ALPS User Tutorial (Base Camp)

Michael Karo

mek@cray.com

CUG 2007

CRAY
THE SUPERCOMPUTER COMPANY

Topics

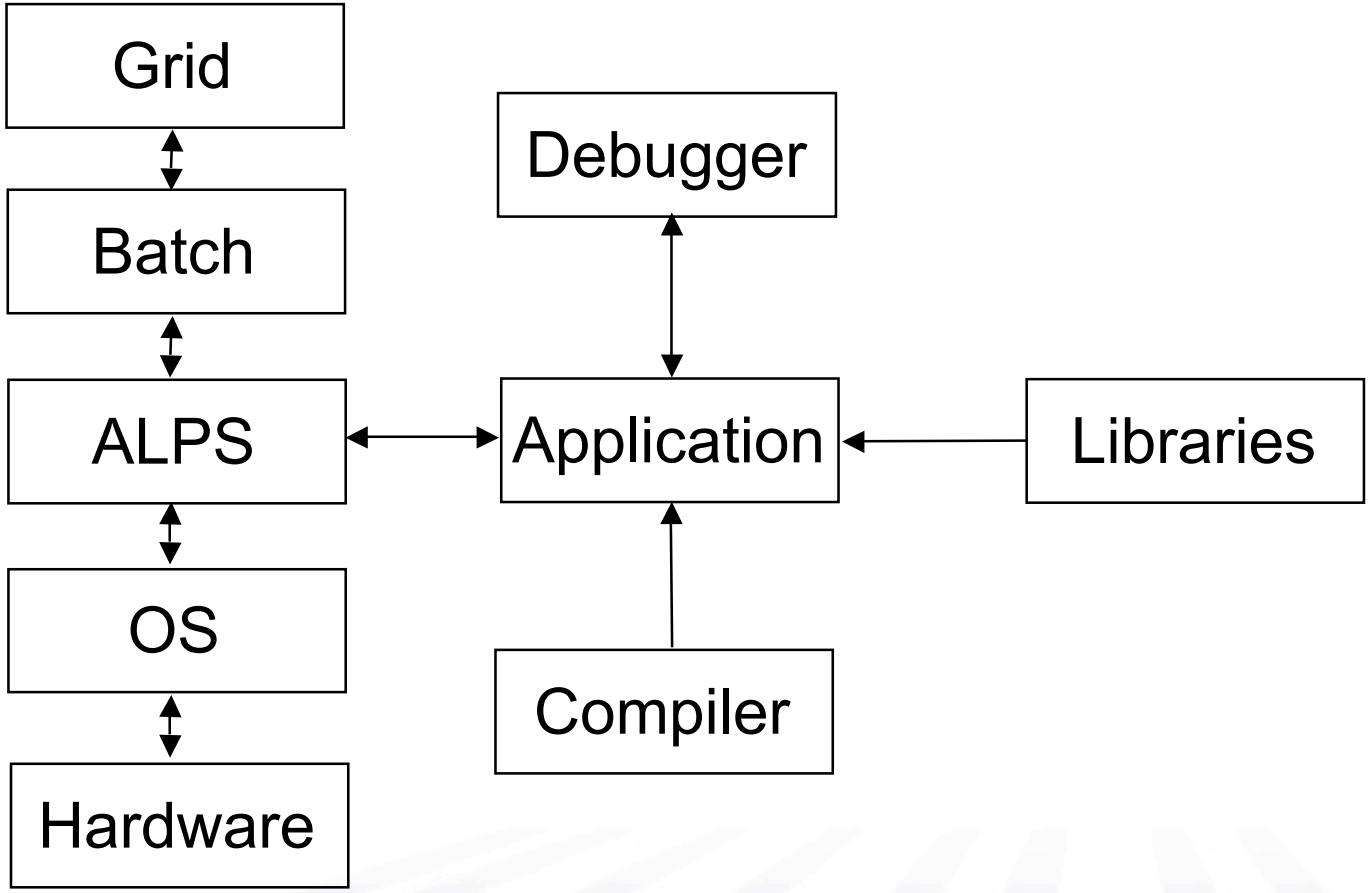
- ALPS Overview
- Launching applications
- Node attributes
- Application status information
- ALPS and PBS Pro
- ALPS, MPI, and OpenMP
- Debugging Applications
- MPMD application launch
- Heterogeneous systems
- Troubleshooting ALPS

Base Camp

Summit

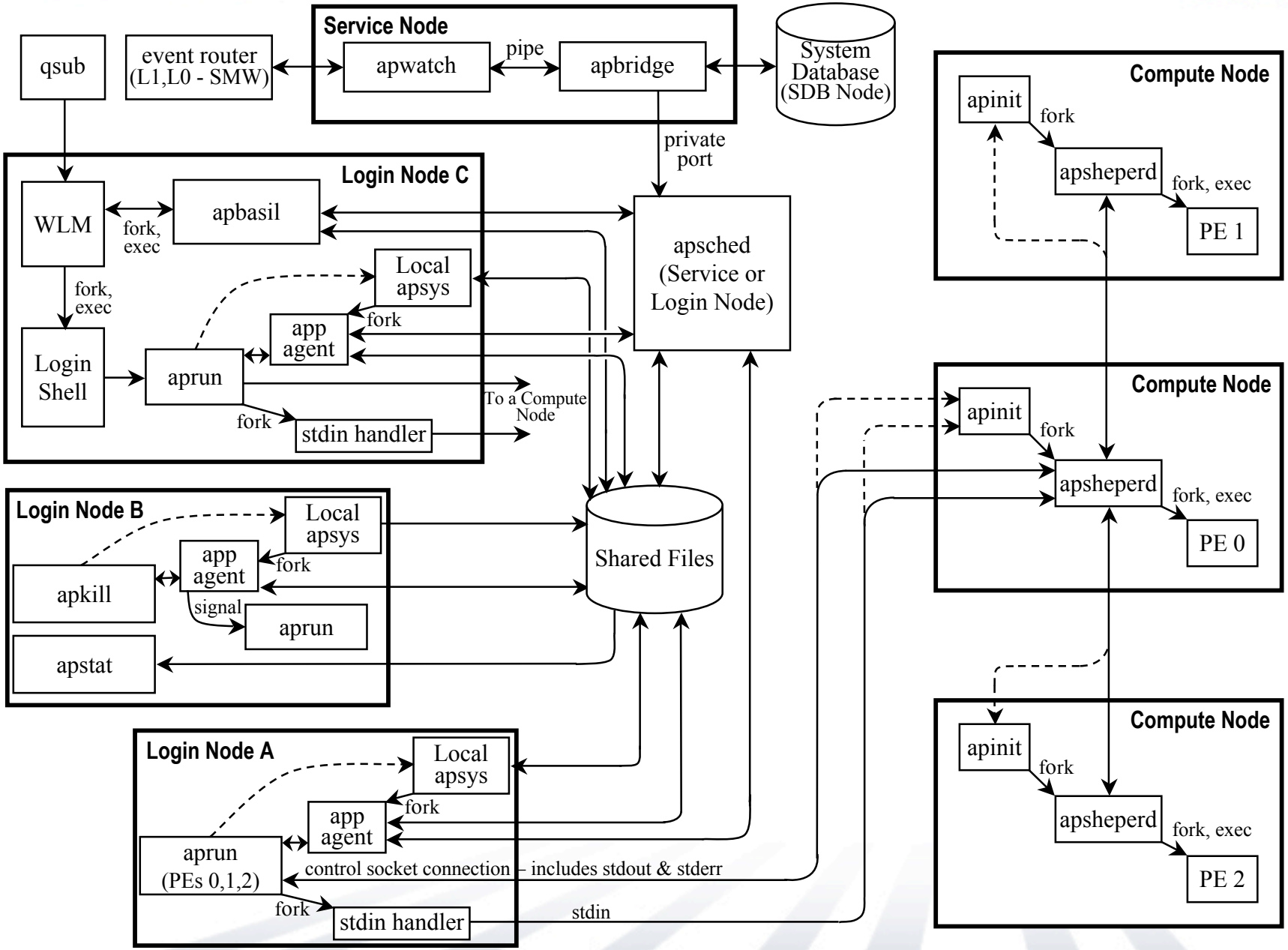
ALPS Overview

- ALPS = **A**pplication **L**evel **P**lacement **S**cheduler



The ALPS Suite

- Clients
  - aprun – Application submission
  - apstat – Application status
  - apkill – Signal delivery
  - apmgr – Administration interface
  - apbasil – Workload manager interface
- Servers
  - apsys – Client interaction on login nodes
  - apinit – Process management on compute nodes
  - apsched – Reservations and placement
  - apbridge – System data collection
  - apwatch – Event monitoring
- And now, the big picture…



ALPS “hostname” Launch

- Use the aprun client with the following parameters:
  - -b = Bypass binary transfer
  - -n 4 = Specify a width of four PEs

```
$ ssh crayadm@pike
    Welcome back to pike...
    The system is now running the XT 2.0 OS
/home/crayadm> aprun -b -n 4 /bin/hostname
aprun: [NID 24]Exec /bin/hostname failed:
                    chdir /home/crayadm No such file or directory
/home/crayadm> echo $?
1
/home/crayadm>
```

- What happened?

ALPS “hostname” Launch (again)

- What went wrong…
  - Application initialization includes changing to the current directory.
  - /home/crayadm did not exist on the compute node.
  - We must be in /tmp or a Lustre mounted directory.
- Try again…

```
/home/crayadm> cd /tmp
/tmp> aprun -b -n 4 /bin/hostname
aprun: [NID 25]Apid 122: cannot execute: exit(107) exec failed
/tmp> echo $?
1
/tmp>
```

- Now what?

ALPS “hostname” Launch (or something like it)

- What went wrong…
  - CNL is very lightweight, no “hostname” command is installed.
  - Instead of “hostname”, we can acquire the NID.
- One more time…

```
/home/crayadm> cd /tmp
/tmp> aprun -b -n 4 /bin/cat /proc/cray_xt/nid
44
45
46
47
Application 123 resources: utime 0, stime 0
/tmp> echo $?
0
/tmp>
```

- Eureka!

ALPS “hostname” Launch (the real thing)

- But, I really want to launch hostname!
- Take the “-b” off and ALPS will distribute the binary

```
/tmp> aprun -n 4 /bin/hostname
nid00044
nid00045
nid00046
nid00047
Application 124 resources: utime 0, stime 0
/tmp> ldd /bin/hostname
        libc.so.6 => /lib64/tls/libc.so.6 (0x00002b5288722000)
$
```

- Caution 1: If the binary requires dynamic libraries, they must be installed on the compute nodes.
- Caution 2: This works for x86_64 compute nodes only!

Preparing an Application

- Choose your compiler
  - gcc
  - PathScale
  - PGI

```
/tmp> PS1="\$ "
$ module purge
$ module load Base-opts PrgEnv-gnu xtpe-target-cnl
$ module swap PrgEnv-gnu PrgEnv-pathscale
$ module swap PrgEnv-pathscale PrgEnv-pgi
$
```

A Simple MPI “Hello, world!”

```
$ cat hello.c

#include "mpi.h"

int main(int argc, char *argv[])
{
    int rank, nid;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    PMI_CNOS_Get_nid(rank, &nid);
    printf("Hello from rank %d on nid%05d\n", rank, nid);
    MPI_Finalize();
    return(0);
}

$
```

Compile the Application

```
$ cc -g -o hello hello.c
/opt/xt-pe/2.1/bin/snos64/cc: INFO: linux target is being used
hello.c:
$ file hello
hello: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV),
       for GNU/Linux 2.4.1, statically linked, not stripped
$ ls -l hello
-rwxr-xr-x  1 crayadm crayadm 4827204 Apr 30 10:40 hello
$ strip hello
$ file hello
hello: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV),
       for GNU/Linux 2.4.1, statically linked, stripped
$ ls -l hello
-rwxr-xr-x  1 crayadm crayadm 1127976 Apr 30 10:45 hello
$
```

- Stripping binary reduces file size.

Launch the Application

- Invoke aprun with the following parameters:
  - Specify a width of four PEs (-n 4)
  - Do not specify “-b”, ALPS will distribute the binary.

```
$ aprun -n 4 ./hello
Hello from rank 0 on nid00024
Hello from rank 1 on nid00025
Hello from rank 2 on nid00026
Hello from rank 3 on nid00027
Application 124 resources: utime 0, stime 0
$
```

The aprun Client, Explained

```
$ man aprun
Reformatting aprun(1), please wait...
aprun(1)                                                    aprun(1)



NAME
       aprun -- Launches an application

SYNOPSIS
       aprun [-a sys] [-b] [-d depth] [-D value | -q]
            [-h hard_label | -s soft_label]
            [-L nodes] [-m size] [-n procs]
            [-N pes] [-S] [-t sec] executable [options]

       [: -n pes [-d depth] [-N pes] [-a sys] [-L nodes] executable_2

       [: -n pes [-d depth] [-N pes] [-a sys] [-L nodes] executable_3 [:...]]]

IMPLEMENTATION
       UNICOS/lc systems with CNL
...
```

Terminology

- Node
  - All resources managed by a single CNL instance
- Processing Element (PE)
  - ALPS launched binary invocation on a compute node
- Width (aprun -n)
  - Number of PEs to Launch
  - Applies to both single-core and multi-core systems
- Depth (aprun -d)
  - Number of threads per PE (OpenMP)
  - Applies to multi-core systems only
- PEs Per Node / PPN (aprun -N)
  - Number of PEs per CNL instance (multiple MPI ranks per node)
  - Applies to multi-core systems only
- Node List (aprun -L)
  - A user supplied list of candidate nodes to constrain placement

Application Width (aprun -n)

```
-n pes    Specifies number of PEs needed. Default
          is 1. The application will be allocated the
          number of processor cores determined by the
          depth multiplied by pes.
```

- Specifies number of PEs
- MPI rank is derived from PE count
- λ = (width * depth)
- ALPS reserves λ processor cores for use by the application
- Applies to both single-core and multi-core systems

Application Depth (aprun -d)

```
-d depth   Specifies the depth (number of threads) of
           each PE. The meaning of this option is
           dependent on the programming model. Default
           is 1. The application will be allocated the
           number of processors determined by the depth
           multiplied by the pes. This option should be
           specified when using OpenMP code.
```

- Specifies number of threads per PE
- Not part of MPI rank assignment
- (width * depth) ≅ (pes * threads)
- ALPS invokes width instances of the binary
- Application spawns (depth - 1) additional threads per PE
- Applies only to multi-core systems
- Compute nodes must have at least depth cores

Application PEs Per Node (aprun -N)

```
-N pes    Specifies number of processing elements per node.
```

- Specifies number of PEs per node
- Not part of MPI rank assignment
- When specified…
  - Places specified number of PEs per node
  - A sufficient number of cores must exist on each node
- When not specified…
  - Allows ALPS to pack PEs tightly
  - Behavior dependent upon application and system resources
- Applies only to multi-core systems
- ALPS assigns the same number of PEs to all nodes, regardless of whether PPN is specified. (DM requirement)

Node List (aprun -L)

```
-L node_list     Specifies a user-defined candidate node list.
                 The syntax allows a comma-separated list of
                 nodes (node[,node), a range of nodes
                 (node1-node2]...), and a combination of both
                 formats. Node values can be expressed in
                 decimal, octal, and hexadecimal. The first
                 number in a range must be less than the second
                 number (i.e., 8-6 is invalid). A complete node
                 list is required. If the candidate node list is
                 too short for the -n, -d, and -N options, a
                 fatal error is produced. This option can be
                 specified multiple times.
```

- Specifies a list of NIDs as candidates for placement
- The node list is a superset of the placement list
- Multiple lists may be specified, but only one is used
- Applies to both single-core and multi-core systems

The Test System

```
$ apstat -n
  NID Arch St CPUs  PgSz     Avl    Conf  Placed  PEs Apids
   44  XT3 UP -       4K  256000       0       0    0
   45  XT3 UP -       4K  256000       0       0    0
   46  XT3 UP -       4K  256000       0       0    0
   47  XT3 UP -       4K  256000       0       0    0
   48  XT3 UP -       4K  256000       0       0    0
   49  XT3 UP -       4K  256000       0       0    0
   50  XT3 UP -       4K  256000       0       0    0
   51  XT3 UP -       4K  256000       0       0    0
   52  XT3 UP -       4K  256000       0       0    0
   53  XT3 UP -       4K  256000       0       0    0
   54  XT3 UP -       4K  256000       0       0    0
   55  XT3 UP -       4K  256000       0       0    0
   56  XT3 UP --      4K  768000       0       0    0
   57  XT3 UP --      4K  768000       0       0    0
   58  XT3 UP --      4K  768000       0       0    0
   59  XT3 UP --      4K  768000       0       0    0
   60  XT3 UP --      4K  512000       0       0    0
   61  XT3 UP --      4K  512000       0       0    0
   62  XT3 UP --      4K  512000       0       0    0
   63  XT3 UP --      4K  512000       0       0    0
Compute node summary: up: 20 idle: 20
$
```

Specifying Width

```
$ aprun -n 4 ./hello
Hello from rank 1 on nid00056
Hello from rank 0 on nid00056
Hello from rank 3 on nid00057
Hello from rank 2 on nid00057
Application 125 resources: utime 0, stime 0
$
```

- ALPS finds the “best” place.
- Packs tightly, minimizes node count.

Specifying PEs Per Node

```
$ aprun -n 4 -N 1 ./hello
Hello from rank 0 on nid00044
Hello from rank 1 on nid00045
Hello from rank 2 on nid00046
Hello from rank 3 on nid00047
Application 129 resources: utime 0, stime 0
$ aprun -n 4 -N 2 ./hello
Hello from rank 0 on nid00056
Hello from rank 1 on nid00056
Hello from rank 3 on nid00057
Hello from rank 2 on nid00057
Application 130 resources: utime 0, stime 0
$ aprun -n 4 -N 4 ./hello
aprun: -N * -d values are invalid for this system
$
```

- Try packing one, two, and four PEs per node
- No quad-core nodes… -N 4 failed, as expected

MPI and OpenMP Enabled hello.c

```
$ cat hello.c

#include <mpi.h>
#include <omp.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int rank, nid, thread;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    PMI_CNOS_Get_nid(rank, &nid);
    #pragma omp parallel private(thread)
    {
        thread = omp_get_thread_num();
        #pragma omp barrier
        printf("Hello from rank %d (thread %d) on nid%05d",
                rank, thread, nid);
        if (thread == 0)
            printf(" <-- MASTER\n");
        else
            printf(" <-- slave\n");
    }
    MPI_Finalize();
    return(0);
}

$
```

Specifying Depth

```
$ cc -mp -g -o hello hello.c
/opt/xt-pe/2.0.03/bin/snos64/cc: INFO: linux target is being used
hello.c:
$ strip hello
$ export OMP_NUM_THREADS=2
$ aprun -n 2 -d 2 ./hello
Hello from rank 0 (thread 0) on nid00060 <-- MASTER
Hello from rank 0 (thread 1) on nid00060 <-- slave
Hello from rank 1 (thread 0) on nid00061 <-- MASTER
Hello from rank 1 (thread 1) on nid00061 <-- slave
Application 136 resources: utime 0, stime 0
$
```

- Compile application with OpenMP awareness
- Set OMP_NUM_THREADS
- Specify a corresponding depth for aprun

Specifying Node Lists

```
$ aprun -n 8 -d 2 -L56-59 ./hello
aprun: at least one command's user NID list is short
$ aprun -n 2 -d 2 -L56-59 ./hello
Hello from rank 0 (thread 0) on nid00056 <-- MASTER
Hello from rank 0 (thread 1) on nid00056 <-- slave
Hello from rank 1 (thread 0) on nid00057 <-- MASTER
Hello from rank 1 (thread 1) on nid00057 <-- slave
Application 159 resources: utime 0, stime 0
$
```

- First attempt fails because width is greater than node list length
- Second attempt succeeds with width smaller than node list length
- ALPS uses node list to constrain placement candidates

Application Memory Requirements

```
 -m size    Specifies the per processing element maximum
            Resident Set Size memory limit in megabytes.
            K|M|G suffixes are supported (16 = 16M = 16
            megabytes). Any truncated or full spelling of
            unlimited is recognized. See the NOTES section
            on how to determine the -m default value.

NOTES: The -m option specifies the application per processing
       element maximum Resident Set Size (RSS) memory limit.
       If the -m option is not specified, the default value
       assigned is set to the minimum compute node memory
       size divided by the maximum number of processors on
       any compute nodes.
```

- Memory requirements are per PE
- Aggregate OS memory usage not enforced by CNL
- XT restriction: one application per node due to lack of enforceability
- More on this when we discuss X2 compute nodes

Specifying Memory

```
$ OMP_NUM_THREADS=1
$ aprun -n 1 -m 1000m ./hello
Hello from rank 0 (thread 0) on nid00044 <-- MASTER
Application 175 resources: utime 0, stime 0
$ aprun -n 1 -m 3000m ./hello
Hello from rank 0 (thread 0) on nid00056 <-- MASTER
Application 176 resources: utime 0, stime 0
$ aprun -n 1 -m 3001m ./hello
aprun: request exceeds max memory
$
```

- First launch placed on smaller memory node.
- Second launch placed on larger memory node.
- Third launch fails, memory requirement cannot be met.

Specifying Memory And PPN

```
$ aprun -n 2 -N 2 -m 1501m ./hello
aprun: request exceeds max CPUs, memory
$ aprun -n 2 -N 2 -m 1500m ./hello
Hello from rank 0 (thread 0) on nid00056 <-- MASTER
Hello from rank 1 (thread 0) on nid00056 <-- MASTER
Application 181 resources: utime 0, stime 0
$ aprun -n 2 -N 2 -m 500m ./hello
Hello from rank 1 (thread 0) on nid00056 <-- MASTER
Hello from rank 0 (thread 0) on nid00056 <-- MASTER
Application 182 resources: utime 0, stime 0
$
```

- First launch fails… (2 * 1501) = 3002
- Second launch placed on larger memory node.
- Third launch uses larger memory node due to PPN.

Node Attributes

- Node attributes do…
  - Provide additional details about compute nodes
    - Architecture
    - OS class
    - Clock speed
    - Memory page size and total pages
    - Labels
  - Help to address issues surrounding heterogeneity
  - Allow users to take more control of application placement
- Node attributes don't…
  - Describe the HSN topology or routing
  - Get set dynamically (yet)
  - Remind you that Mother's Day is Sunday

SDB Node Attributes Table

```
mysql> describe attributes;
+----------+------------------+------+-----+---------+-------+
| Field    | Type             | Null | Key | Default | Extra |
+----------+------------------+------+-----+---------+-------+
| nodeid   | int(32) unsigned |      | PRI | 0       |       |
| archtype | int(4) unsigned  |      |     | 2       |       |
| osclass  | int(4) unsigned  |      |     | 1       |       |
| coremask | int(4) unsigned  |      |     | 1       |       |
| availmem | int(32) unsigned |      |     | 0       |       |
| pageszl2 | int(32) unsigned |      |     | 12      |       |
| clockmhz | int(32) unsigned | YES  |     | NULL    |       |
| label0   | varchar(32)      | YES  |     | NULL    |       |
| label1   | varchar(32)      | YES  |     | NULL    |       |
| label2   | varchar(32)      | YES  |     | NULL    |       |
| label3   | varchar(32)      | YES  |     | NULL    |       |
+----------+------------------+------+-----+---------+-------+
11 rows in set (0.00 sec)

mysql>
```

Displaying Node Attributes

```
$ xtprocadmin --attrsall
Connected
   NID    (HEX)    NODENAME     TYPE ARCH        OS CORES AVAILMEM   PAGESZ CLOCKMHZ
    32     0x20  c0-0c1s0n0  service   xt (service)     1     2000     4096     2400
    35     0x23  c0-0c1s0n3  service   xt (service)     1     2000     4096     2400
    36     0x24  c0-0c1s1n0  service   xt (service)     1     2000     4096     2400
    39     0x27  c0-0c1s1n3  service   xt (service)     1     2000     4096     2400
    40     0x28  c0-0c1s2n0  service   xt (service)     2     2000     4096     2400
    43     0x2b  c0-0c1s2n3  service   xt (service)     2     2000     4096     2400
    44     0x2c  c0-0c1s3n0  compute   xt       CNL     1     1000     4096     2000
    45     0x2d  c0-0c1s3n1  compute   xt       CNL     1     1000     4096     2000
    46     0x2e  c0-0c1s3n2  compute   xt       CNL     1     1000     4096     2000
    47     0x2f  c0-0c1s3n3  compute   xt       CNL     1     1000     4096     2000
    48     0x30  c0-0c1s4n0  compute   xt       CNL     1     1000     4096     2000
    49     0x31  c0-0c1s4n1  compute   xt       CNL     1     1000     4096     2000
    50     0x32  c0-0c1s4n2  compute   xt       CNL     1     1000     4096     2000
    51     0x33  c0-0c1s4n3  compute   xt       CNL     1     1000     4096     2000
    52     0x34  c0-0c1s5n0  compute   xt       CNL     1     1000     4096     2000
    53     0x35  c0-0c1s5n1  compute   xt       CNL     1     1000     4096     2000
    54     0x36  c0-0c1s5n2  compute   xt       CNL     1     1000     4096     2000
    55     0x37  c0-0c1s5n3  compute   xt       CNL     1     1000     4096     2000
    56     0x38  c0-0c1s6n0  compute   xt       CNL     2     3000     4096     2400
    57     0x39  c0-0c1s6n1  compute   xt       CNL     2     3000     4096     2400
    58     0x3a  c0-0c1s6n2  compute   xt       CNL     2     3000     4096     2400
    59     0x3b  c0-0c1s6n3  compute   xt       CNL     2     3000     4096     2400
    60     0x3c  c0-0c1s7n0  compute   xt       CNL     2     2000     4096     2400
    61     0x3d  c0-0c1s7n1  compute   xt       CNL     2     2000     4096     2400
    62     0x3e  c0-0c1s7n2  compute   xt       CNL     2     2000     4096     2400
    63     0x3f  c0-0c1s7n3  compute   xt       CNL     2     2000     4096     2400
$
```

Using Node Attributes

- Use the “cnselect” (compute node select) command to…
  - List available fields and values
  - Generate candidate node lists for aprun
  - Constrain apsched node selection

```
$ man cnselect
Reformatting cnselect(1), please wait...
cnselect(1)                                                cnselect(1)

NAME
       cnselect -- Returns a list of compute nodes based on user-
       specified compute node attributes

SYNOPSIS
       cnselect [-c] [-U] [-V] [-D] [-y] [-l] [-L fieldname|[-e] expression]
…
```

Using the cnselect Command

```
$ module load MySQL
$ cnselect
44-63
$ cnselect coremask.eq.1
44-55
$ cnselect coremask.gt.1
56-63
$ cnselect clockmhz.ge.2400
56-63
$ cnselect clockmhz.lt.2400
44-55
$ cnselect availmem.lt.2000
44-55
$ cnselect availmem.eq.2000
60-63
$ cnselect availmem.gt.2000
56-59
$ cnselect availmem.lt.3000
44-55,60-63
$ cnselect availmem.lt.3000.and.coremask.eq.1
44-55
$ cnselect availmem.lt.3000.and.coremask.gt.1
60-63
$
```

Using cnselect with aprun

```
$ NODES=$(cnselect availmem.lt.2000 .and. coremask.eq.1)
$ echo $NODES
44-55
$ aprun -n 2 -d 1 -L $NODES ./hello
Hello from rank 0 (thread 0) on nid00044 <-- MASTER
Hello from rank 1 (thread 0) on nid00045 <-- MASTER
Application 1406 resources: utime 0, stime 0
$
```

- Use cnselect to construct the node list
- Pass node list to aprun via the -L parameter

Behind the Scenes of cnselect

```
$ cnselect -D availmem.lt.3000.and.coremask.gt.1
SELECT nodeid FROM processor, attributes WHERE
     processor_id = nodeid AND processor_type='compute' AND
     ( availmem<3000 AND coremask>1 ) ;
$ head -6 /opt/xt-os/2.0.03/bin/snos64/cnselect
#!/bin/bash
#
# Copyright 2007 Cray Inc.
#
# simple query interface to node attributes
#
$
```

- Convenient MySQL interface to attributes table
- Why is cnselect a shell script?
  - Not for portability (must be run from a login node)
  - Extensible for site enhancements and customizations
  - Supported “as is”
  - Contributions encouraged

ALPS Status

- ALPS apstat command provides status information for…
  - Reservations
  - Applications
  - Compute nodes
  - Scheduler statistics

```
$ man apstat
Reformatting apstat(1), please wait...
apstat(1)                                                    apstat(1)

NAME
       apstat -- Provides status information for Cray XT series
       systems with CNL or Cray X2 applications

SYNOPSIS
       apstat [-a] [-n] [-p] [-r][-s] [-v] [apid [apid...]] [-X]
…
```

Node Status

```
$ apstat -n
  NID Arch St CPUs  PgSz     Avl    Conf  Placed  PEs Apids
   44  XT3 UP P      4K  256000  128000  128000    1 1413
   45  XT3 UP P      4K  256000  128000  128000    1 1413
   46  XT3 UP P      4K  256000  128000  128000    1 1413
   47  XT3 UP P      4K  256000  128000  128000    1 1413
   48  XT3 UP P      4K  256000  128000  128000    1 1413
   49  XT3 UP P      4K  256000  128000  128000    1 1413
   50  XT3 UP P      4K  256000  128000  128000    1 1413
   51  XT3 UP P      4K  256000  128000  128000    1 1413
   52  XT3 UP C      4K  256000  128000       0    0
   53  XT3 UP -      4K  256000       0       0    0
   54  XT3 UP -      4K  256000       0       0    0
   55  XT3 UP -      4K  256000       0       0    0
   56  XT3 UP PP     4K  768000  128000  128000    1 1411
   57  XT3 UP PP     4K  768000  128000  128000    1 1411
   58  XT3 UP PP     4K  768000  128000  128000    1 1412
   59  XT3 UP PP     4K  768000  128000  128000    1 1412
   60  XT3 UP PP     4K  512000  128000  128000    1 1412
   61  XT3 UP PP     4K  512000  128000  128000    1 1412
   62  XT3 UP --     4K  512000       0       0    0
   63  XT3 UP --     4K  512000       0       0    0
Compute node summary: up: 20 idle: 6
$
```

Application Status

```
$ apstat -av
Placed  Apid ResId     User    PEs Nodes    Age   State Command
        1411   311  crayadm     2     2   0h04m  run   sleep
        1412   312  crayadm     4     4   0h03m  run   sleep
        1413   313  crayadm     8     8   0h03m  run   sleep

Application detail
Ap[0]: apid 1411, pagg 31312, resId 311, user crayadm,
       gid 14901, account 12795, time 0, normal
  Number of commands 1, control network fanout 32
  Cmd[0]: sleep -n 2 -d 2 -N 0, memory 500MB, type XT3, nodes 2
  Placement list entries: 2
    PE 0, cmd 0, nid 56, CPU map 0x3 - PE 1, cmd 0, nid 57, CPU map 0x3
Ap[1]: apid 1412, pagg 31312, resId 312, user crayadm,
       gid 14901, account 12795, time 0, normal
  Number of commands 1, control network fanout 32
  Cmd[0]: sleep -n 4 -d 2 -N 0, memory 500MB, type XT3, nodes 4
  Placement list entries: 4
    PE 0, cmd 0, nid 58, CPU map 0x3 - PE 1, cmd 0, nid 59, CPU map 0x3
    PE 2, cmd 0, nid 60, CPU map 0x3 - PE 3, cmd 0, nid 61, CPU map 0x3
Ap[2]: apid 1413, pagg 31312, resId 313, user crayadm,
       gid 14901, account 12795, time 0, normal
  Number of commands 1, control network fanout 32
  Cmd[0]: sleep -n 8 -d 1 -N 0, memory 500MB, type XT3, nodes 8
  Placement list entries: 8
    PE 0, cmd 0, nid 44, CPU map 0x1 - PE 1, cmd 0, nid 45, CPU map 0x1
    PE 2, cmd 0, nid 46, CPU map 0x1 - PE 3, cmd 0, nid 47, CPU map 0x1
    PE 4, cmd 0, nid 48, CPU map 0x1 - PE 5, cmd 0, nid 49, CPU map 0x1
    PE 6, cmd 0, nid 50, CPU map 0x1 - PE 7, cmd 0, nid 51, CPU map 0x1
Ap[3]: apid 1414, pagg 0, resId 314, user crayadm,
       gid 14901, account 12795, time 0, normal
  Number of commands 1, control network fanout 32
  Cmd[0]: BASIL -n 1 -d 1 -N 0, memory 500MB, type XT3, nodes 1
  Placement list entries: 0
$
```

Reservation Status

```
$ apstat -rv
 ResId      ApId  From Arch    PEs N d Memory State
   311      1411 aprun  XT3      2 0 2    500 atomic,conf,claim
   312      1412 aprun  XT3      4 0 2    500 atomic,conf,claim
   313      1413 aprun  XT3      8 0 1    500 atomic,conf,claim
   314      1414 batch  XT3      1 0 1    500 conf
$
```

- The first three reservations are “claimed” by applications
  - Atomic = Reservation is from an interactive launch (not batch)
  - Conf = Reservation is confirmed
  - Claim = Application has claimed the confirmed reservation
- The last reservation is “confirmed” by a batch job

Scheduler Status

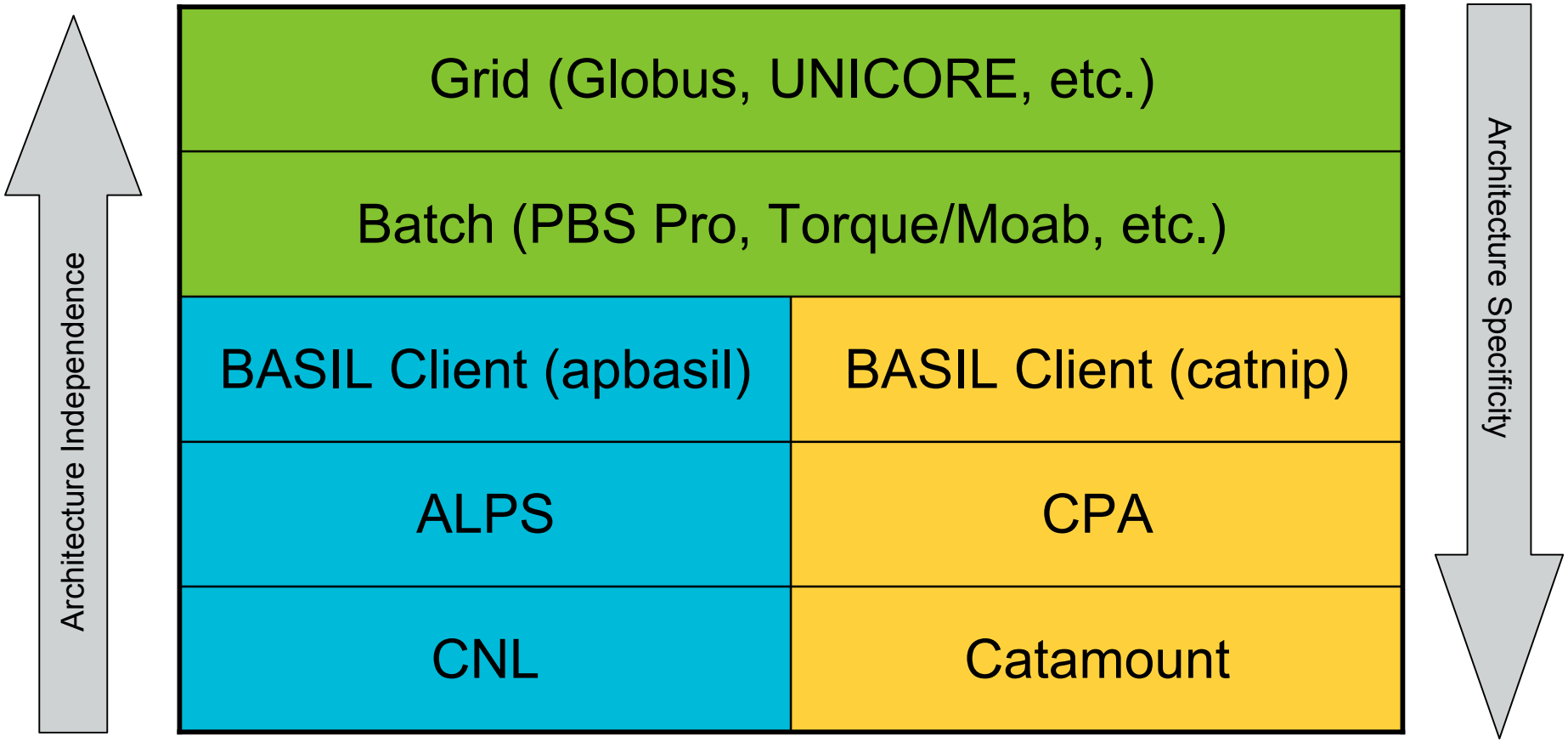
```
$ apstat -s
ALPS scheduler up for 12h40m. Build: Apr 26 2007 10:30:06
Applications placed       267
             exited       264
Resources    confirmed    268
Information last updated: Wed May  2 08:46:21 2007
$
```

- 267 - 264 = 3 claimed (in execution)
- 268 - 267 = 1 confirmed

ALPS and PBS Pro

- Batch and Application Scheduler Interface Layer (BASIL)
  - XML interface between ALPS and the batch system
  - Four basic methods:
    - Inventory
    - Reservation creation
    - Reservation confirmation
    - Reservation cancellation
- ALPS apbasil command implements BASIL for CNL
- CPA catnip command implements BASIL for Catamount
- One PBS Pro release supports both CNOS environments

BASIL Hierarchy



BASIL Inventory

```
$ cat basil_inventory_request
<?xml version="1.0"?>
<BasilRequest protocol="1.0" method="QUERY" type="INVENTORY"/>
$ cat basil_inventory_request | apbasil
<?xml version="1.0"?>
<BasilResponse protocol="1.0">
 <ResponseData status="SUCCESS" method="QUERY">
  <Inventory>
   <NodeArray>
    <Node node_id="44" name="c0-0c1s3n0" architecture="XT" role="BATCH" state="UP">
     …
    </Node>
    …
   </NodeArray>
   <ReservationArray>
    <Reservation reservation_id="316" user_name="crayadm" account_name="DEFAULT"/>
    <Reservation reservation_id="317" user_name="crayadm" account_name="DEFAULT"/>
    <Reservation reservation_id="318" user_name="crayadm" account_name="DEFAULT"/>
    <Reservation reservation_id="319" user_name="crayadm" account_name="DEFAULT"/>
   </ReservationArray>
  </Inventory>
 </ResponseData>
</BasilResponse>
$
```

BASIL Inventory - Node Detail

```
…
<Node node_id="60" name="c0-0c1s7n0" architecture="XT" role="BATCH" state="UP">
 <ProcessorArray>
  <Processor ordinal="0" architecture="x86_64" clock_mhz="2400">
   <ProcessorAllocation reservation_id="317"/>
  </Processor>
  <Processor ordinal="1" architecture="x86_64" clock_mhz="2400">
   <ProcessorAllocation reservation_id="317"/>
  </Processor>
 </ProcessorArray>
 <MemoryArray>
  <Memory type="OS" page_size_kb="4" page_count="512000">
   <MemoryAllocation reservation_id="317" page_count="128000"/>
  </Memory>
 </MemoryArray>
 <LabelArray/>
</Node>
…
```

- Node attributes are included
- Processor and memory allocations are identified
- Data is passed from PBS MOM to the scheduler

BASIL Reservation

```
$ cat basil_request_reserve
<?xml version="1.0"?>
<BasilRequest protocol="1.0" method="RESERVE">
  <ReserveParamArray user_name="crayadm">
    <ReserveParam architecture="XT" width="4" depth="1" nppn="1"/>
  </ReserveParamArray>
</BasilRequest>
$ cat basil_request_reserve | apbasil
<?xml version="1.0"?>
<BasilResponse protocol="1.0">
 <ResponseData status="SUCCESS" method="RESERVE">
  <Reserved reservation_id="339" admin_cookie="0" alloc_cookie="0"/>
 </ResponseData>
</BasilResponse>
$
```

- Reserve 4 PEs with depth and PPN of one
- The PBS MOM daemon does this during job initialization

BASIL Confirmation

```
$ cat basil_request_confirm
<?xml version="1.0"?>
<BasilRequest protocol="1.0" method="CONFIRM" job_name="foo"
                  reservation_id="339" admin_cookie="7707"/>
$ cat basil_request_confirm | apbasil
<?xml version="1.0"?>
<BasilResponse protocol="1.0">
 <ResponseData status="SUCCESS" method="CONFIRM">
  <Confirmed/>
 </ResponseData>
</BasilResponse>
$
```

- Confirm reservation with identifier 339
- PBS MOM daemon does this after creating reservation
- Two step process to support creation within batch scheduler

BASIL Claim

```
$ apstat -r
 ResId      ApId  From Arch    PEs N d Memory State
   339      1441 batch  XT3      4 1 1    500 conf
$
```

- Confirmed reservation can now be claimed
- User invokes aprun within the batch job to claim
- More than one aprun may claim the confirmed resources in serial or in parallel
- Aggregate resources claimed may not exceed those of the confirmed reservation
- Ensures batch job has constant and immediate access to reserved resources

BASIL Cancellation

```
$ cat basil_request_release
<?xml version="1.0"?>
<BasilRequest protocol="1.0" method="RELEASE" reservation_id="339"/>
$ cat basil_request_release | apbasil
<?xml version="1.0"?>
<BasilResponse protocol="1.0">
 <ResponseData status="SUCCESS" method="RELEASE">
  <Released/>
 </ResponseData>
</BasilResponse>
$
```

- Cancel reservation with identifier 339
- PBS MOM daemon does this after job completes
- ALPS frees resources associated with reservation

PBS Pro MPP Resources

- Specialized PBS resources map to aprun parameters

aprun	qsub	Resource Description
-n 4	-l mppwidth=4	Width
-d 2	-l mppdepth=2	Depth
-N 1	-l mppnppn=1	Number of PEs Per Node
-L 5,6,7	-l mppnodes=\"5,6,7\"	Node List
-m 1000m	-l mppmem=1000mb	Memory Per PE
-t 3600	-l mpptime=1:00:00	CPU Time Limit
-a xt	-l mpparch=XT	Target Architecture
	-l mpplabels=\"foo,bar\"	Node Attribute Labels
	-l mpphost=pike	Target MPP Host

Submitting PBS Pro Jobs

```
$ module load pbs
$ cat myjob
#PBS -j oe
#PBS -l mppwidth=4
#PBS -l mppdepth=2
#PBS -l mppmem=1000mb
sleep 30
cd /tmp
aprun -n 4 -d 2 ./sleep 30
$ qsub myjob
5.sdb
$ qstat -a

sdb:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time   S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
5.sdb           crayadm  workq    myjob       13327   1   1    --     --  R 00:00

$
```

- Job allows us to see both the confirm and claim states

PBS Pro Job Status Data

```
$ qstat -f
Job Id: 5.sdb
    Job_Name = myjob
    Job_Owner = crayadm@nid00040
    …
    qtime = Wed May  2 16:17:45 2007
    Rerunable = True
    Resource_List.mpparch = XT
    Resource_List.mppdepth = 2
    Resource_List.mppmem = 1000mb
    Resource_List.mppwidth = 4
    Resource_List.ncpus = 1
    Resource_List.nodect = 1
    Resource_List.place = pack
    Resource_List.select = 1
    stime = Wed May  2 16:17:45 2007
    session_id = 13327
    job_dir = /home/crayadm
    …
    comment = Job run at Wed May 02 at 16:17 on (login0:ncpus=1)
    etime = Wed May  2 16:17:45 2007

$
```

ALPS Reservation Data

```
$ apstat -r
 ResId     ApId  From Arch   PEs N d Memory State
   537     1485 batch  XT3     4 0 2   1000 conf
$ apstat -a
No placed applications are present
$ apstat -r
 ResId     ApId  From Arch   PEs N d Memory State
   537     1485 batch  XT3     4 0 2   1000 conf,claim
A  537     1486 batch  XT3     4 - -    500 conf,claim
$ apstat -a
Placed  Apid ResId     User    PEs Nodes     Age    State Command
        1486   537  crayadm     4     4   0h00m  run   sleep
$
```

- The first two commands were run during the initial sleep
- Subsequent commands were run after aprun was invoked
- “A” stands for Allocation against a batch reservation

Why Is My MPP Job Stuck?

```
$ tracejob 0

Job: 0.sdb

05/02/2007 07:53:55  L    Considering job to run
05/02/2007 07:53:55  L    Evaluating subchunk: ncpus=1
05/02/2007 07:53:55  L    Evaluating MPP resource requirements.
05/02/2007 07:53:55  L    Failed to satisfy subchunk: 1:ncpus=1
05/02/2007 07:53:55  S    Job Queued at request of crayadm@nid00040, owner =
                          crayadm@nid00040, job name = myjob, queue = workq
05/02/2007 07:53:55  S    Job Modified at request of Scheduler@nid00035
05/02/2007 07:53:55  L    MPP nodes on "pike": 20 total, 0 fit, failed: 0 arch,
                          0 state, 20 role, 0 procs, 0 mem, 0 nodelist 0 labels
05/02/2007 07:53:55  L    No matching MPP host found.
05/02/2007 07:53:55  L    No available resources on nodes
…

$
```

- Output from tracejob on the SDB (pbs_sched) node
- Note that 20 nodes were excluded based on role

SDB Node Allocation Mode

```
$ xtshowmesh
Compute Processor Allocation Status as of Thu May  3 09:18:20 2007

        C 0 (X dir)

  Z dir-> 01234567
  Y dir 0 SSS;;;;;
        1    ;;;;;
        2    ;;;;;
        3 SSS;;;;;

 Legend:
   nonexistent node                  S  service node
;  free interactive compute CNL      -  free batch compute node CNL
A  allocated, but idle compute node  ?  suspect compute node
X  down compute node                 Y  down or admindown service node
Z  admindown compute node            R  node is routing

Available compute nodes:      20 interactive,     0 batch

$
```

Setting Node Allocation Mode to “Batch”

```
$ su
Password:
# module load xt-boot
# i=44; while [ $i -lt 64 ]; do \
      xtprocadmin -k m batch -n $i; \
      let i=i+1; \
  done
Connected
   NID     (HEX)     NODENAME      TYPE          MODE
    44      0x2c   c0-0c1s3n0   compute        batch
…
Connected
   NID     (HEX)     NODENAME      TYPE          MODE
    63      0x3f   c0-0c1s7n3   compute        batch
# exit
$
```

- Use xtprocadmin (as root) to modify node allocation mode

Viewing the Configuration Change

```
$ xtshowmesh
Compute Processor Allocation Status as of Thu May  3 09:18:20 2007

        C 0 (X dir)

  Z dir-> 01234567
  Y dir 0 SSS-----
        1    -----
        2    -----
        3 SSS-----

 Legend:
   nonexistent node                  S  service node
;  free interactive compute CNL      -  free batch compute node CNL
A  allocated, but idle compute node  ?  suspect compute node
X  down compute node                 Y  down or admindown service node
Z  admindown compute node            R  node is routing

Available compute nodes:      0 interactive,     20 batch

$
```

Reasons Why Jobs Get “Stuck”

- Arch - Nodes that do not satisfy mpparch
- State - Nodes not in “available” state
- Role - Not enough “batch” nodes
- Procs - Nodes not satisfying mppwidth/mppdepth/mppnppn
- Mem - Nodes not satisfying mppmem
- Nodelist - Nodes not present in mppnodes
- Labels - Nodes not providing requested mpplabels

```
MPP nodes on "pike": 20 total, 0 fit, failed: 0 arch, 0 state,
                     20 role, 0 procs, 0 mem, 0 nodelist 0 labels
```

Node Labels

- Used to “tag” nodes
- Node labels defined by administrator
- Use cnselect to generate corresponding node list

- Prepare to assign node labels:

```
$ su
Password:
# module load xt-boot
#
```

View Existing Labels

```
# xtprocadmin -a label0
Connected
   NID    (HEX)    NODENAME     TYPE                                    LABEL0
    32     0x20   c0-0c1s0n0   service
    35     0x23   c0-0c1s0n3   service
    36     0x24   c0-0c1s1n0   service
    39     0x27   c0-0c1s1n3   service
    40     0x28   c0-0c1s2n0   service
    43     0x2b   c0-0c1s2n3   service
    44     0x2c   c0-0c1s3n0   compute
    45     0x2d   c0-0c1s3n1   compute
    46     0x2e   c0-0c1s3n2   compute
    47     0x2f   c0-0c1s3n3   compute
    48     0x30   c0-0c1s4n0   compute
    49     0x31   c0-0c1s4n1   compute
    50     0x32   c0-0c1s4n2   compute
    51     0x33   c0-0c1s4n3   compute
    52     0x34   c0-0c1s5n0   compute
    53     0x35   c0-0c1s5n1   compute
    54     0x36   c0-0c1s5n2   compute
    55     0x37   c0-0c1s5n3   compute
    56     0x38   c0-0c1s6n0   compute
    57     0x39   c0-0c1s6n1   compute
    58     0x3a   c0-0c1s6n2   compute
    59     0x3b   c0-0c1s6n3   compute
    60     0x3c   c0-0c1s7n0   compute
    61     0x3d   c0-0c1s7n1   compute
    62     0x3e   c0-0c1s7n2   compute
    63     0x3f   c0-0c1s7n3   compute
#
```

Assign Labels “red”, “green”, and “blue”

```
# xtprocadmin -a label0=red -n 44
Connected
   NID     (HEX)     NODENAME      TYPE                                LABEL0
    44      0x2c   c0-0c1s3n0   compute                                   red
…
# xtprocadmin -a label0=yellow -n 48
Connected
   NID     (HEX)     NODENAME      TYPE                                LABEL0
    48      0x30   c0-0c1s4n0   compute                                yellow
…
# xtprocadmin -a label0=blue -n 52
Connected
   NID     (HEX)     NODENAME      TYPE                                LABEL0
    52      0x34   c0-0c1s5n0   compute                                  blue
#
```

View New Labels

```
# xtprocadmin -a label0
Connected
   NID     (HEX)     NODENAME      TYPE                                LABEL0
    32      0x20   c0-0c1s0n0   service
    35      0x23   c0-0c1s0n3   service
    36      0x24   c0-0c1s1n0   service
    39      0x27   c0-0c1s1n3   service
    40      0x28   c0-0c1s2n0   service
    43      0x2b   c0-0c1s2n3   service
    44      0x2c   c0-0c1s3n0   compute                                 red
    45      0x2d   c0-0c1s3n1   compute                                 red
    46      0x2e   c0-0c1s3n2   compute                                 red
    47      0x2f   c0-0c1s3n3   compute                                 red
    48      0x30   c0-0c1s4n0   compute                              yellow
    49      0x31   c0-0c1s4n1   compute                              yellow
    50      0x32   c0-0c1s4n2   compute                              yellow
    51      0x33   c0-0c1s4n3   compute                              yellow
    52      0x34   c0-0c1s5n0   compute                                blue
    53      0x35   c0-0c1s5n1   compute                                blue
    54      0x36   c0-0c1s5n2   compute                                blue
    55      0x37   c0-0c1s5n3   compute                                blue
    56      0x38   c0-0c1s6n0   compute
    57      0x39   c0-0c1s6n1   compute
    58      0x3a   c0-0c1s6n2   compute
    59      0x3b   c0-0c1s6n3   compute
    60      0x3c   c0-0c1s7n0   compute
    61      0x3d   c0-0c1s7n1   compute
    62      0x3e   c0-0c1s7n2   compute
    63      0x3f   c0-0c1s7n3   compute
#
```

Creating Node Lists Based On Labels

```
$ cnselect label0.eq.\"red\"
44-47
$ cnselect label0.eq.\"yellow\"
48-51
$ cnselect label0.eq.\"blue\"
52-55
$
```

- These node lists may be passed to…
  - ALPS as an argument to aprun -L
  - PBS as an argument to mppnodes

ALPS for BlackWidow

- Platform specific apinit daemon
  - DM placement support (NTT, RTT, processor/node granularity)
  - Uses apstart for application initialization
  - PE utilizes DM for IPC
- Placement scheduler (apsched) enhancements
  - Architecture specific placement for DM
  - High radix fat tree reduces placement restrictions
- Client specific enhancements
  - Launch client (aprun) recognizes binary format
  - Status client (apstat) distinguishes between architectures
- Architecture and application data management
  - Support existing (XT/BW), planned (Scorpio) and future architectures
  - Bridge gathers configuration data from SDB, Mazama, etc.
  - Heterogeneous and extensible by design

ALPS BlackWidow Usage

- Interactive use automatically determines binary format
- Batch use requires user/queue to specify architecture
- User may override architecture with aprun -a parameter
- Currently supports launch to one architecture (SPMD)
- Planned support for multiple architectures (MPMD)
  - Requires PE support
  - Additional ALPS development
- Multiple applications may currently communicate via files, pipes, and sockets
  - aprun -n 16 my_bw_app | aprun -n 32 my_xt_app

Scaling To 100K Nodes

- Batch systems use an all-to-one model for multi-node jobs
- ALPS uses a fanout tree for multi-node jobs
- Spread out connections to improve scalability

Tree Radix

Tree Depth	2	4	8	16	32
1	1	1	1	1	1
2	3	5	9	17	33
3	7	21	73	273	1057
4	15	85	585	4369	33825
5	31	341	4681	69905	1082401