

Best Practices for Queuing and Running Jobs on Theta

Adrian Pope
Christopher Knight
Misha Salim

Outline

<https://www.alcf.anl.gov/user-guides>

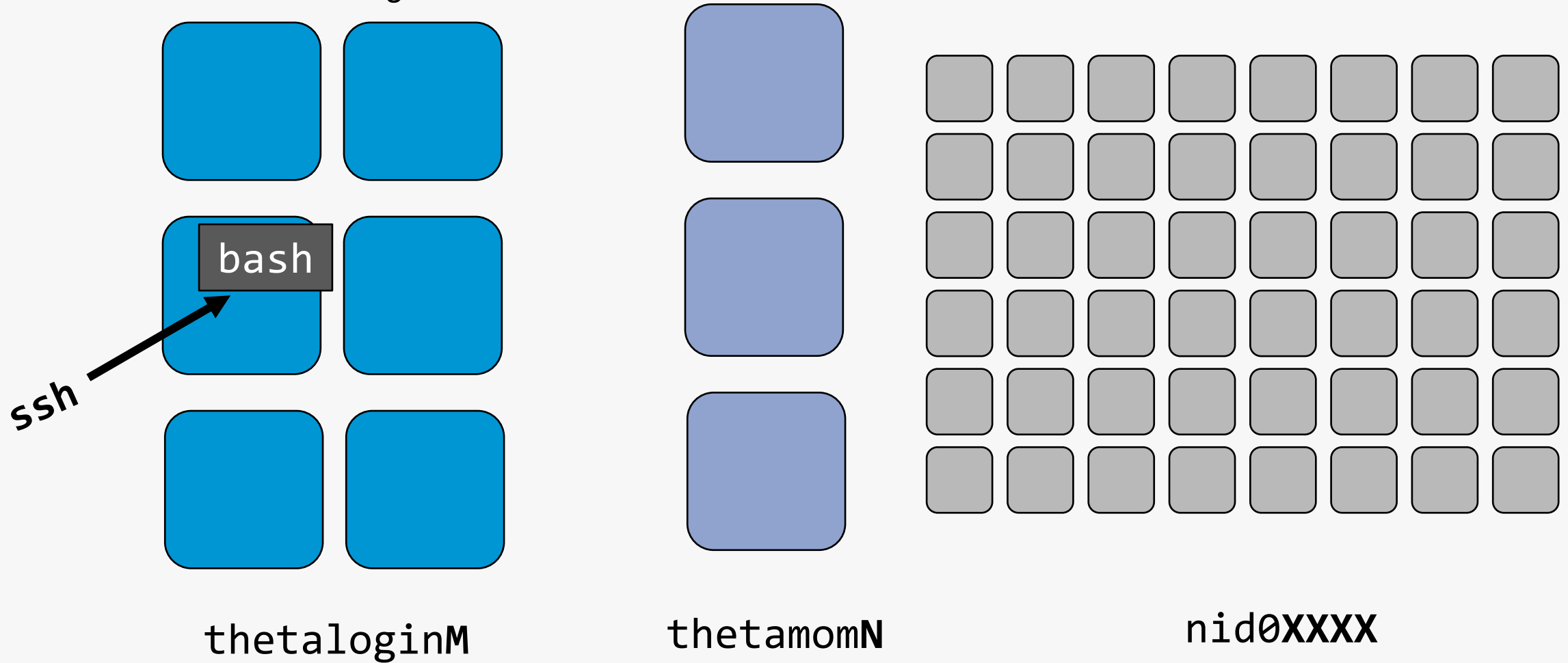
- Scheduling Policies & Cobalt
 - Job Priorities
 - Cobalt attributes
- Tips for Short & Interactive Jobs
 - General tips for submitting jobs
 - How to find idle nodes
 - Interactive jobs
- Workflows and Ensembles
 - Tips for simple ensembles
 - Balsam



Section I: Scheduling Policies & Cobalt

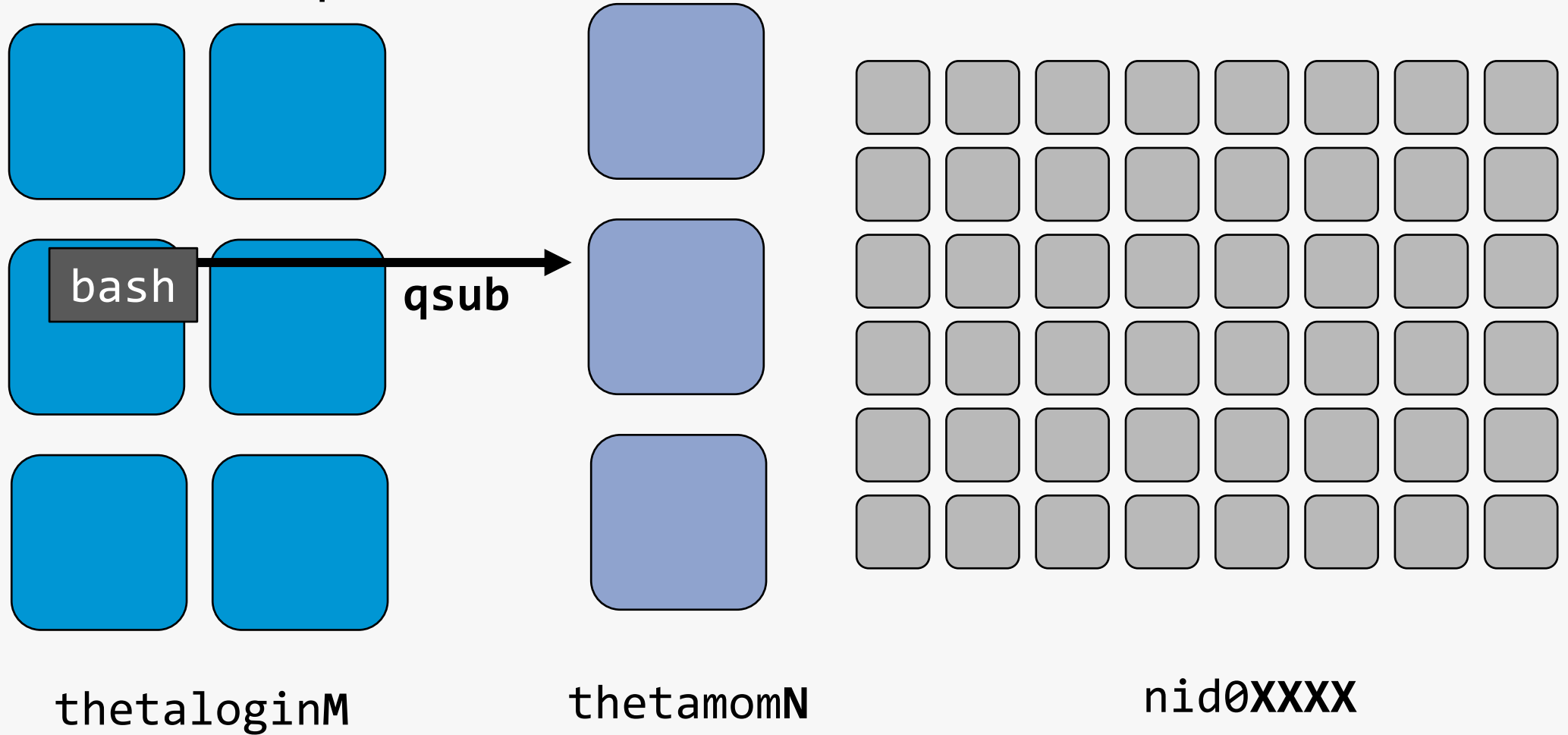
Theta System Overview

Users SSH to Theta Login Nodes



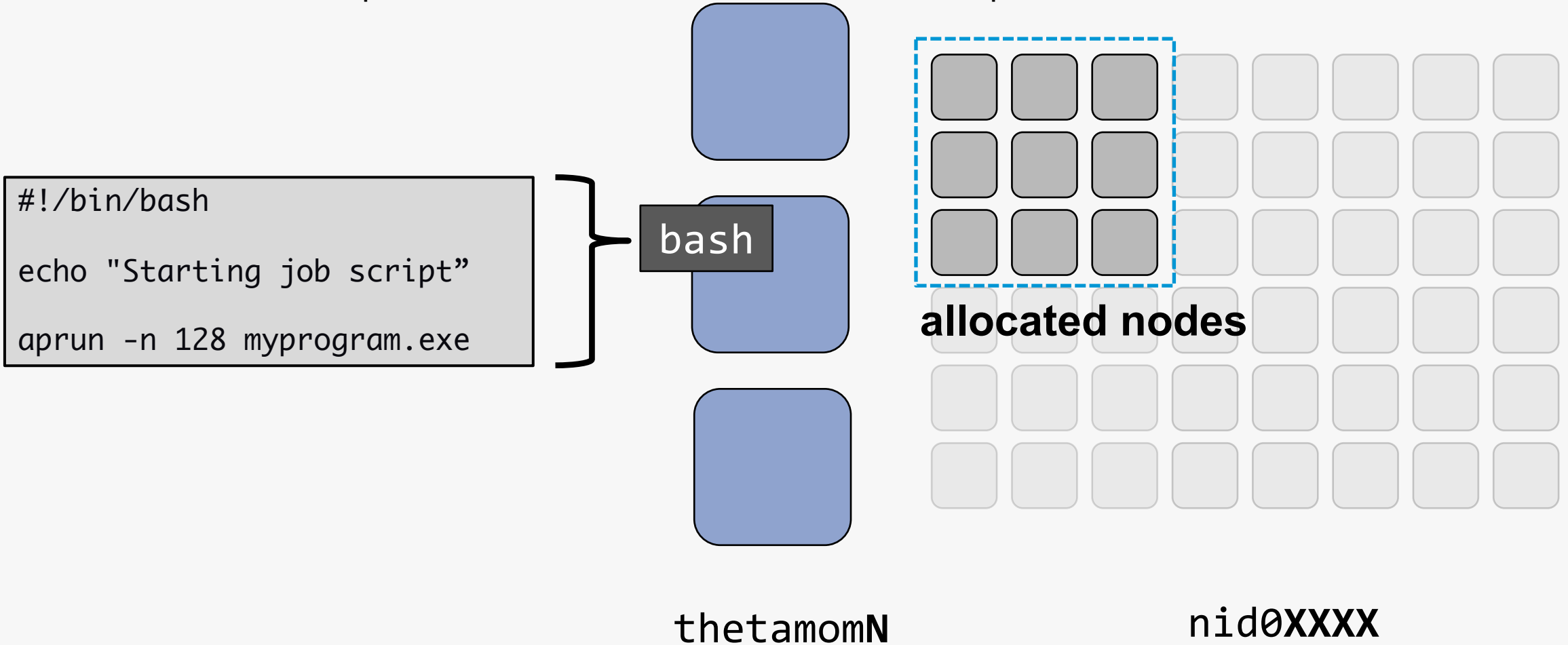
Theta System Overview

Submit Jobs to Cobalt via `qsub`



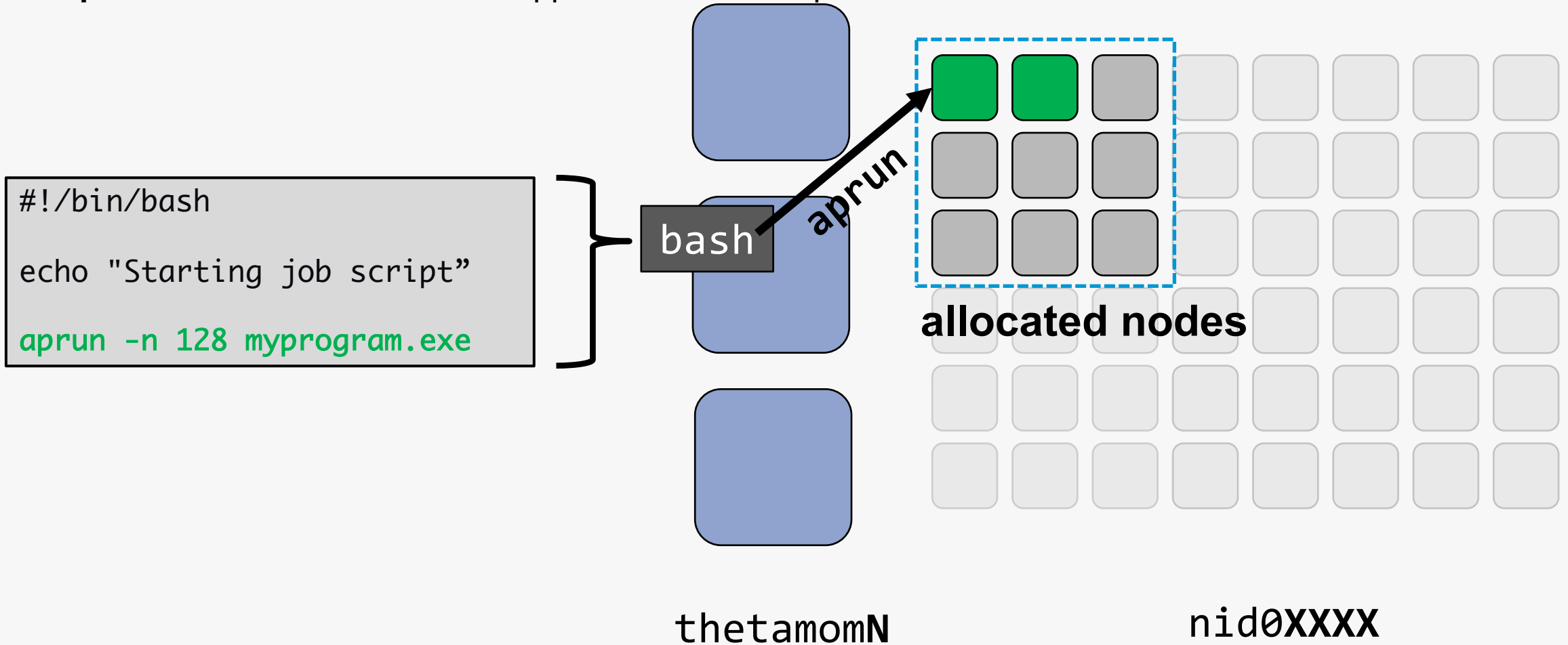
Theta System Overview

Cobalt allocates compute nodes and launches the submitted script



Theta System Overview

`aprun` on MOM node launches application onto compute nodes



Submitting and Running Jobs on Theta

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

<https://www.alcf.anl.gov/support-center/theta/running-jobs-and-submission-scripts>

- **Cobalt** (ALCF)
 - Manages job queue, decides what job will run on which nodes when
 - User writes a job script, submits script to queue, script eventually runs on thetamom node
 - Commands: qsub, qstat, qdel, qalter, qhold, qrls

<https://www.alcf.anl.gov/support-center/theta/theta-memory-modes>

<https://www.alcf.anl.gov/support-center/theta/affinity-theta>

- **ALPS** (Cray)
 - aprun in job script runs on thetamom, launches executable onto compute nodes
 - MPI details, processor affinity, etc.

Theta - Submitting Script Jobs

- Executable is invoked within script (bash, csh, ...)
- `aprun` is used to launch executables on compute nodes

```
> cat myscript.sh
```

```
#!/bin/sh
```

```
#COBALT -A <project_name> -t 10 -n 16 -O <prefix_name> -q default
```

```
#COBALT --attrs mcdram=cache:numa=quad
```

```
echo "Starting Cobalt job script"
```

```
aprun -n 1024 -N 64 -d 1 -j 1 --cc depth <app> <app_args>
```

MPI Ranks

Ranks per node

Affinity

Memory Mode

```
> qsub myscript.sh
```

```
123456
```

Cobalt: notable qsub attributes

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

<https://www.alcf.anl.gov/support-center/theta/running-jobs-and-submission-scripts>

- Specify memory mode: `--attrs mcdram=<mcdram_mode>:numa=<numa_mode>`
 - Default is cache/quad: `--attrs mcdram=cache:numa=quad`
- Submit to specific nodes: `--attrs location=<list_of_nodes>`
- Enable access to SSDs: `--attrs ssds=required:ssd_size=<size_in_GB>`
 - Current maximum is 128 GB: `--attrs ssds=required:ssd_size=128`
- Enable SSH access to nodes: `--attrs enable_ssh=1`
 - Retrieve compute node ids from `$COBALT_PARTNAME`
 - Prepend node id with 'nid' and zeroes to span 5-digits
 - SSH to compute node from MOM node: `ssh nid00001`

Theta - Now That Your Job Is Queued

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

- Check status of submitted jobs with `qstat`
- Format of output can be customized with `--header`

```
> qstat --header JobID:User:WallTime:Nodes:State:Queue
```

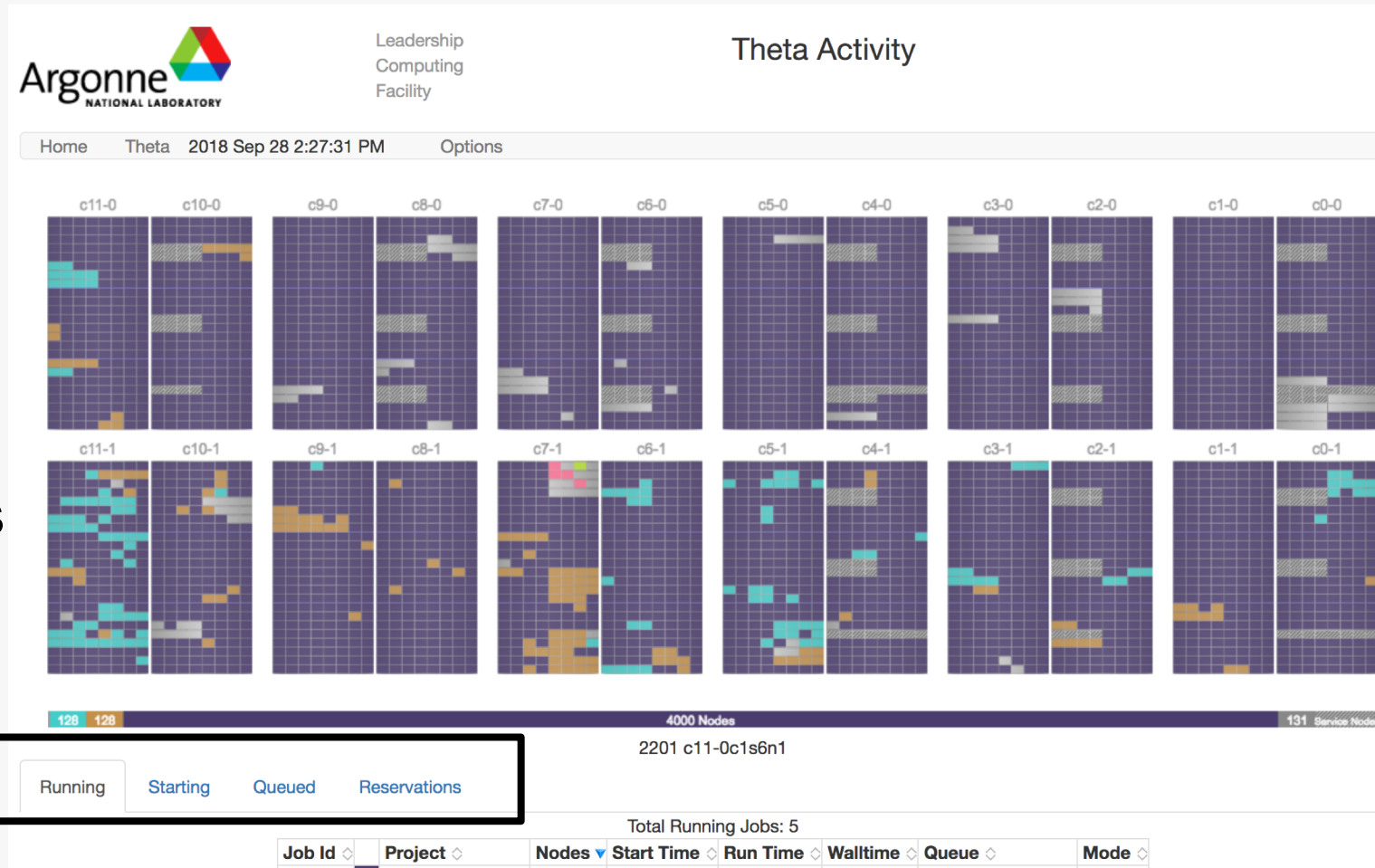
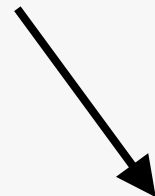
JobID	User	WallTime	Nodes	State	Queue
123456	user1	24:00:00	4000	running	default
123457	user2	03:00:00	2048	queued	default
123458	user3	03:00:00	128	running	default
123459	user1	00:30:00	1	running	debug-cache-quad
123460	user4	00:30:00	64	queued	training

Theta - Checking Status of Job

<https://www.alcf.anl.gov/support-center/theta/gronkulator-job-status-display>

<https://status.alcf.anl.gov/theta/activity>

Running
Starting
Queued
Reservations



Theta's Queues

<https://www.alcf.anl.gov/support-center/theta/job-scheduling-policy-theta>

- **debug-cache-quad**: 16 nodes, max size 8, max length 1 hr, max queued 1 job
- **debug-flat-quad**: 16 nodes, max size 8, max length 1 hr, max queued 1 job
- **default**: production jobs (see next slide)
- **backfill**: unexpired projects that have exhausted their allocation can still submit jobs
 - Usually has access to same nodes as default
 - Always lower priority than default jobs
 - Will run when there are scheduling windows that no default job can use
 - Shorter jobs have a better chance of running sooner

Default (Production) Queue

<https://www.alcf.anl.gov/support-center/theta/job-scheduling-policy-theta>

- Maximum 20 jobs queued per user
- Maximum 10 jobs running per user
- Minimum wall clock 30 minutes (0:30:00 hours)
- Maximum wall clock
 - node count \geq 128 nodes (minimum allocation): maximum 3:00:00 hours
 - node count \geq 256 nodes : maximum 6:00:00 hours
 - node count \geq 384 nodes : maximum 9:00:00 hours
 - node count \geq 640 nodes : maximum 12:00:00 hours
 - node count \geq 802 nodes : maximum 24:00:00 hours

Cobalt: Scheduler and Resource Manager

- **Queue:** After a job is submitted, it enters a queue of jobs waiting to use the system
- **Cobalt:** Manages the queue and determines which jobs will run on which nodes and when
- **Score:** Jobs are ordered in the queue by the score they have accrued, greatest to least
- **Wait Time:** Can be roughly divided into 2 factors
 - **Rate** that a job accrues score according to the score function (see next slide)
 - **Score required** for a job of a particular size/length to run in current queue conditions
 - Large jobs usually need to get near the top of the queue to run
 - Small/Short jobs can sometimes find windows to run in between larger/longer jobs and do not always need to get to the top of the queue
 - Deeper queues generally result in larger scores being accrued before jobs run

Cobalt: Score Function

- **Initial Score:** same value for all jobs in default queue (currently 51.0)
- **Functional Form:** scores for all jobs have the same functional dependence on time
 - Super-linear in time, dS/dt starts very small and increases
 - Jobs initially gain score very slowly, but will “build up steam” later
- **Coefficient** in front of time function depends on job parameters
 - Jobs with larger coefficients will start at same initial value but rise faster and pass other jobs
 - **Project type:** INCITE/ALCC accrue score more quickly than DD
 - **Number of nodes:** Larger jobs accrue score more quickly
 - Encourages leadership computing
 - **Wall clock:** Shorter jobs accrue score more quickly
 - Encourages users to accurately estimate job times, which helps scheduling efficiency
- **Holds:** Score and dS/dt are frozen when jobs are on hold (*user_hold*, *dep_hold*, *admin_hold*)
 - Score and dS/dt are functions of **EligibleWaitTime** = QueuedTime - HoldTime
 - Tools show **QueuedTime**, scores can look confusing for jobs that have been on hold

Deep Queues

- **Depth:** Theta's queue has been historically deep by ALCF standards since ~May 2020
 - Typically ~5-10x deeper than in 2019
- **Estimate** how long jobs are waiting as a function of size/length from queued jobs
 - `qstat` or status webpage for current queue, `sbank` to look up job histories
 - Project type - Adrian estimates that a DD job waits ~2x as long as equivalent INCITE/ALCC
- **Optimizing** overall project throughput is subject to particular constraints of project
 - **Large** jobs accrue score more quickly, and use many core-hrs per job
 - **Small** jobs might have lower latency when they can find windows, but shorter wall clock available and fewer core-hrs per job
 - The middle-ground and cross-overs are not always clear
- **Job Organization Tools:** ensembles, dependencies, workflows (see following slides)
- **Help/Advice**
 - INCITE/ADSP projects have catalysts, ALCC and DD projects may have an ALCF contact
 - Email support@alcf.anl.gov if you get stuck

Ensembles: Overview

<https://www.alcf.anl.gov/support-center/theta/running-jobs-and-submission-scripts>

- **Efficiency:** pack jobs together and wait in the queue once
 - Simple cases can be handled directly in an “ensemble” cobalt script
- **Serial**
 - Situation: a number of jobs that are similar in size that are short compared to wall clock limit
 - Solution: run a sequence of aprun in the same script
- **Parallel**
 - Situation: a number of independent jobs that are similar in length
 - Solution: run multiple aprun at the same time
 - Benefit: bundling small jobs together into a larger job can access longer wall clock limits
- **Complexity**
 - Can mix strategies in a single cobalt script
 - Workflow might be better for complicated mixes of sizes/lengths and dependencies

Ensembles: Serial

<https://www.alcf.anl.gov/support-center/theta/running-jobs-and-submission-scripts>

- Straightforward shell script
- Later aprun can depend on output of previous

```
#!/bin/bash  
echo "Starting Cobalt job script"  
aprun -n 128 -N 64 myprogram.exe arg1  
aprun -n 128 -N 64 myprogram.exe arg2  
aprun -n 128 -N 64 myprogram.exe arg3
```

Ensembles: Parallel

<https://www.alcf.anl.gov/support-center/theta/running-jobs-and-submission-scripts>

- Background each aprun so that script can start more apruns before first returns
- “sleep 1” between aprun to allow resource manager to keep track of node use
- ”wait” at end for all backgrounded aprun to finish before exiting script
- Limitations: no more than 1000 aprun at a time; if you need more, use a workflow

```
#!/bin/bash
echo "Starting Cobalt job script"
aprun -n 192 -N 64 myprogram.exe arg1 &      #background
sleep 1                                       #give launch ctrl a second
aprun -n 256 -N 64 myprogram.exe arg1 &
sleep 1
aprun -n 64 -N 64 myprogram.exe arg1 &
wait                                          #wait to finish before exit
```

Dependencies: Overview

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

- **Run a sequence of jobs in a particular order**
 - Example: a single long simulation that requires more wall clock than allowed for an individual job, and the code employs a checkpoint/restart mechanism
- **Dependencies:** Cobalt supports “dependencies” between jobs
 - Next job will only start when job on which it depends has finished
 - Guarantees **order of jobs**, and that only 1 job will run at a time
 - A job in a dependency chain will **inherit some fraction of the score** that the previous job accrued by the time it ran, **decreasing the queue wait time** in between jobs after first job
- **Details:** Knowing some details can help with user experience

Dependencies: Details

- **Step-by-Step**
 - Job 1 is submitted and begins accruing score, the job state is *queued*
 - Job 2 is submitted with a dependency on the Job 1 and is put in *dep_hold* state
 - A job in *dep_hold* state does not accrue score or accumulate eligible wait time
 - Job 1 accrues enough score to run
 - When Job 1 finishes, Job 2 is released from *dep_hold* and inherits a fraction of the score that Job 1 had accrued when it launched (default is usually 50%)
 - Job 2, now in a *queued* state, starts accumulating eligible wait time and accrues score
- **What can go wrong?**
 - If Job 1 **returns a non-zero exit code**, Job 2 will go into a *dep_fail* state
 - The user can release Job 2 from *dep_fail* into a *queued* state, but the **score transfer is lost**
`qrls --dependencies <jobid>`

Dependencies: nofail

Theta: /home/rloy/public/scripts/nofail

- If a job runs out of wall clock and is interrupted by the scheduler, the return code is non-zero, the chain will be interrupted, and score will not be inherited
- If your code has a checkpoint/restart mechanism, you may not care if the first job was interrupted by running out of time since you can restart from last checkpoint
- Cobalt scripts are shell scripts, so you can trap signals and return zero anyway
- “nofail” script is an example of how to do this

```
cp /home/rloy/public/scripts/nofail .
```

```
qsub -t 5 -n 64 --mode script nofail <your_script> arg1 arg2 ...
```

- Not an officially supported product, but has been used successfully
- Any #COBALT lines from <your_script> are not visible to cobalt
 - Put values in command line flags, or edit your copy of nofail to include your #COBALT lines

Special Cases

<https://www.alcf.anl.gov/support-center/theta/machine-reservations-theta>

- Occasionally projects have needs that fall outside of the normal queueing policies
- **Form:** We ask those projects to fill out the form linked above to give us the necessary information to consider their request and determine whether to take any action, and we may need up to 5 business days to decide what we will do
- Potential actions:
 - **Score boosts:** Scores for a few jobs can be boosted to help meet deadlines for conferences, proposals, journal submissions, and similar.
 - **Reservations:** A set of nodes can be reserved in advance for a particular time. Typically requires stronger justification than score boosts because it is more disruptive to scheduling.
 - **Special queues:** If a project will regularly need to run outside normal queue limits, it is sometimes possible to set up a limited-access queue with properties that differ from default. This is rarely used and requires extremely strong justification and approval at a high level in ALCF management.

Maximum Job Size

- Theta nominally has 4360 nodes available for the default queue
- Rarely a node will not show up after maintenance
 - `nodelist | grep default | wc -l`
- Nodes that are “down” are also not available for jobs
 - `nodelist | grep default | grep down | wc -l`
- Reservations may make some default nodes unavailable to other jobs
 - Use the status webpage reservations tab or `showres`
 - Turn the node ranges into a count with a command line utility on Theta:
`/soft/cobalt/tools/expand-nodes.py 3356-3823,3840-4371 | wc -w`
- COVID-19 research currently has a standing reservation of 260 nodes, in which case the baseline for maximum job size is 4100 nodes
- Cobalt allows users to queue jobs up to 4360 nodes to default even if some nodes are down or there is a very long reservation, so very large jobs can appear to get stuck at the top of the queue if not enough resources are available

Other Policies Relevant to Project Planning

<https://www.alcf.anl.gov/support-center/theta/job-scheduling-policy-theta>

- **Overburn (INCITE/ALCC)**
 - Projects can use up to 125% of the allocation for capability-scale jobs (802+ nodes)
 - Sub-capability jobs will go to backfill when a project reaches 100%
 - All jobs will go to backfill when a project reaches 125%
 - Available for the first 11 months of an allocation year
 - INCITE: Jan 1 – Nov 30
 - ALCC: July 1 – May 31



ANY QUESTIONS?

Section II: Tips for Short and Interactive Jobs

Preparing to Submit Job

<https://www.alcf.anl.gov/user-guides/allocation-accounting-sbank>

- Check available disk space
 - \$HOME directory: [myquota](#)
 - Project directories: [myprojectquotas](#)
 - Project directories should be used for production work
- Check that your project has core-hours available
 - Use [sbank](#) command to query allocation details
 - Allocation available to project: [sbank l a -p <project_name>](#)
 - Charges against project by user: [sbank l u -p <project_name> -u <user>](#)
 - Charges on Theta are based on number of nodes
 - Jobs smaller than 128 nodes are allocated 128 nodes

Why Hasn't My Job Started?

- There is a reservation which delays your job from starting
 - List all reservations currently in place: [showres](#)
- Job on Theta is in "starting" state; nodes being rebooted into memory mode requested.
- There are no available nodes for the requested queue
 - Nodes may be down, busy running other jobs, draining next job, or reserved
 - Check queue status: [qstat](#)
 - Check machine status: <http://status.alcf.anl.gov>
 - Check "ALCF Weekly Updates" for training, reservation, and maintenance notices
- List status of nodes on Theta: [nodelist](#)

When will my job start?

<https://www.alcf.anl.gov/>

- The question every user asks after submitting a job
- Difficult question to answer: depends on dynamic state of queue
- Output of qstat can offer one estimate
 - based on total machine hours queued/running ahead of job

Example of alias
to add to user
.bashrc file



```
[knight@thetalogin4:~> qstat --header JobID:Score:User:WallTime:QueuedTime:RunTime:TimeRemaining:Nodes:State:Mode:Procs:Queue:Est_Start_Time --reverse
```

JobID	Score	User	WallTime	QueuedTime	RunTime	TimeRemaining	Nodes	State	Mode	Procs	Queue	Est_Start_Time
434178	10.0	M sspatel	24:00:00	1915:26:14	17:31:13	06:28:46	256	running	script	256	Q.GEO_test	-
452118	1.1	M lovato	03:00:00	422:38:02	N/A	N/A	2048	queued	script	2048	default	2020-07-25 10:47:25
450995	1.1	M jasonhc	08:00:00	563:20:50	N/A	N/A	2048	queued	script	2048	default	2020-07-25 12:11:21
450996	1.1	M jasonhc	08:00:00	563:20:08	N/A	N/A	2048	queued	script	2048	default	2020-07-25 15:55:11
450997	1.1	M jasonhc	08:00:00	563:19:30	N/A	N/A	2048	queued	script	2048	default	2020-07-25 19:39:00
450998	1.1	M jasonhc	08:00:00	563:18:49	N/A	N/A	2048	queued	script	2048	default	2020-07-25 23:22:50
450999	1.1	M jasonhc	08:00:00	563:18:24	N/A	N/A	2048	queued	script	2048	default	2020-07-26 03:06:40
451000	1.1	M jasonhc	08:00:00	563:17:43	N/A	N/A	2048	queued	script	2048	default	2020-07-26 06:50:29
451001	1.1	M jasonhc	08:00:00	563:17:03	N/A	N/A	2048	queued	script	2048	default	2020-07-26 10:34:19
452561	1.1	M antoniov	06:00:00	372:40:57	04:20:02	01:39:57	2092	running	script	2092	default	-
441081	896.5	K dvartany	24:00:00	1508:57:28	N/A	N/A	2048	queued	script	2048	default	2020-07-26 14:18:08
451280	871.6	K knomura	24:00:00	529:15:34	N/A	N/A	4096	queued	script	4096	default	2020-07-27 01:29:37
451515	823.7	K knomura	24:00:00	519:35:12	N/A	N/A	4096	queued	script	4096	default	2020-07-27 23:52:34
451516	823.7	K knomura	24:00:00	519:34:59	N/A	N/A	4096	queued	script	4096	default	2020-07-28 22:15:31
446013	811.0	K jola	24:00:00	1058:37:41	N/A	N/A	1000	queued	script	1000	default	2020-07-29 20:38:28
449217	809.2	K jrpete	24:00:00	753:38:37	N/A	N/A	2048	queued	script	2048	default	2020-07-30 02:06:20
446042	807.0	K ars527	24:00:00	1057:02:13	N/A	N/A	1000	queued	script	1000	default	2020-07-30 13:17:48

Tips for submitting short test jobs

<https://www.alcf.anl.gov/>

- Short walltime jobs are sometimes needed
 - Debugging issue in code/script
 - Run multiple tests without having to resubmit job
 - Test result of optimizations and profiling
 - Testing new capability
- Debug queues available for small runs (1-8 nodes)
- What about jobs larger than 8 nodes?
 - Need to submit within default queue
 - Jobs smaller than 128 nodes are allocated 128 nodes
- Opportunity to sneak in short-large jobs ahead of maintenance windows
 - Pay attention to ALCF weekly updates for Maintenance Mondays and other reservations

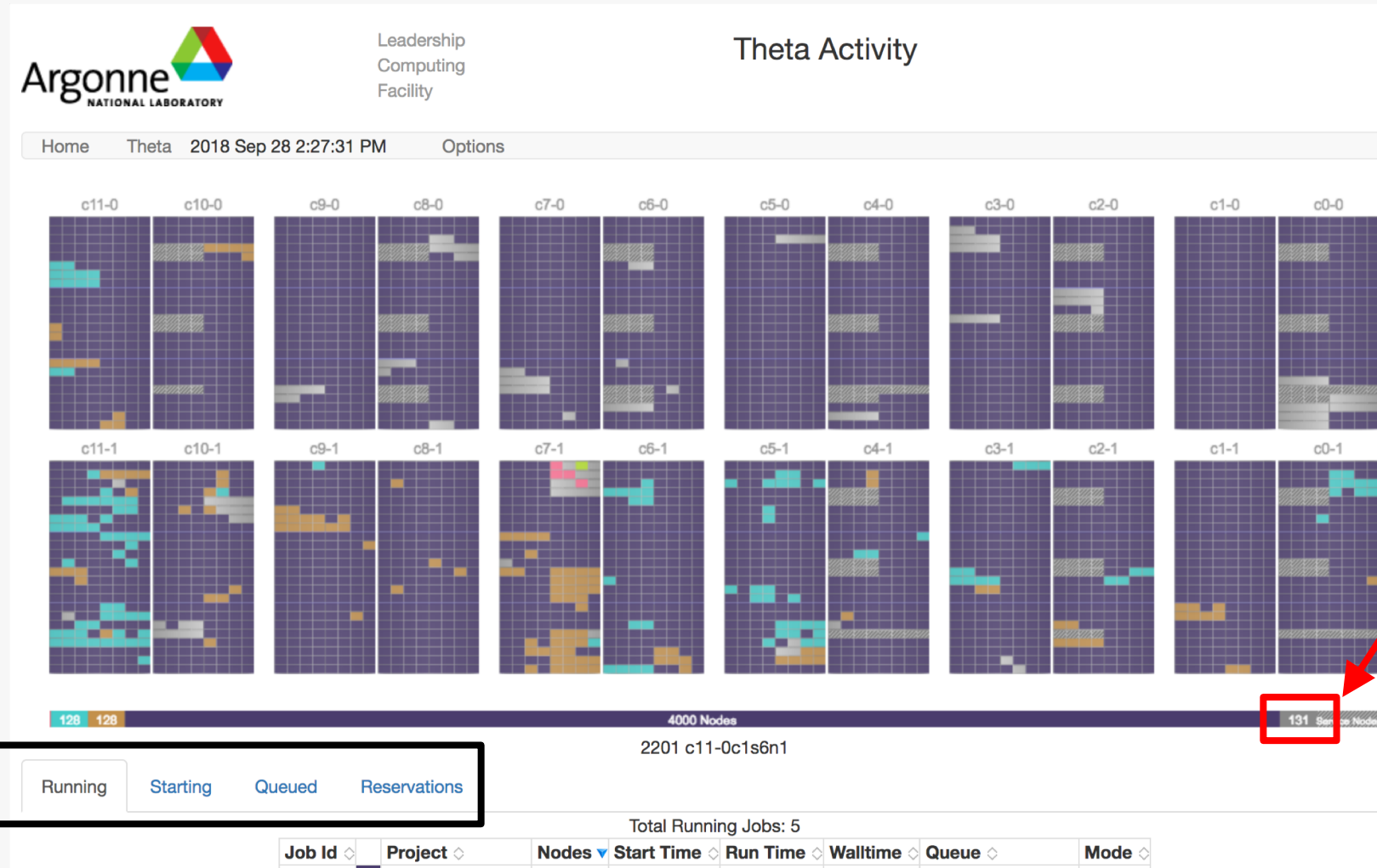
Improving turnaround in debug queues

<https://www.alcf.anl.gov/>

- Two 16-node debug queues available: debug-cache-quad & debug-flat-quad
 - Choose whichever queue has nodes readily available!
- Flat mode
 - Default memory allocations to DDR (192 GB)
 - If need <16 GB, then use numactl to allocate HBM for performance
 - Example: `aprun -n 128 -N 64 -d 1 -j 1 --cc depth numactl -m 1 <app> <app_args>`
- Can job fit within debug queue limits?
 - Maximum of 256 MPI ranks per node (if fit within memory)
 - Increase number of ranks per node to decrease number of nodes requested

Checking availability of nodes

<https://status.alcf.anl.gov/theta/activity>



Running
Starting
Queued
Reservations

Idle nodes

Tools: nodelist

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

- nodelist
 - List resources on system
 - Helpful to discover “backfill” nodes to run jobs on

```
nodelist
```

Node_id	Name	Queues	Status	MCDRAM	NUMA	Backfill
{...}						
20	c0-0c0s5n0	default	cleanup-pending	flat	quad	4:59:44
21	c0-0c0s5n1	default	cleanup-pending	flat	quad	4:59:44
22	c0-0c0s5n2	default	busy	flat	quad	4:59:44
24	c0-0c0s6n0	default	busy	flat	quad	4:59:44
25	c0-0c0s6n1	default	busy	flat	quad	4:59:44
26	c0-0c0s6n2	default	busy	flat	quad	4:59:44
27	c0-0c0s6n3	default	busy	flat	quad	4:59:44
28	c0-0c0s7n0	default	idle	flat	quad	4:59:44
29	c0-0c0s7n1	default	idle	flat	quad	4:59:44
30	c0-0c0s7n2	default	idle	flat	quad	4:59:44
31	c0-0c0s7n3	default	idle	flat	quad	4:59:44
32	c0-0c0s8n0	default	idle	flat	quad	4:59:44
33	c0-0c0s8n1	default	idle	flat	quad	4:59:44
34	c0-0c0s8n2	default	idle	flat	quad	4:59:44
{...}						

Tools: nodelist

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

- Desire for quick turnaround on short debugging jobs
- Find batch of idle nodes that could be backfilled
 - Ignore nodes assigned to active reservation (e.g. CVD_Research)

```
[knight@thetalogin4:~> nodelist | awk '{print $3" "$4" "$7}' | grep idle | grep default | sort | uniq -c
 59 backfill-cache-quad:cache-quad:all-nodes:backfill-all-nodes:default:backfill:gpfs-test:balsam:analysis:CVD_Research idle 29:00:42:16
 563 backfill-cache-quad:cache-quad:all-nodes:backfill-all-nodes:default:backfill:gpfs-test:balsam:analysis idle 46:04
 73 backfill-flat-quad:flat-quad:all-nodes:backfill-all-nodes:default:backfill:balsam:analysis idle 46:04
 2 default:backfill:balsam:analysis idle 46:04
 1 default:CSC249AD0A01_walltime idle -
 5 default:Q.GEO_test idle 2:57:05
[knight@thetalogin4:~> qsub -I -n 512 -t 30 -q default -A Catalyst
Connecting to thetamom1 for interactive qsub...
Job routed to queue "default".
Wait for job 454531 to start...
Opening interactive session to 90-109,190-199,210-219,270-279,290-299,340-349,404,480-489,500-509,554-555,557-559,600-607,609,618,632-639,662-663,740-759,840-843,852
-859,900-907,916-919,1001,1020-1025,1027-1039,1050-1059,1100-1109,1113,1115,1118-1119,1200,1202-1207,1210-1219,1278-1279,1376,1440-1442,1444,1446-1459,1511,1530,1584
-1585,1587,1590-1591,1593,1595-1597,1599,1620-1629,1653,1655,1690-1699,1722-1723,1730-1734,1736-1739,1780-1781,1791,1798,1840-1841,1843-1847,1850-1859,1897,1900-1909
,1930-1935,1947,1949,2038,2040-2049,2136-2139,2146-2147,2160-2175,2189,2260-2269,2310-2319,2389-2399,2401,2406-2408,2416,2433,2440-2443,2463,2540-2541,2543,2545-2559
,2595,2600-2617,2619,2669,2680,2682-2689,2751,2761-2763,2772-2779,2803,2836,2860-2863,2866-2867,2890-2899,2912,2917-2919,2940-2949,2952-2957,2965,3001-3007,3024,3061
-3062
[knight@thetamom1:/gpfs/mira-home/knight>
[knight@thetamom1:/gpfs/mira-home/knight>
[knight@thetamom1:/gpfs/mira-home/knight> exit
exit
Exiting interactive job 454531
Connection to thetamom1 closed.
```

Interactive Jobs

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

- Desire to interactively work on compute nodes
- qsub options
 - Interactive job: `-I` or `--interactive`
 - E-mail notifications: `-M <email_address1>:<email_address2>`
- Need to wait for nodes allocated and job to start
- Important to remember shell is executed on launch node, not compute node
 - aprun needed to launch commands on compute nodes

```
[knight@thetalogin4:~]$ qsub -I -n 512 -t 20 -q default -A Catalyst
Connecting to thetamom1 for interactive qsub...
Job routed to queue "default".
Wait for job 454531 to start...
Opening interactive session to 90-109,190-199,210-219,270-279,290-299,340-349,404,480-489,500-509,554-555,557-559,600-607,609,618,632-639,662-663,740-759,840-843,852
-859,900-907,916-919,1001,1020-1025,1027-1039,1050-1059,1100-1109,1113,1115,1118-1119,1200,1202-1207,1210-1219,1278-1279,1376,1440-1442,1444,1446-1459,1511,1530,1584
-1585,1587,1590-1591,1593,1595-1597,1599,1620-1629,1653,1655,1690-1699,1722-1723,1730-1734,1736-1739,1780-1781,1791,1798,1840-1841,1843-1847,1850-1859,1897,1900-1909
,1930-1935,1947,1949,2038,2040-2049,2136-2139,2146-2147,2160-2175,2189,2260-2269,2310-2319,2389-2399,2401,2406-2408,2416,2433,2440-2443,2463,2540-2541,2543,2545-2559
,2595,2600-2617,2619,2669,2680,2682-2689,2751,2761-2763,2772-2779,2803,2836,2860-2863,2866-2867,2890-2899,2912,2917-2919,2940-2949,2952-2957,2965,3001-3007,3024,3061
-3062
[knight@thetamom1:~/gpfs/mira-home/knight]$
[knight@thetamom1:~/gpfs/mira-home/knight]$
[knight@thetamom1:~/gpfs/mira-home/knight]$ exit
exit
Exiting interactive job 454531
Connection to thetamom1 closed.
```

Interactive Jobs

<https://www.alcf.anl.gov/support-center/theta/submit-job-theta>

- Desire to interactively work on compute nodes
- qsub options
 - Interactive job: `-I` or `--interactive`
 - E-mail notifications: `-M <email_address1>:<email_address2>`
- Need to wait for nodes allocated and job to start
- Important to remember shell is executed on launch node, not compute node
 - aprun needed to launch commands on compute nodes

```
[knight@thetalogin4:~> qsub -I -n 1 -t 30 -q debug-flat-quad -A Catalyst -M knightc@anl.gov
Connecting to thetamom3 for interactive qsub...
Job routed to queue "debug-flat-quad".
Memory mode set to flat quad for queue debug-flat-quad
Wait for job 454678 to start...
Opening interactive session to 2
[knight@thetamom3:/gpfs/mira-home/knight>
```

```
knight@thetamom3:/gpfs/mira-home/knight> cat /proc/cpuinfo
processor      : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 79
model name   : Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz
```

```
knight@thetamom3:/gpfs/mira-home/knight> aprun -n 1 cat /proc/cpuinfo
processor      : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 87
model name   : Intel(R) Xeon Phi(TM) CPU 7230 @ 1.30GHz
```

Interactive Jobs: Environment

<https://www.alcf.anl.gov/>

- Environment on MOM node may need to be refreshed in some cases
 - Dynamic linking
 - ``ldd <exe>`` results in "... error while loading shared libraries ..."
- Manually update needed environment variables
 - `export LD_LIBRARY_PATH=<value from login node>`
- Alternatively, can reload environment using `module`
 - Login node: `module save my_defaults`
 - Creates `~/.module_snapshots/my_defaults` file
 - Mom node: `module restore my_defaults`

```
[knight@thetalogin4:~> qsub -I -n 1 -t 15 -q debug-cache-quad -A Catalyst
Connecting to thetamom2 for interactive qsub...
Job routed to queue "debug-cache-quad".
Memory mode set to cache quad for queue debug-cache-quad
Wait for job 455265 to start...

Opening interactive session to 3827
knight@thetamom2:/gpfs/mira-home/knight>
[knight@thetamom2:/gpfs/mira-home/knight> echo $LD_LIBRARY_PATH

[knight@thetamom2:/gpfs/mira-home/knight> module restore my_defaults
[knight@thetamom2:/gpfs/mira-home/knight> echo $LD_LIBRARY_PATH
/soft/perftools/darshan/darshan-3.2.1/lib:/opt/cray/pe/papi/6.0.0.1/lib64:/opt.
```

Interactive Jobs: Jupyter Hub

<https://www.alcf.anl.gov/support-center/theta/jupyter-hub>

- Work within Python and R notebooks or open terminal in browser
- Login (using ALCF credentials): <https://jupyter.alcf.anl.gov/theta/hub/login>
- Open new terminal

Running earlier 'nodelist' command

```
jupyter Logout Control Panel  
[knight@jupyter02 knight]$  
[knight@jupyter02 knight]$ ./bin/find_free_nodes.sh  
  
Default queue:  
 249 backfill-cache-quad:cache-quad:all-nodes:backfill-all-nodes:default:backfill:gpfs-test:balsam:analysis:CVD_Research  
idle 29:23:24:02  
  1 backfill-cache-quad:cache-quad:all-nodes:backfill-all-nodes:default:backfill:gpfs-test:balsam:analysis idle -  
  1 default:Q.GEO_test idle -  
  
Debug queue:  
  6 debug-cache-quad idle -  
  2 debug-flat-quad:build idle -  
  4 debug-flat-quad idle -  
[knight@jupyter02 knight]$  
[knight@jupyter02 knight]$ qsub -I -n 2 -t 15 -q debug-flat-quad -A Catalyst  
Job routed to queue "debug-flat-quad".  
Memory mode set to flat quad for queue debug-flat-quad  
Wait for job 455252 to start...  
Opening interactive session to 20,23  
[knight@jupyter02 knight]$
```




ANY QUESTIONS?

Section III: Ensembles

The background of the slide features a large, stylized letter 'E' composed of numerous thin, light blue lines. Scattered throughout the scene are many small, semi-transparent blue spheres, some of which appear to be connected to the lines, creating a complex, network-like or molecular structure. The overall color palette is a range of blues, from dark navy to light sky blue.

Theta Ensemble Jobs

```
#!/bin/bash  
Job scripts run on MOM  
(Broadwell) nodes  
myApp="/path/to/app --input="
```

Compute (KNL) Nodes

nid00001

nid00002

nid00003

nid00004

nid00005

Theta Ensemble Jobs

```
#!/bin/bash  
Job scripts run on MOM  
(Broadwell) nodes  
myApp="/path/to/app --input="  
aprun -n 64 -N 64 $myApp input1 >& run1.out &  
sleep 1
```

aprun

Compute (KNL)
Nodes

nid00001

nid00002

nid00003

nid00004

nid00005

Theta Ensemble Jobs

```
#!/bin/bash  
Job scripts run on MOM  
(Broadwell) nodes  
myApp="/path/to/app --input="  
aprun -n 64 -N 64 $myApp input1 >& run1.out &  
sleep 1  
aprun -n 128 -N 64 $myApp input2 >& run2.out &  
sleep 1
```

aprun

aprun

Compute (KNL)
Nodes

nid00001

nid00002

nid00003

nid00004

nid00005

Theta Ensemble Jobs

```
#!/bin/bash  
Job scripts run on MOM  
(Broadwell) nodes  
myApp="/path/to/app --input="  
  
aprun -n 64 -N 64 $myApp input1 >& run1.out &  
sleep 1  
  
aprun -n 128 -N 64 $myApp input2 >& run2.out &  
sleep 1  
  
aprun -n 128 -N 64 $myApp input3 >& run3.out &
```

aprun

aprun

aprun

Compute (KNL)
Nodes

nid00001

nid00002

nid00003

nid00004

nid00005

alcf.anl.gov/user-guides/running-jobs-xc40#bundling-multiple-runs-into-a-script-job

Theta Ensemble Jobs

Efficiently packing jobs across nodes and time requires a workflow manager

```
#!/bin/bash
Job scripts run on MOM
(Broadwell) nodes

myApp="/path/to/app --input="

aprun -n 64 -N 64 $myApp input1 >& run1.out &
sleep 1

aprun -n 128 -N 64 $myApp input2 >& run2.out &
sleep 1

aprun -n 128 -N 64 $myApp input3 >& run3.out &

aprun -n 128 -N 64 $myApp input3 >& run3.out &
```

aprun

aprun

aprun



Compute (KNL)
Nodes

nid00001

nid00002

nid00003

nid00004

nid00005

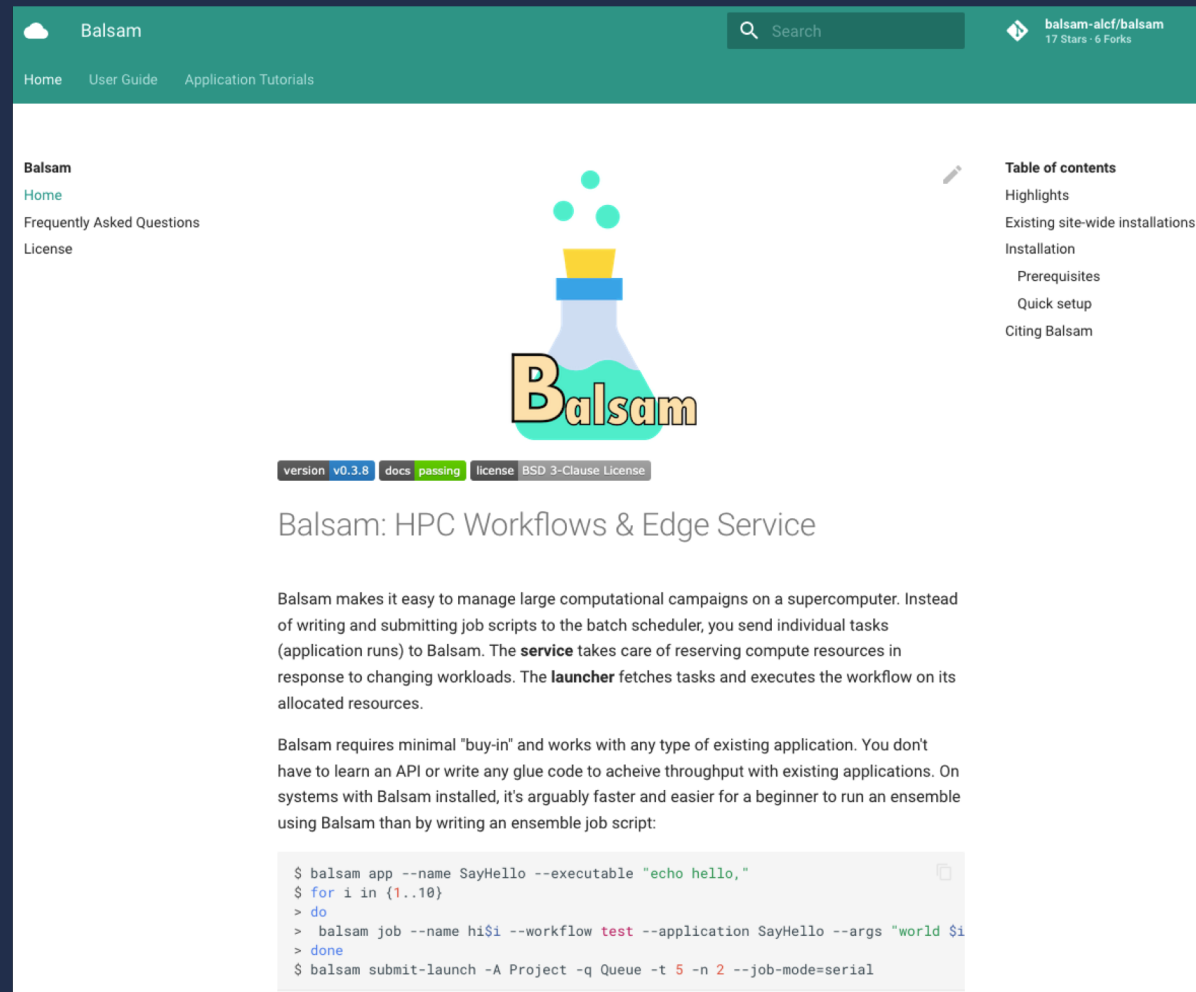
alcf.anl.gov/user-guides/running-jobs-xc40#bundling-multiple-runs-into-a-script-job

Balsam

Automated scheduling and execution for HPC workflows

- Submit unlimited application runs to a private task database
- **Service** component automates queue submission
- **Launcher** component pulls tasks for load-balanced execution
 - Resilient to task-level faults
 - Automatic retry or custom handling of timed-out, failed jobs
 - Runs **unmodified** user applications or Singularity containers
- Workflow status and project statistics available at-a-glance

Balsam in production @ ALCF



The screenshot shows the Balsam project documentation website. The header is green with the Balsam logo, a search bar, and navigation links for Home, User Guide, and Application Tutorials. The main content area features the Balsam logo (a flask with green liquid and bubbles) and the text "Balsam: HPC Workflows & Edge Service". Below this, there is a description of Balsam's capabilities and a code block showing how to use Balsam to run a workflow.

Balsam
Home
Frequently Asked Questions
License

Table of contents
Highlights
Existing site-wide installations
Installation
Prerequisites
Quick setup
Citing Balsam

version **v0.3.8** docs **passing** license **BSD 3-Clause License**

Balsam: HPC Workflows & Edge Service

Balsam makes it easy to manage large computational campaigns on a supercomputer. Instead of writing and submitting job scripts to the batch scheduler, you send individual tasks (application runs) to Balsam. The **service** takes care of reserving compute resources in response to changing workloads. The **launcher** fetches tasks and executes the workflow on its allocated resources.

Balsam requires minimal "buy-in" and works with any type of existing application. You don't have to learn an API or write any glue code to achieve throughput with existing applications. On systems with Balsam installed, it's arguably faster and easier for a beginner to run an ensemble using Balsam than by writing an ensemble job script:

```
$ balsam app --name SayHello --executable "echo hello,"  
$ for i in {1..10}  
> do  
> balsam job --name hi$i --workflow test --application SayHello --args "world $i"  
> done  
$ balsam submit-launch -A Project -q Queue -t 5 -n 2 --job-mode=serial
```

balsam.readthedocs.io

Theta Module

```
$ module load balsam
```

**Sets PATH to include:
PostgreSQL 9.6 binaries
Python 3.6 environment bin/ (with Balsam installed)**

Command line interface

```
$ balsam
```

```
usage: balsam [-h]
```

```
       {app,job,dep,ls,modify,rm,killjob,mkchild,launcher,submit-launch,init,se
```

```
Balsam 0.3.5
```

```
Command line interface:
```

app	add a new application definition
job	add a new Balsam job
dep	add a dependency between two existing jobs
ls	list jobs, applications, or jobs-by-workflow

Start a new Balsam DB

Use `balsam init` to create a new database directory

```
balsam init ~/myproject
```

```
*****  
Successfully created Balsam DB at: /home/user/myProject  
Use `source balsamactivate myProject` to begin working.  
*****
```

Start a new Balsam DB

```
source balsamactivate <db-name>  
starts server (if not running), sets environment
```

```
$ . balsamactivate myProject  
Launching Balsam DB server  
waiting for server to start.... done  
server started  
[BalsamDB: myProject] $
```

Balsam Hello World

`balsam app :`
Register new applications with Balsam

```
[BalsamDB: myProject] $ balsam app --name say-hello \  
--executable "echo Hello, "
```

Application 1:

name:	say-hello
description:	
executable:	echo Hello,

Added app to database

Balsam Hello World

`balsam job :`
Add a new task

```
[BalsamDB: myProject] $ balsam job --name test1 --workflow test \  
--app say-hello --args "World 1!"
```

```
[BalsamDB: myProject] $ for i in {2..10}  
> do  
> balsam job --name test$i --workflow test \  
--app say-hello --args "World $i!" --yes  
> done
```

BalsamJob 0796bd50-adbc-424b-bd26-476f2c00275b

```
-----  
workflow:                test  
name:                    test1  
description:  
parents:                 []  
input_files:            *  
num_nodes:              1  
ranks_per_node:         1  
environ_vars:  
application:         say-hello  
args:                World 1!  
auto_timeout_retry:     True  
  *** Executed command:  echo Hello, World 1!  
  *** Working directory: ~/myProject/data/test/test1_0796bd50
```

Confirmation shows task details and adjustable fields

Confirm adding job to DB [y/n]: y

Balsam Hello World

`balsam ls :`
View tasks in database

```
[BalsamDB: myProject] $ balsam ls
```

job_id	name	workflow	application	state
0796bd50-adbc-424b-bd26-476f2c00275b	test1	test	say-hello	CREATED
421e6df8-4984-423f-b44f-c58c6e2e8307	test2	test	say-hello	CREATED
d62b194a-e20b-4111-a407-3669fb4c89e6	test3	test	say-hello	CREATED
e73325df-3104-4e7f-aa1e-9ba61840ecc6	test4	test	say-hello	CREATED
79becbd6-8ab9-4012-9828-6dc98157eb5c	test5	test	say-hello	CREATED
c7ed41fd-6aa4-4a29-957e-bf91cfef3453	test6	test	say-hello	CREATED
dda7cdd3-0098-4fe7-9827-ca78a73d7be9	test7	test	say-hello	CREATED
2e6c4914-a1a7-4ea1-be5e-994fc6ca7830	test8	test	say-hello	CREATED
4a12cb47-cbe1-4f83-97d2-ee97eefc9343	test9	test	say-hello	CREATED
f4d77a25-1d48-4fc9-be2b-cc0e1af61473	test10	test	say-hello	CREATED

Balsam Hello World

`balsam submit-launch` :
Shortcut for Cobalt job submission (template in `~/.balsam`)

```
[BalsamDB: myProject] $ balsam submit-launch -n 2 -t 5 \  
-q debug-cache-quad -A datascience --job-mode mpi
```

```
Submit OK: Qlaunch {  
  'command': '~/myProject/qsubmit/qlaunch1.sh',  
  'id': 1,  
  'job_mode': 'mpi',  
  'nodes': 2,  
  'project': 'datascience',  
  'queue': 'debug-cache-quad',  
  'scheduler_id': 333718,  
  'state': 'submitted',  
  'wall_minutes': 5,  
  'wf_filter': ''}
```

Customizable
templated script

Balsam Hello World

If successful, jobs eventually marked
JOB_FINISHED

```
[BalsamDB: myProject] $ balsam ls
```

job_id	name	workflow	application	state
dda7cdd3-0098-4fe7-9827-ca78a73d7be9	test7	test	say-hello	JOB_FINISHED
f4d77a25-1d48-4fc9-be2b-cc0e1af61473	test10	test	say-hello	JOB_FINISHED
79becbd6-8ab9-4012-9828-6dc98157eb5c	test5	test	say-hello	JOB_FINISHED
c7ed41fd-6aa4-4a29-957e-bf91cfef3453	test6	test	say-hello	JOB_FINISHED
e73325df-3104-4e7f-aa1e-9ba61840ecc6	test4	test	say-hello	JOB_FINISHED
4a12cb47-cbe1-4f83-97d2-ee97eefc9343	test9	test	say-hello	JOB_FINISHED
421e6df8-4984-423f-b44f-c58c6e2e8307	test2	test	say-hello	JOB_FINISHED
2e6c4914-a1a7-4ea1-be5e-994fc6ca7830	test8	test	say-hello	JOB_FINISHED
0796bd50-adbc-424b-bd26-476f2c00275b	test1	test	say-hello	JOB_FINISHED
d62b194a-e20b-4111-a407-3669fb4c89e6	test3	test	say-hello	JOB_FINISHED

Balsam Hello World

Job working directories are created as:

`data/<workflow>/<name>_<id>`

```
[BalsamDB: myProject] $ ~/myProject/data/test> ls  
  
test1_0796bd50    test2_421e6df8    test4_e73325df    test6_c7ed41fd    test8_2e6c4914  
test10_f4d77a25  test3_d62b194a    test5_79becbd6    test7_dda7cdd3    test9_4a12cb47
```



ANY QUESTIONS?

When Things Go Wrong Running...

<https://www.alcf.anl.gov/user-support>

- Examine core files
- Best to save all three files generated by cobalt
 - <prefix_name>.cobaltlog, <prefix_name>.error, and <prefix_name>.output
- Retain important information
 - Jobid, machine name, copy/location of all files, exact error message
- Contact us
 - Your ALCF contact
 - Email: support@alcf.anl.gov



HAPPY COMPUTING!