Using Gold, Moab, and Torque to Run a Campus-wide Allocation-based HPC Service

Andrew Caird Matthew Britt

11 April 2012

Who We Are

- College of Engineering "centralized" HPC support
- Been trying this for 15+ years
- We are four full-time CoE employees, two full-time campus employees, and much support from Engineering Central IT
- We also get business and data center support from U-M Central IT
- We also get end-user support from many Colleges and Departments at U-M

What

- Using Moab/Torque/Gold for several years
- Condo-style clusters for even longer
- Two years ago, we started an allocation-based service

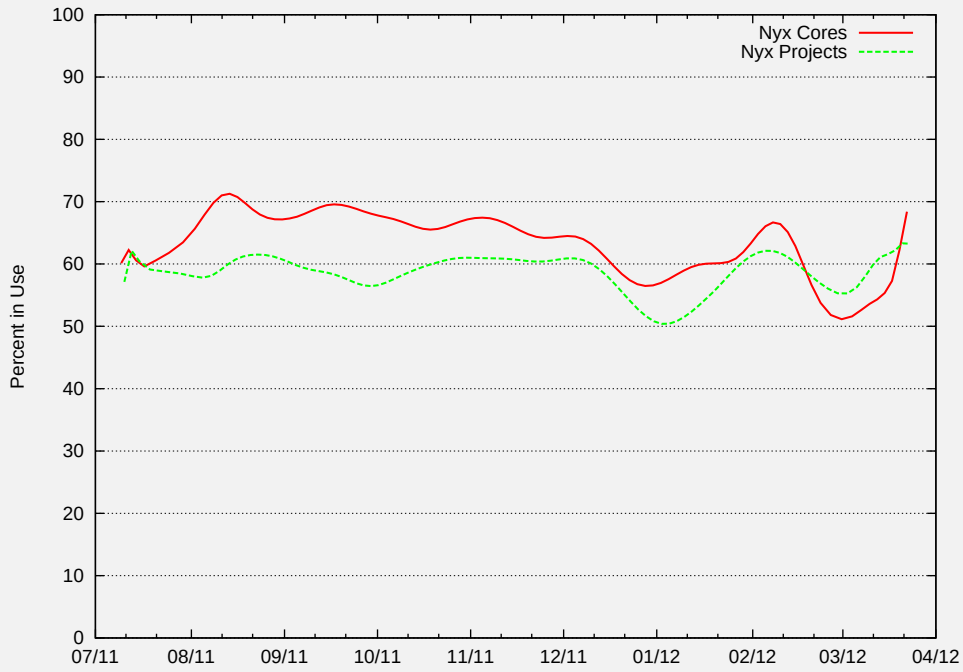
disclaimer: this is what we do, but that doesn't make it right

What we started with

This isn't *really* what we started with, but what we have now that stems from what we started with. It's not that different, though.

- condo-style cluster of 778 nodes providing $\sim 5,700$ cores
- 40+ owners who own from 1 node (8 cores) to 159 nodes (1288 cores)
- $\sim 60\%$ utilization by core
- $\sim 60\%$ of projects running jobs at any given time
- Moab (for lots of custom scheduling), Torque (just because), Gold (for record keeping, but no policy enforcement)

Recent Condo Usage



The good and bad

This is good because:

- 60–70% isn't all bad
- everyone gets their own computers as they want them
- it supports "one-time" funds for capital purchases

This is bad because:

- 30–40% of $\sim$ 6000 cores is:
  - 1800–2400 cores
  - 150–200 computers or more
  - 6–8 racks or more
  - $60,000–$80,000 / year or more in facilities costs
- provisioning is slow and requires a lot of staff time

Good, Bad, but mostly Good

This was mostly good, though, because

- we were supporting 40 research groups with 4 FTE, pretty much
- we gave the impression that we could scale
- researchers were pretty happy

So... the rest of campus asked us to extend our services to them. Our estimates were that there was another 12,000 cores of demand outside of the College of Engineering.

However... we knew better than to have $\sim 6,000$ idle cores on campus.

What we did

We started with a few goals

- we wanted something we could over-subscribe
- we wanted something researchers could "rent" in small time increments
- we wanted it to feel, at least a little, like getting a computer (even though it totally isn't like that)
- we needed to do *full cost recovery*

Then we started another condo as part of our existing cluster

- except it was a partition, which the others aren't
- and it a name ("Flux") to distinguish its policy set

Then we called Adaptive

Hi, Scott!

Extensions to Gold

To satisfy our goals

- we knew we needed allocations (not standing reservations), so we had to use Gold as it was intended to be used
- we wanted to constrain the allocations to guard against uneven usage which would compromise our ability to honor commitments
  - this required an extension to Gold's `g_allocation` table called MaxProc (`g_max_proc`)
- we also added `g[ls|mk|rm|ch]qos` commands to get everything into Gold

Gold ↔ Moab

The Moab → Gold conversation is pretty well described and we had been doing that for a while:

Moab → Gold

```
AMCFG[bank] SERVER=gold://nyx JOBFAILUREACTION=HOLD,HOLD
CHARGEPOLICY=DEBITALLBLOCKED TIMEOUT=30
```

The Gold → Moab conversation was new to us, but is similar:

Gold → Moab

```
IDCFG[cacid] SERVER=file:///opt/moab/etc/id.cfg
REFRESHPERIOD=minute CREATECRED=True
```

All you need is a file called `id.cfg`

id.cfg: a simple example

A simple `id.cfg` looks like:

user

user:acaird alist=project1_flux qlist=project1_flux

account

account:project1_flux qdef:project1_flux maxproc=12

qos

qos:project1_flux

id.cfg

Our `id.cfg` is produced by a Perl script and contains lines of three types that look like these

user

user:acaird alist=project1_flux,project2_flux,project3 adef=project1_flux qlist=project1_flux,project2_flux,project3 qdef=project1_flux emailaddress=acaird@umich.edu

account

account:project1_flux qdef:project1_flux maxproc=12,12 maxmem=49152 maxps=" fstarget=0.19 plist=flux

qos

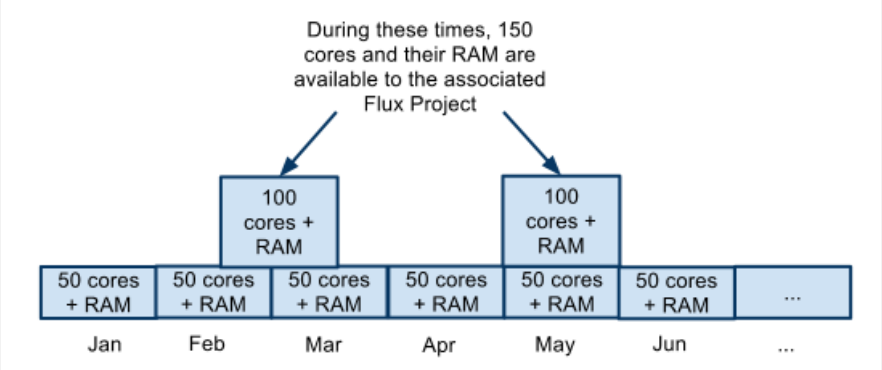
qos:project1_flux

Everything in each type comes from Gold except `fstarget`, `maxproc` hard limit, and `maxmem` which are computed by the Perl script (sometimes called "business logic")

Allocations

Allocations can do a limited number of things:

- be created, along with their projects and accounts if needed
- be added to other allocations
- be ended



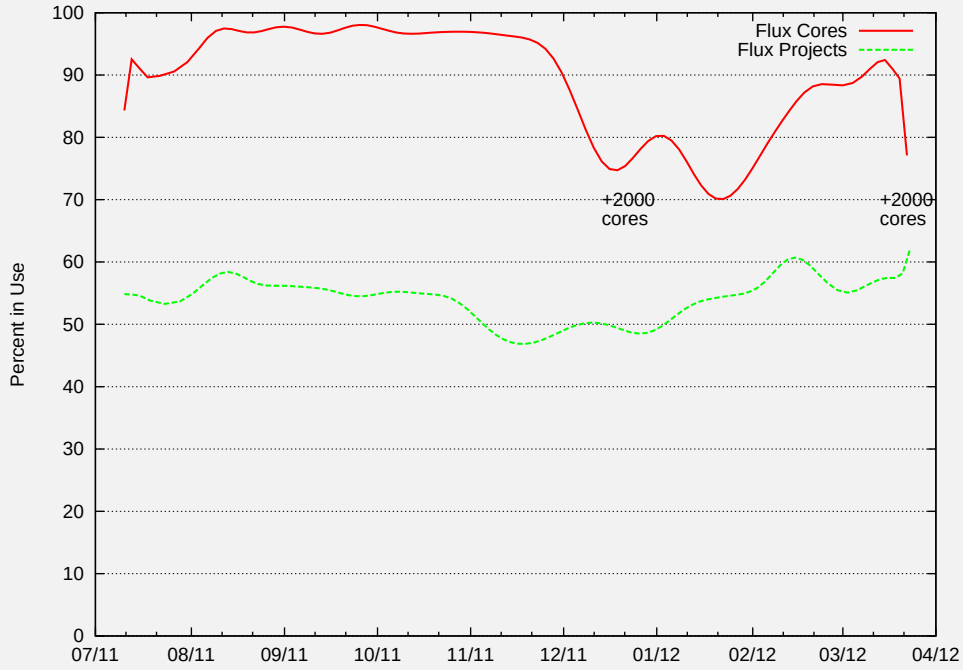
Those three actions have been sufficient for us for 18 months and over 800 allocations ($\sim$ 2 per business day).

What we got

Today our allocation-based cluster is

- 508 nodes providing $\sim$ 6000 cores
- 175+ projects (research groups and classes) in total over 24 months
- $\sim$ 90% utilization by core
- $\sim$ 60% of allocations running jobs at any given time
- Moab (for pretty basic scheduling), Torque (just because), Gold (for record keeping and policy enforcement)

Recent Allocation Usage



The good and bad

This is good because:

- 90% is pretty good
- very fast provisioning
- adaptable to research or course demands
- supporting 4x the researchers with little more staff is nice
- nice integration with U-M business processes for billing and reporting

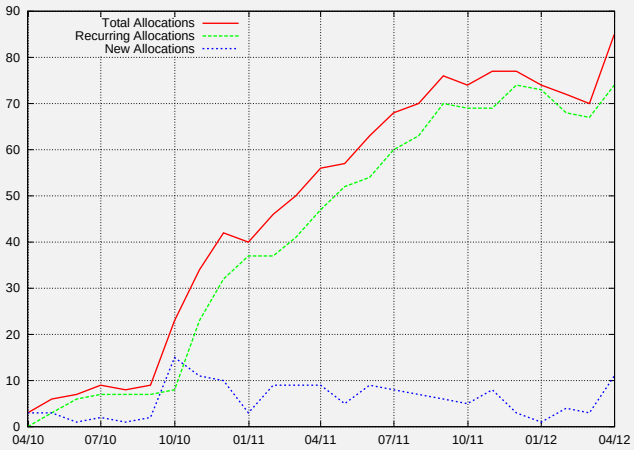
This is bad because:

- on-going costs are difficult for some grants
- the over-subscribed system can introduce queue delays
- homogeneous hardware may not be perfect for everyone

Good, Bad, but mostly Good

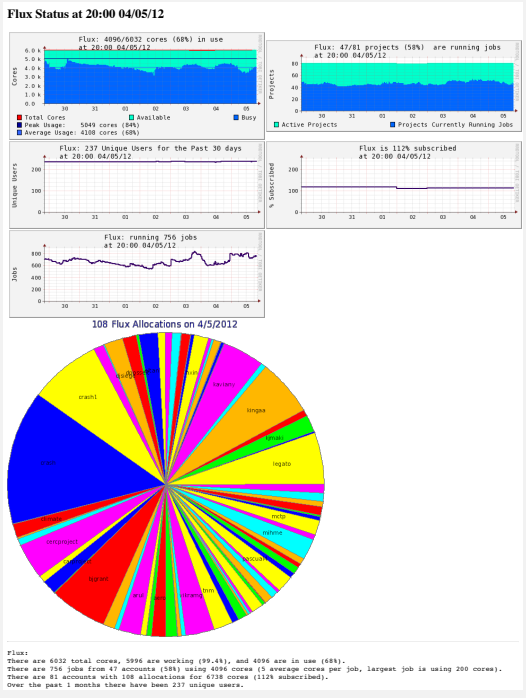
This is mostly good

- we have added 6 new projects (research groups or classes) every month for the last 12
- we have 81 active projects with 108 allocations
- we have added 4,000 cores since we started due to demand



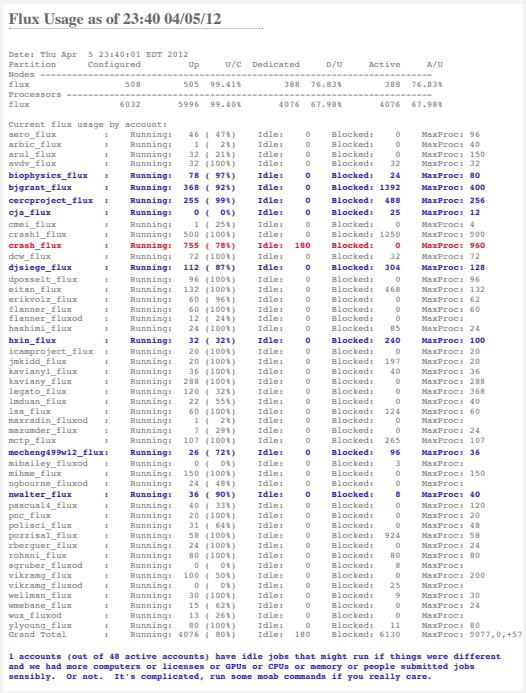
Some details

- the allocation-based cluster, “Flux”, is a partition in our one cluster
- one Torque server, one Moab instance, one Gold instance for 1,286 nodes providing 11,732 cores
- since January, we start a job every 15 seconds, on average



Some details

- we also have an "on-demand" product that is pay-for use
- we are adding a larger memory product with a different rate, but the same policies
- we use allocation state to manage scratch space
- we learned memory is important to control



Flux Usage as of 23:40 04/05/12

```
Date: Thu Apr  5 23:40:01 EDT 2012
Partition      Configured        Up     U/C  Dedicated     D/U      Active     A/U
Nodes ------------------------------------------------------------------------------
flux                508       505  99.41%         388  76.83%         388  76.83%
Processors -------------------------------------------------------------------------
flux               6032      5996  99.40%        4076  67.98%        4076  67.98%

Current flux usage by account:
aero_flux       :    Running:   46 ( 47%)    Idle:    0    Blocked:    0    MaxProc: 96
arbic_flux      :    Running:    1 (  2%)    Idle:    0    Blocked:    0    MaxProc: 40
arul_flux       :    Running:   32 ( 21%)    Idle:    0    Blocked:    0    MaxProc: 150
avdv_flux       :    Running:   32 (100%)    Idle:    0    Blocked:   32    MaxProc: 32
biophysics_flux :    Running:   78 ( 97%)    Idle:    0    Blocked:   24    MaxProc: 80
bjgrant_flux    :    Running:  368 ( 92%)    Idle:    0    Blocked: 1392    MaxProc: 400
cercproject_flux:    Running:  255 ( 99%)    Idle:    0    Blocked:  488    MaxProc: 256
cja_flux        :    Running:    0 (  0%)    Idle:    0    Blocked:   25    MaxProc: 12
cmei_flux       :    Running:    1 ( 25%)    Idle:    0    Blocked:    0    MaxProc: 4
crashl_flux     :    Running:  500 (100%)    Idle:    0    Blocked: 1250    MaxProc: 500
crash_flux      :    Running:  755 ( 78%)    Idle:  180    Blocked:    0    MaxProc: 960
dcw_flux        :    Running:   72 (100%)    Idle:    0    Blocked:   32    MaxProc: 72
djsiege_flux    :    Running:  112 ( 87%)    Idle:    0    Blocked:  304    MaxProc: 128
dposselt_flux   :    Running:   96 (100%)    Idle:    0    Blocked:    0    MaxProc: 96
eitan_flux      :    Running:  132 (100%)    Idle:    0    Blocked:  468    MaxProc: 132
erikvolz_flux   :    Running:   60 ( 96%)    Idle:    0    Blocked:    0    MaxProc: 62
flanner_flux    :    Running:   60 (100%)    Idle:    0    Blocked:    0    MaxProc: 60
flanner_fluxod  :    Running:   12 ( 24%)    Idle:    0    Blocked:    0    MaxProc:
hashimi_flux    :    Running:   24 (100%)    Idle:    0    Blocked:   85    MaxProc: 24
hxin_flux       :    Running:   32 ( 32%)    Idle:    0    Blocked:  240    MaxProc: 100
icamproject_flux:    Running:   20 (100%)    Idle:    0    Blocked:    0    MaxProc: 20
jmkidd_flux     :    Running:   20 (100%)    Idle:    0    Blocked:  197    MaxProc: 20
kavianyl_flux   :    Running:   36 (100%)    Idle:    0    Blocked:   40    MaxProc: 36
kaviany_flux    :    Running:  288 (100%)    Idle:    0    Blocked:    0    MaxProc: 288
legato_flux     :    Running:  120 ( 32%)    Idle:    0    Blocked:    0    MaxProc: 368
lmduan_flux     :    Running:   22 ( 55%)    Idle:    0    Blocked:    0    MaxProc: 40
lsa_flux        :    Running:   60 (100%)    Idle:    0    Blocked:  124    MaxProc: 60
maxradin_fluxod :    Running:    1 (  2%)    Idle:    0    Blocked:    0    MaxProc:
mazumder_flux   :    Running:    7 ( 29%)    Idle:    0    Blocked:    0    MaxProc: 24
mctp_flux       :    Running:  107 (100%)    Idle:    0    Blocked:  265    MaxProc: 107
mecheng499w12_flux:  Running:   26 ( 72%)    Idle:    0    Blocked:   96    MaxProc: 36
mibailey_fluxod :    Running:    0 (  0%)    Idle:    0    Blocked:    3    MaxProc:
mihme_flux      :    Running:  150 (100%)    Idle:    0    Blocked:    0    MaxProc: 150
ngbourne_fluxod :    Running:   24 ( 48%)    Idle:    0    Blocked:    0    MaxProc:
nwalter_flux    :    Running:   36 ( 90%)    Idle:    0    Blocked:    8    MaxProc: 40
pascual4_flux   :    Running:   40 ( 33%)    Idle:    0    Blocked:    0    MaxProc: 120
pnc_flux        :    Running:   20 (100%)    Idle:    0    Blocked:    0    MaxProc: 20
polisci_flux    :    Running:   31 ( 64%)    Idle:    0    Blocked:    0    MaxProc: 48
pozzisal_flux   :    Running:   58 (100%)    Idle:    0    Blocked:  924    MaxProc: 58
rberguer_flux   :    Running:   24 (100%)    Idle:    0    Blocked:    0    MaxProc: 24
rohani_flux     :    Running:   80 (100%)    Idle:    0    Blocked:   80    MaxProc: 80
sgruber_fluxod  :    Running:    0 (  0%)    Idle:    0    Blocked:    8    MaxProc:
vikramg_flux    :    Running:  100 ( 50%)    Idle:    0    Blocked:    0    MaxProc: 200
vikramg_fluxod  :    Running:    0 (  0%)    Idle:    0    Blocked:   25    MaxProc:
wellman_flux    :    Running:   30 (100%)    Idle:    0    Blocked:    9    MaxProc: 30
wmebane_flux    :    Running:   15 ( 62%)    Idle:    0    Blocked:    0    MaxProc: 24
wux_fluxod      :    Running:   13 ( 26%)    Idle:    0    Blocked:    0    MaxProc:
ylyoung_flux    :    Running:   80 (100%)    Idle:    0    Blocked:   11    MaxProc: 80
Grand Total     :    Running: 4076 ( 80%)    Idle:  180    Blocked: 6130    MaxProc: 5077,0,+57

1 accounts (out of 48 active accounts) have idle jobs that might run if things were different
and we had more computers or licenses or GPUs or CPUs or memory or people submitted jobs
sensibly.  Or not.  It's complicated, run some moab commands if you really care.
```


Acknowledgments

- my co-workers, who actually did this
- the Vice President for Research CyberInfrastructure, Dr. Dan Atkins, who actually funded this
- Adaptive Computing's support and professional services, who decoded what the developers did so we could pay for and do this

Questions?

Andy	Matt
acaird@umich.edu	msbritt@umich.edu