Balsam Workflows

github.com/balsam-alcf/balsam

Misha Salim
Argonne Leadership Computing Facility
msalim@anl.gov

HPC needs better throughput tools

Sometimes a few scripts is enough:

(20 runs) (1024 nodes) (12 hours) = 245,760 node-hours

```
-----myjob.sh------
#!/bin/sh
echo "Starting Cobalt job script"
aprun -n 128 -N 64 run1.exe arg1 &
sleep 1
aprun -n 256 -N 64 run1.exe arg1 &
sleep 1
aprun -n 512 -N 64 run1.exe arg1 &
wait

-----end myjob.sh---
```

HPC needs better throughput tools

Sometimes a few scripts is enough:

(20 runs) (1024 nodes) (12 hours) = 245,760 node-hours

```
-----myjob.sh------
#!/bin/sh
echo "Starting Cobalt job script"
aprun -n 128 -N 64 run1.exe arg1 &
sleep 1
aprun -n 256 -N 64 run1.exe arg1 &
sleep 1
aprun -n 512 -N 64 run1.exe arg1 &
wait

-----end myjob.sh---
```

Human effort scales unfavorably with # of runs:

(2,457,600 runs) (1 node) (6 minutes) = 245,760 node-hours

HPC needs better throughput tools

Restrictive queue policy

Lacking job packing / MPMD execution

Cumbersome error & timeout handling

Human effort scales unfavorably with # of runs:

```
(2,457,600 runs) (1 node) (6 minutes) = 245,760 node-hours
```

Balsam

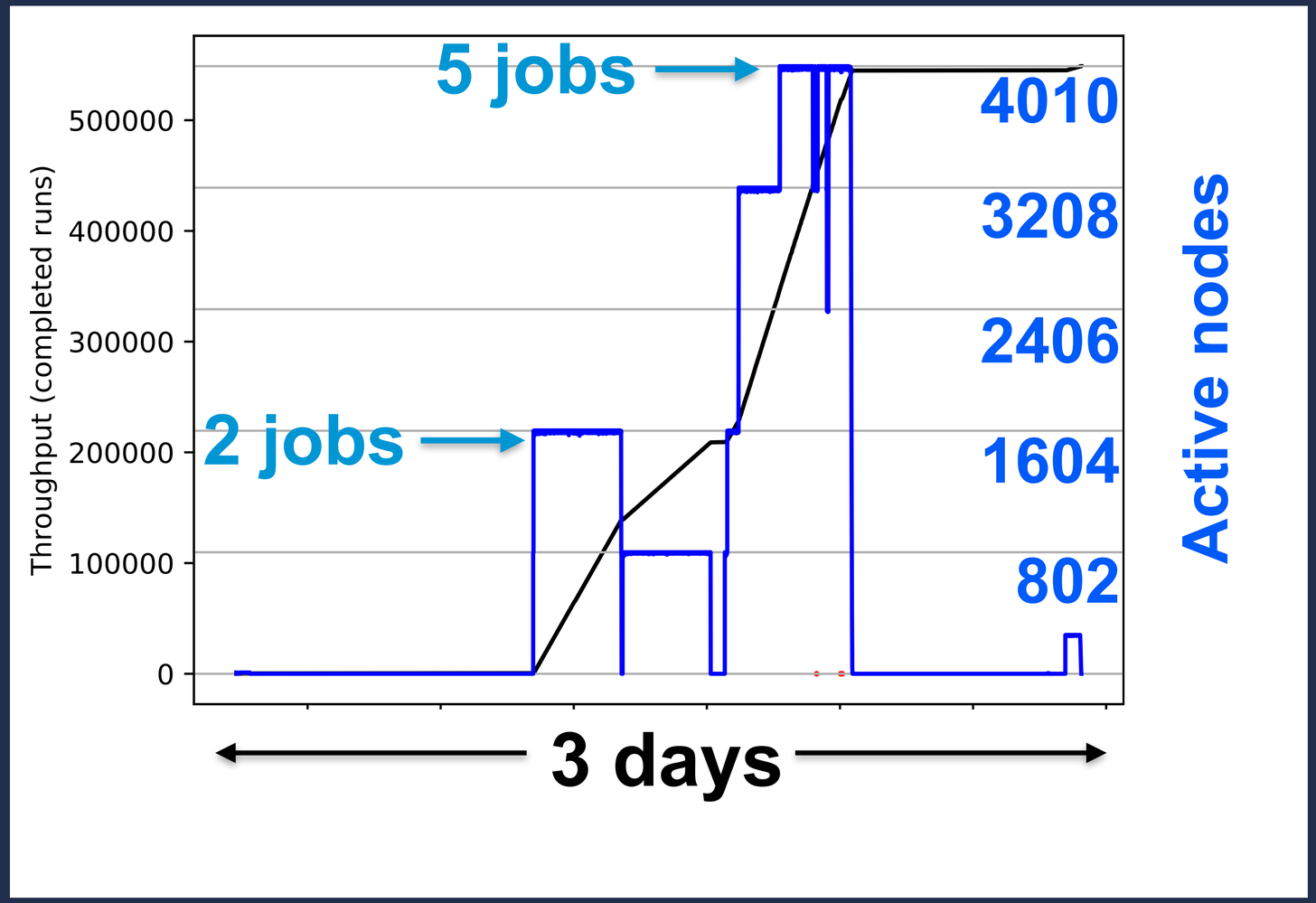
Workflows, scheduling, and execution for HPC

- **Plan unlimited application runs in a stateful task database**
- **Balsam automates packing, scheduling, and MPMD task execution**
  - **no modification to user applications**
  - **strong fault tolerance at task level**
- **Program dynamic workflows with Python API**
- **Workflow status and project statistics available at-a-glance**

Scaled to 91% of Theta, 1.2M+ tasks

Constructing and Navigating Polymorphic Landscapes of Molecular Crystals (PI: Alexandre Tkatchenko)

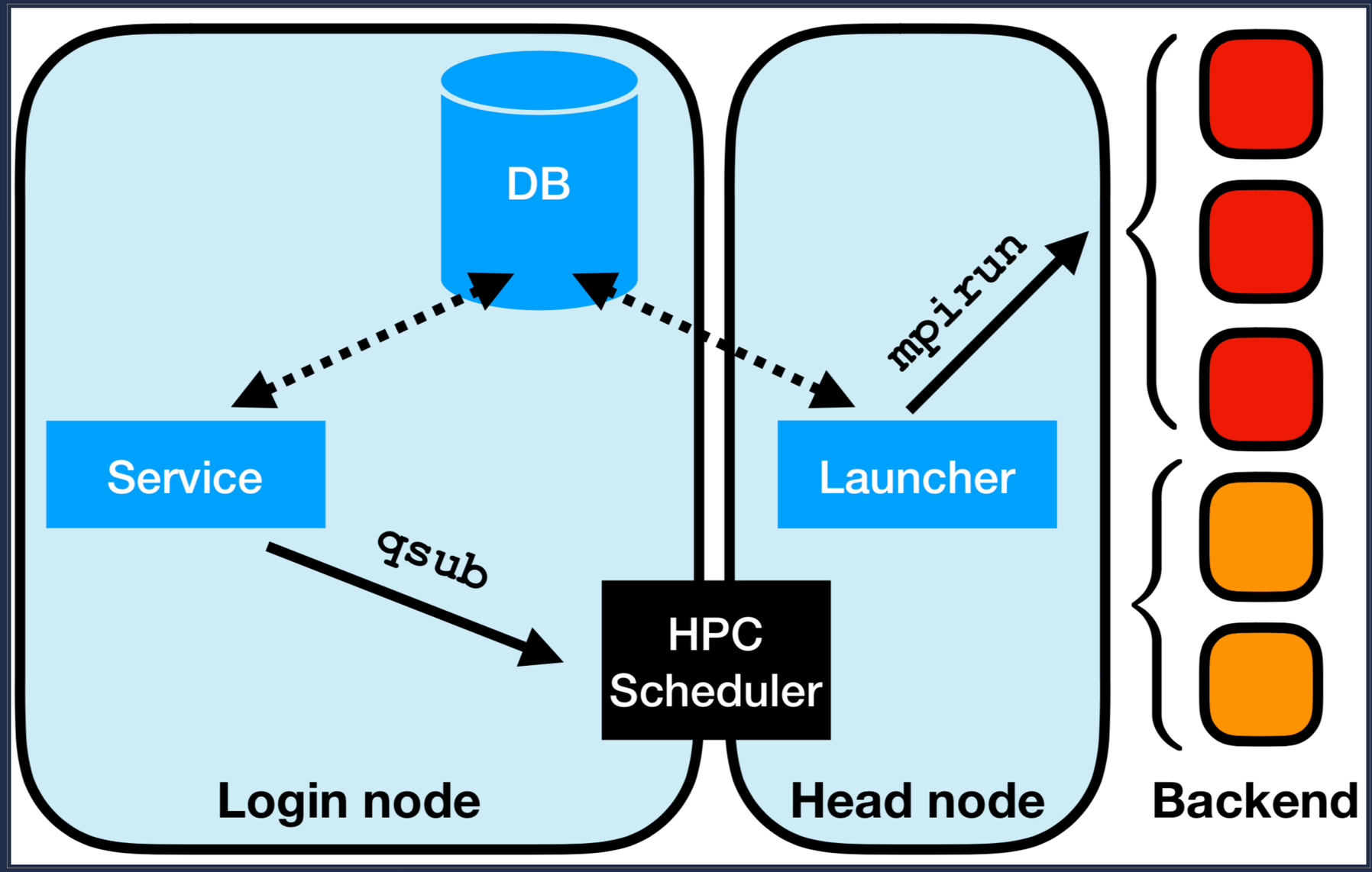
- 7M+ core hours of DFT
- Up to 5 concurrent batch jobs consuming tasks from the same database



Release in production @ ALCF


```
module load balsam
```

Balsam components



Database

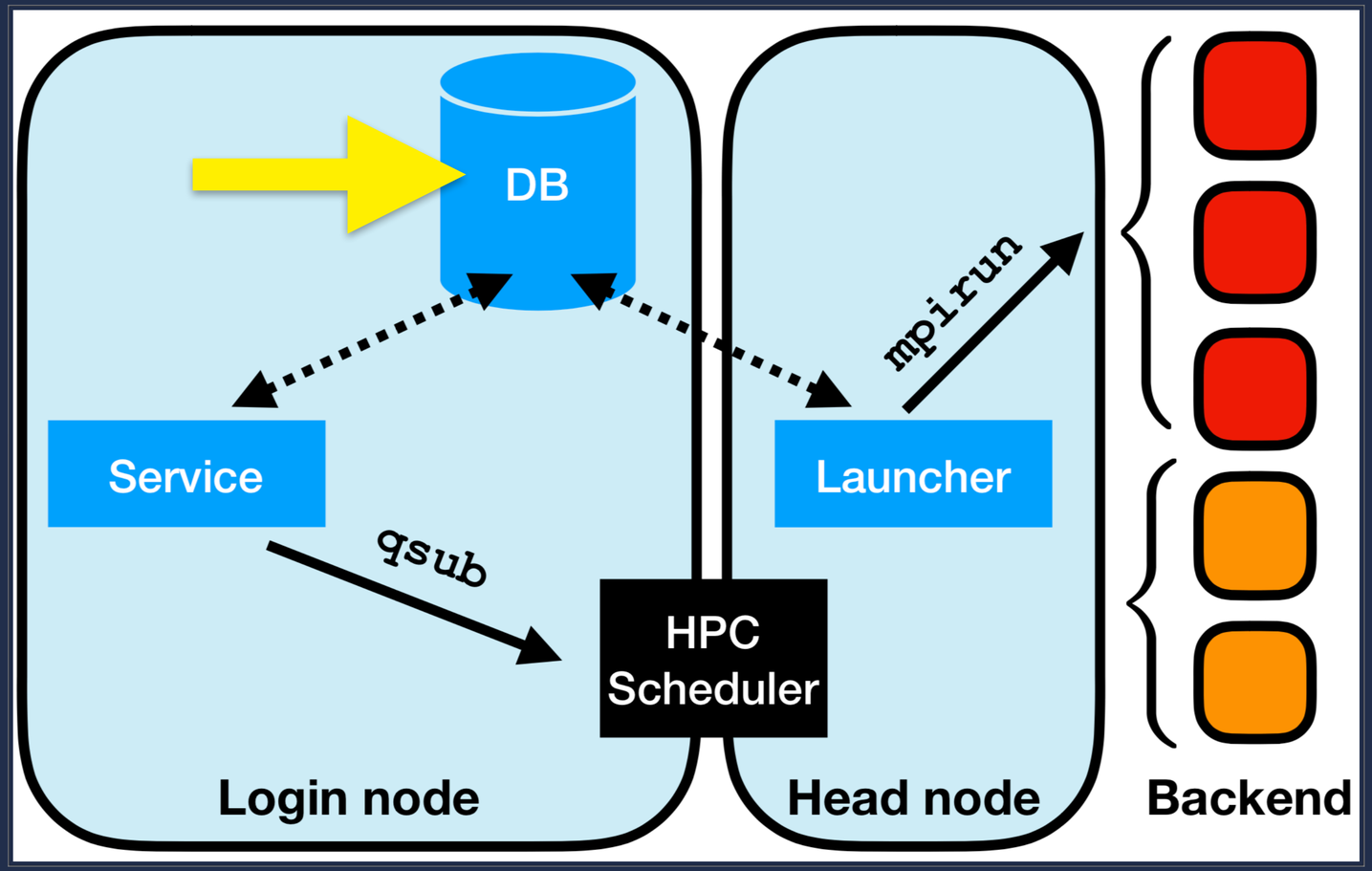
BalsamJob table:

one row per task

PostgreSQL backend

High concurrency

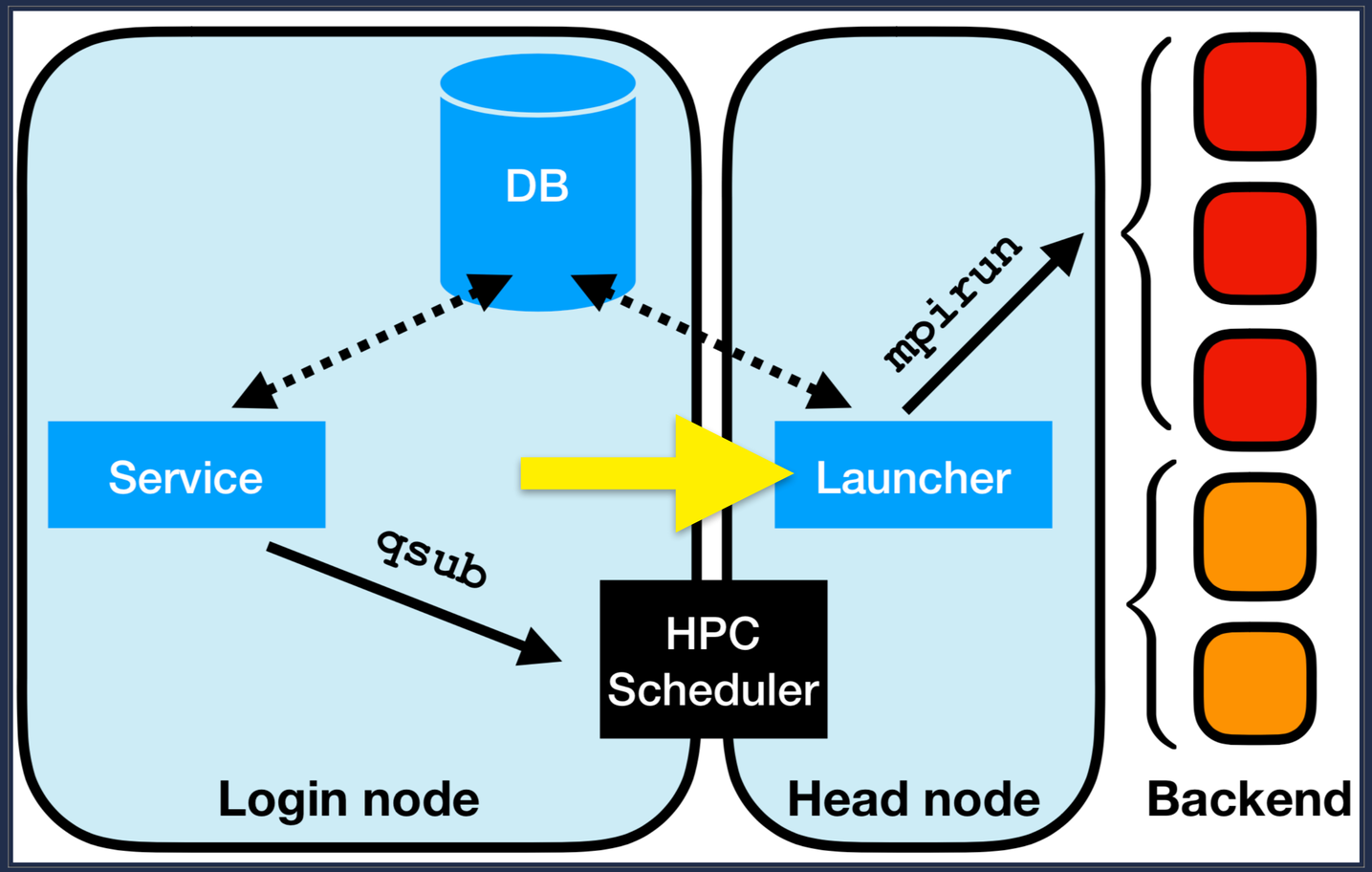
Django API



Launcher

Pilot application running inside Cobalt job

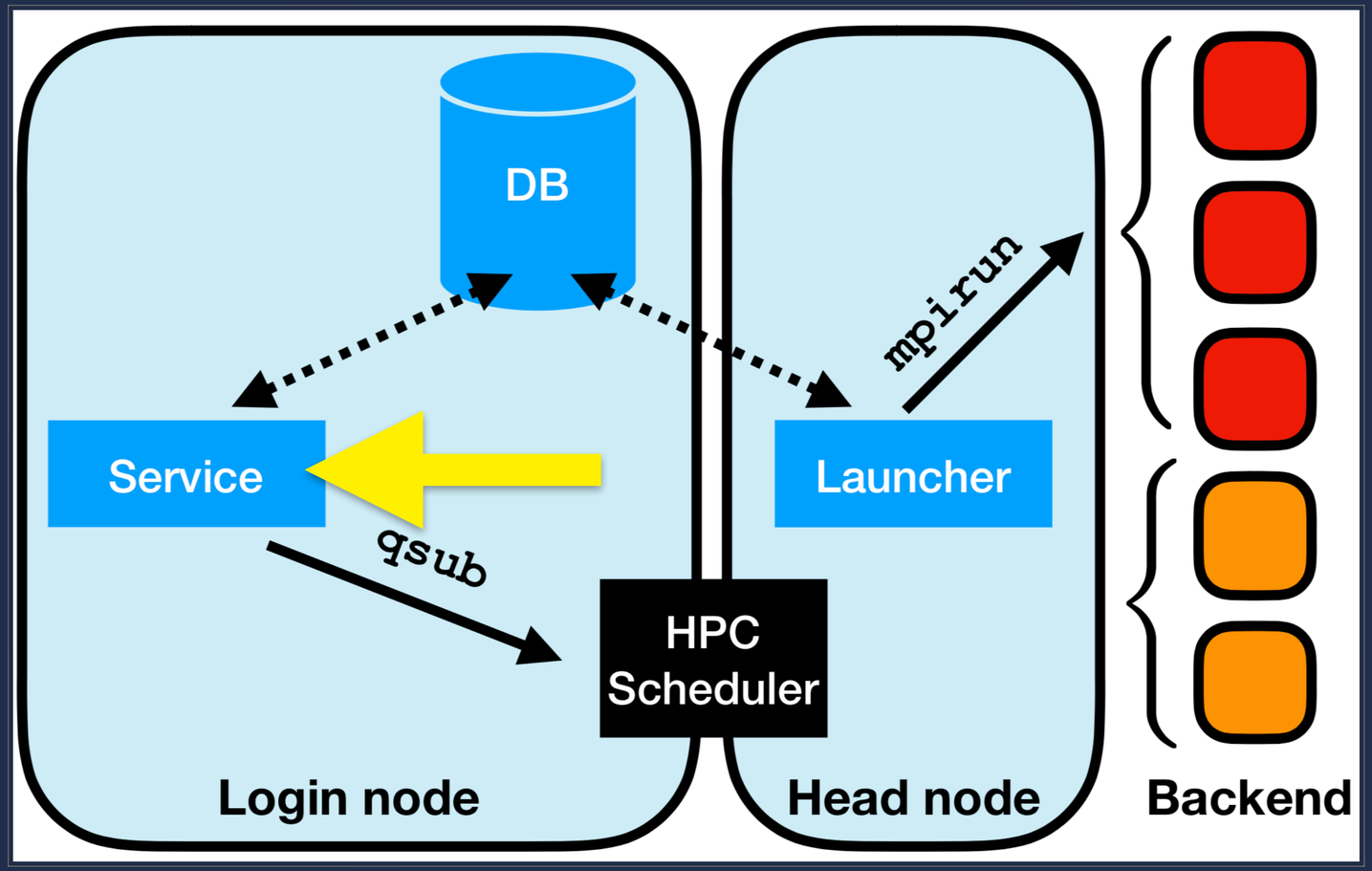
Dynamic task pull and execution



Service

Automated queue submission (optional)

Elastic scheduling of launcher jobs under varying workload



BalsamJob fields: storage & retrieval

Name

Workflow Tag

Description

Working directory

JSON data

BalsamJob fields: dependencies & data movement

Parents

Stage-out URL

Stage-in URL

Input file patterns

BalsamJob fields: application parameters

Executable

Environment variables

Command line arguments

Pre/post-process scripts

MPI ranks

ranks / node

Django: flexible queries from Python

```
from balsam.launcher.dag import BalsamJob


BalsamJob.objects.filter(
    num_nodes__lte=128,
    name__contains="sim",
    state="JOB_FINISHED"
).update(state="RESTART_READY")
```

CLI encapsulates database configuration

Automated setup

```
balsam init ~/myproject
```

Virtualenv style context switching

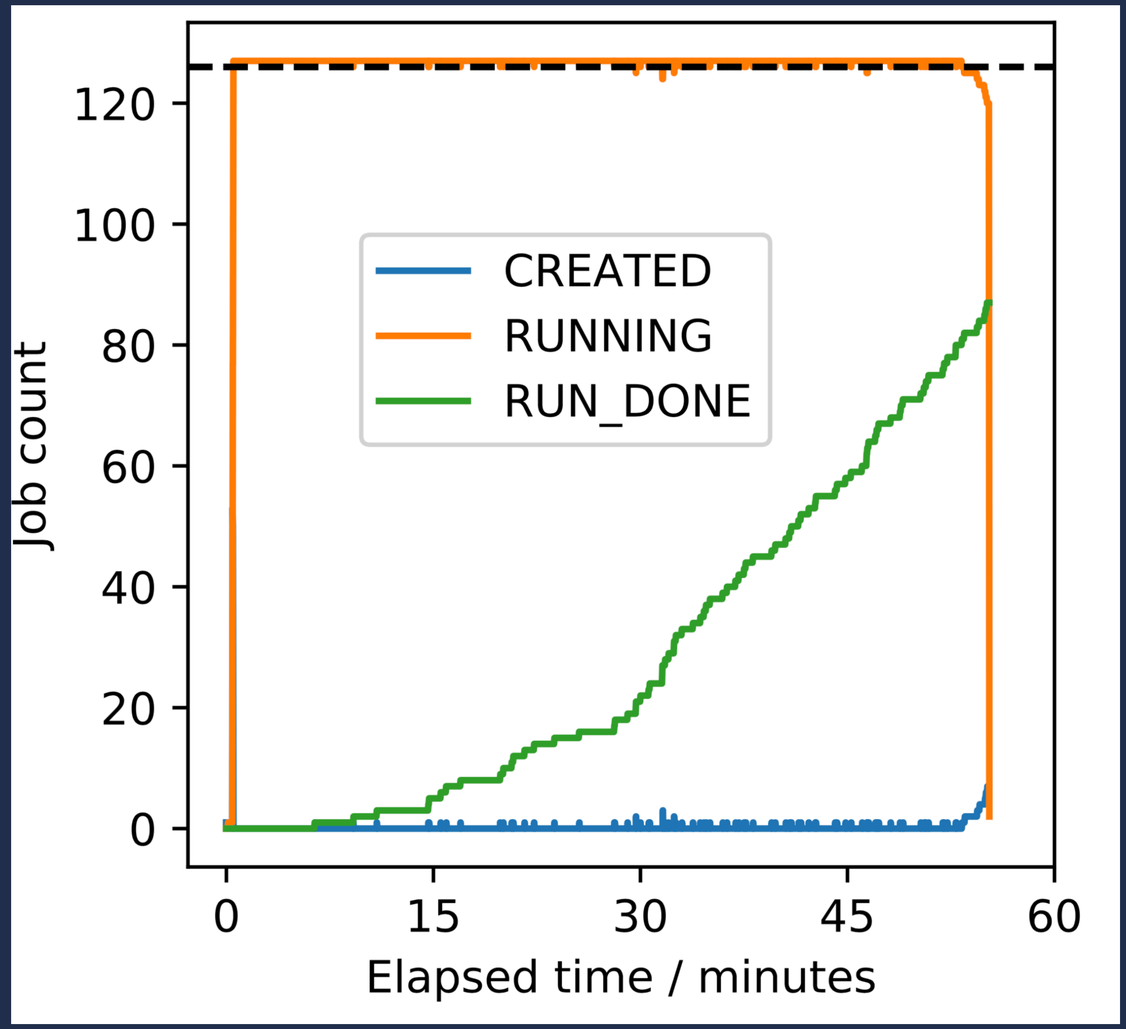
```
. balsamactivate myproject
```

Check load balance from launcher metadata

Async Bayesian hyperparameter optimization: 64 nodes, 1 model-per-GPU

Load-balanced, flexible MPMD execution

Execution metadata available for post-run analysis



Launcher

Resilient to task faults; errors logged in database

```
[BalsamDB: test-db] msalim@thetalogin6:~/test-db> balsam ls --state FAILED
--history
Job testfail [fab575a3-01db-41b5-b70d-c396c17ef10d]
------------------------------------------------

[10-03-2018 19:34:38.379895 CREATED]
[10-03-2018 19:38:24.490910 PREPROCESSED]
[10-03-2018 19:38:24.701099 RUNNING]
[10-03-2018 19:38:30.618931 RUN_ERROR]     Traceback (most recent call last):
   Hello from rank 2
   Hello from rank 1
      File "/gpfs/mira-home/msalim/test-db/fail.py", line 5, in <module>
           raise RuntimeError("simulated error")
```

Theta Walkthrough

Balsam on Theta — Walkthrough

- **Theta module requires Python 3.6**

```
msalim@thetalogin6:~> module load balsam
msalim@thetalogin6:~> balsam
Balsam requires Python version >= 3.6
```

Command line interface

Load 3.6

```
msalim@thetalogin6:~> module load cray-python/3.6.1.1
msalim@thetalogin6:~> balsam
usage: balsam [-h]
              {app,job,dep,ls,modify,rm,killjob,mkchild,launcher,submit-launch,
mmies,which,log,server}
              ...

Balsam command line interface

optional arguments:
  -h, --help            show this help message and exit

Subcommands:
  {app,job,dep,ls,modify,rm,killjob,mkchild,launcher,submit-launch,init,service
log,server}
    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
    modify              alter job or application
    rm                  remove jobs or applications from the database
    killjob             Kill a job without removing it from the DB
    mkchild             Create a child job of a specified job
    launcher            Start a local instance of the balsam launcher
```

CLI subcommands

Start a new Balsam DB

Use `balsam init` to create a new database directory

```
msalim@thetalogin6:~> balsam init ~/test-db
```

```
**********************************************************************
  Successfully created Balsam DB at: /home/msalim/test-db
  Use `source balsamactivate test-db` to begin working.
**********************************************************************
```

Start a new Balsam DB

`source balsamactivate <db-name>` **: starts server, if not already running, sets environment for Balsam**

```
msalim@thetalogin6:~> . balsamactivate test-db
Launching Balsam DB server
waiting for server to start.... done
server started
[BalsamDB: test-db] msalim@thetalogin6:~>
```

Hello World

`balsam app` :

Register new applications with Balsam

```
[BalsamDB: test-db] msalim@thetalogin6:~> balsam app --name say-hello --executable "echo Hello,"
Application 1:
---------------------------
Name:           say-hello
Description:
Executable:     echo Hello,
Preprocess:
Postprocess:

Added app to database
```

Hello World

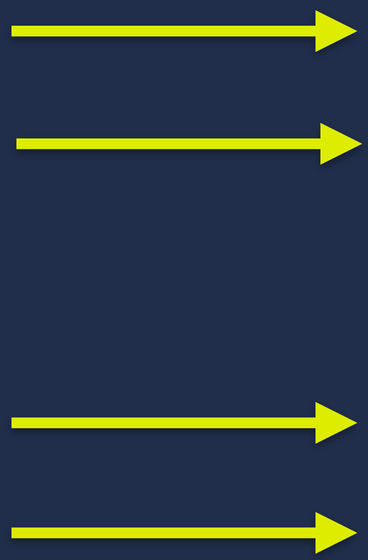
`balsam job` :

Add a new run (or task) to the database

```
[BalsamDB: test-db] msalim@thetalogin6:~> balsam job --name test1 --workflow test \
> --app say-hello --args "world!"
```

Hello World

Confirmation shows task details and names of adjustable fields



```
BalsamJob 7df86c8b-f94b-4349-8ab6-a8afc85d8a9f
-----------------------------------------------
workflow:                       test
name:                           test1
description:
lock:
parents:                        []
input_files:                    *
stage_in_url:
stage_out_files:
stage_out_url:
wall_time_minutes:              1
num_nodes:                      1
coschedule_num_nodes:           0
ranks_per_node:                 1
cpu_affinity:                   none
threads_per_rank:               1
threads_per_core:               1
node_packing_count:             1
environ_vars:
application:                    say-hello
args:                           world!
user_workdir:
wait_for_parents:               True
post_error_handler:             False
post_timeout_handler:           False
```

Hello World

Confirmation shows task details and names of adjustable fields

```
auto_timeout_retry:             True
state:                          CREATED
queued_launch_id:               None
data:                           {}
  *** Executed command:         echo Hello, world!
  *** Working directory:        /gpfs/mira-home/msalim/test-db/data/test/test1_7df86c8b
```

Hello World

`balsam ls` :

View tasks in database

```
[BalsamDB: test-db] msalim@thetalogin6:~> balsam ls
                                  job_id |   name | workflow | application |   state
-------------------------------------------------------------------------------------
7df86c8b-f94b-4349-8ab6-a8afc85d8a9f | test1  | test     | say-hello   | CREATED
ab58c816-0665-4cd3-b3f2-e16312e77887 | test2  | test     | say-hello   | CREATED
ff23e57b-6829-4abc-966d-82e2c6a90d9e | test3  | test     | say-hello   | CREATED
5bee0827-99bb-4bbb-ac7a-a7dd15e4bda3 | test4  | test     | say-hello   | CREATED
d4705a0a-bf6d-4f7e-a4fb-146a9bfe55a1 | test5  | test     | say-hello   | CREATED
b830798e-a4f8-45b1-b7bd-dd62adfb92ac | test6  | test     | say-hello   | CREATED
51e23e81-0655-436e-a1f0-550b6a1b3102 | test7  | test     | say-hello   | CREATED
6a87c9cf-19d5-4796-8d3b-1c6115067ad5 | test8  | test     | say-hello   | CREATED
cac08656-8056-452f-b709-bdadd21ccf46 | test9  | test     | say-hello   | CREATED
14f94e50-9076-4db5-ac3f-de7977717b8c | test10 | test     | say-hello   | CREATED
```

Hello World

`balsam submit-launch` :

Shortcut for Cobalt job submission (template in ~/.balsam)

```
[BalsamDB: test-db] msalim@thetalogin6:~> balsam submit-launch -n 2 -t 5 -q debug-cache-quad \
> -A SDL_Workshop -q training --job-mode=serial
Submit OK: Qlaunch {   'command': '/gpfs/mira-home/msalim/test-db/qsubmit/qlaunch1.sh',
    'from_balsam': True,
    'id': 1,
    'job_mode': 'serial',
    'nodes': 2,
    'prescheduled_only': False,
    'project': 'SDL_Workshop',
    'queue': 'training',
    'scheduler_id': 278079,
    'state': 'submitted',
    'wall_minutes': 5,
    'wf_filter': ''}
```

Hello World

If successful, jobs eventually marked JOB_FINISHED

```
[BalsamDB: test-db] msalim@thetalogin6:~/test-db/qsubmit> balsam ls
                                job_id |   name | workflow | application |        state
-------------------------------------------------------------------------------------------
cac08656-8056-452f-b709-bdadd21ccf46 | test9  | test     | say-hello   | JOB_FINISHED
b830798e-a4f8-45b1-b7bd-dd62adfb92ac | test6  | test     | say-hello   | JOB_FINISHED
ff23e57b-6829-4abc-966d-82e2c6a90d9e | test3  | test     | say-hello   | JOB_FINISHED
ab58c816-0665-4cd3-b3f2-e16312e77887 | test2  | test     | say-hello   | JOB_FINISHED
7df86c8b-f94b-4349-8ab6-a8afc85d8a9f | test1  | test     | say-hello   | JOB_FINISHED
6a87c9cf-19d5-4796-8d3b-1c6115067ad5 | test8  | test     | say-hello   | JOB_FINISHED
51e23e81-0655-436e-a1f0-550b6a1b3102 | test7  | test     | say-hello   | JOB_FINISHED
14f94e50-9076-4db5-ac3f-de7977717b8c | test10 | test     | say-hello   | JOB_FINISHED
d4705a0a-bf6d-4f7e-a4fb-146a9bfe55a1 | test5  | test     | say-hello   | JOB_FINISHED
5bee0827-99bb-4bbb-ac7a-a7dd15e4bda3 | test4  | test     | say-hello   | JOB_FINISHED
```

Where did the output go?

By default, everything goes into DB directory

```
[BalsamDB: test-db] msalim@thetalogin6:~/test-db> ls
balsamdb  data  log  qsubmit  server-info
```

Job working directories are created as:

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

```
[BalsamDB: test-db] msalim@thetalogin6:~/test-db> ls data/test/
test10_14f94e50  test2_ab58c816  test4_5bee0827  test6_b830798e  test8_6a87c9cf  testfail_fab575a3
test1_7df86c8b   test3_ff23e57b  test5_d4705a0a  test7_51e23e81  test9_cac08656
```

Error States

```
[BalsamDB: test-db] msalim@thetalogin6:~/test-db/data/test/testfail_fab575a3> balsam ls
                             job_id |     name | workflow | application |        state
------------------------------------------------------------------------------------------
fab575a3-01db-41b5-b70d-c396c17ef10d | testfail | test     | failer      | FAILED
cac08656-8056-452f-b709-bdadd21ccf46 | test9    | test     | say-hello   | JOB_FINISHED
b830798e-a4f8-45b1-b7bd-dd62adfb92ac | test6    | test     | say-hello   | JOB_FINISHED
ff23e57b-6829-4abc-966d-82e2c6a90d9e | test3    | test     | say-hello   | JOB_FINISHED
ab58c816-0665-4cd3-b3f2-e16312e77887 | test2    | test     | say-hello   | JOB_FINISHED
7df86c8b-f94b-4349-8ab6-a8afc85d8a9f | test1    | test     | say-hello   | JOB_FINISHED
6a87c9cf-19d5-4796-8d3b-1c6115067ad5 | test8    | test     | say-hello   | JOB_FINISHED
51e23e81-0655-436e-a1f0-550b6a1b3102 | test7    | test     | say-hello   | JOB_FINISHED
14f94e50-9076-4db5-ac3f-de7977717b8c | test10   | test     | say-hello   | JOB_FINISHED
d4705a0a-bf6d-4f7e-a4fb-146a9bfe55a1 | test5    | test     | say-hello   | JOB_FINISHED
5bee0827-99bb-4bbb-ac7a-a7dd15e4bda3 | test4    | test     | say-hello   | JOB_FINISHED
```

Error States

`balsam ls –state` :
Filter jobs by state

```
[BalsamDB: test-db] msalim@thetalogin6:~/test-db/data/test/testfail_fab575a3> balsam ls --state FAILED --history
Job testfail [fab575a3-01db-41b5-b70d-c396c17ef10d]
------------------------------------------

[10-03-2018 19:34:38.379895 CREATED]
[10-03-2018 19:38:24.490910 PREPROCESSED]
[10-03-2018 19:38:24.701099 RUNNING] Not scheduled by service
[10-03-2018 19:38:30.618931 RUN_ERROR]    Traceback (most recent call last):
    Hello from rank 2
    Hello from rank 1
      File "/gpfs/mira-home/msalim/test-db/qsubmit/fail.py", line 5, in <module>
        raise RuntimeError("simulated error")
    RuntimeError: simulated error
    Hello from rank 4
    Hello from rank 3
    Hello from rank 5
    Application 5762994 exit codes: 1
    Application 5762994 resources: utime ~2s, stime ~4s, Rss ~19956, inblocks ~22664, outblocks ~0

[10-03-2018 19:38:34.516193 FAILED]
```

Modifying Tasks

Modify BalsamJob fields from command line:

```
balsam modify fab5 state RESTART_READY
```

Or with more flexible Python API:

```
>>> from balsam.launcher.dag import BalsamJob
>>> BalsamJob.objects.filter(num_nodes__lte=128, name__contains="test", state="JOB_FINISHED").update(
... state="RESTART_READY")
10
```

Managing large job counts

Populating the Database

```
def prep_job(name, workflow, xyz_path):
    return BalsamJob(name=name,
                     workflow = workflow,
                     stage_in_url = xyz_path,
                     application = 'fhi-aims',
                     ranks_per_node=64,
                     cpu_affinity='depth',
                     environ_vars="OMP_NUM_THREADS=1",
                     )
```

Populating the Database

```
for (dirpath, dirnames, filenames) in os.walk(inbox_path):
    xyz_files = [f for f in filenames if f.endswith('.xyz')]
    new_jobs = []
    for f in xyz_files:
        name = os.path.splitext(f)[0]
        workflow = os.path.basename(dirpath)
        xyz_path = os.path.join(dirpath, f)
        job = prep_job(name, workflow, xyz_path)
        new_jobs.append(job)
    BalsamJob.objects.bulk_create(new_jobs)
    print("Created", len(new_jobs), "new jobs in DB")
```

The Launcher job template

~/.balsam/settings.json

```
"SCHEDULER_CLASS": "CobaltScheduler",
"SCHEDULER_SUBMIT_EXE": "/usr/bin/qsub",
"SCHEDULER_STATUS_EXE": "/usr/bin/qstat",
"DEFAULT_PROJECT": "datascience",
"SERVICE_PERIOD": 1,

"NUM_TRANSITION_THREADS": 5,
"MAX_CONCURRENT_MPIRUNS": 1000,

"LOG_HANDLER_LEVEL": "INFO",
"LOG_BACKUP_COUNT": 5,
"LOG_FILE_SIZE_LIMIT": 104857600,

"QUEUE_POLICY": "theta_policy.ini",
"JOB_TEMPLATE": "job-templates/theta.cobaltscheduler.tmpl"
```

The Launcher job template

```
#!/bin/bash -x
#COBALT -A {{ project }}
#COBALT -n {{ nodes }}
#COBALT -q {{ queue }}
#COBALT -t {{ time_minutes }}
#COBALT --attrs ssds=required:ssd_size=128
```

...

```
source balsamactivate {{ balsam_db_path }}
sleep 2

balsam launcher --{{ wf_filter }} --job-mode={{ job_mode }} --time-limit-minutes={{ time_minutes-2 }}

source balsamdeactivate
```

Configurable Queue Submission Policy

```
[debug-flat-quad]
submit-jobs = on
max-queued = 1
policy =  [
            {"min-nodes": 1,
             "max-nodes": 16,
             "min-time": 0,
             "max-time": 1
            }
          ]
```

Monitoring Progress

`balsam ls –by-states :`

Group by state

```
msalim@thetalogin6:/projects/CrystalsADSP/msalim> BALSAM_LS_LIMIT=2 balsam ls --by-states
```

Monitoring Progress

```
JOB_FINISHED  (573989 BalsamJobs)
                              job_id |                  name |         workflow | application |        state
-------------------------------------------------------------------------------------------------------------
7cceb07b-ea1b-4567-8dbc-6d0b4f8ba51e | gdb7-m4101-i2-c1-d67 | gdb7-m4101-i2-c1 | fhi-aims    | JOB_FINISHED
02b806d7-ddf4-42e8-9367-e6676541da55 | gdb7-m4012-i1-c6-d57 | gdb7-m4012-i1-c6 | fhi-aims    | JOB_FINISHED
(573993 more BalsamJobs not shown...)

PREPROCESSED  (72409 BalsamJobs)
                              job_id |                  name |         workflow | application |        state
-------------------------------------------------------------------------------------------------------------
91725337-6226-4302-8ddd-f97044612b1a | gdb7-m4002-i1-c6-d43 | gdb7-m4002-i1-c6 | fhi-aims    | PREPROCESSED
91827748-7550-41a4-bdb9-88ac1d2b131d | gdb7-m3492-i7-c1-d52 | gdb7-m3492-i7-c1 | fhi-aims    | PREPROCESSED
(72402 more BalsamJobs not shown...)

RESTART_READY  (5396 BalsamJobs)
                              job_id |                   name |          workflow | application |         state
-----------------------------------------------------------------------------------------------------------------
2c03e5cd-2c94-4376-8c1c-5cd8abb6adbf | gdb7-m4449-i2-c7-d97  | gdb7-m4449-i2-c7  | fhi-aims    | RESTART_READY
2c0325ac-3135-424c-b332-17909e4e6c13 | gdb7-m5074-i1-c10-d70 | gdb7-m5074-i1-c10 | fhi-aims    | RESTART_READY
(5394 more BalsamJobs not shown...)

RUN_DONE  (17 BalsamJobs)
                              job_id |                  name |         workflow | application |    state
-------------------------------------------------------------------------------------------------------------
34020491-3c31-4dc2-a9a4-5aa622fc517d | gdb7-m4960-i2-c4-d75 | gdb7-m4960-i2-c4 | fhi-aims    | RUN_DONE
251ac1f1-566f-446b-9245-0a1079a701f0 | gdb7-m3695-i1-c5-d96 | gdb7-m3695-i1-c5 | fhi-aims    | RUN_DONE
(10 more BalsamJobs not shown...)

RUNNING  (797 BalsamJobs)
```

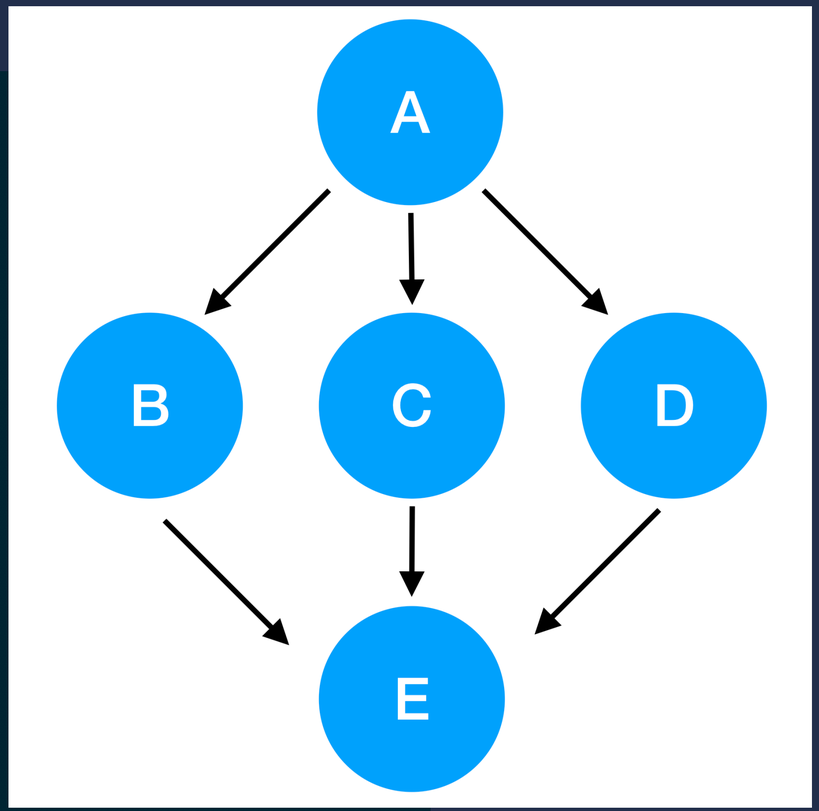
Dependencies and Dynamic Workflows

Defining Dependencies

```
from balsam.launcher.dag import (
        add_job, add_dependency)

A = add_job(name="A", application="generate")
B,C,D = [
        add_job(
            name=name,
            application="simulate",
            input_files=name+".inp"
        )
        for name in "BCD"
        ]
E = add_job(name="E", application="reduce", input_files="*.out")

for job in B,C,D:
    add_dependency(A, job)
    add_dependency(job,E)
```



Dynamic Workflows

- `balsam.launcher.dag` **enables context-aware processing**
  - `dag.current_job`

- **Attach pre/post-processing scripts to application**
  - **Inspect** `dag.current_job`
  - **Modify DB in response to workflow outcomes**

- `dag.kill(job)`
  - **launcher terminates and replaces tasks in near-realtime**

Dynamic Workflows

- **In more tightly-coupled workflows, a “master” app running under Balsam may itself spawn tasks for parallel, asynchronous execution**

- `balsam.launcher.async` **facilitates programmatic polling/ processing of BalsamJobs**

concurrent.futures-inspired polling

```
from balsam.launcher.dag import add_job
from balsam.launcher.async import wait, FutureTask
```

```
def on_done(job):
    return job.read_file_in_workdir(f'{job.name}.out')

futures = []
for i in range(10):
    j = add_job(name=f'task{i}',
                application='say-hello',
                args=f'world {i}!'
                )
    futures.append(FutureTask(j, on_done))

results = wait(futures, return_when='ANY_COMPLETED', timeout=30)
```

Thank you!

- https://github.com/balsam-alcf/balsam

- Tom Uram
- Taylor Childers
- Venkat Vishwanath
- Prasanna Balaprakash
- ADSP teams

Argonne Leadership Computing Facility
DOE Office of Science
Supported under Contract DE-AC02-06CH11357