CUG 2008 HELSINKI • MAY 5–8, 2008
CROSSING THE BOUNDARIES

# ALPS Tutorial "Ascent"

Michael Karo
mek@cray.com

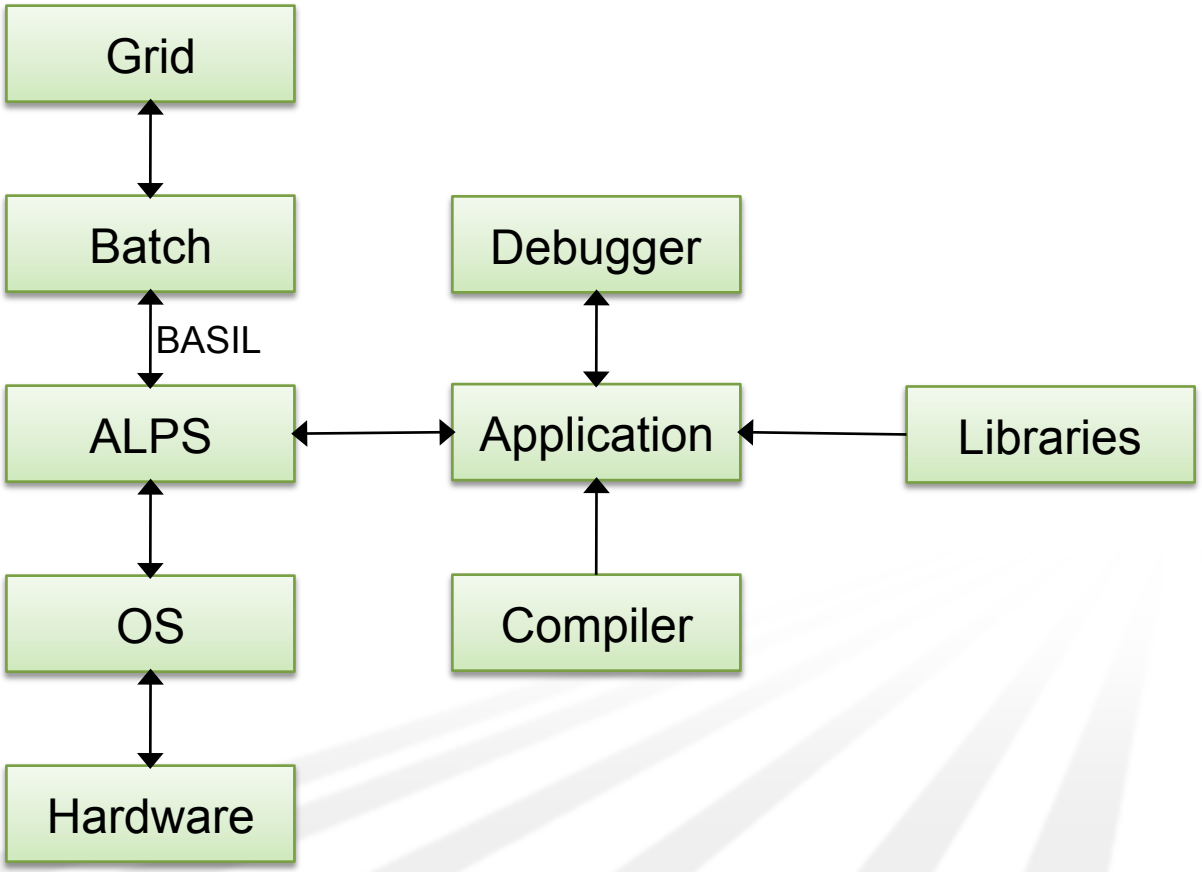
CRAY
THE SUPERCOMPUTER COMPANY

# Topics

- A look back at “Base Camp”
- ALPS for Cray XT5 systems
  - Multisocket nodes
  - Accounting and auditing
  - Checkpoint / Restart
  - Huge pages
- ALPS for Cray XT5h systems
  - X2 quadrant support
  - MPMD launch
  - Context switching
- BASIL 1.1
- ALPS troubleshooting
- CSA

# ALPS Overview

- ALPS = Application Level Placement Scheduler
- BASIL = Batch Application Scheduler Interface Layer



# Terminology

- Node
  - All resources managed by a single Cray Linux Environment (CLE) instance
- Processing Element (PE)
  - ALPS launched binary invocation on a compute node
- Width (aprun -n)
  - Number of PEs to Launch
- Depth (aprun -d)
  - Number of threads per PE (OpenMP)
- PEs Per Node / PPN (aprun -N)
  - Number of PEs per CNL instance (multiple MPI ranks per node)
- Node List (aprun -L)
  - A user supplied list of candidate nodes to constrain placement
- Node Attributes
  - Characteristics of a node described in the SDB

# ALPS for Cray XT5 Systems

- Support for multisocket nodes
  - NUMA domains
  - Processor core affinity
  - Memory affinity
- Application Checkpoint / Restart (CPR)

# NUMA Domains

- Increased processor core density per node
  - Multiple sockets per node
  - Multiple dies per socket
- Increasingly complex intranode topology
  - XT3/XT4 – One NUMA domain per OS instance
  - XT5 – Two NUMA domains per OS instance
  - Beyond XT5 – Expect density to increase
- NUMA domains provide a mechanism to:
  - increase machine utilization
  - assign multiple applications per node
  - utilize OS features to shield processes from one another
- The batch system decides when to use the mechanisms
- Linux cpusets provide the underlying OS implementation

# SDB Segment Table

- node_id – Node identifier mapping to processor table
- socket_id – Processor socket ordinal
- die_id – Processor die ordinal
- coremask – Processor core mask
- mempgs – number of pages local to memory controller

```
mysql> describe segment;
+-----------+---------------------+------+-----+---------+-------+
| Field     | Type                | Null | Key | Default | Extra |
+-----------+---------------------+------+-----+---------+-------+
| node_id   | int(10) unsigned    | NO   | MUL |         |       |
| socket_id | tinyint(3) unsigned | NO   |     |         |       |
| die_id    | tinyint(3) unsigned | NO   |     | 0       |       |
| coremask  | int(10) unsigned    | NO   |     |         |       |
| mempgs    | int(10) unsigned    | NO   |     |         |       |
+-----------+---------------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

# NUMA Domain Support

- One application per NUMA domain
  - Multiple NUMA domains per node allow multiple applications per node
  - Pro: Potentially higher overall resource utilization
  - Con: Cannot mitigate contention for SeaStar bandwidth
- Quality of service guarantees
  - Process aggregates (paggs) provide inescapable container
  - CPU affinity enforced by the kernel
  - Memory affinity enforced by cpusets

# Test System Configuration

- Heterogeneous mix of XT4 and XT5 compute nodes

```
$ apstat -nv
  NID Arch State HW Rv Pl  PgSz     Avl    Conf  Placed  PEs Apids
...
   52   XT UP  I  4  -  -    4K 2048000       0       0    0
   53   XT UP  I  4  -  -    4K 2048000       0       0    0
   54   XT UP  I  4  -  -    4K 2048000       0       0    0
   55   XT UP  I  4  -  -    4K 2048000       0       0    0
   56   XT UP  I  8  -  -    4K 4096000       0       0    0
   57   XT UP  I  8  -  -    4K 4096000       0       0    0
   58   XT UP  I  8  -  -    4K 4096000       0       0    0
   59   XT DN  I  8  -  -    4K 4096000       0       0    0
...
Compute node summary
   arch config     up    use   held  avail   down
     XT     19     18      0      0     18      1
$
```

# Updated hello.c (1 of 3)

- Similar to hello.c from “Base Camp”
- Reports for each process:
  - MPI rank
  - OpenMP thread
  - hostname of compute node
  - CPU affinity list
- Three parts: front matter, support function, main function

```
#define _GNU_SOURCE

#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <sched.h>
#include <mpi.h>
#include <omp.h>
```

# Updated hello.c (2 of 3)

```
/* Borrowed from util-linux-2.13-pre7/schedutils/taskset.c */
static char *cpuset_to_cstr(cpu_set_t *mask, char *str)
{
    char *ptr = str;
    int i, j, entry_made = 0;
    for (i = 0; i < CPU_SETSIZE; i++) {
          if (CPU_ISSET(i, mask)) {
              int run = 0;
              entry_made = 1;
              for (j = i + 1; j < CPU_SETSIZE; j++) {
                    if (CPU_ISSET(j, mask)) run++;
                    else break;
              }
              if (!run)
                    sprintf(ptr, "%d,", i);
              else if (run == 1) {
                    sprintf(ptr, "%d,%d,", i, i + 1);
                    i++;
              } else {
                    sprintf(ptr, "%d-%d,", i, i + run);
                    i += run;
              }
              while (*ptr != 0) ptr++;
          }
    }
    ptr -= entry_made;
    *ptr = 0;
    return(str);
}
```

# Updated hello.c (3 of 3)

```
int main(int argc, char *argv[])
{
    int rank, thread;
    cpu_set_t coremask;
    char clbuf[7 * CPU_SETSIZE], hnbuf[64];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    memset(clbuf, 0, sizeof(clbuf));
    memset(hnbuf, 0, sizeof(hnbuf));
    (void)gethostname(hnbuf, sizeof(hnbuf));
    #pragma omp parallel private(thread, coremask, clbuf)
    {
         thread = omp_get_thread_num();
         (void)sched_getaffinity(0, sizeof(coremask), &coremask);
         cpuset_to_cstr(&coremask, clbuf);
         #pragma omp barrier
         printf("Hello from rank %d, thread %d, on %s. (core affinity = %s)\n",
                 rank, thread, hnbuf, clbuf);
    }
    MPI_Finalize();
    return(0);
}
```

# Compiling and running hello.c

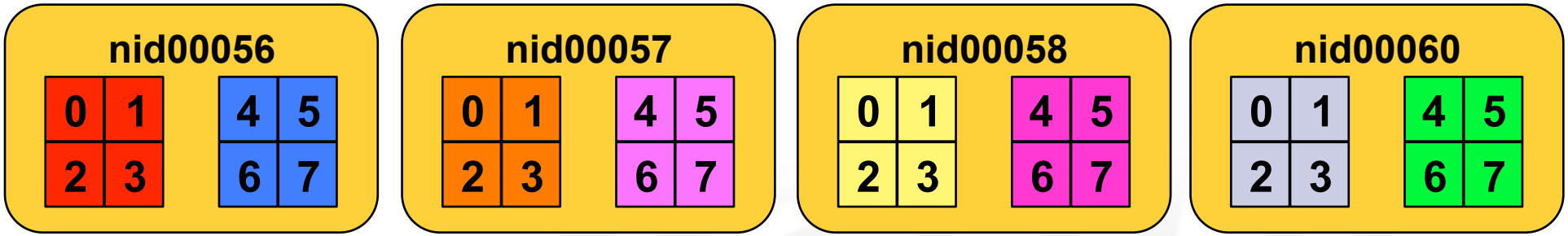
```
$ cd /tmp
$ cc -mp -g -o hello hello.c ; strip hello
/opt/xt-asyncpe/1.0/bin/cc: INFO: linux target is being used
hello.c:
$ aprun -N 1 -n 18 -cc none ./hello
Hello from rank 0, thread 0, on nid00044. (core affinity = 0,1)
Hello from rank 1, thread 0, on nid00045. (core affinity = 0,1)
Hello from rank 2, thread 0, on nid00046. (core affinity = 0,1)
Hello from rank 3, thread 0, on nid00048. (core affinity = 0,1)
Hello from rank 4, thread 0, on nid00049. (core affinity = 0,1)
Hello from rank 5, thread 0, on nid00050. (core affinity = 0,1)
Hello from rank 6, thread 0, on nid00051. (core affinity = 0,1)
Hello from rank 7, thread 0, on nid00052. (core affinity = 0-3)
Hello from rank 8, thread 0, on nid00053. (core affinity = 0-3)
Hello from rank 9, thread 0, on nid00054. (core affinity = 0-3)
Hello from rank 10, thread 0, on nid00055. (core affinity = 0-3)
Hello from rank 11, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 12, thread 0, on nid00057. (core affinity = 0-7)
Hello from rank 13, thread 0, on nid00058. (core affinity = 0-7)
Hello from rank 14, thread 0, on nid00060. (core affinity = 0-7)
Hello from rank 15, thread 0, on nid00061. (core affinity = 0-7)
Hello from rank 16, thread 0, on nid00062. (core affinity = 0-7)
Hello from rank 17, thread 0, on nid00063. (core affinity = 0-7)
Application 43132 resources: utime 0, stime 0
$
```

# New NUMA Domain Parameters

- aprun -S *pes_per_numa_domain*
  - Specifies PEs per NUMA domain (must be ≤ PEs per node)
  - Up to four with quad core
- aprun -sn *numa_domains_per_node*
  - Limits number of NUMA domains per node
  - Only one for XT3/XT4; one or two for XT5
- aprun -sl *list_of_numa_domains*
  - Specifies restricted list of NUMA domains for placement
  - comma separated list or dash separated range
- aprun -ss
  - Specifies strict memory affinity per NUMA domain
  - Affinity policy is local NUMA domain only
  - Alternative is node exclusive
- Specified per binary for MPMD launch

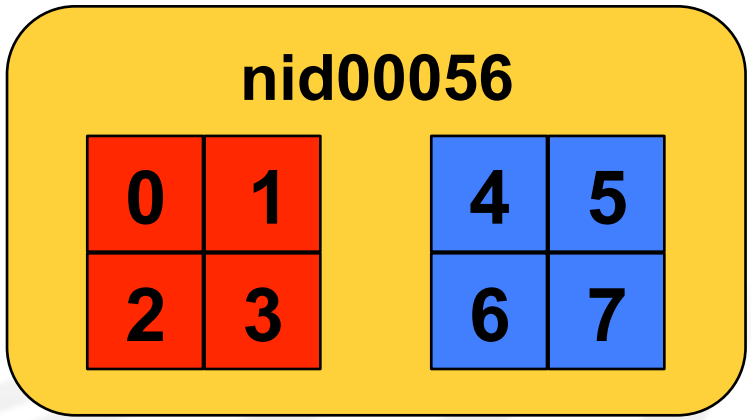
# aprun -S *pes_per_numa_domain* (1 of 2)

```
$ aprun -S 1 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 2, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 4, thread 0, on nid00058. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid00058. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00060. (core affinity = 0-3)
Hello from rank 7, thread 0, on nid00060. (core affinity = 4-7)
$
```



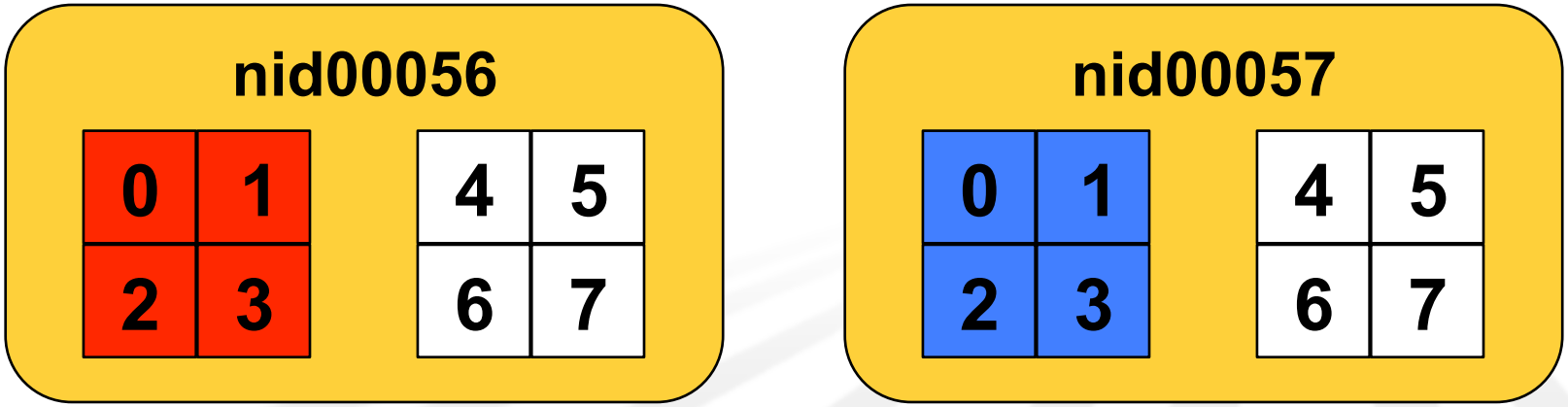
# aprun -S *pes_per_numa_domain* (2 of 2)

```
$ aprun -S 4 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 4-7)
$
```



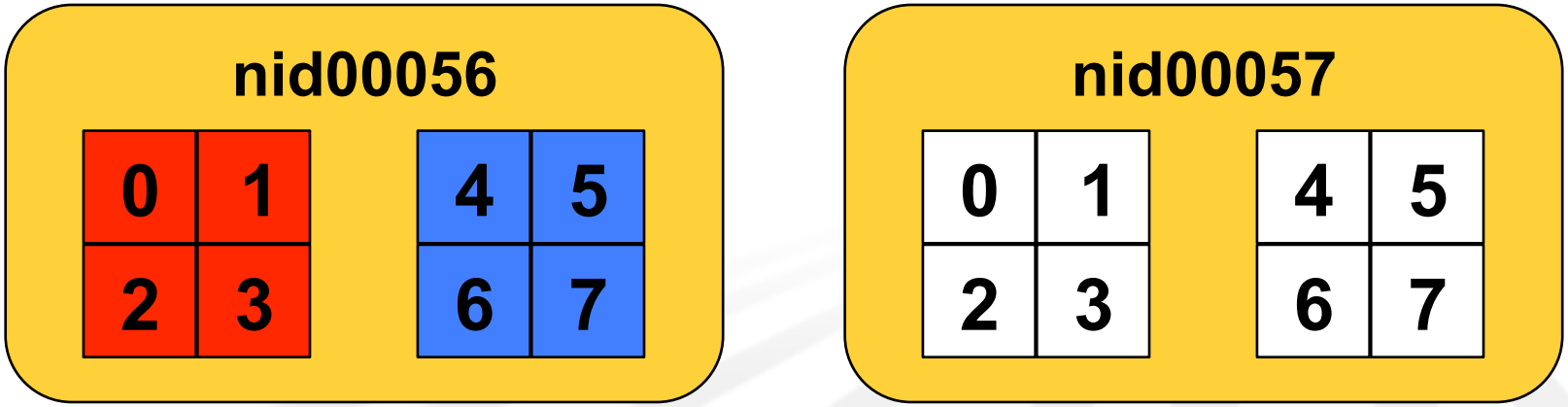
# aprun -sn *numa_domains_per_node* (1 of 2)

```
$ aprun -sn 1 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 6, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 7, thread 0, on nid00057. (core affinity = 0-3)
$
```



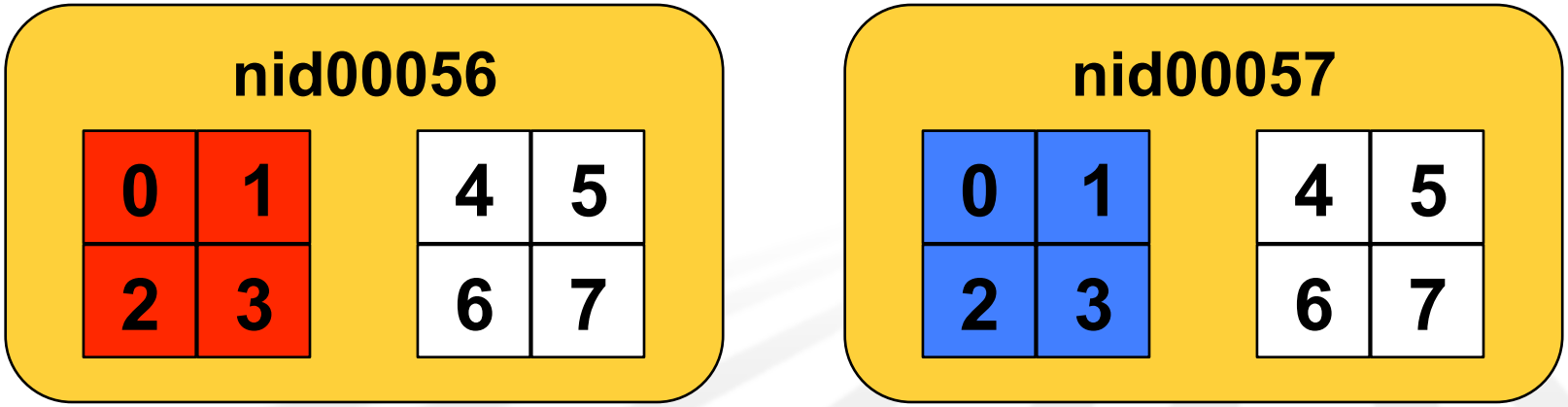
# aprun -sn *numa_domains_per_node* (2 of 2)

```
$ aprun -sn 2 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 4-7)
$
```



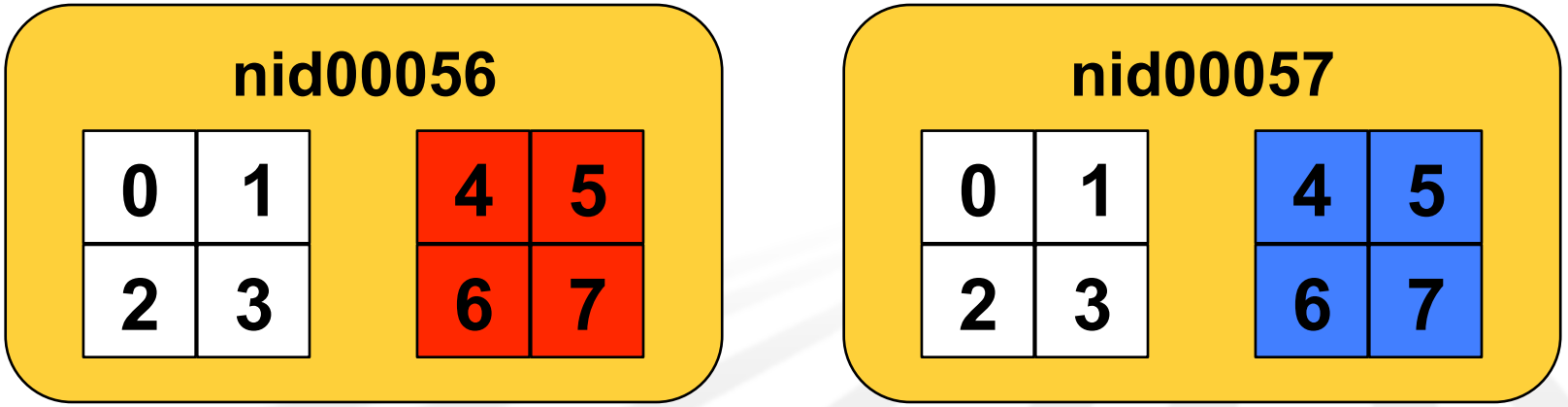
# aprun -sl *list_of_numa_domains* (1 of 3)

```
$ aprun -sl 0 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 6, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 7, thread 0, on nid00057. (core affinity = 0-3)
$
```



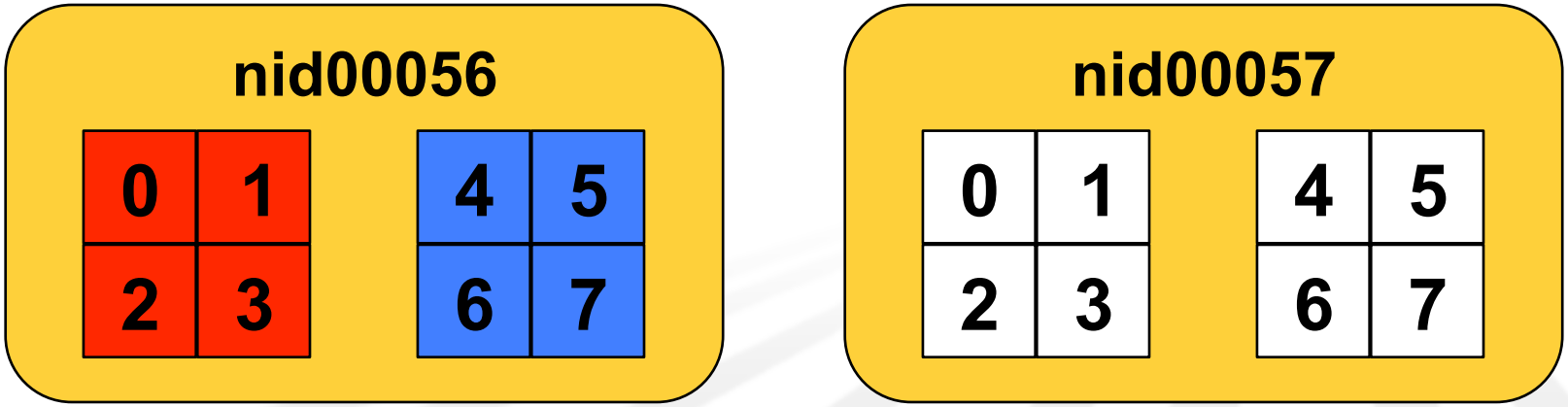
# aprun -sl *list_of_numa_domains* (2 of 3)

```
$ aprun -sl 1 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 1, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 2, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 3, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 4, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00057. (core affinity = 4-7)
$
```



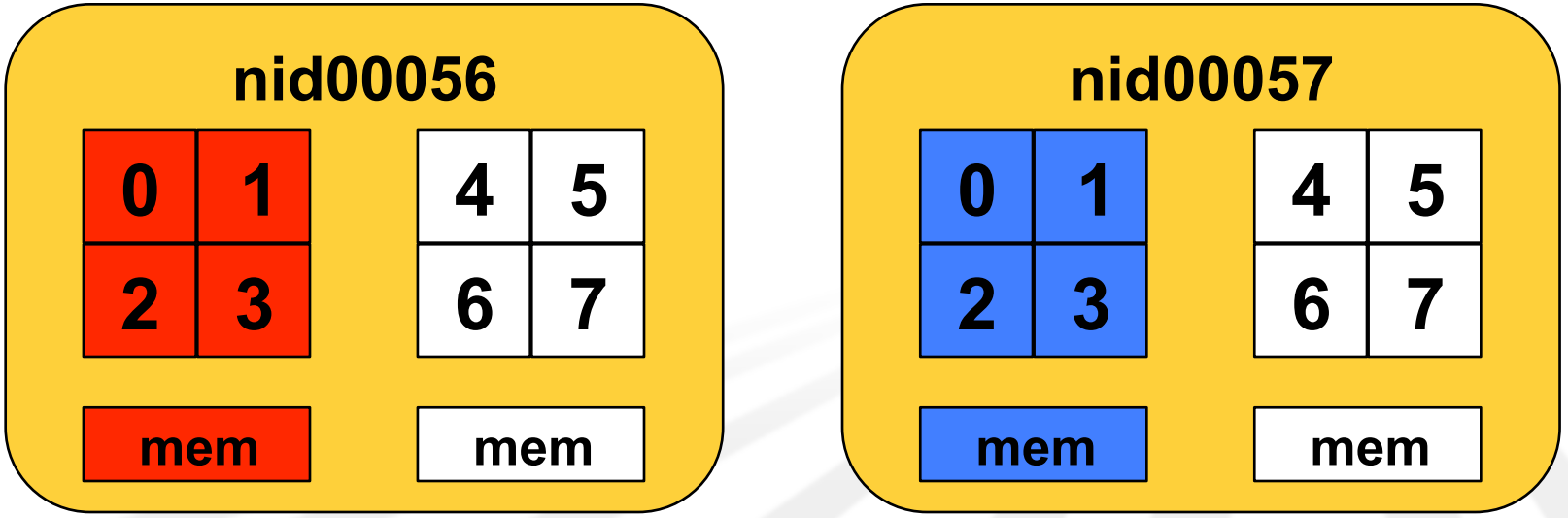
# aprun -sl *list_of_numa_domains* (3 of 3)

```
$ aprun -sl 0,1 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00057. (core affinity = 4-7)
$
```



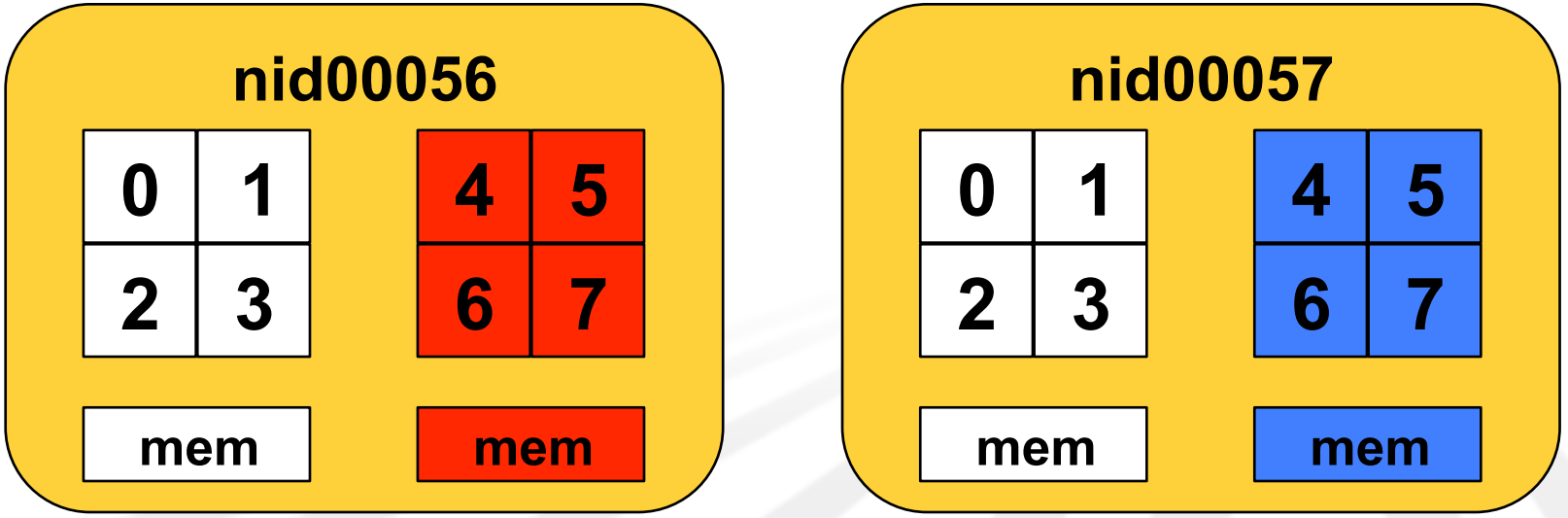
# aprun -ss (1 of 3)

```
$ aprun -ss -sl 0 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 6, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 7, thread 0, on nid00057. (core affinity = 0-3)
$
```



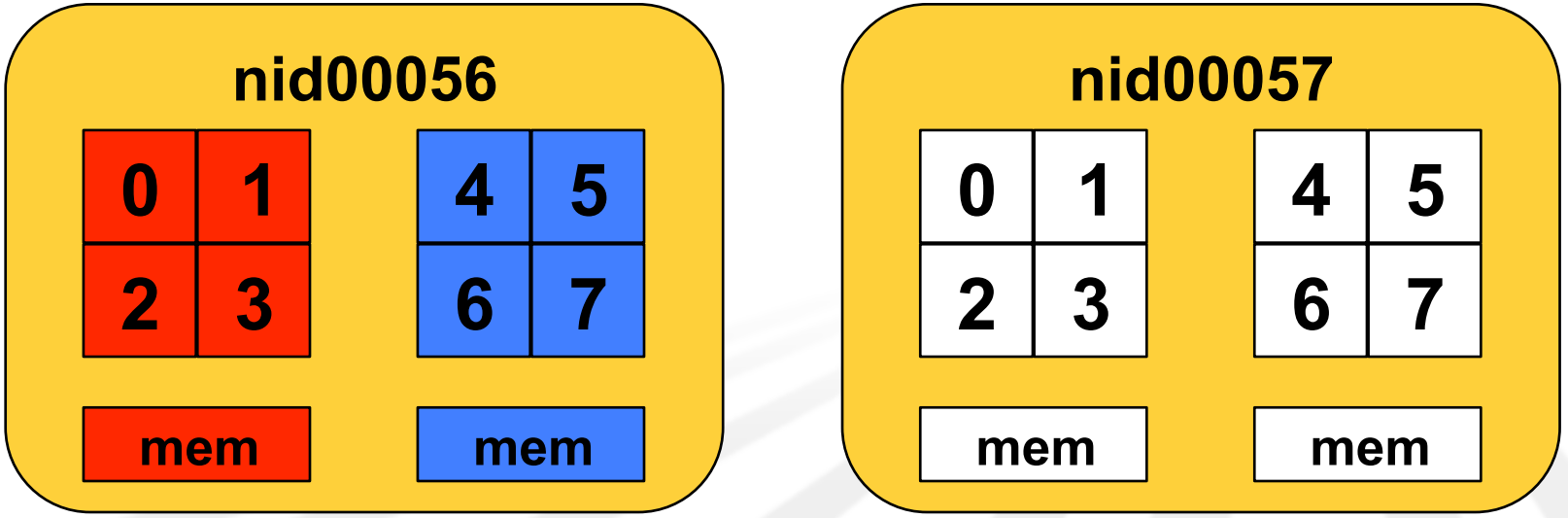
# aprun -ss (2 of 3)

```
$ aprun -ss -sl 1 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 1, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 2, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 3, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 4, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00057. (core affinity = 4-7)
$
```



# aprun -ss (3 of 3)

```
$ aprun -ss -sl 0,1 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 4-7)
$
```

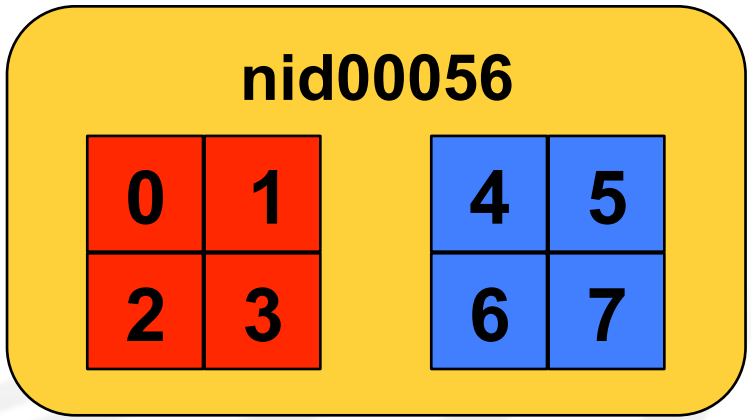


# New Core Affinity Parameters

- aprun -cc {segment | cpu | none | 0,1,2}
  - Bind processes to one or more cores
  - Restricts behavior of Linux process scheduler
  - Default is NUMA domain (segment)
- aprun -cp cpu_placement_file_name
  - Used for more complex specifications
  - File must be accessible from the compute nodes
  - Deferred implementation
- Specified per binary for MPMD launch

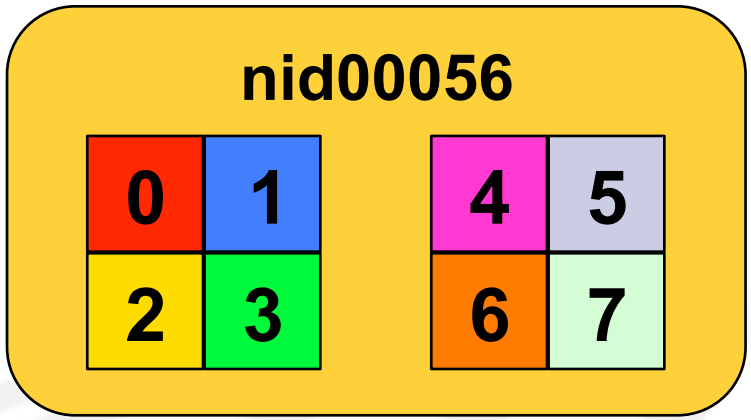
# aprun -cc segment

```
$ aprun -cc segment -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 4-7)
$
```



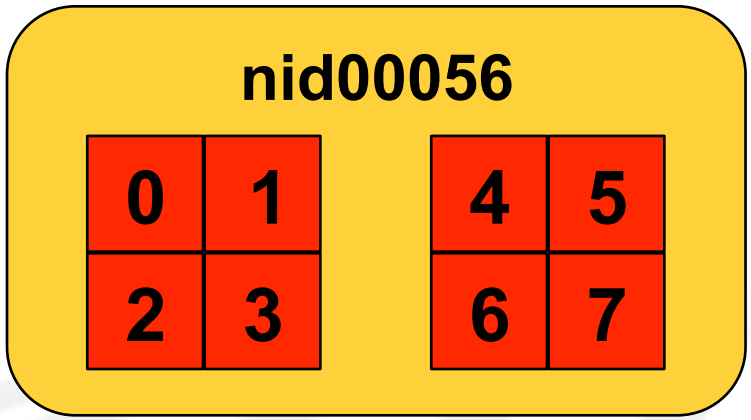
# aprun -cc cpu

```
$ aprun -cc cpu -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0)
Hello from rank 1, thread 0, on nid00056. (core affinity = 1)
Hello from rank 2, thread 0, on nid00056. (core affinity = 2)
Hello from rank 3, thread 0, on nid00056. (core affinity = 3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4)
Hello from rank 5, thread 0, on nid00056. (core affinity = 5)
Hello from rank 6, thread 0, on nid00056. (core affinity = 6)
Hello from rank 7, thread 0, on nid00056. (core affinity = 7)
$
```



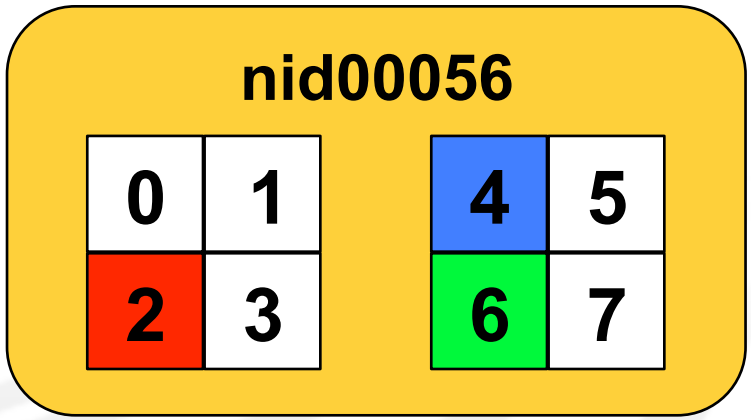
# aprun -cc none

```
$ aprun -cc none -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 4, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 0-7)
$
```



# aprun -cc list

```
$ aprun -cc 2,4,6 -n 8 -L 56-63 -q ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 2)
Hello from rank 1, thread 0, on nid00056. (core affinity = 4)
Hello from rank 2, thread 0, on nid00056. (core affinity = 6)
Hello from rank 3, thread 0, on nid00056. (core affinity = 2)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4)
Hello from rank 5, thread 0, on nid00056. (core affinity = 6)
Hello from rank 6, thread 0, on nid00056. (core affinity = 2)
Hello from rank 7, thread 0, on nid00056. (core affinity = 4)
$
```



# New Verbose Status Information

```
$ apstat -rvvv
 ResId     ApId  From          Arch   PEs N d Memory State
  4369    47722 batch:0          XT     8 0 1   1000 conf

Reservation detail
Res[0]: apid 47722, pagg 0, resId 4369, user crayadm,
       gid 14901, account 12795, time 0, normal
  Number of commands 1, control network fanout 32
  Cmd[0]: BASIL -n 8 -d 1 -N 0 -S 0 -sn 0 -sl 0x2 -a XT, mem
1000MB, nodes 2
  Reservation list entries: 8
    PE 0, cmd 0, nid 56, CPU 0xf0
    PE 1, cmd 0, nid 56, CPU 0xf0
    PE 2, cmd 0, nid 56, CPU 0xf0
    PE 3, cmd 0, nid 56, CPU 0xf0
    PE 4, cmd 0, nid 57, CPU 0xf0
    PE 5, cmd 0, nid 57, CPU 0xf0
    PE 6, cmd 0, nid 57, CPU 0xf0
    PE 7, cmd 0, nid 57, CPU 0xf0
$
```

# Use Case 1

- MPI application
- Allow placement on any node
- Default CPU binding is per NUMA domain

```
$ aprun -q -n 8 ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 4-7)
$
```

# Use Case 2

- Alter the CPU affinity to be per core

```
$ aprun -q -n 8 -cc cpu ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0)
Hello from rank 1, thread 0, on nid00056. (core affinity = 1)
Hello from rank 2, thread 0, on nid00056. (core affinity = 2)
Hello from rank 3, thread 0, on nid00056. (core affinity = 3)
Hello from rank 4, thread 0, on nid00056. (core affinity = 4)
Hello from rank 5, thread 0, on nid00056. (core affinity = 5)
Hello from rank 6, thread 0, on nid00056. (core affinity = 6)
Hello from rank 7, thread 0, on nid00056. (core affinity = 7)
$
```

# Use Case 3

- Allow Linux to migrate processes between CPUs

```
$ aprun -q -n 8 -cc none ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 2, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 3, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 4, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 5, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 6, thread 0, on nid00056. (core affinity = 0-7)
Hello from rank 7, thread 0, on nid00056. (core affinity = 0-7)
$
```

# Use Case 4

- MPI application
- Four PEs per node
- May be XT4 or XT5 nodes
- Memory affinity is local NUMA domain

```
$ aprun -q -n 8 -N 4 -cc segment ./hello | sort
Hello from rank 0, thread 0, on nid00052. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00052. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00052. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid00052. (core affinity = 0-3)
Hello from rank 4, thread 0, on nid00053. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid00053. (core affinity = 0-3)
Hello from rank 6, thread 0, on nid00053. (core affinity = 0-3)
Hello from rank 7, thread 0, on nid00053. (core affinity = 0-3)
$
```

# Use Case 5

- MPI application
- Four PEs per node
- Two PEs per NUMA domain
- Must be XT5 nodes

```
$ aprun -q -n 8 -N 4 -S 2 ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid00056. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 3, thread 0, on nid00056. (core affinity = 4-7)
Hello from rank 4, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid00057. (core affinity = 0-3)
Hello from rank 6, thread 0, on nid00057. (core affinity = 4-7)
Hello from rank 7, thread 0, on nid00057. (core affinity = 4-7)
$
```

# Use Case 6

- MPI application
- Two PEs per NUMA domain
- Stay off CPUs 0 and 1 of each NUMA domain

```
$ aprun -q -n 8 -S 2 -cc 2,3,6,7 ./hello | sort
Hello from rank 0, thread 0, on nid00056. (core affinity = 2)
Hello from rank 1, thread 0, on nid00056. (core affinity = 3)
Hello from rank 2, thread 0, on nid00056. (core affinity = 6)
Hello from rank 3, thread 0, on nid00056. (core affinity = 7)
Hello from rank 4, thread 0, on nid00057. (core affinity = 2)
Hello from rank 5, thread 0, on nid00057. (core affinity = 3)
Hello from rank 6, thread 0, on nid00057. (core affinity = 6)
Hello from rank 7, thread 0, on nid00057. (core affinity = 7)
$
```

# ALPS Tool Helper

- Provides mechanism to launch a helper process
- One helper launched per node of an application
- Controlling (login node) process provided with
  - ALPS application ID
  - Placement list
- Helper (compute node) processes provided with
  - ALPS fanout tree data
  - Local process IDs associated with application
- Helper processes establish their own communication paths
- Private interface used for integration with
  - Application debuggers
  - Checkpoint / Restart

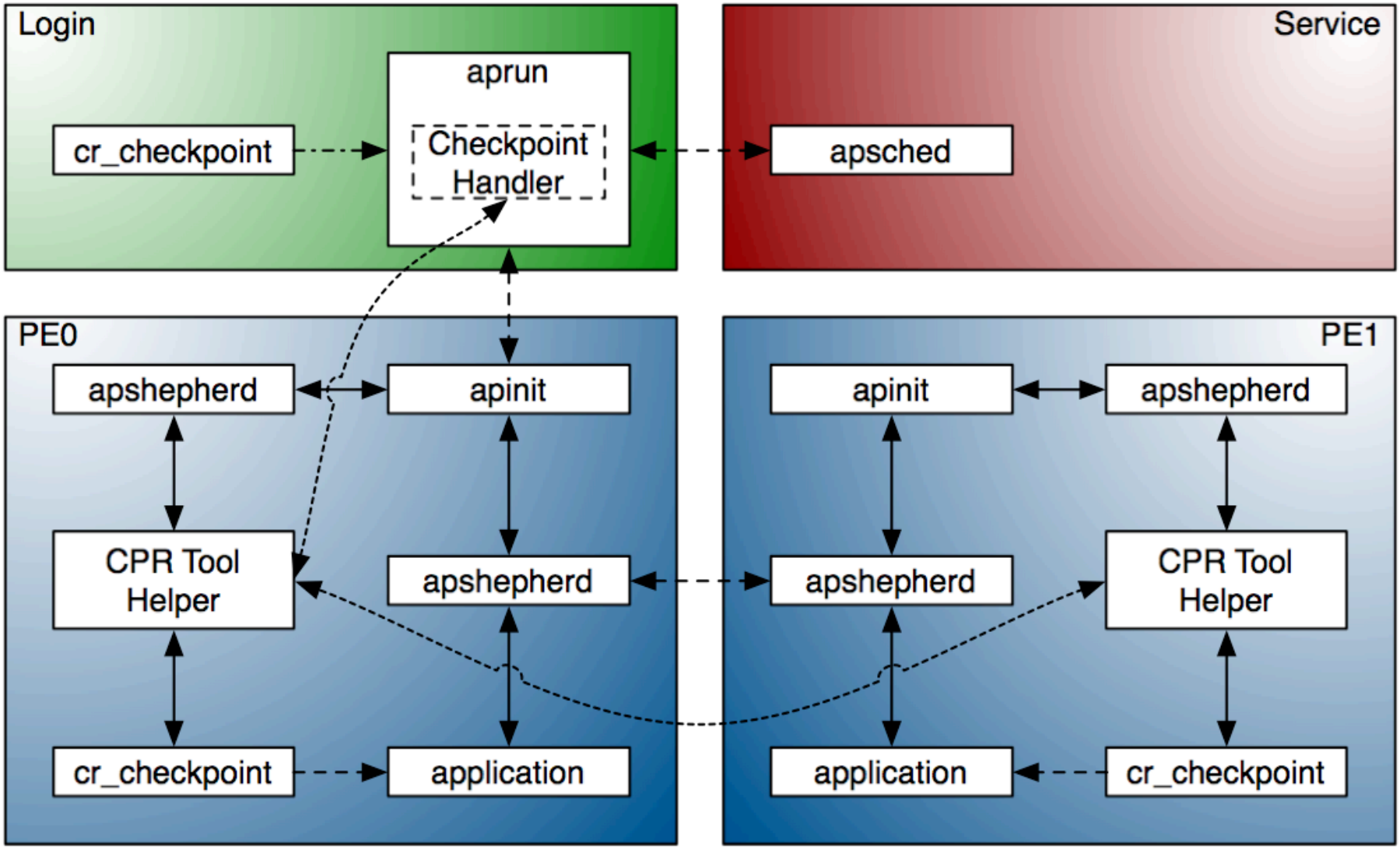
# Checkpoint / Restart Overview

- ALPS integration with BLCR
  - Berkeley Labs Checkpoint Restart
- Enhances recoverability when nodes fail
- Allows for preemptive scheduling
  - System maintenance
  - High priority jobs
- Relies on Lustre as backing store
- Utilizes ALPS tool helper interface
- Supported on XT compute nodes running CLE
  - No X2 support
  - No Catamount support
- Limited availability summer 2008

# Checkpoint Tasks

- aprun modifications
  - Checkpoint capable
    - Multithreaded (main and handler threads)
    - Register BLCR checkpoint handler
  - BLCR checkpoint handler
    - Called when checkpoint requested, determines if periodic or kill
    - ALPSMSG_CHKPNT begin sent to apsys, status updated
    - Tool helper launched to perform compute node checkpoint
    - aprun waits for compute node checkpoints to complete
    - ALPSMSG_CHKPNT end sent to apsys, status updated
- Checkpoint helper
  - Triggers and coordinates checkpoint activities on compute nodes
  - Returns status to checkpoint handler in aprun

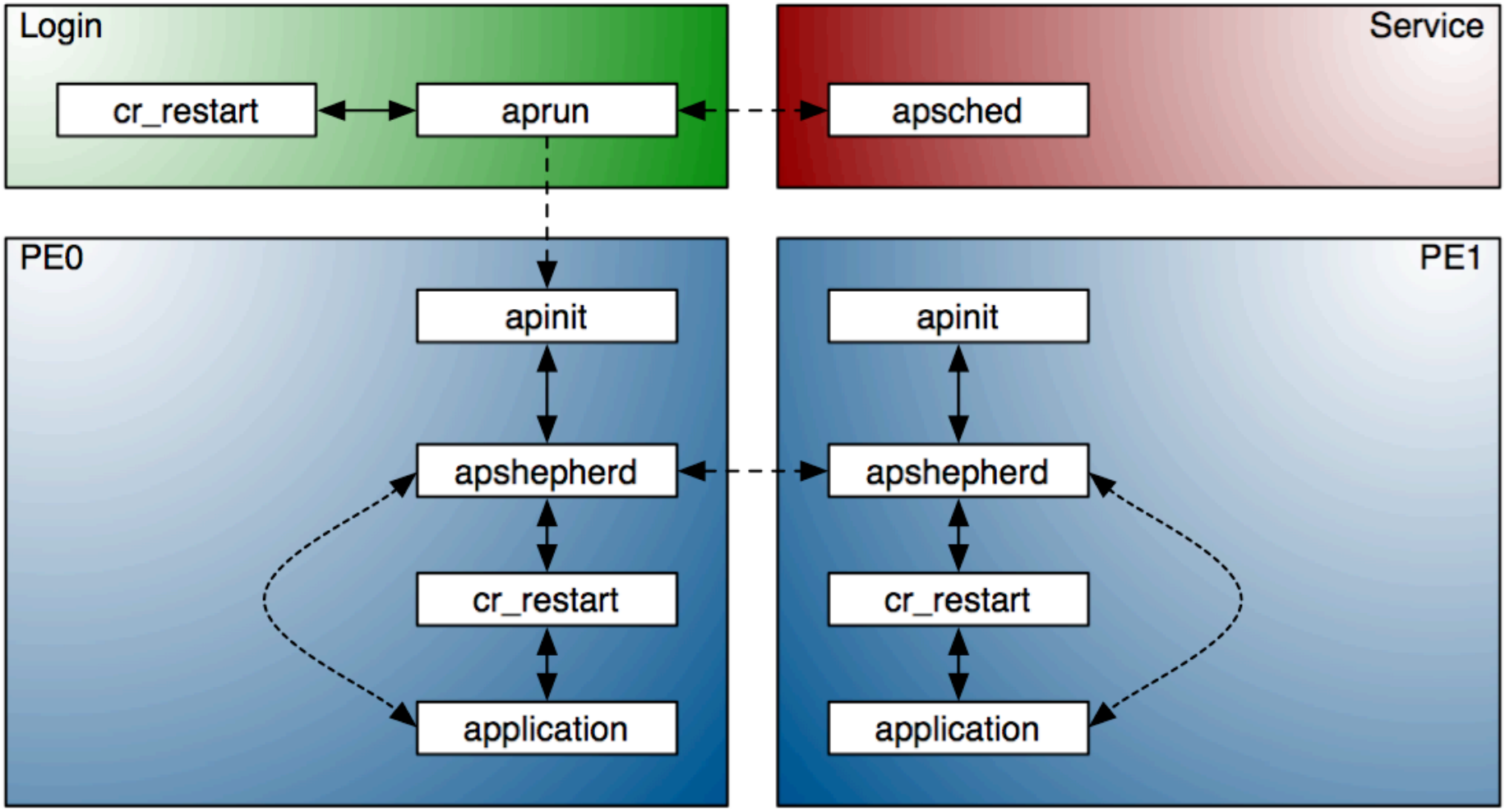
# Checkpoint Illustration



# Restart Tasks

- aprun checkpoint handler thread resumes execution
  - Obtain restart command from BLCR
  - Set bypass transfer bit (restart command present on compute node)
  - Set restart bit indicating a restart is underway
  - Yield control back to main thread
- aprun main thread resumes execution
  - Check the restart bit and prepare for restart
  - aprun files a new placement request indicating restart
  - apsched assigns a new application ID and placement list
  - aprun restarts stdin handler
  - launch of restart command proceeds
  - restart command waits for child to complete

# Restart Illustration



# ALPS for Cray XT5h Systems

- Quadrant support
  - Allows up to four node spanning applications per node
  - Allows oversubscription of CPUs, but not memory
  - Applications are context switched by ALPS
- Context switching
  - Supports CPU oversubscription
  - No gang scheduling interval (this is not gang scheduling)
  - Not supported for XT, processor to memory ratio is too high

# ALPS for X2 (1 of 2)

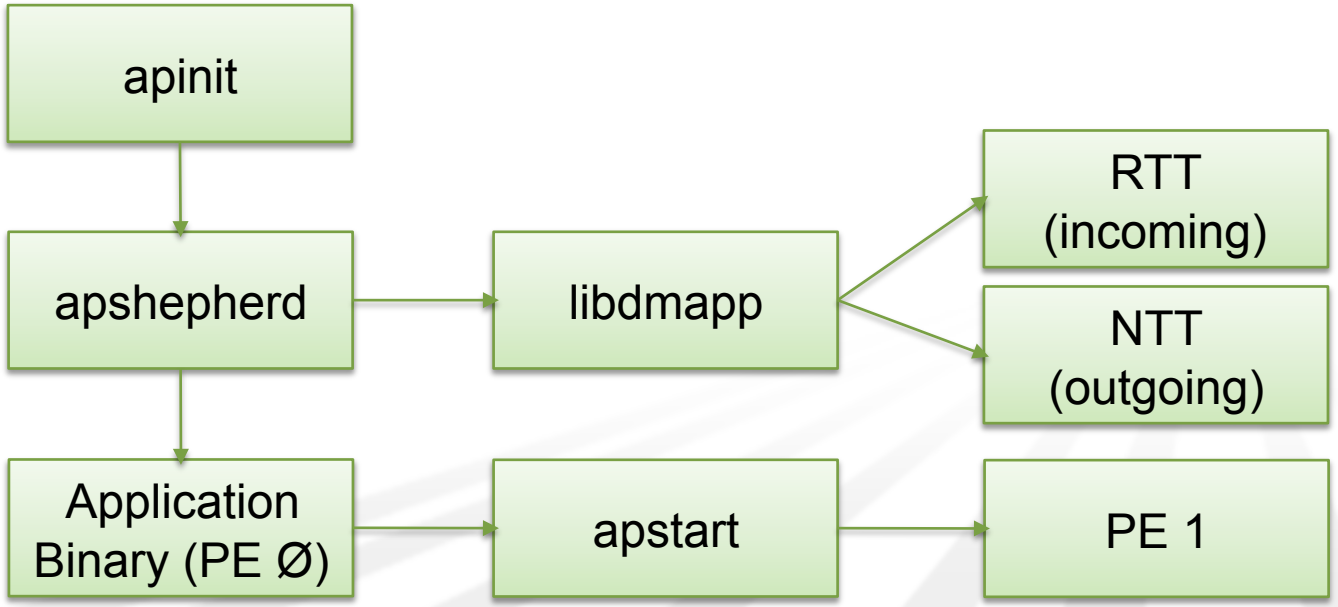
- PEs utilize DM for IPC
- Platform specific apinit daemon
  - DM placement support (NTT, RTT, processor/node granularity)
    - NTT maps virtual PE to physical endpoint
    - RTT similar to TLB, maps incoming requests
  - Uses apstart for application initialization
    - Allows ALPS to remain agnostic to programming environment
    - MPI and shmem both supported with no changes to ALPS
- 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

# ALPS for X2 (2 of 2)

- Multiple architecture support
  - Support existing (XT/X2) and future architectures
  - Bridge gathers configuration data from SDB, Mazama, etc.
  - Heterogeneous and extensible by design
  - Interactive use automatically determines binary format
  - Batch use requires user/queue to specify architecture
  - User may override architecture with aprun -a parameter
- Multiple applications may currently communicate via
  - files
  - pipes
  - sockets
- aprun -n 16 my_bw_app | aprun -n 32 my_xt_app

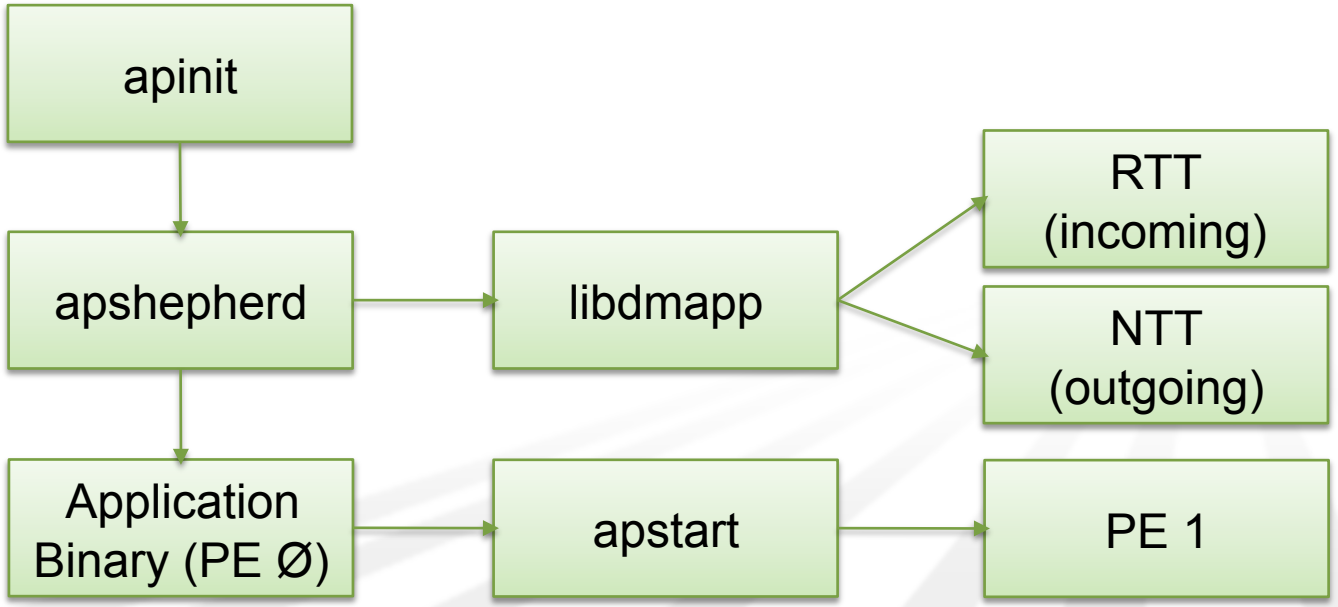
# Application Initialization for X2 (1 of 2)

- apinit forks a shepherd for the application
- apshepherd uses libdmapp to prepare DM tables
  - RTT handles incoming DM references
  - NTT handles external DM references
- apshepherd does fork/exec of target binary



# Application Initialization for X2 (2 of 2)

- Linker references apstart routine
  - Performs clone for remaining PEs on node
  - Reparents PEs to apshepherd
  - Maps huge page memory for application
  - These steps do not happen for commands, only applications
- Application begins execution



# Compiling an X2 Application

- Load appropriate modules
- Compile the application
- Strip the binary

```
$ module purge
$ module use /opt/ctl/modulefiles
$ module load PrgEnv-x2
$ cc -h omp -g -o hello_x2 hello.c
CC-7907 cc: WARNING File = hello.c, Line = 1
  The "-hscalar" level has been changed from 0 to 1 for OpenMP
processing in one
          or more functions.
$ strip hello_x2
$
```

# MPMD Application Launch

```
$ aprun -n 4 ./hello_xt
Hello from rank 0, thread 0, on nid00016. (core affinity = 0)
Hello from rank 1, thread 0, on nid00017. (core affinity = 0)
Hello from rank 2, thread 0, on nid00018. (core affinity = 0)
Hello from rank 3, thread 0, on nid00019. (core affinity = 0)
Application 409799 resources: utime 0, stime 0
$ aprun -n 4 ./hello_x2
Hello from rank 0, thread 0, on nid02048. (core affinity = 0-3)
Hello from rank 3, thread 0, on nid02048. (core affinity = 0-3)
Hello from rank 2, thread 0, on nid02048. (core affinity = 0-3)
Hello from rank 1, thread 0, on nid02048. (core affinity = 0-3)
Application 409800 resources: utime 9, stime 6
$ aprun -n 4 ./hello_xt : -n 4 ./hello_x2
Hello from rank 0, thread 0, on nid00016. (core affinity = 0)
Hello from rank 1, thread 0, on nid00017. (core affinity = 0)
Hello from rank 2, thread 0, on nid00018. (core affinity = 0)
Hello from rank 3, thread 0, on nid00019. (core affinity = 0)
Hello from rank 4, thread 0, on nid02048. (core affinity = 0-3)
Hello from rank 7, thread 0, on nid02048. (core affinity = 0-3)
Hello from rank 6, thread 0, on nid02048. (core affinity = 0-3)
Hello from rank 5, thread 0, on nid02048. (core affinity = 0-3)
Application 409801 resources: utime 9, stime 6
$
```

# BASIL 1.1

- Changes to
  - Inventory method
  - Reservation creation (batch_id added)
  - Reservation confirmation (job_name removed)

# BASIL 1.1 Inventory Request

- Updated protocol version in request

```
<?xml version="1.0"?>
<BasilRequest protocol="1.1" method="QUERY" type="INVENTORY"/>
```

- Backward compatibility
  - 1.1 requests return 1.1 responses
  - 1.0 requests return 1.0 responses
- Response includes
  - Additional node data
    - SegmentArray and Segment elements
  - Additional reservation information
    - CommandArray and Command elements
    - batch_id

# BASIL 1.1 Segment Arrays

```
<Node node_id="62" name="c0-0c1s7n2" architecture="XT" role=”BATCH” state="UP">
 <SegmentArray>
  <Segment ordinal="0">
   <ProcessorArray>
    <Processor ordinal="0" architecture="x86_64" clock_mhz="1900"/>
    <Processor ordinal="1" architecture="x86_64" clock_mhz="1900"/>
    <Processor ordinal="2" architecture="x86_64" clock_mhz="1900"/>
    <Processor ordinal="3" architecture="x86_64" clock_mhz="1900"/>
   </ProcessorArray>
   <MemoryArray>
    <Memory type="OS" page_size_kb="4" page_count="2048000"/>
   </MemoryArray>
   <LabelArray/>
  </Segment>
  <Segment ordinal="1">
   <ProcessorArray>
    <Processor ordinal="0" architecture="x86_64" clock_mhz="1900"/>
    <Processor ordinal="1" architecture="x86_64" clock_mhz="1900"/>
    <Processor ordinal="2" architecture="x86_64" clock_mhz="1900"/>
    <Processor ordinal="3" architecture="x86_64" clock_mhz="1900"/>
   </ProcessorArray>
   <MemoryArray>
    <Memory type="OS" page_size_kb="4" page_count="2048000"/>
   </MemoryArray>
   <LabelArray/>
  </Segment>
 </SegmentArray>
</Node>
```

# BASIL 1.1 Application Data

```
<ReservationArray>
 <Reservation reservation_id="3" user_name="me" account_name="DEFAULT"
                                        time_stamp="1209577894" batch_id="4321">
  <ApplicationArray>
   <Application application_id="49398" user_id=”12345" group_id="1049"
                                                    time_stamp="1209577894">
    <CommandArray>
     <Command width="1" depth="8" nppn="0" memory="1000" architecture="XT"
                                                    cmd="BASIL"/>
    </CommandArray>
   </Application>
   <Application application_id="49399" user_id=”12345" group_id="1049"
                                                    time_stamp="1209578763">
    <CommandArray>
     <Command width="1" depth="1" nppn="0" memory="1000" architecture="XT"
                                                    cmd="hello"/>
    </CommandArray>
   </Application>
  </ApplicationArray>
 </Reservation>
</ReservationArray>
```

# BASIL 1.1 Reservation Request

- Required batch_id field
  - Replaces job_name from confirm method in BASIL 1.0
  - ALPS stores numeric portion
  - Batch ID present in inventory to correlate with ALPS reservation ID
- New resource types

| aprun | BASIL | Description |
|---|---|---|
| -S | npps | PEs per NUMA domain |
| -sn | nspn | NUMA domains per node |
| -sl | segments | NUMA domain list |

```
<?xml version="1.0"?>
<BasilRequest protocol="1.1" method="RESERVE">
  <ReserveParamArray user_name="me" batch_id="4321.sdb">
    <ReserveParam architecture="XT" width="2" depth="1" npps="1"/>
  </ReserveParamArray>
</BasilRequest>
```

# Troubleshooting ALPS

- Configuration file parameters
- Tracking down problems
- Common problems

# /etc/sysconfig/alps (1 of 2)

- Present in boot root and shared root

| Parameter | Description |
|---|---|
| ALPS_MASTER_NODE | (Required) Specifies the node name (uname -n) of the service node that runs apsched. Cray recommends that the SDB node be used as the ALPS_MASTER_NODE.<br>For example: ALPS_MASTER_NODE="nid00003" |
| ALPS_BRIDGE_NODE | (Required) Specifies the node name (uname -n) of the service node that runs apbridge. This is usually the boot node. If no value is set and there is network connectivity between the master node and the SMW, the default value ALPS_MASTER_NODE is used. (Such connectivity is guaranteed to exist from the boot node.) This default value is enforced in /etc/init.d/alps.<br>For example: ALPS_BRIDGE_NODE="boot001" |
| ALPS_MOUNT_SHARED_FS | Specifies the shared file system. If a separate file system is mounted at ALPS startup to hold control data, set to yes. Default is no. Use of separate file system space is optional for configurations using a single login node. For configurations using multiple login nodes, a shared file system is required, and this parameter must be set to yes.<br>For example: ALPS_MOUNT_SHARED_FS="yes" |

# /etc/sysconfig/alps (2 of 2)

| Parameter | Description |
|---|---|
| ALPS_SHARED_DIR_PATH | (Required) Specifies the directory path to the file that contains ALPS control data. If ALPS_MOUNT_SHARED_FS is set to yes, this is assumed to be a mount point. Default is /ufs/alps_shared.<br>For example: ALPS_SHARED_DIR_PATH="/ufs/alps_shared" |
| ALPS_SHARED_DEV_NAME | Specifies the device to mount at ALPS start-up. If it is null and ALPS_MOUNT_SHARED_FS is yes, the device is determined by /etc/fstab. This parameter is not used unless yes is specified for ALPS_MOUNT_SHARED_FS.<br>For example: ALPS_SHARED_DEV_NAME="ufs:/ufs/alps_shared" |
| ALPS_SHARED_MOUNT_OPTIONS | Specifies the shared mount options. Set this parameter only if ALPS_MOUNT_SHARED_FS is yes and ALPS_SHARED_DEV_NAME is not null.<br>For example: ALPS_SHARED_MOUNT_OPTIONS="-t nfs -o tcp,rw" |
| ALPS_IP_PREFIX | Specifies the first two octets for IP addresses on the high-speed network (HSN). These are internal addresses within the HSN.<br>For example: ALPS_IP_PREFIX="192.168” |

# /etc/alps.conf (1 of 2)

- Present in the ALPS shared root

| Parameter | Description |
|---|---|
| bridge | Enables the apbridge daemon to provide dynamic rather than static information about the system node configuration to apsched. Cray strongly recommends setting the bridge parameter to use the apbridge daemon. By default, it is set to 1 (enabled). |
| alloc | If 0 or not specified, the distinction between batch and interactive nodes is enforced; if nonzero, no distinction is made. By default, it is set to 0. |
| debug | This field is set to a default level of 1 for both apsched and apsys. For information about valid values, see the apsched(8) and apsys(8) man pages. |
| fanout | This field is set to a default level of 32. This value controls the width of the ALPS TCP/IP network fan-out tree used by apinit on the compute nodes for ALPS application launch, transfer, and control messages. |

# /etc/alps.conf (2 of 2)

- Configuration example

```
$ cat /etc/alps.conf
# ALPS configuration file
# See the system admin guide for more information
# on possible settings and values

apsched
       alloc          0
       bridge         1
       fanout         32
       debug          1
/apsched

apsys
       debug          1
/apsys
$
```

# Tracking Problems

- ALPS log files
  - /var/log/alps/apschedMMDD on the SDB node
  - /var/log/alps/apbridgeMMDD on the boot node
  - /var/log/alps/apsysMMDD on the login nodes
  - /var/log/alps/apinitMMDD.NID on compute nodes
- Event logs
- Console logs
- HSS logs
- System dumps

# Common Problem 1

- Scenario: apstat is taking a long time to respond
- Probable cause: HSN or ALPS shared file system problems
- Discussion: ALPS utilizes memory mapped files over NFS to store reservation and application data. ALPS clients such as apstat may read data from these files without having to query an ALPS daemon. However, problems with the network or the underlying file system can lead to significant delays or failures when invoking apstat.
- Solution: Address the underlying HSN or NFS issues

# Common Problem 2

- Scenario: apstat shows a node up, but applications fail to launch claiming the node is unavailable
- Probable cause: HSN or apwatch problems
- Discussion: The ALPS apwatch daemon runs on the boot node and subscribes to events indicating node failure, forwarding them to apsched. If apwatch is down, these events will not be seen by apsched. Alternatively, problems with the HSN can lead to hung system calls on the compute nodes. This can lead to nodes becoming unresponsive as they wait for network requests to complete.
- Solution: Restart ALPS on the boot node or diagnose and address HSN issues

# Common Problem 3

- Scenario: aprun failure "before app startup barrier"
- Probable cause: Programming environment failure
- Discussion: ALPS shares information with the programming environment through an API called the ALPS Low Level Interface (ALPS LLI). This message is seen when there is a problem with this exchange of data. The main() function has not yet been called. The problem is most likely the result of an unhealthy node or HSN problems.
- Solution: Try using a different set of nodes to launch your application.

# Common Problem 4

- Scenario: aprun failure "No such file or directory"
- Probable cause: aprun invoked from non-Lustre file system
- Discussion: As part of application initialization, ALPS initializes the user's environment on the compute node to match that of the login node where aprun was invoked. This includes per-process limits, environment variables, and the current working directory. If aprun is invoked from a directory that is not visible on the compute nodes, this failure will occur.
- Solution: Launch the application from a Lustre mounted file system that is visible on the compute nodes. Alternatively, launching an application from /tmp will also work.

# Common Problem 5

- Scenario: node counts from apstat don't seem to add up
- Probable cause: Simple misunderstanding
- Discussion: Only placed applications with a claim against a reservation show up in the apstat -nv display. ALPS reservations created for batch jobs may not have claims against them. These reservations can be seen using the apstat -r display.
- Solution: Use apstat –r to see reserved resources

# Common Problem 6

- Scenario: ALPS not starting properly
- Probable cause: portmapper conflict
- Discussion: At Linux boot time, privileged ports are assigned by the portmapper daemon in consecutive order starting with port 600. This can cause a conflict when ALPS tries to bind to ports at startup.
- Solution: Configure the portmapper blacklist file…

```
$ cat /etc/bindresvport.blacklist
# This file contains a list of port numbers between 600 and 1024,
# which should not be used by bindresvport. bindresvport is mostly
# called by RPC services. This mostly solves the problem, that a
# RPC service uses a well known port of another service.
606         # ALPS
607         # ALPS
608         # ALPS
631         # cups
636         # ldaps
774         # rpasswd
921         # lwresd
993         # imaps
995         # pops
$
```

# Comprehensive System Accounting (CSA)

- Customized open source implementation
- Kernel patches expand data collection beyond BSD 4
- Linux process aggregates (paggs) and jobs
- Features:
  - Project based accounting
  - ALPS integration
  - Batch system integration
  - Shared file system for collection/reporting
  - Report generation

# Cray Enhancements to CSA

- Additional fields for process accounting records:
  - ALPS application ID
  - Node location (cname) and NID
  - Node architecture
  - Controlling terminal
  - Parent job ID (aprun pagg job ID from login node)
- Additional fields for start/end of job records:
  - Node location (cname) and NID
  - Node architecture
  - Parent job ID
- Additional fields for accounting configuration records:
  - Node location (cname)
  - Node architecture

# CSA at Application Launch

- Service nodes
  - ALPS collects pagg job ID and account ID, defines application ID
    - Pagg job ID acquired by batch system or PAM module
    - Account ID may be changed using account(1) command
  - Data forwarded to compute nodes during application launch
- Compute nodes
  - apshepherd ioctl() calls via libjob…
    - Set pagg job ID and parent job ID
    - Set account ID
    - Set ALPS application ID
  - Process accounting records written to local /var/csa/day/pacct file

# CSA at Application Exit

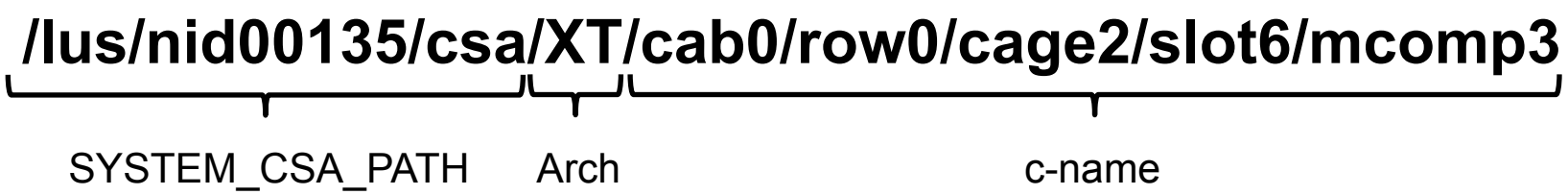
- apinit calls csanodeacct(8) indirectly
- csanodeacct(8) does the following:
  - calls csaswitch(8) to rotate current accounting file
  - Determine path to destination file based on cname
  - Evaluates COMPUTE_NODE_PROC_ACCT
    - Create application summary record OR
    - Transfers all accounting records
  - csanodesum(8) is called with pathname and summary option
    - validates all accounting records
    - forms summary records if specified
    - transfers records to shared file system

# CSA Service Node Requirements

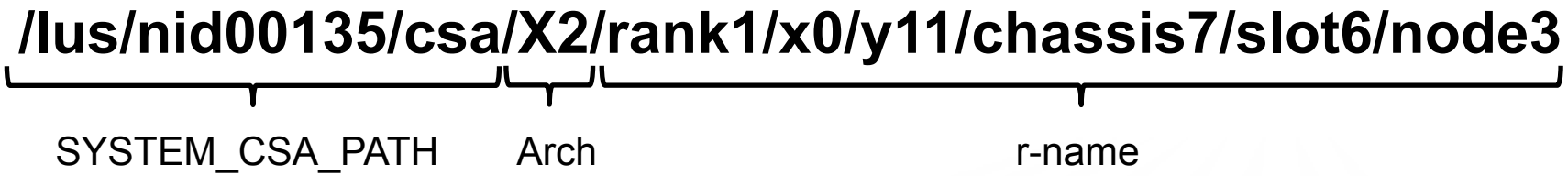
- Prerequisites
  - Persistent /var must be configured on service nodes
  - /etc/csa.conf must be edited for compute and service nodes
- Operational
  - Each login node
    - Uses cron to periodically run csanodeacct(8)
      - csanodeacct(8) calls csanodesum(8) to move the data
  - One login node
    - Invokes csarun(8) to prepare pacct files for processing
    - csarun(8) runs csanodemerg(8) to consolidate data
      - csanodemerg(8) calls csanodesum(8) to move the data
    - csaperiod(8) used to generate periodic accounting reports
- Reports are generated based on data in Lustre

# CSA Data Files

- XT example




- X2 example



# CSA Configuration for /etc/csa.conf

| Name | Value | Description |
|---|---|---|
| COMPUTE_NODE_PROC_ACCOUNT | ON \| OFF | Enables collection of individual process accounting records. Can be set differently for shared root and compute node images. |
| ACCT_SIO_NODES | 1-99 | Defines number of mount points for accounting file systems. |
| ACCT_FILE_SYSTEM_## | _lus_nid00007 | Must be one ACCT_FILE_SYSTEM_## defined for each ACCT_SIO_NODE that is configured. Each one defines an account file system mount point. Note: "_" must be used to represent "/" in pathname. This example defines a mount point on /lus/nid00007. |
| _lus_nid00007_csa_XT | c0-0c0n0s0--c0-0c2c7c3 | Defines a path for Cray XT accounting files described by the c-name range shown. These files will be written in subdirectories under the following pathname: /lus/nid00007/csa/XT |
| _lus_nid00007_csa_X2 | r10-11c0s0n0--r10-13c2s7n3 | Defines a path for Cray X2 accounting files described by the r-name range shown. These files will be written in subdirectories under the following pathname: /lus/nid00007/csa/X2 |
| SYSTEM_CSA_PATH | /lus/nid00007/csa | Defines the pathname where CSA will maintain its working directories, and also where the accounting reports are saved. |

# Sample /etc/csa.conf (1 of 2)

```
# Create only Application summary records for compute nodes (Recommended)
# Note, it may be desirable to create application summary records for
# compute nodes, and to save all process accounting records for service
# nodes. This can be done by having different settings for the
# COMPUTE_NODE_PROC_ACCOUNT parameter on the shared root versus the
# compute node image.

COMPUTE_NODE_PROC_ACCOUNT   OFF

# Define 3 SIO nodes to handle accounting files

ACCT_SIO_NODES  3

# Define the file system mount points for these SIO nodes for the
# following 3 SIO nodes:
#   /lus/nid00011
#   /lus/nid00128
#   /lus/nid00335

ACCT_FILE_SYSTEM_00   _lus_nid00011
ACCT_FILE_SYSTEM_01   _lus_nid00128
ACCT_FILE_SYSTEM_02   _lus_nid00335
```

# Sample /etc/csa.conf (2 of 2)

```
# Write accounting files to these file systems as follows:
# All cabinet 0 and 1 files to /lus/nid00011
# All cabinet 2 files to /lus/nid00128
# All cabinet 3 and 4 files to /lus/nid00135
# Make sure all rages of possible node cnames are covered by the
# configuration.
# Make sure that there is no overlap between the different file systems.

_lus_nid00011_csa_XT        c0-0c0s0n0--c1-0c2s7n3
_lus_nid00128_csa_XT        c2-0c0s0n0--c2-0c2s7n3
_lus_nid00335_csa_XT        c3-0c0s0n0--c4-0c2s7n3

# Setup up the system wide CSA accounting file and the CSA working
# directories on /lus/nid00128
SYSTEM_CSA_PATH     /lus/nid00128/csa
```

# Using csacom

- Searches and prints CSA accounting files
- One entry per process, per node
- Extended options
  - -l1 prints NID
  - -l2 prints c-name
  - -l3 prints NID and c-name

```
$ csacom -P -J -l2 pacct
ACCOUNTING RECORDS FROM:   Mon Apr  7 13:33:33 2008
COMMAND                    START    END          REAL       CPU                          PROJECT     NODE PHYSICAL NODE
NAME       USER      TTY   TIME     TIME        (SECS)    (SECS)          JOB ID          ID         LOCATION TYPE
#gunzip    root      0     17:39:17 17:39:17      0.23      0.06          0x2437           0       c0-0c1s0n2    1
#cpubound  beh       0     17:39:17 17:39:47     30.01     30.00          0x2437        8036       c0-0c1s0n2    1
#apinit    root      0     17:39:17 17:39:47     30.14      0.03          0x2437        8036       c0-0c1s0n2    1
#gunzip    root      0     17:39:17 17:39:17      0.23      0.07          0x2437           0       c0-0c1s0n3    1
#cpubound  beh       0     17:39:17 17:39:47     30.00     30.00          0x2437        8036       c0-0c1s0n3    1
#apinit    root      0     17:39:17 17:39:47     30.14      0.00          0x2437        8036       c0-0c1s0n3    1
#gunzip    root      0     17:39:16 17:39:16      0.23      0.05          0x2437           0       c0-0c1s1n0    1
#cpubound  beh       0     17:39:17 17:39:47     30.00     30.00          0x2437        8036       c0-0c1s1n0    1
#apinit    root      0     17:39:16 17:39:46     30.13      0.00          0x2437        8036       c0-0c1s1n0    1
#gunzip    root      0     17:39:17 17:39:17      0.23      0.06          0x2437           0       c0-0c1s2n0    1
#cpubound  beh       0     17:39:17 17:39:47     30.00     30.00          0x2437        8036       c0-0c1s2n0    1
#apinit    root      0     17:39:17 17:39:47     30.13      0.01          0x2437        8036       c0-0c1s2n0    1
...
```

# Thank You!

- The ALPS development team
  - Marlys Kohnke
  - Carl Albing
  - Jim Nordby
  - Jason Coverston
- The CSA development team
  - Don Hankins
- Group manager
  - Blaine Ebeling
- Technical Lead / Chief Procrastinator
  - Michael Karo

- Questions, comments, feedback, and discussion…