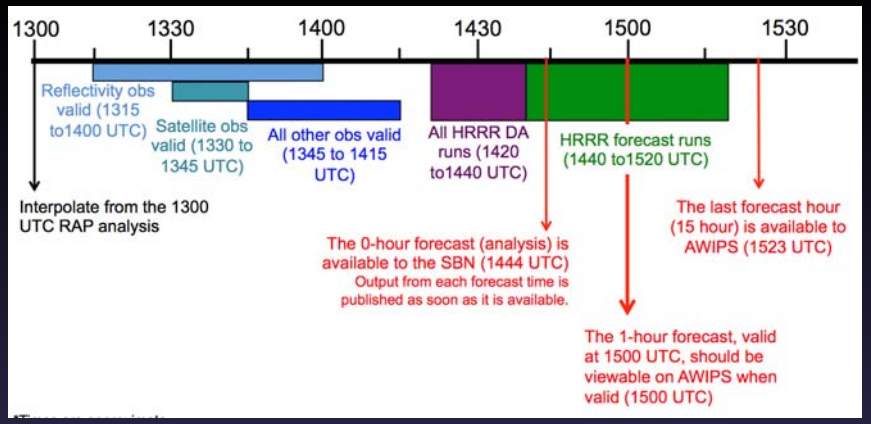
Supporting Deadline Driven Science in Research and Development High Performance Computing Environments

Craig Tierney[1]
Nathan Dauchy[2]
Chris Harrop[1]
Forrest Hobbs[3]

[1] Cooperative Institute for Research in Environmental Sciences, University of Colorado at Boulder
[2] Computer Sciences Corporation
[3] National Oceanic and Atmospheric Administration, Earth Science Research Laboratory, Global Systems Division

What is Deadline Driven Science?

- Deadline for completion is critical to value of workflow completion
  - Real-time experiments
  - Guidance products

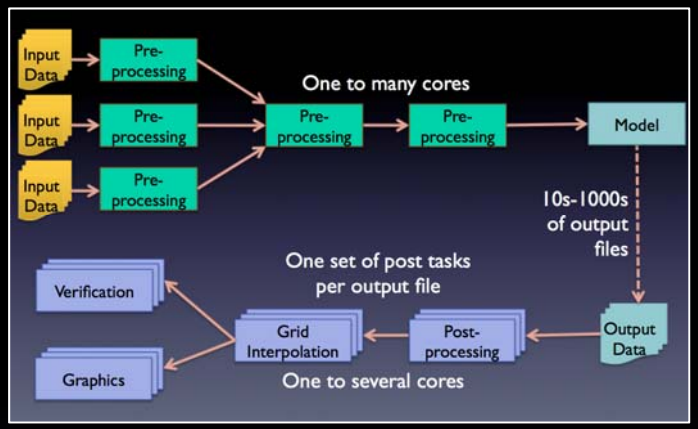


- Similar to operational, except
  - No guarantees provided to product users
  - No impact to life and property when runs are missed

What Are the Challenges?

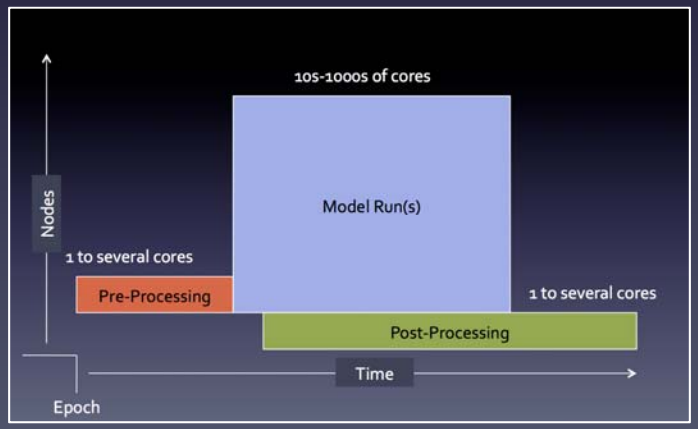
- Most R&D HPC Systems
  - FIFO queue, possibly with fair-share
  - Large mix of users, job sizes, varying operating modes
- Complex time, file, and job dependencies
- Need guarantees to meet deadlines
- Need reliable/resilient/robust workflow management
- No operational staff to monitor job completion

Solutions needs to meet our philosophy of ***Portability***

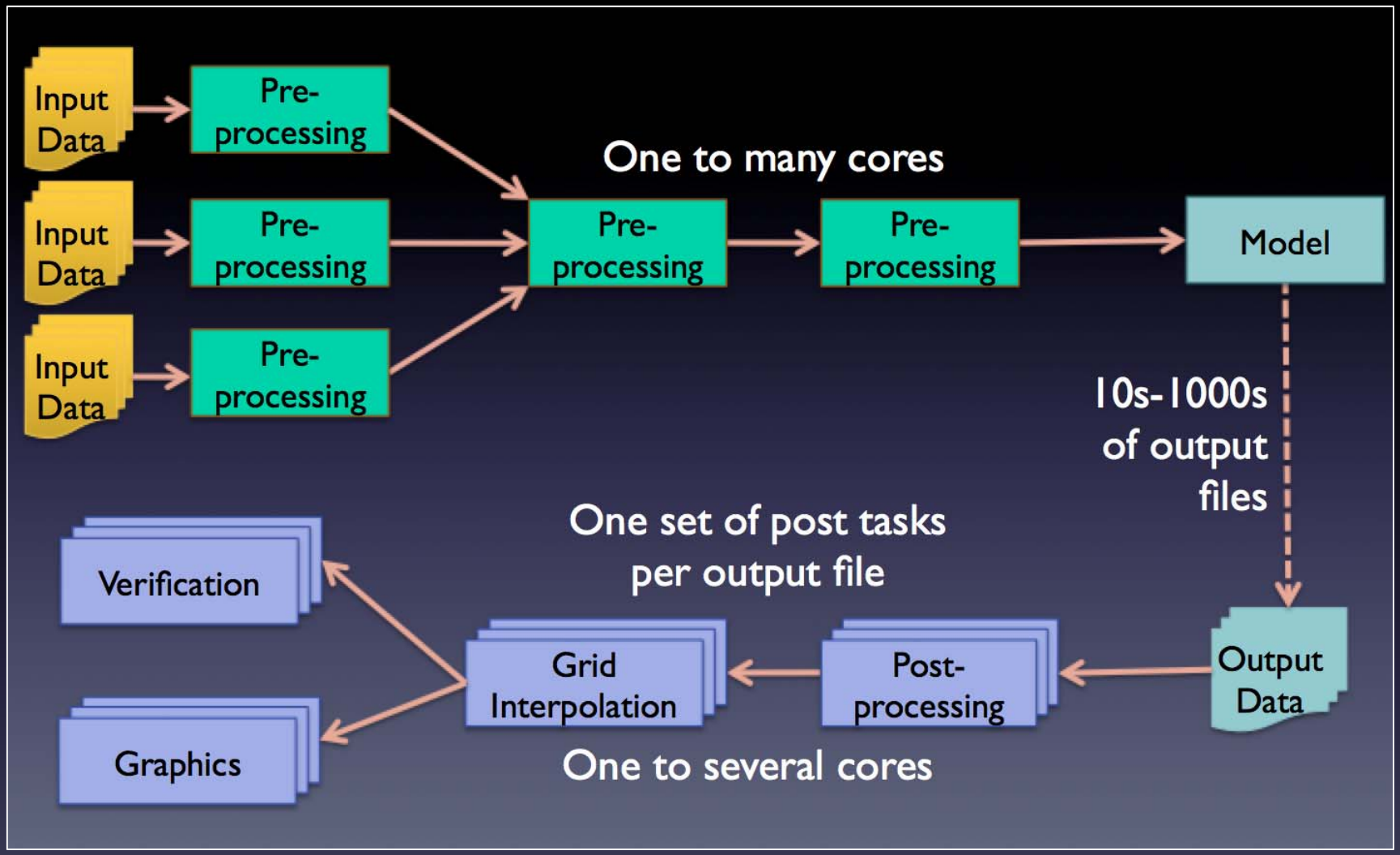


Workflow Management

Distributed CRON



Standing Reservations



Workflow Management with Rocoto

Chris Harrop

What is Workflow Management?

What is Workflow Management?

Describe and manage the execution of a collection of tasks in a scientific application.

What is Workflow Management?

Describe and manage the execution of a collection of tasks in a scientific application.

That's Easy!!!

What is Workflow Management?

What is Workflow Management?

Ensure completion of workflows with complex dependencies on tasks, files, and times on systems when, not if, component failures happen with no human active job monitoring.

What is Workflow Management?

Ensure completion of workflows with complex dependencies on tasks, files, and times on systems when, not if, component failures happen with no human active job monitoring.

That's Not So Easy…

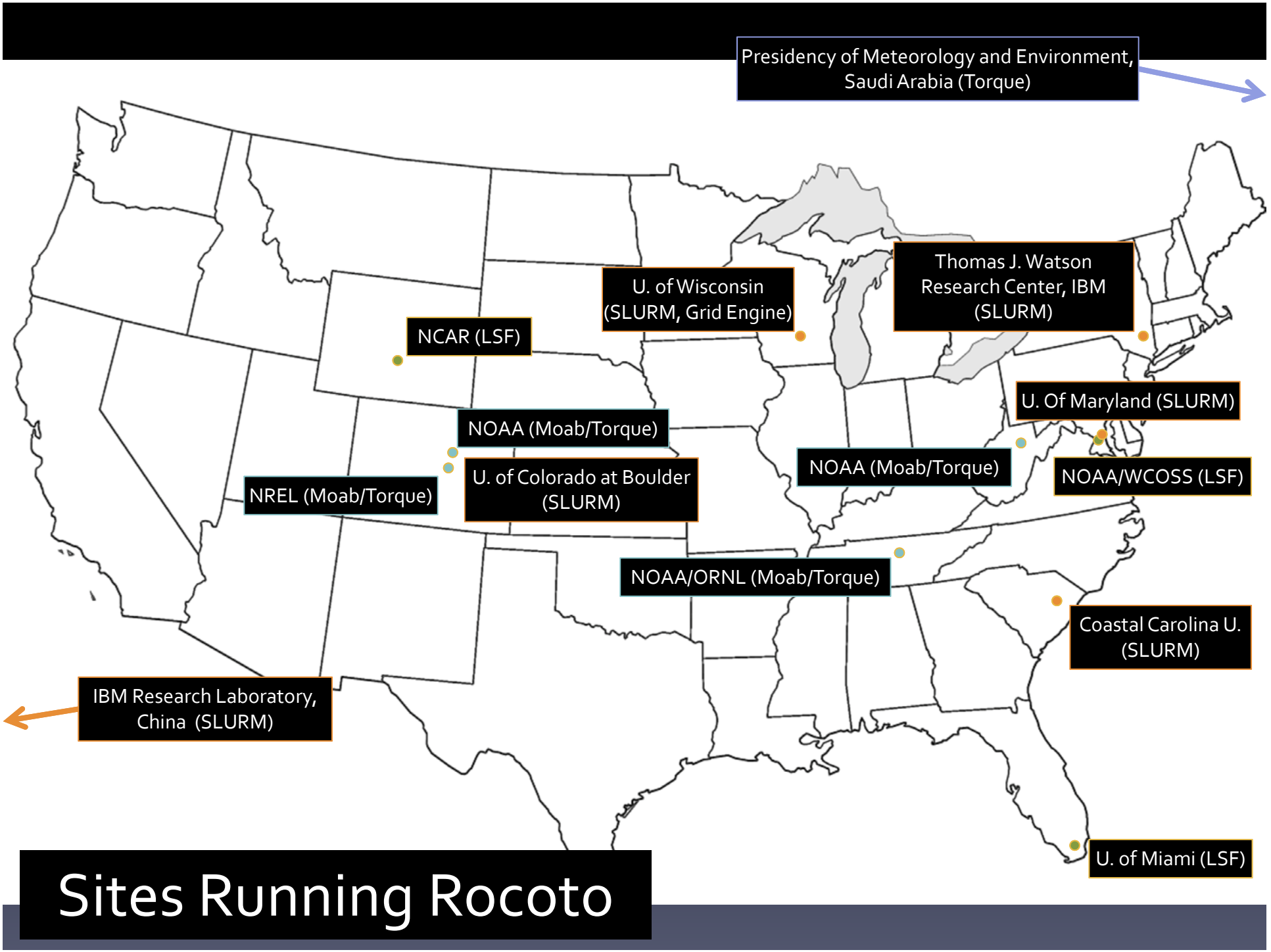
Rocoto

- Supports weather and climate community modeling paradigms
- Runs in user-space
- Portable across many different batch systems
  - Moab/Torque, LSF, Grid Engine, SLURM

ROCOTO manages most all work by the Development Testbed Center
http://www.dtcenter.org/

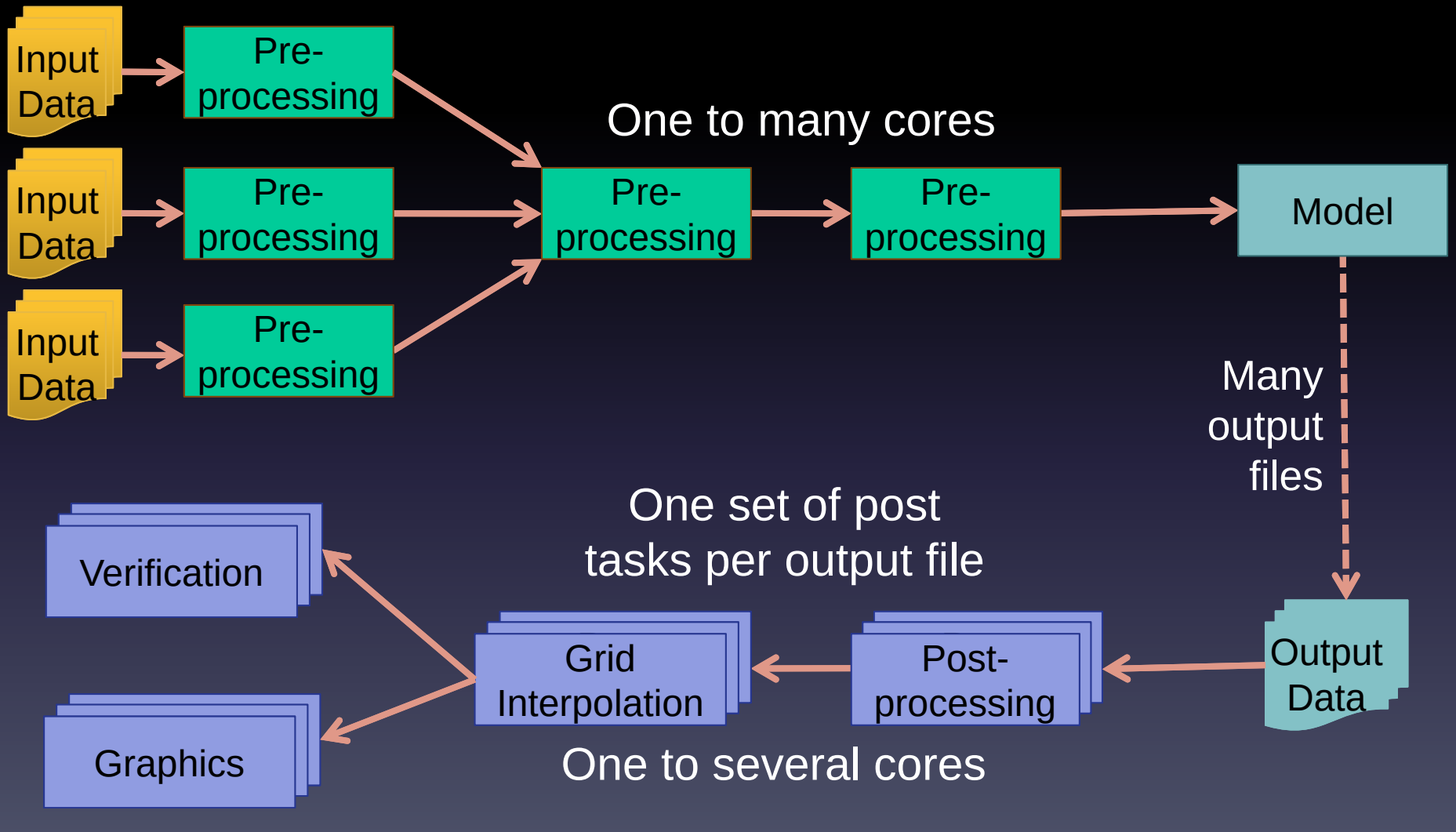
Rocoto – Key Features

- Real-time and retrospective modes
- Fault Tolerance
- Complex dependencies based on Time, File and Task
- Generic and portable batch specifications
- Multi-threaded job submission
- Workflow throttling
- Meta tasks conveniently describe multiple, similar, tasks

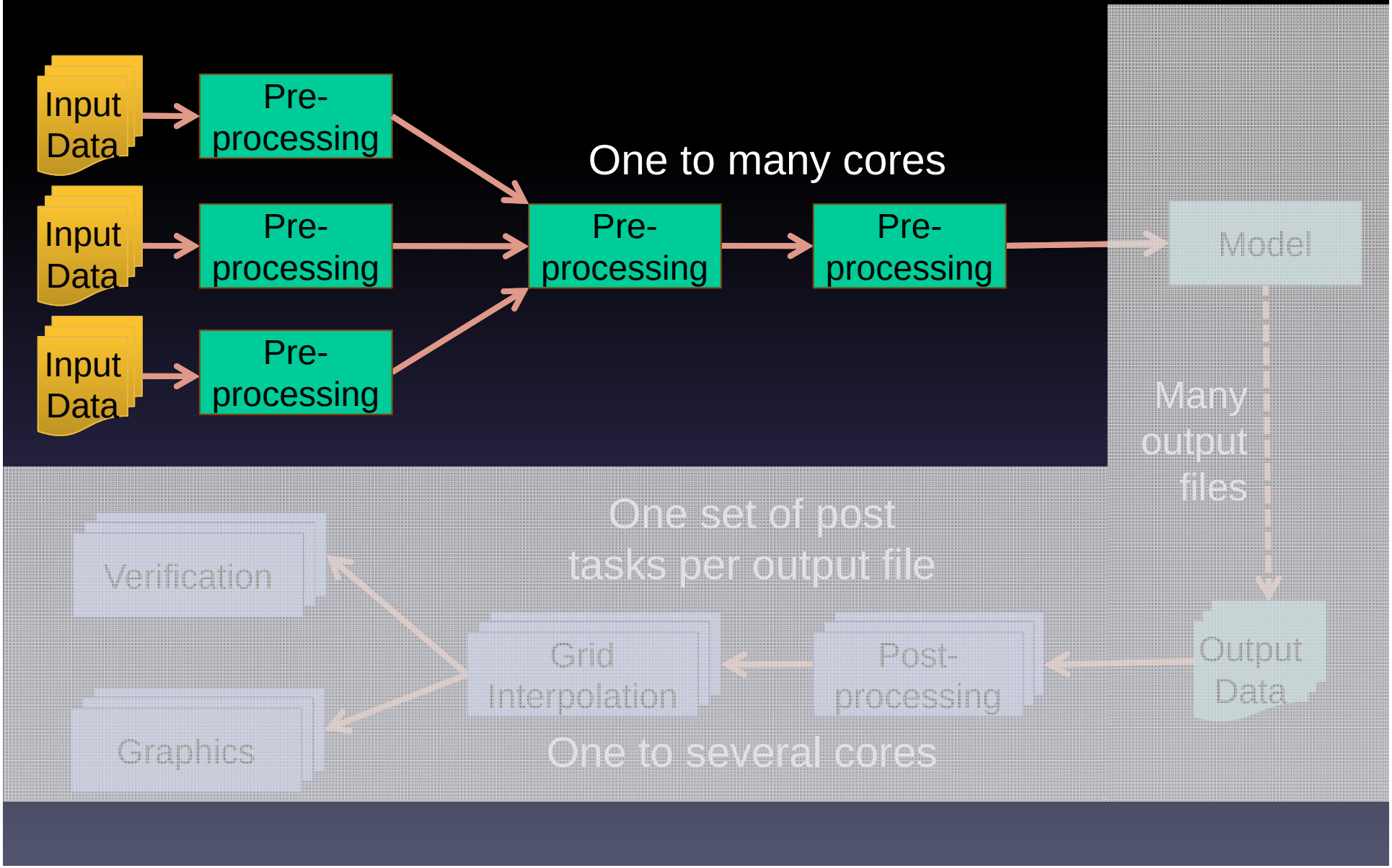


Sites Running Rocoto

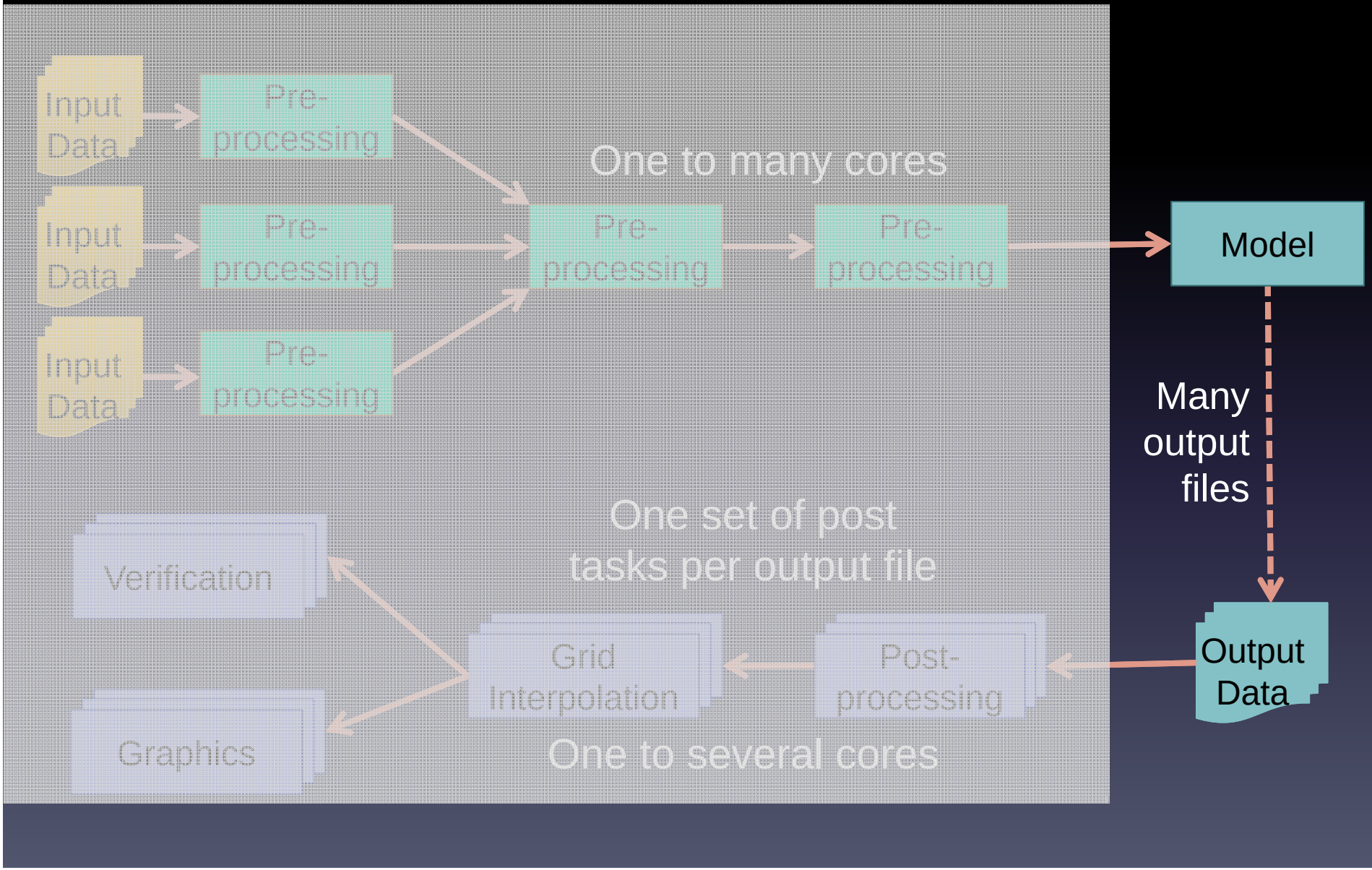
A Typical Workflow



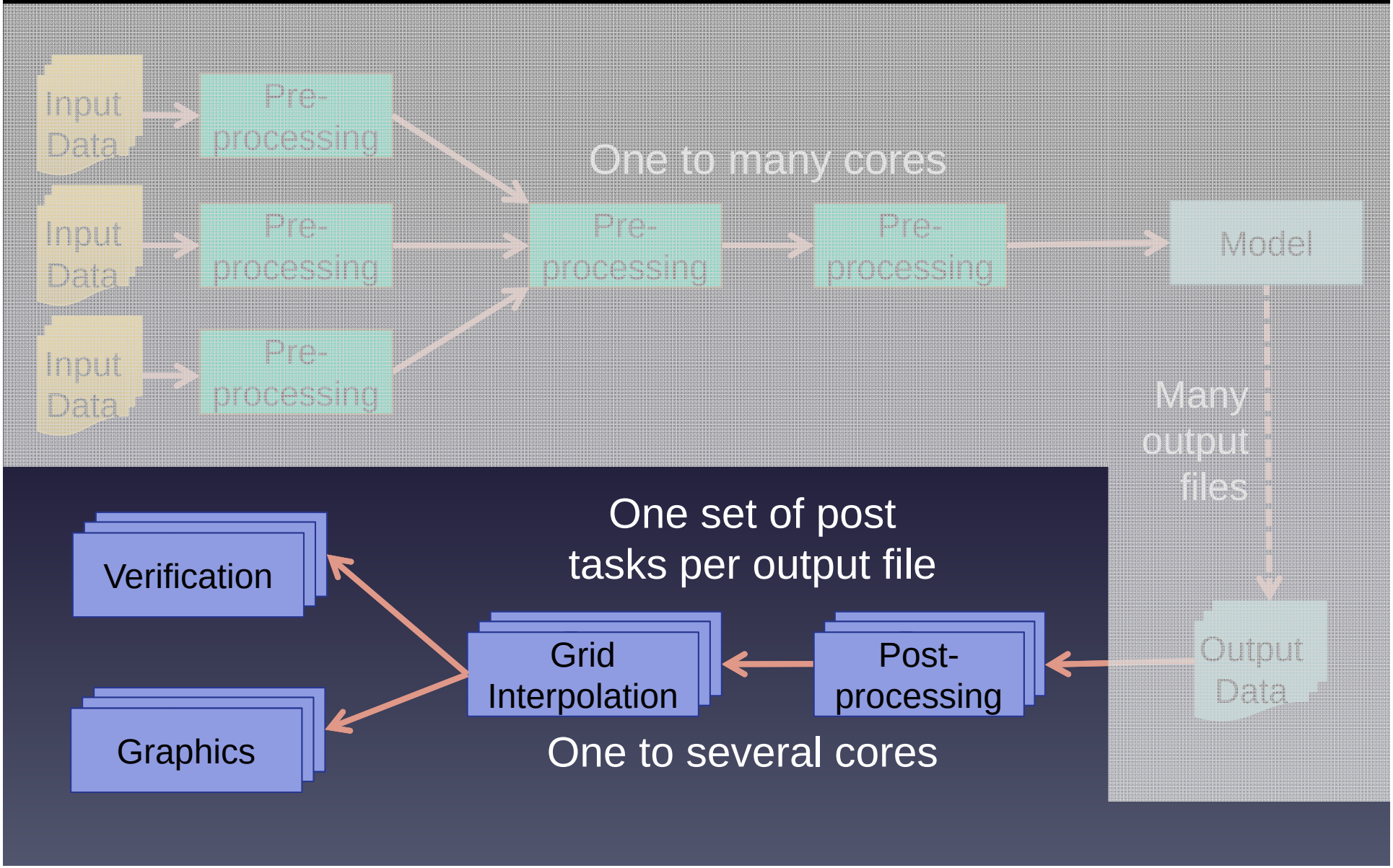
A Typical Workflow



A Typical Workflow

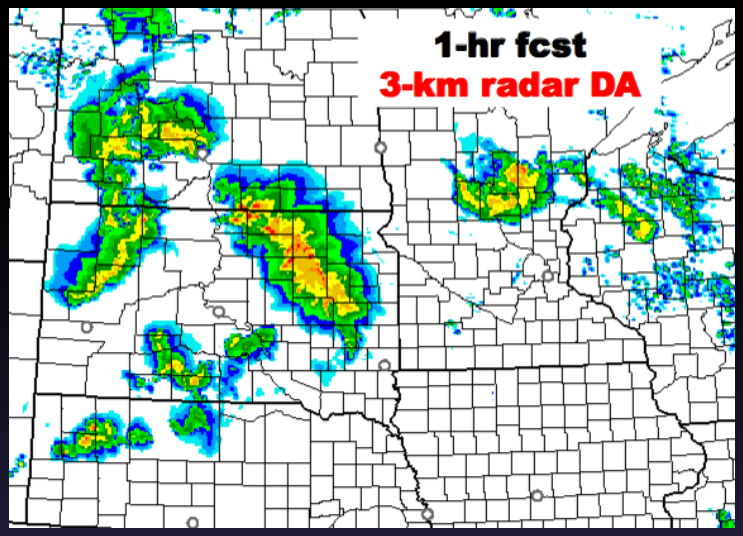


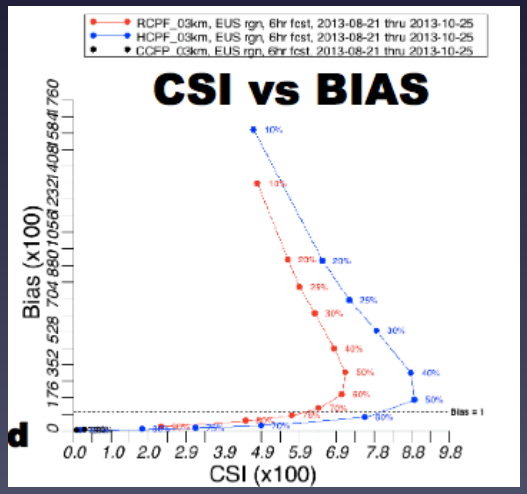
A Typical Workflow



CASE: High Resolution Rapid Refresh

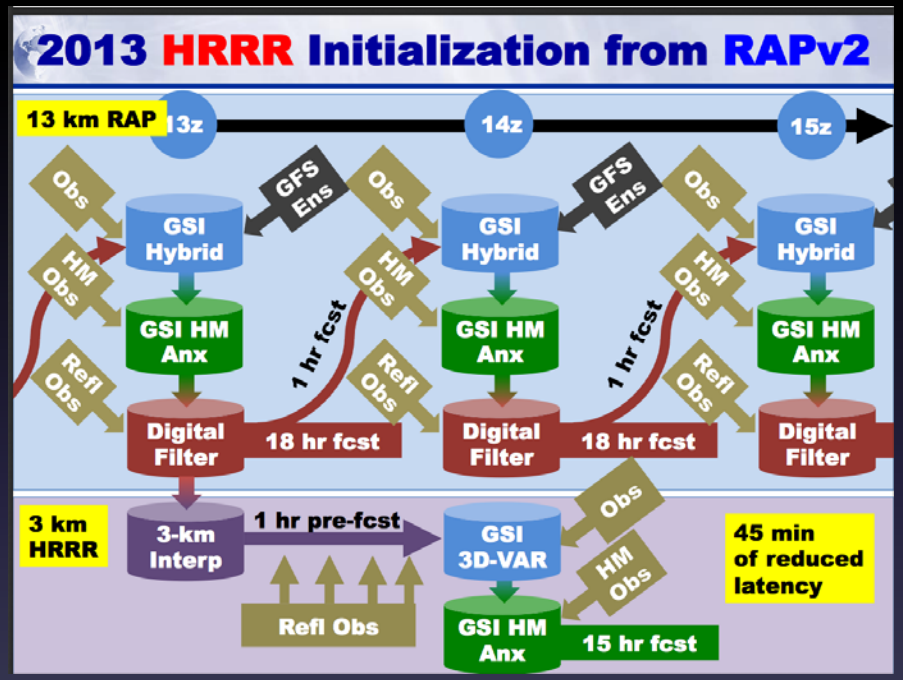
- 15 hour forecast, runs every hour
- 3km resolution
- Continental U.S. domain
- Used in Aviation, Severe Weather, Renewable Energy, Forecasting
- Up to 263 different per run
  - Data Preparation
  - Data Assimilation
  - Model Execution
  - Post Processing and Visualization





CASE: High Resolution Rapid Refresh

- Dependency trees vary depending on start time
- Uses meta-tasks to describe each forecast hour
- Complex dependencies allow workflow to advance in absence of timely data arrival



HRRR was transition to Operations at the National Weather Service in September 2014

Distributed CRON

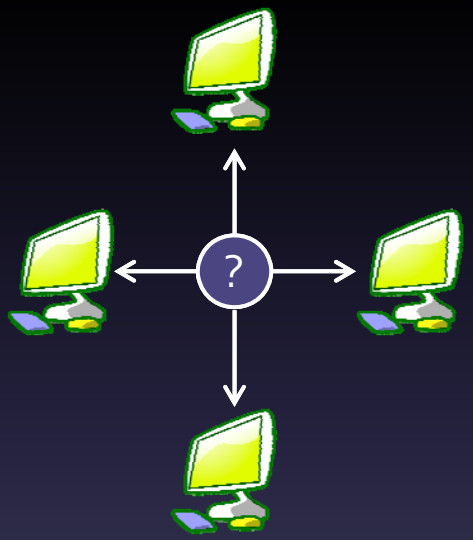
Distributed, Highly-Available, CRON Services

Craig Tierney
Nathan Dauchy

Why we use CRON

- *Weather forecasting is driven by the clock!*
- Model cycles start every 1-6 hours
- Workflow management scripts run every 1-5 minutes
- Input/output data pull/push/sync
- Systems management scripts

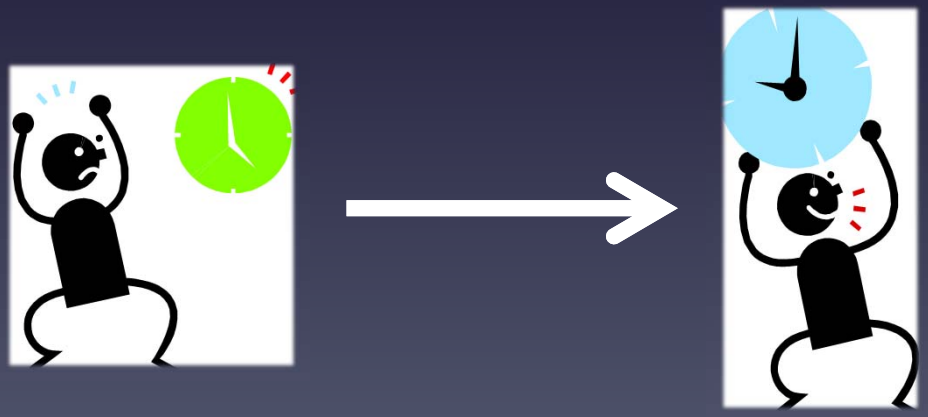
D-CRON – Distributed CRON System



- Provide a unified crontab across the system
- Distribute cron tasks multiple systems
- Peer-to-peer reliability daemon
- Functionality is transparent to the users

D-CRON Secondary Benefits

- Less help tickets about why their workflows did not start or complete
- No more questions about "lost" crontabs
- No longer need to monitor and maintain individual front-end nodes by operations staff



How D-CRON Works

How D-CRON Works

User creates a crontab entry.

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

The user crontab is transparently modified to work with the D-CRON system

How D-CRON Works

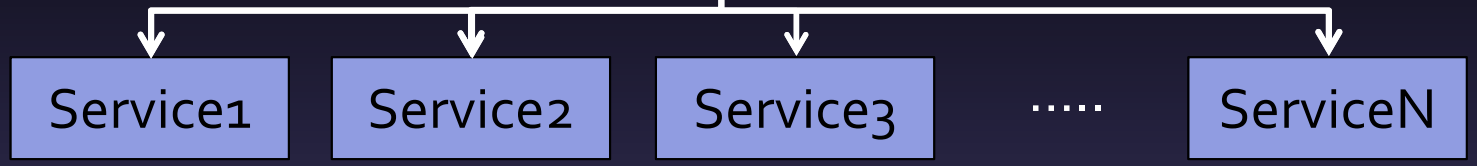
```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Prior to each scheduling iteration,
status of all service nodes is checked.

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

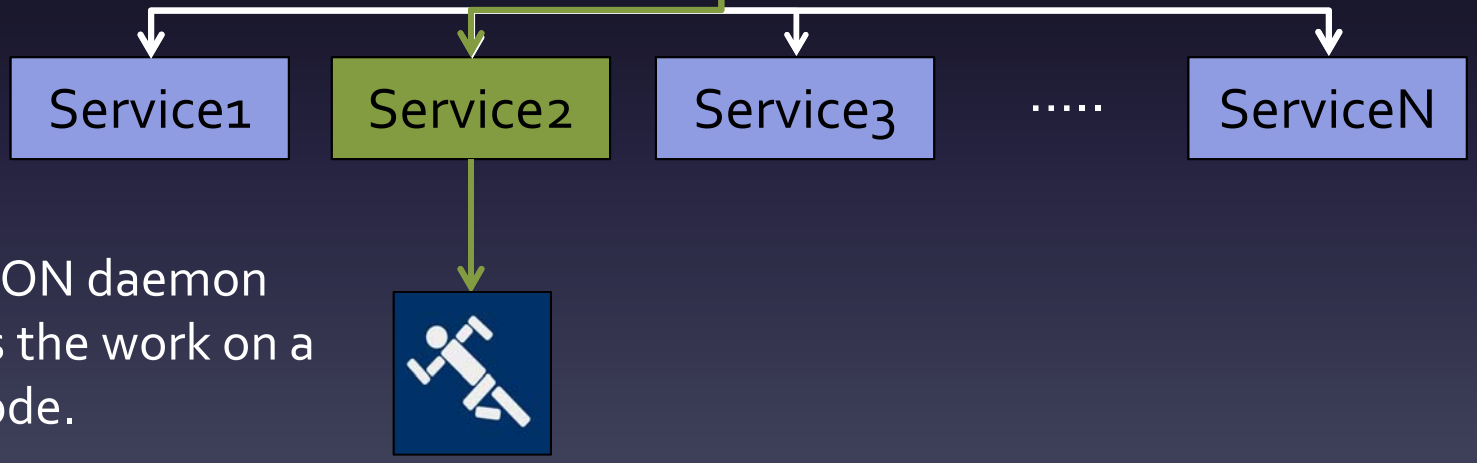
Work is distributed to all service nodes.



How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Hash function is used to determine which node does the work.



Local CRON daemon executes the work on a single node.

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Hash function is used to determine which node does the work.

Local CRON daemon executes the work on a single node.

Work will always be scheduled on the same node unless there is a issue with the service node.

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Prior to each scheduling iteration,
status of all service nodes is checked.

How D-CRON Works

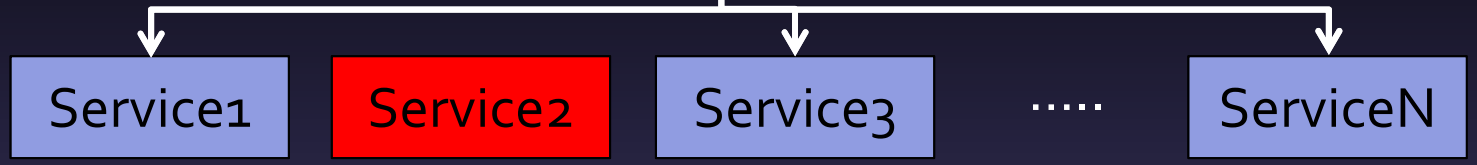
```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Prior to each scheduling iteration,
status of all service nodes is checked.

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

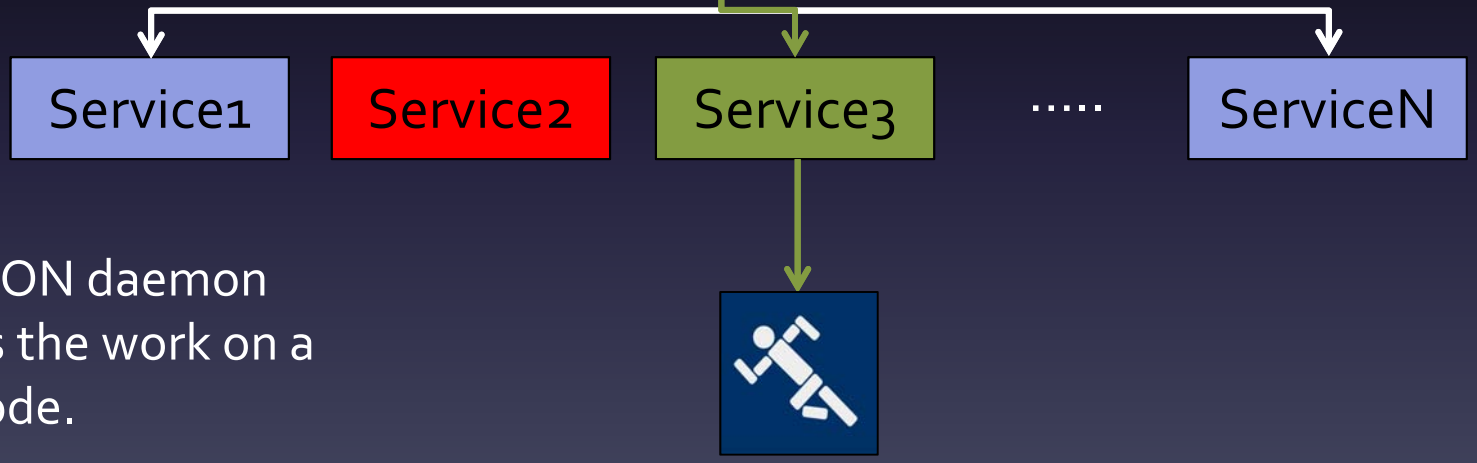
Work is farmed to all service nodes.



How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Hash function is used to determine which node does the work.

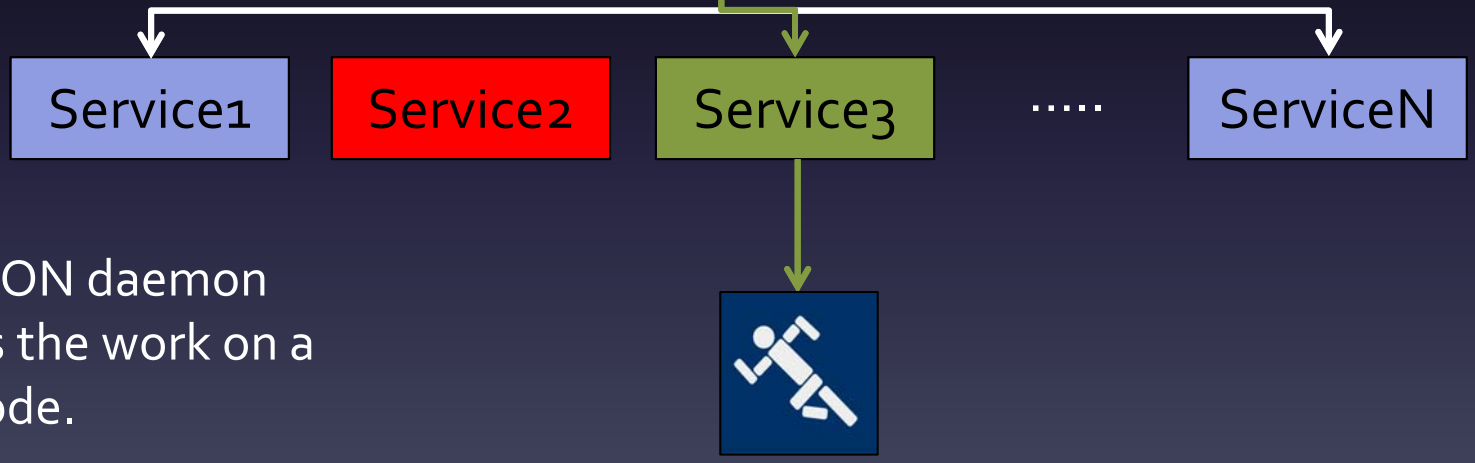


Local CRON daemon executes the work on a single node.

How D-CRON Works

```
# Realtime FIM run
0-57/3 * * * * rocotorun -w FIMG8UJET.xml -d FIMG8UJET.db
```

Hash function is used to determine which node does the work.



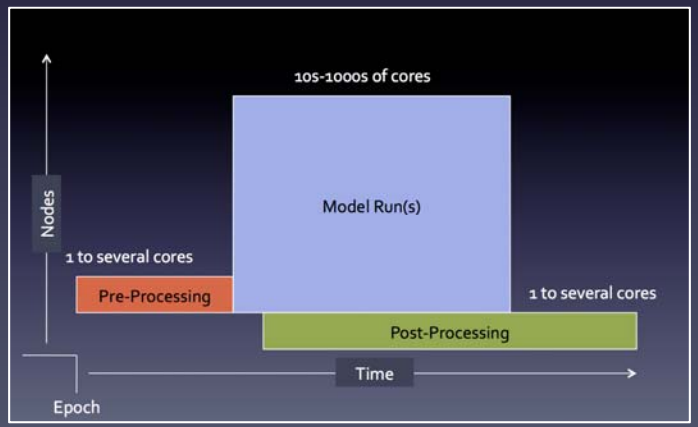
Local CRON daemon executes the work on a single node.

Work will always be scheduled on the same node unless there is a issue with the service node.

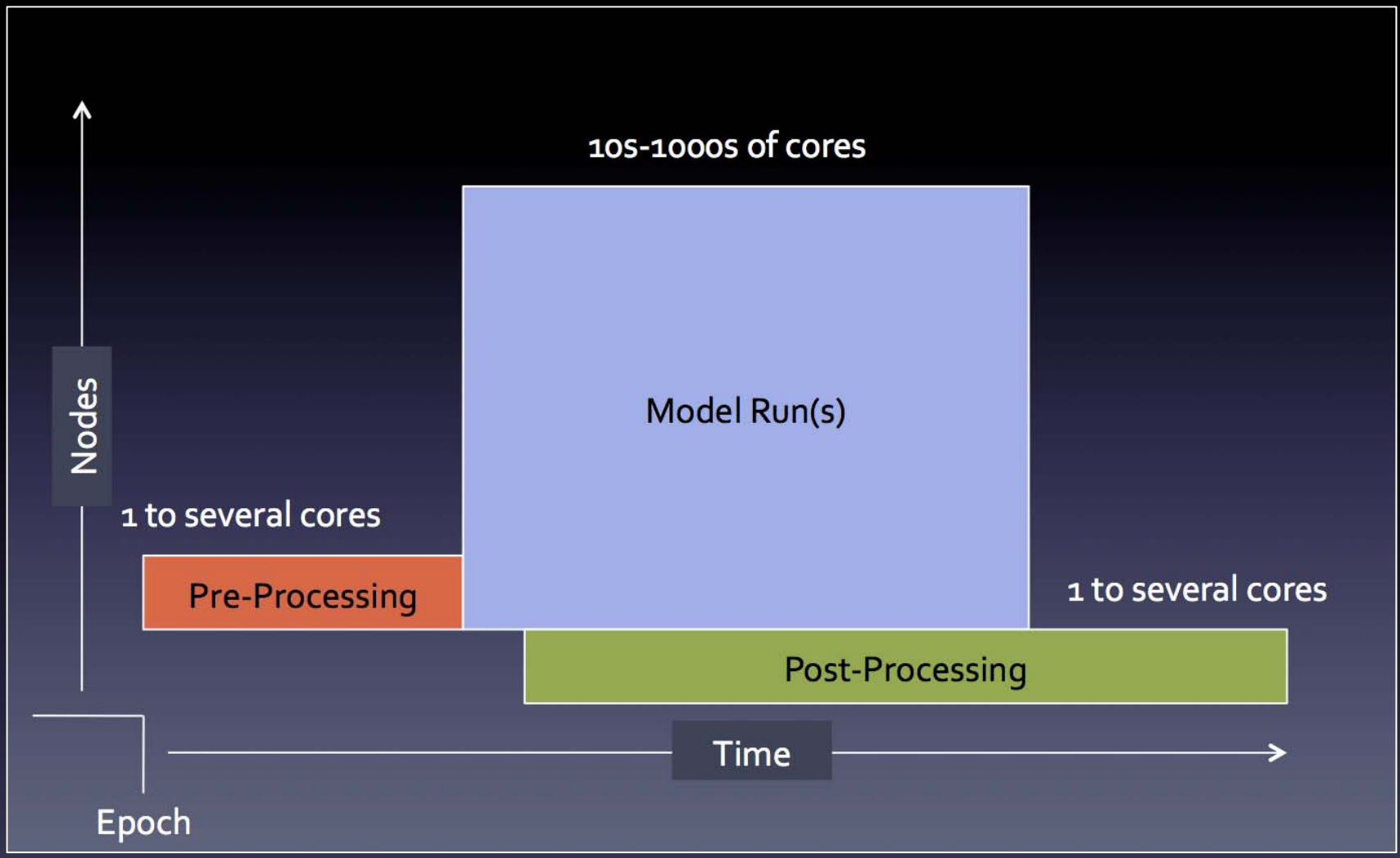
How D-CRON is used

- 81522 CRON tasks launched daily on Jet
  - Versus 48140 batch jobs (Sept. 2014)
- 123785 CRON tasks launched daily on Zeus
  - versus 80239 batch Jobs (Sept. 2014)

Distributed CRON



Standing Reservations

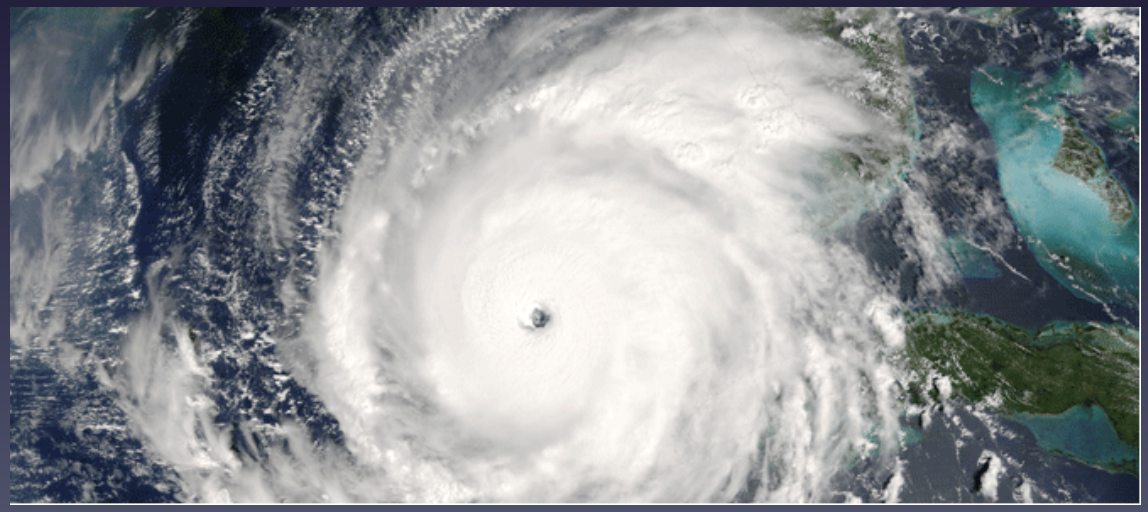


Guaranteeing Resources for Real-Time Experiments

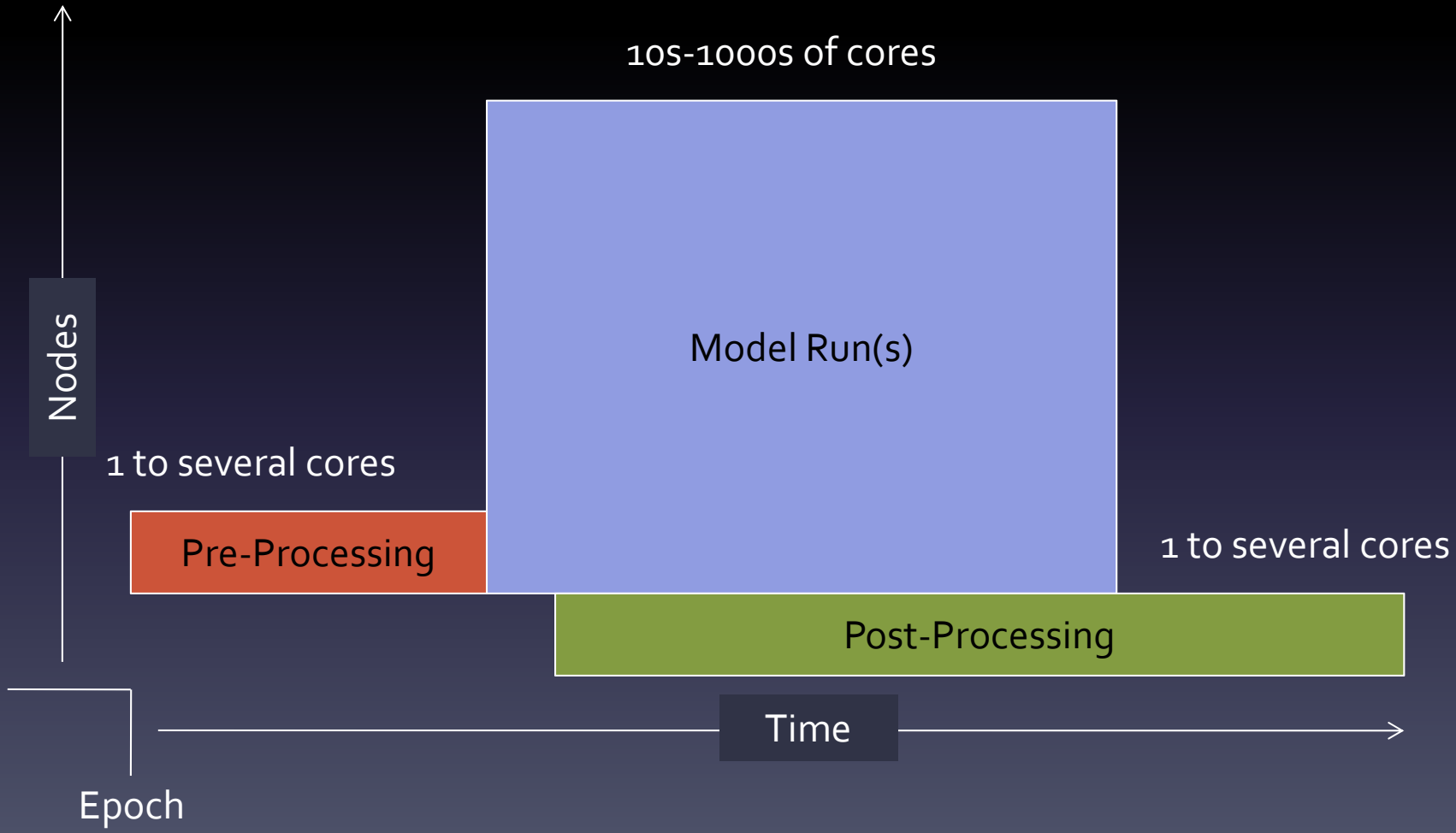
Craig Tierney, CIRES
Christopher Harrop, CIRES

Standing Reservations

- Pre-allocated blocks of system that guarantee availability
- Finite reservation
  - Can be release by user when not needed
- Infinite reservation

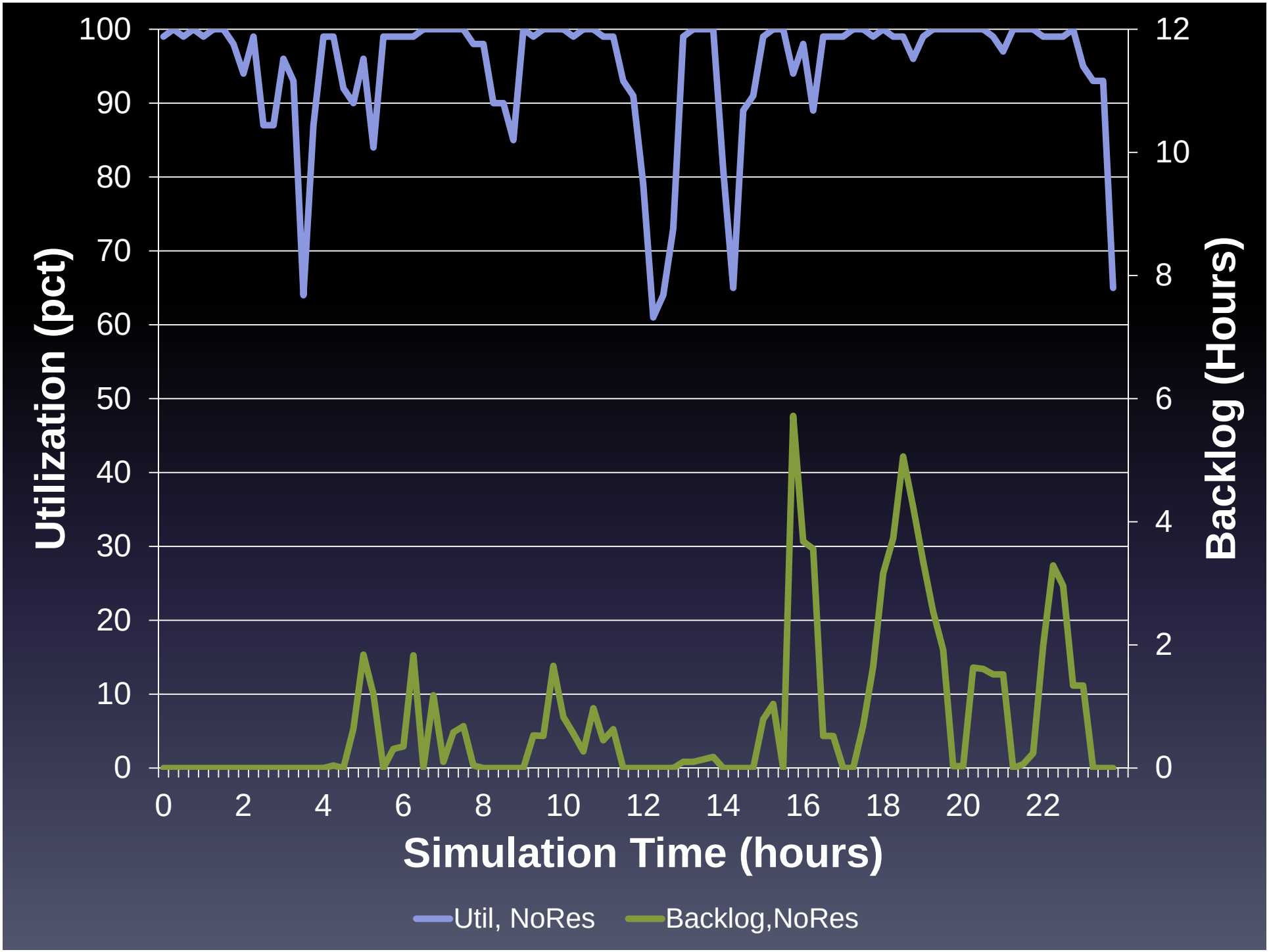


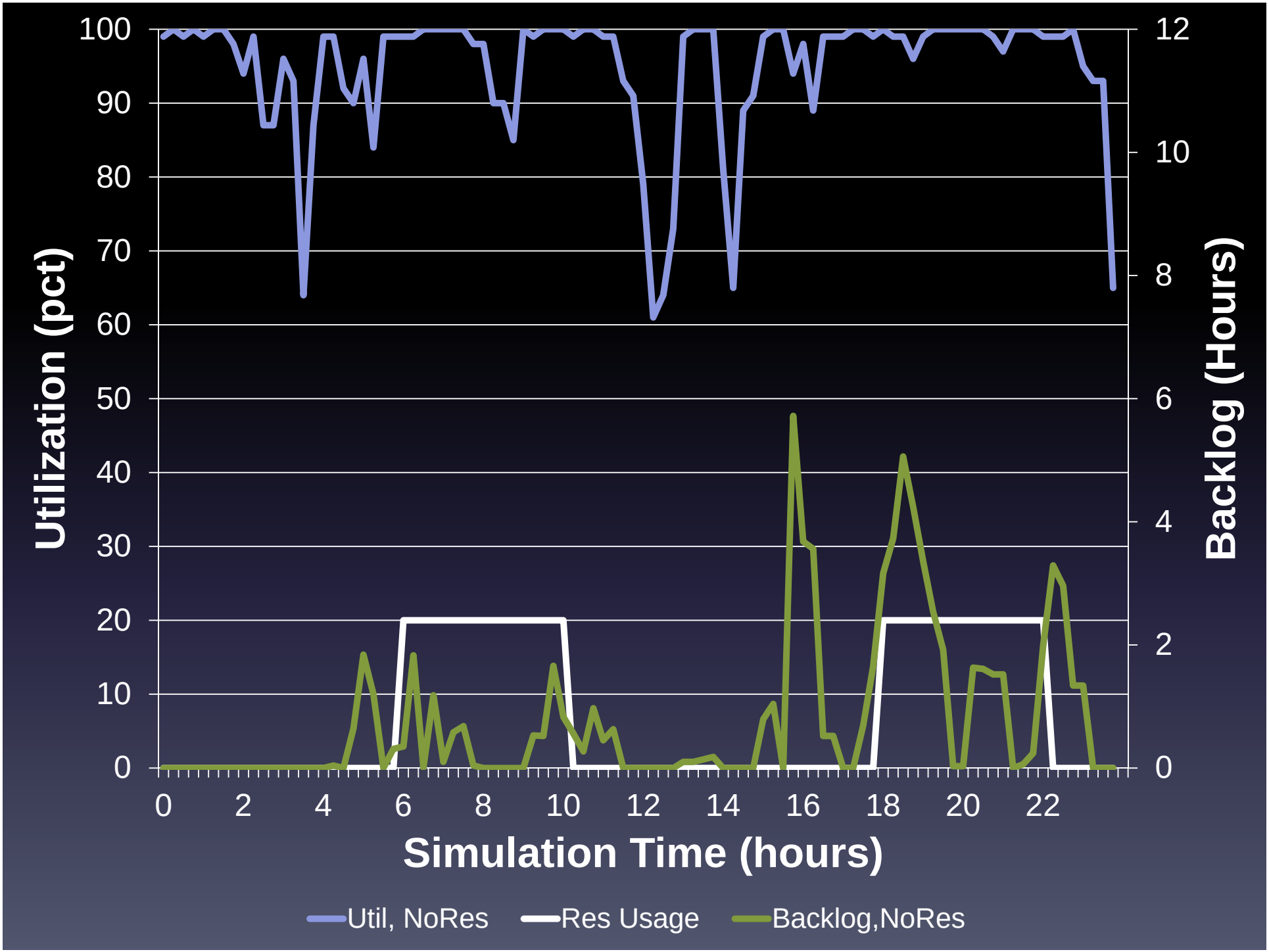
Typical Standing Reservations

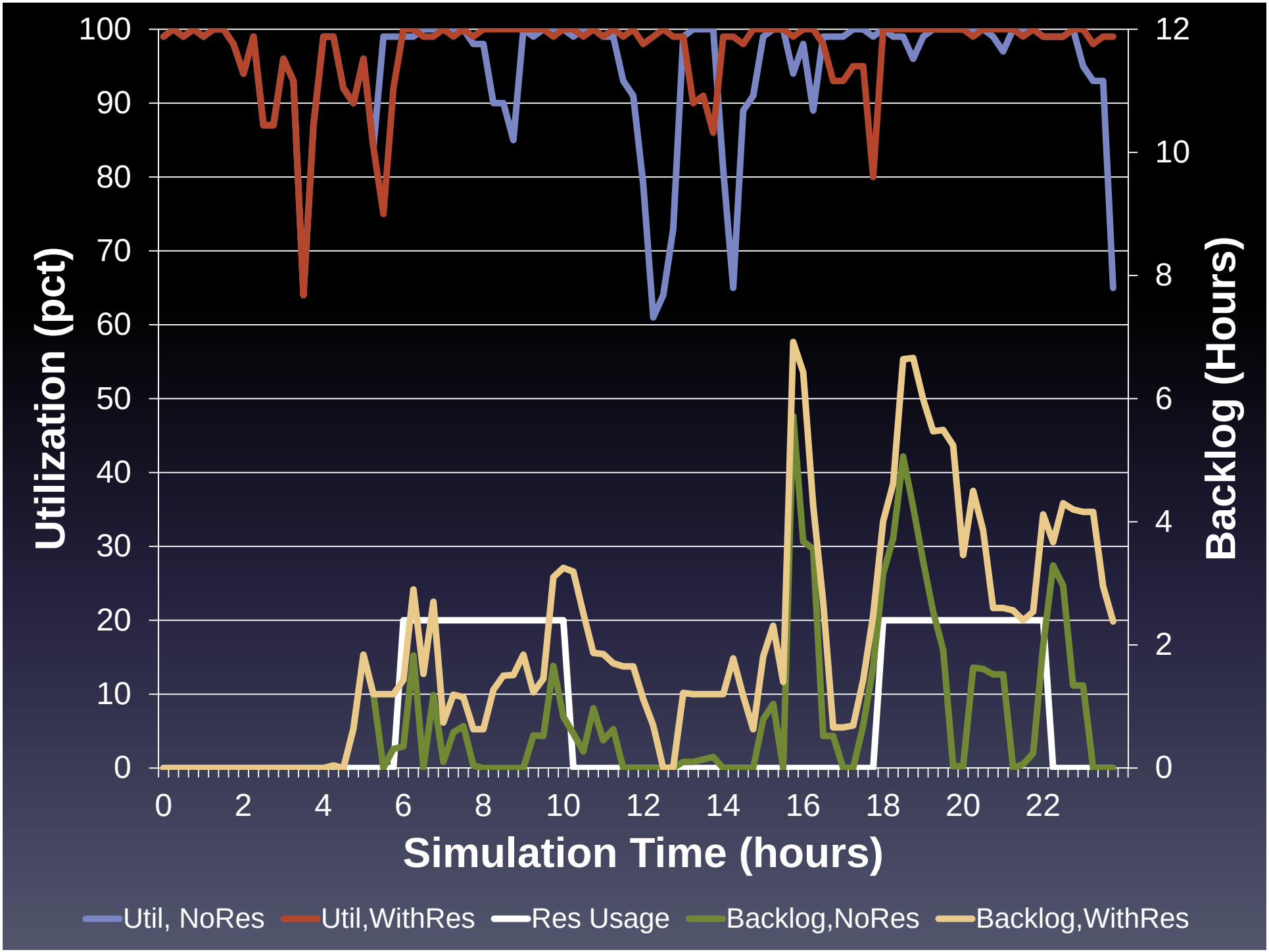


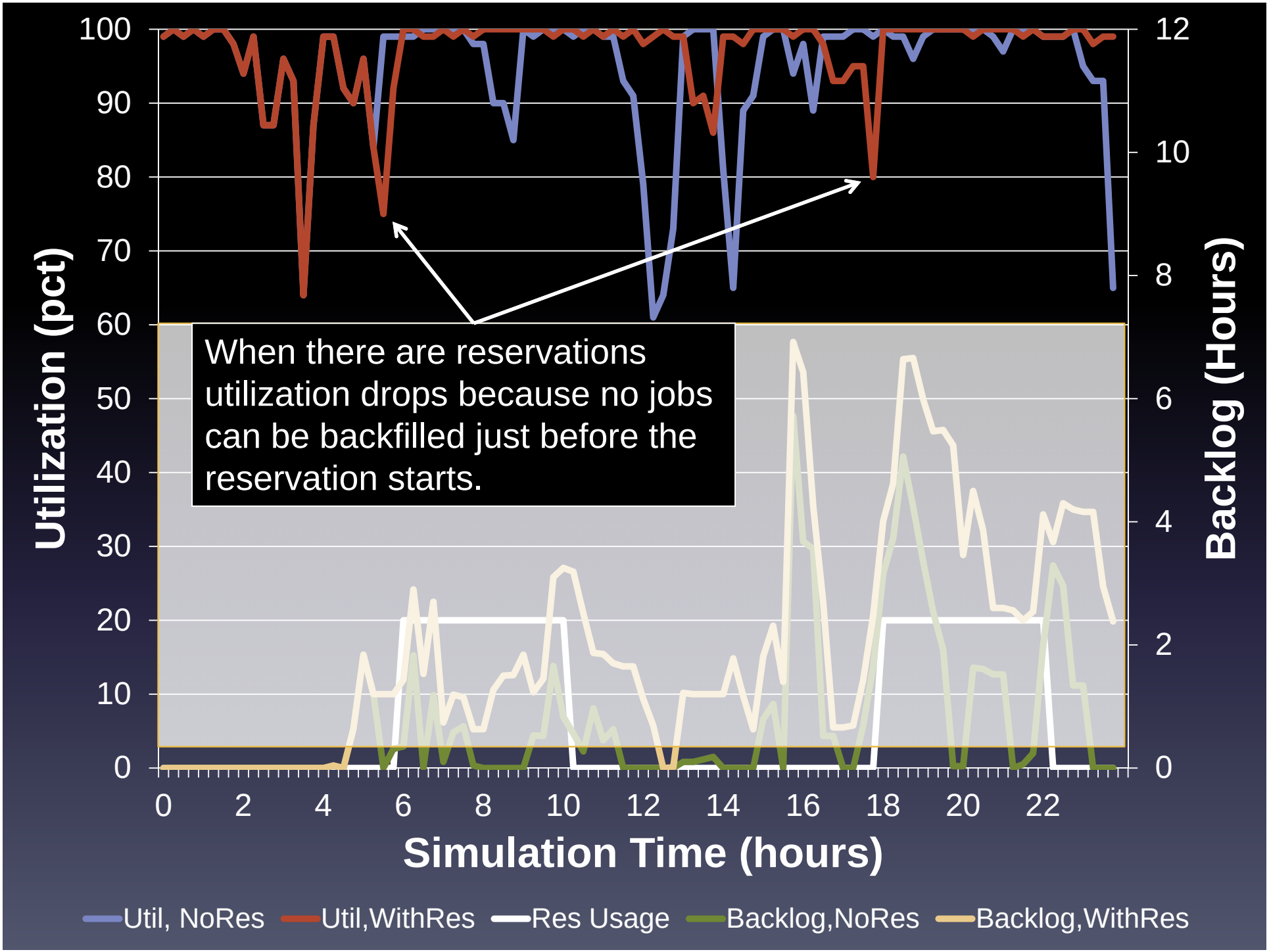
Infinite Reservations (IR)

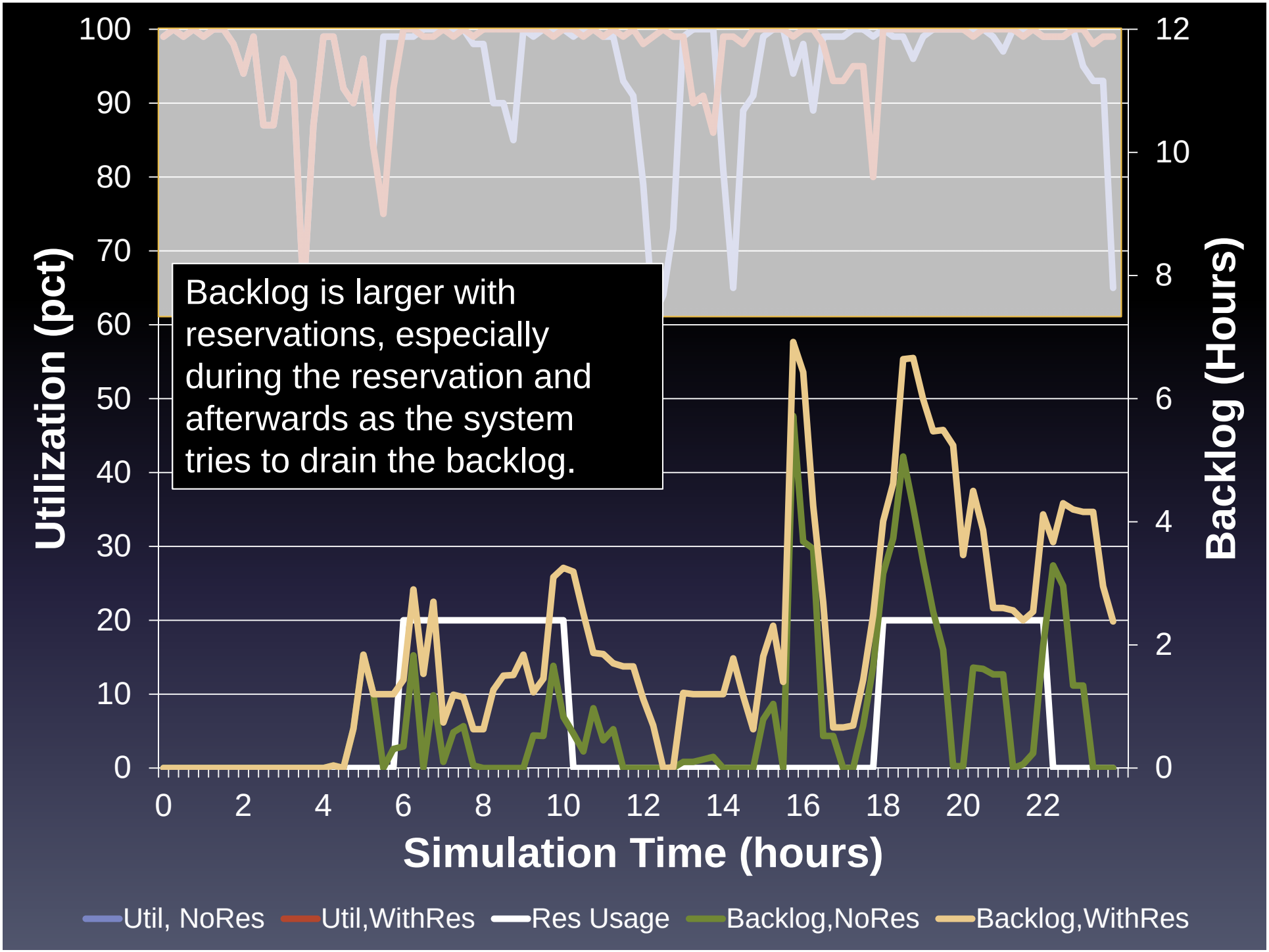
- No end time
- Required when models cannot cold-start
- On our system, IR are stressful on the system
  - Causes problems with the scheduler
  - Often blocks unused resources to non-realtime jobs
- In 2014, we moved to a system based on preemption
  - Reduce stress on the system
  - Allowed for more non-realtime work


Utilization (pct)
Backlog (Hours)
Simulation Time (hours)
100
90
80
70
60
50
40
30
20
10
0
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18
20
22
Util, NoRes
Backlog,NoRes


Utilization (pct)
Backlog (Hours)
Simulation Time (hours)
100
90
80
70
60
50
40
30
20
10
0
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18
20
22
Util, NoRes
Res Usage
Backlog,NoRes


Utilization (pct)
Backlog (Hours)
Simulation Time (hours)
100
90
80
70
60
50
40
30
20
10
0
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18
20
22
Util, NoRes
Util,WithRes
Res Usage
Backlog,NoRes
Backlog,WithRes


100
90
80
70
60
50
40
30
20
10
0
Utilization (pct)
12
10
8
6
4
2
0
Backlog (Hours)
When there are reservations utilization drops because no jobs can be backfilled just before the reservation starts.
0
2
4
6
8
10
12
14
16
18
20
22
Simulation Time (hours)
Util, NoRes
Util,WithRes
Res Usage
Backlog,NoRes
Backlog,WithRes


Backlog is larger with reservations, especially during the reservation and afterwards as the system tries to drain the backlog.
Utilization (pct)
Backlog (Hours)
Simulation Time (hours)
100
90
80
70
60
50
40
30
20
10
0
12
10
8
6
4
2
0
0
2
4
6
8
10
12
14
16
18
20
22
Util, NoRes
Util,WithRes
Res Usage
Backlog,NoRes
Backlog,WithRes

Current Reservation Usage

- 2014 Hurricane Season
  - 25196 total cores
  - 105 reservations per day, 50% of total core hours
  - Maximum of 8332 cores available via preemption (33% of available core hours)

83% of total resources under reservation/preemption

Summary

- Portable and resilient workflow management allows us to reliably complete experiments
- Extending CRON services to be distributed improves fault-tolerance and reduces support requirements
- Using standing reservations allows real-time experiments to reliably finish in traditional R&D HPC environments

Contact:
Craig.Tierney@noaa.gov

Rocoto:
http://rdhpcs.noaa.gov/rocoto/
Christopher.W.Harrop@noaa.gov

Backup Slides