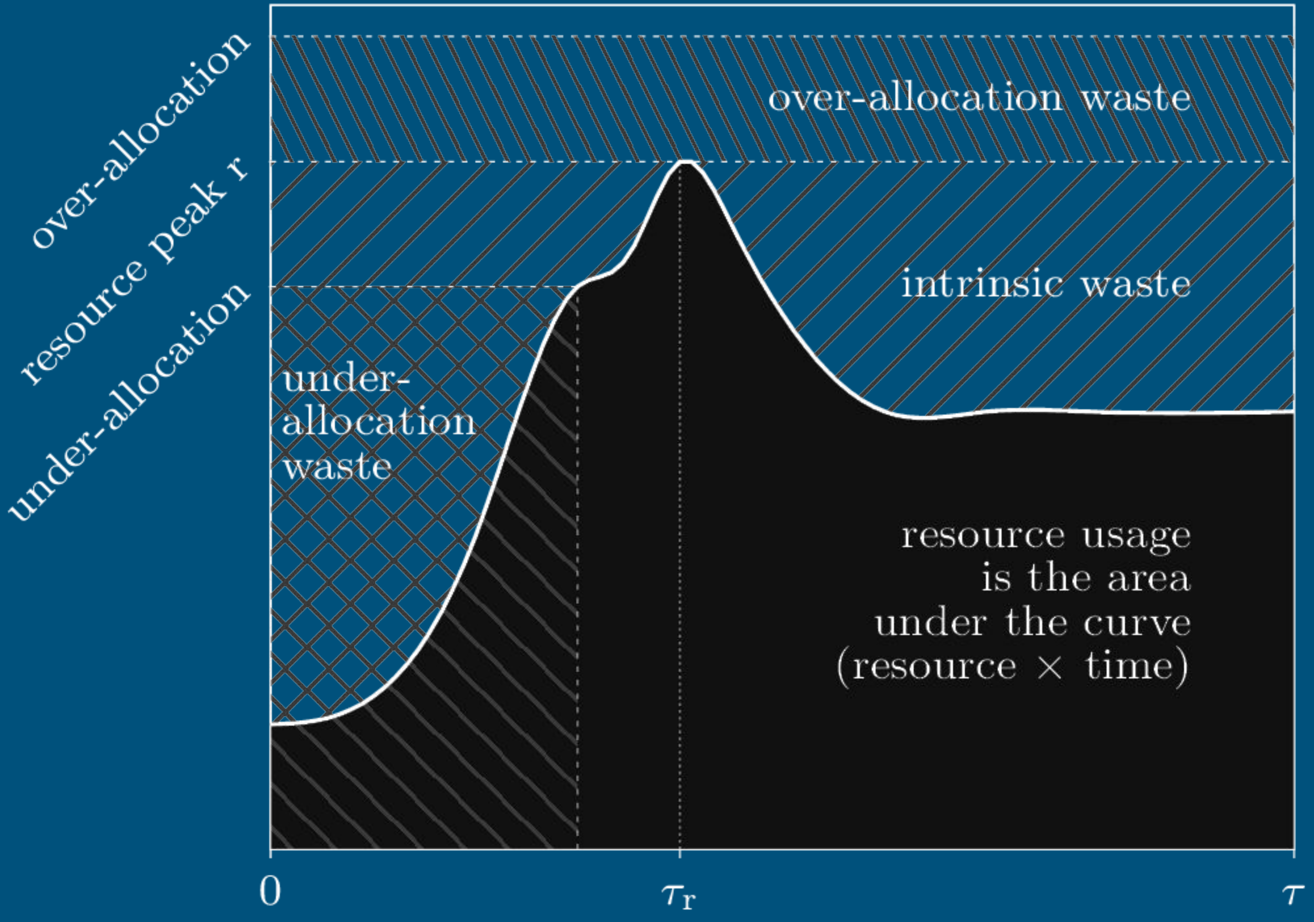
What do we know?

Historical data?
Probability distribution?
Perfect information?
Empirical distribution?

What do we want?

Minimize retries?
Minimize waste?
Maximize throughput?

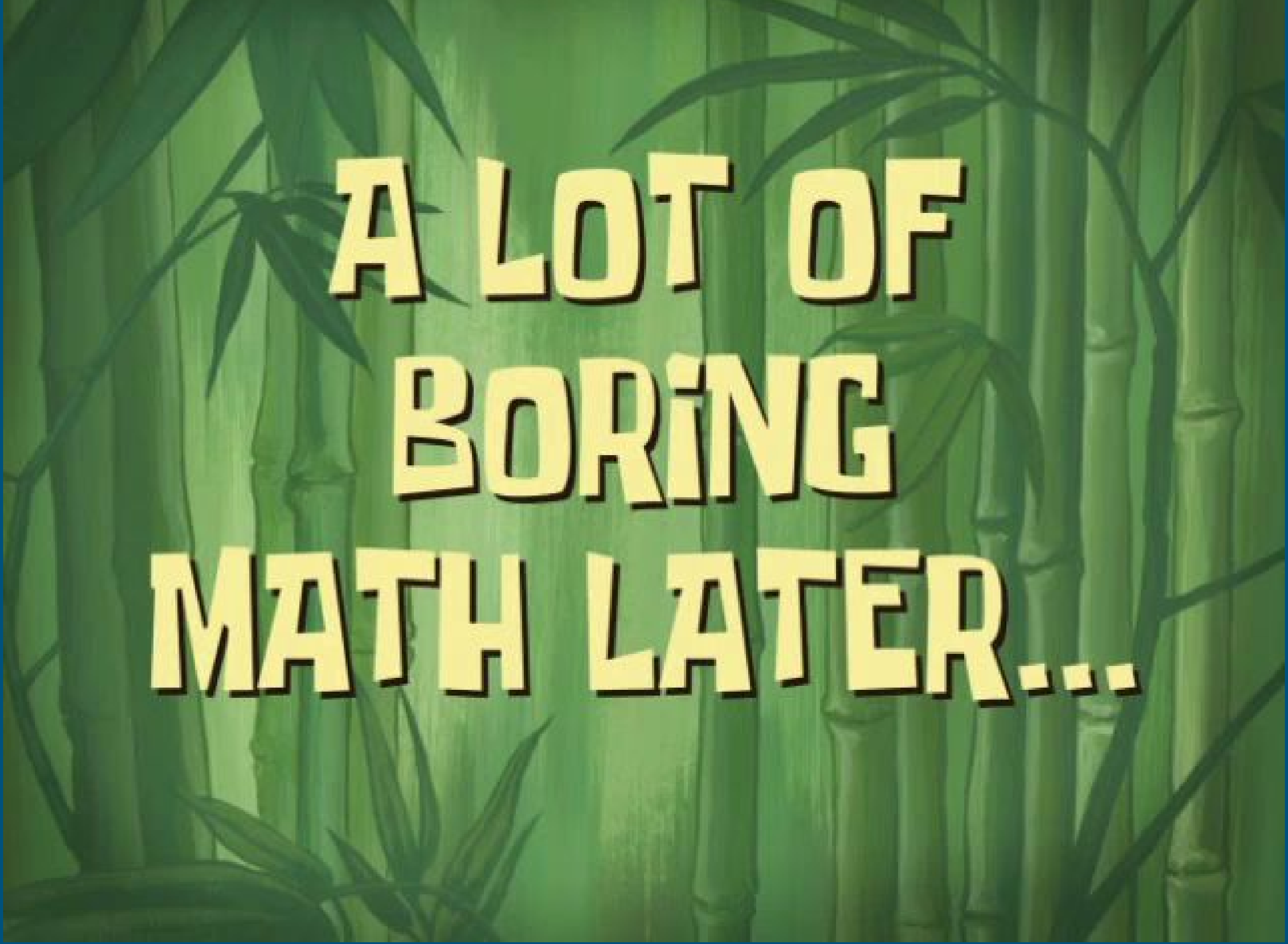
Slow-peaks model



Random variables to describe usage:
Time to completion.
Size of max peak

Resource usage:
time x peak

Slow-peaks:
Resource peaks at the end of execution (conservative assumption)


A LOT OF
BORING
MATH LATER...

Slow-peaks model

Choice of:
maximum throughput
minimum waste.

Optimizations over expectations

O(n) simple arithmetic expressions that use only information available during execution.

$$
\begin{aligned}
\mathrm{E}[\mathrm{waste}(r,\tau,a_1)] &= \int_0^\infty \left(\underbrace{\int_0^{a_1} (a_1 - r)\tau p(r,\tau)\mathrm{d}r}_{\text{first allocation succeds}} \right. \\
&\qquad \left. + \underbrace{\int_{a_1}^{a_m} ((a_m + a_1 - r)\tau p(r,\tau)\mathrm{d}r}_{\text{final allocation succeds}} \right) \mathrm{d}\tau \\
&= a_1 \underbrace{\int_{a_1}^{a_m} \int_0^\infty \tau p(r,\tau)\mathrm{d}\tau\mathrm{d}r}_{\text{mean wall-time for all tasks}} \\
&\quad + a_m \int_{a_1}^{a_m} \underbrace{\int_0^\infty \tau p(\tau|r)\mathrm{d}\tau}_{\text{mean wall-time taks w. peak } r} \quad p(r)\mathrm{d}r \\
&\quad - \underbrace{\int_0^\infty \int_0^\infty r\tau p(r,\tau)\mathrm{d}\tau\mathrm{d}r,}_{\text{used resources}}
\end{aligned}
$$

Integrated in CCTools (next major release)

```
makeflow --max-throughput -Tcondor  myworkflow
```

Activate monitor and allocations

Submit jobs to condor. Allocations in terms of request_cpus, request_memory and request_disk.

```
# unix make style recipes

output.0: input.0 cmd
    ./cmd -i input.0 output.0

output.1: input.1 othercmd
    ./othercmd < input.1  > output.1
```

We need monitoring for all of this

Mechanisms available to unprivileged users

root permissions or loading kernel modules **are a no go**

Tasks as trees of processes

no whole systems or individual processes

High-throughput computing

measure so we can run many tasks at the same time,
not to profile a single instance to make it run faster

We need monitoring for all of this

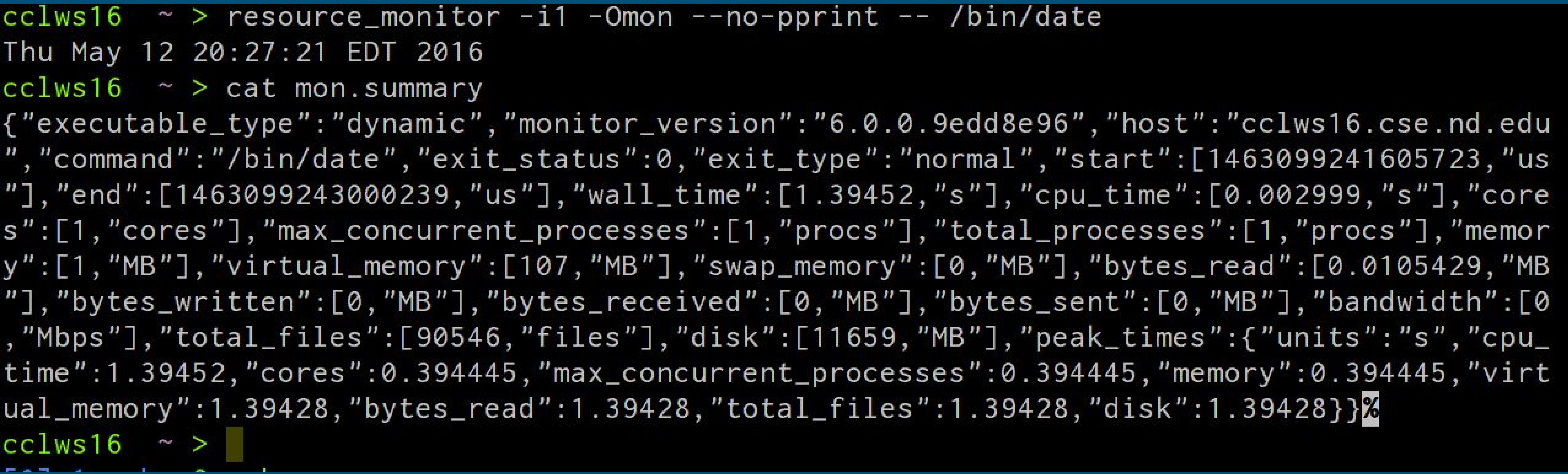
Monitoring as an unprivileged user is hard!

- No permissions
- No ways to add needed kernel support
- What the user wants to measure is different to what a system administrator may care about. (E.g., cpu usage of a single task v.s. system load.)
- Tracking children processes is hard without wrapping the parent process.

Need to measure individual tasks, not individual users or systems.

Integrated in CCTools

```
resource_monitor -L"cores: 4" -L"memory: 4096" -- matlab
```

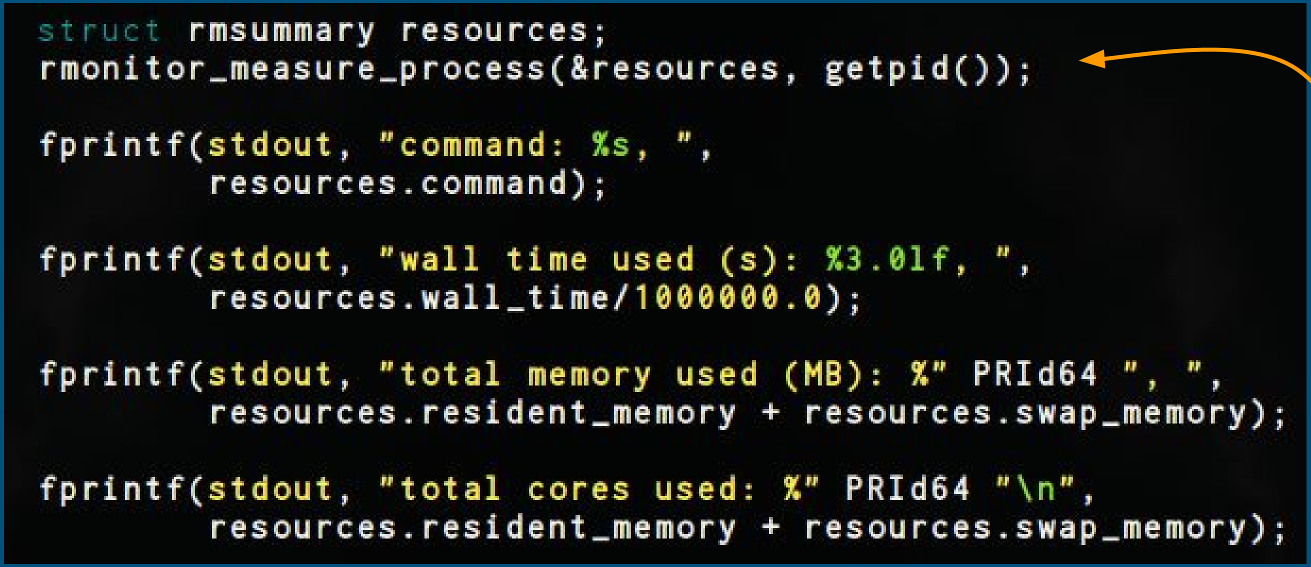


```
cclws16  ~ > resource_monitor -i1 -Omon --no-pprint -- /bin/date
Thu May 12 20:27:21 EDT 2016
cclws16  ~ > cat mon.summary
{"executable_type":"dynamic","monitor_version":"6.0.0.9edd8e96","host":"cclws16.cse.nd.edu
","command":"/bin/date","exit_status":0,"exit_type":"normal","start":[1463099241605723,"us
"],"end":[1463099243000239,"us"],"wall_time":[1.39452,"s"],"cpu_time":[0.002999,"s"],"core
s":[1,"cores"],"max_concurrent_processes":[1,"procs"],"total_processes":[1,"procs"],"memor
y":[1,"MB"],"virtual_memory":[107,"MB"],"swap_memory":[0,"MB"],"bytes_read":[0.0105429,"MB
"],"bytes_written":[0,"MB"],"bytes_received":[0,"MB"],"bytes_sent":[0,"MB"],"bandwidth":[0
,"Mbps"],"total_files":[90546,"files"],"disk":[11659,"MB"],"peak_times":{"units":"s","cpu_
time":1.39452,"cores":0.394445,"max_concurrent_processes":0.394445,"memory":0.394445,"virt
ual_memory":1.39428,"bytes_read":1.39428,"total_files":1.39428,"disk":1.39428}}%
cclws16  ~ >
```

(does not work as well on static executables that fork)

Recent development: Monitoring Library

The resource_monitor main functionality was converted into a library, with C, Python, and perl interfaces.



```
struct rmsummary resources;
rmonitor_measure_process(&resources, getpid());

fprintf(stdout, "command: %s, ",
        resources.command);

fprintf(stdout, "wall time used (s): %3.0lf, ",
        resources.wall_time/1000000.0);

fprintf(stdout, "total memory used (MB): %" PRId64 ", ",
        resources.resident_memory + resources.swap_memory);

fprintf(stdout, "total cores used: %" PRId64 "\n",
        resources.resident_memory + resources.swap_memory);
```

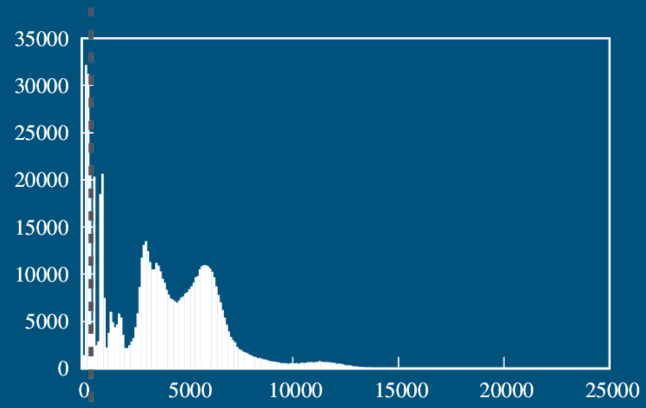
An application can poll its resources usage with a single library call.

(unlike resource_monitor, does not track forks/exits)

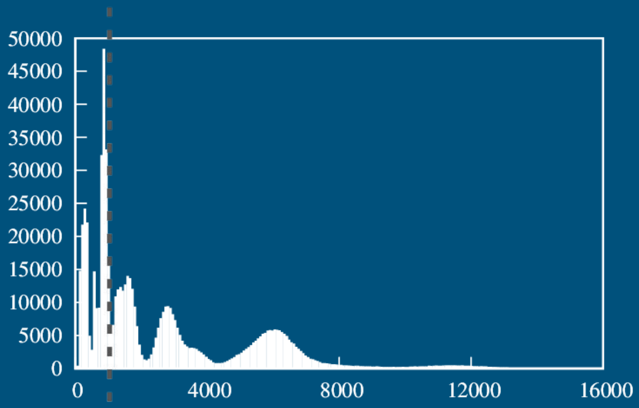
ND CMS workflow distributions

- 681874 tasks on Lobster/WorkQueue/Condor
- Computing allocations takes ~ 0.05 seconds.

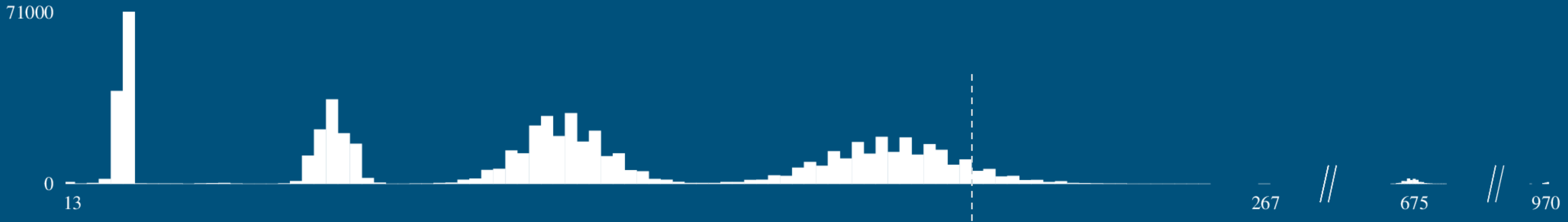
# of tasks



wall-time, peak 25000 s

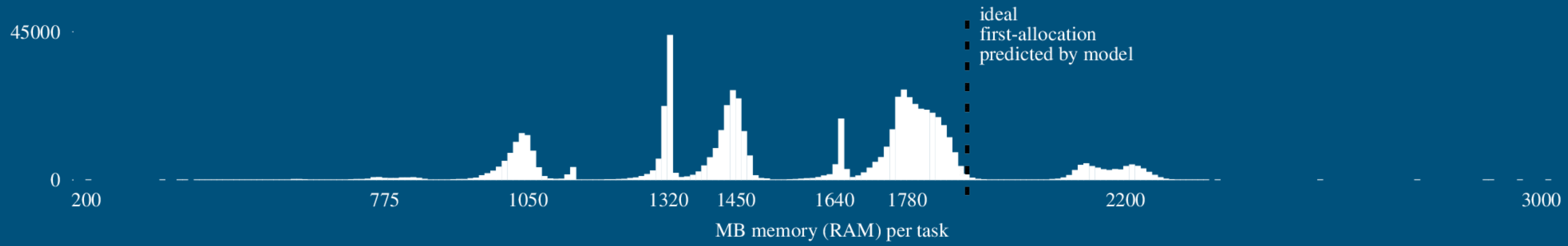


cpu-time peak 6000 s



disk peak at 1GB

ND CMS workflow: Memory bottleneck



	Size	Retries	Proportion wasted	Throughput (norm)
Max peak always	3GB	0%	48%	1.0
Perfect information	-	0%	0%	2.0
Slow-peaks one-guess	1.9GB	9%	28%	1.41

ND CMS workflow: Memory bottleneck

Things are even better if users give coarse information about the workflow. As simple as putting tasks into categories (e.g., merge, analysis recostep, parameter-X, etc.)

	Size	Retries	Proportion wasted	Throughput (norm)
Max peak always	3GB	0%	48%	1.0
Perfect information	-	0%	0%	2.0
Slow-peaks one-guess	1.9GB	9%	28%	1.41
One-guess + categories	(per category)	< 1%	17%	1.64

Questions?

Acknowledgements:

Many thanks to ND CMS group:

Prof. Kevin Lannon
Anna Woodard
Mathias Wolf
Kenyi Hurtado

DOE Grant: ER26110

Downloads:

cctools
http://ccl.cse.nd.edu

btovar@nd.edu

(Paper under current blind-review.
If you are a reviewer,
you are feeling very sleepy...
At the count of three
you will forget all of this...)