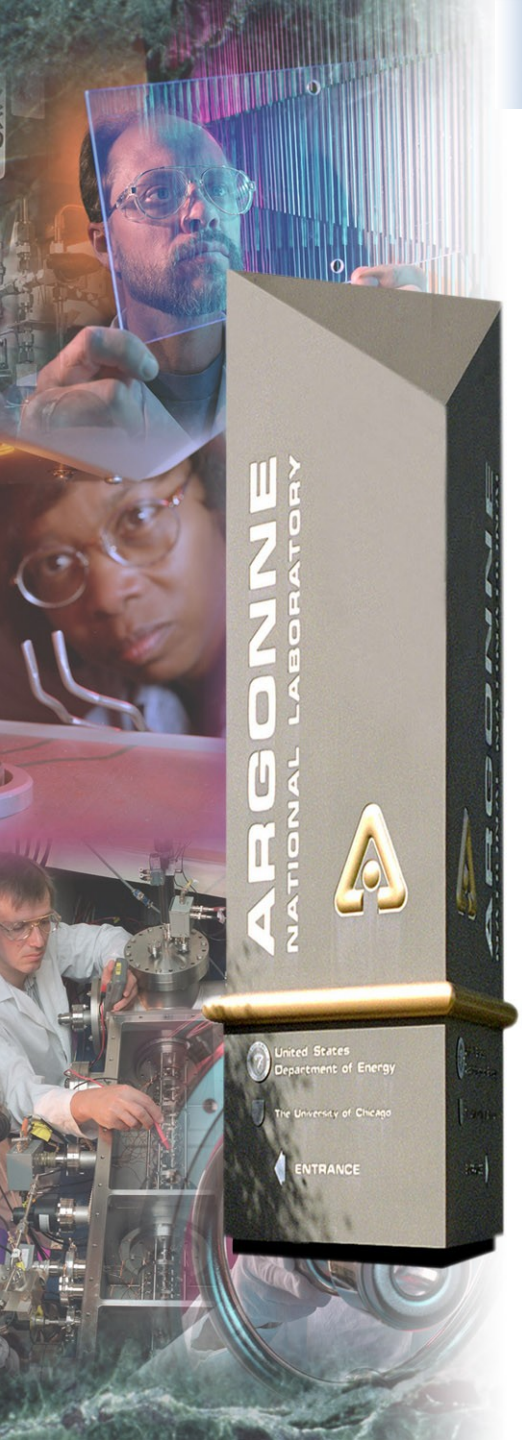


ARGONNE NATIONAL LABORATORY

Cobalt: An Open Source Platform for HPC System Software Research

Edinburgh BG/L System Software Workshop

Narayan Desai
Mathematics and Computer Science Division
Argonne National Laboratory
October 6, 2005

THE UNIVERSITY OF CHICAGO

Office of Science
U.S. DEPARTMENT OF ENERGY

Argonne National Laboratory is managed by
The University of Chicago for the U.S. Department of Energy

Cobalt Research Project

- Goal: Investigate advanced systems management for complex ultra-scale architectures
- Strategy:
  - Build an Open Source platform of parallel tools and components that enable rapid experimentation and exploration of advanced features
- Architecture
  - Components based on the SciDAC Scalable System Software project
    - A collection of interacting components
    - A communication and event system
    - Well-defined interfaces between components
  - Portable
    - *BG/L systems*
    - *Linux and MacOSX Clusters*
  - Initial focus on reconfigurable user environments

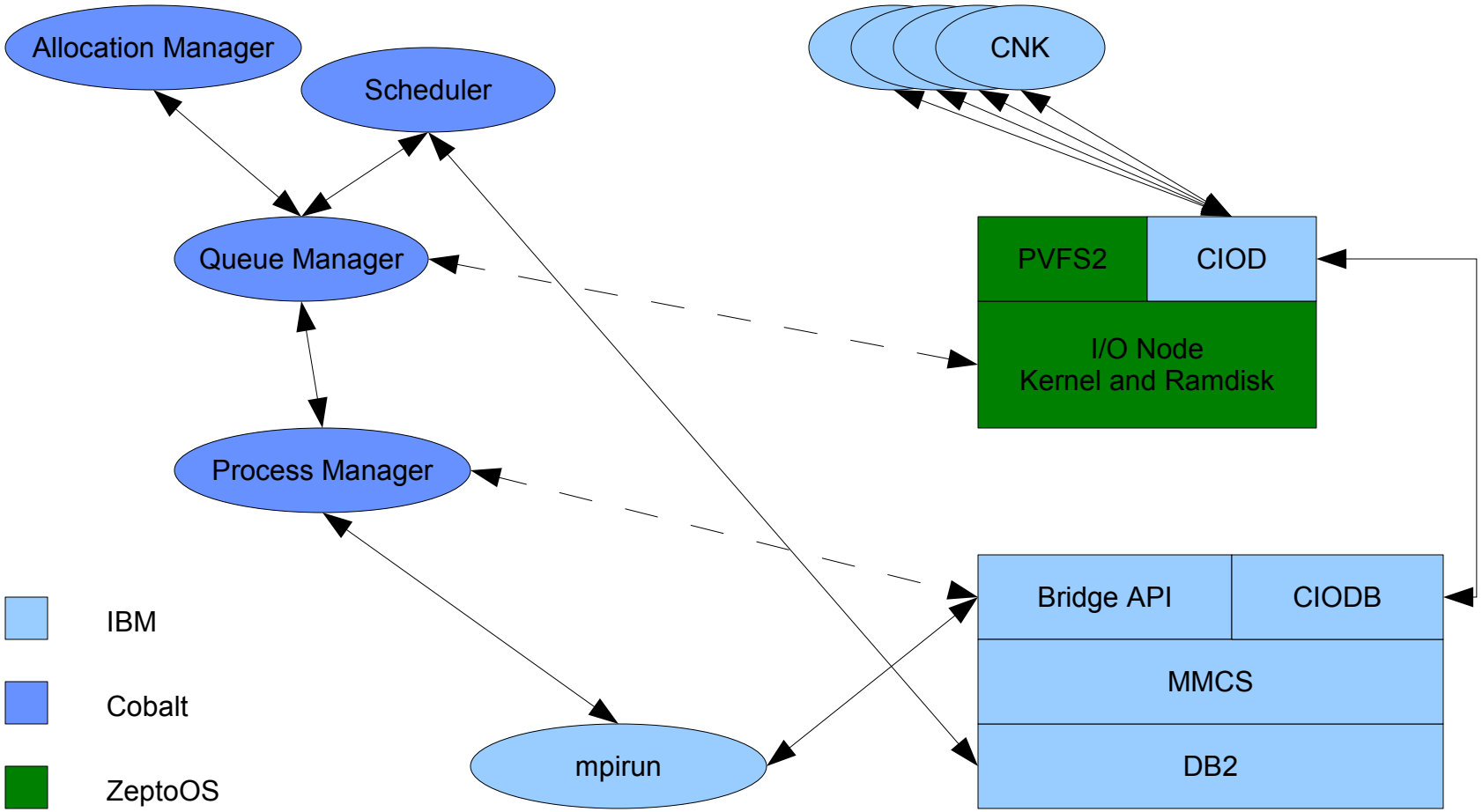
Motivation

- Needed to support both computational and computer science users
- System software developers have different needs from computational scientists
  - System hangs are common, even desired
  - A large variety of configurations are required, sometimes simultaneously
  - The *"application"* can span all software on a node
- System failures are more common during system software research and development
  - System software must deal gracefully with faults
- Most resource managers not suitable for system software research environments

Resource, Job, and Queue Management

- Classic packages: OpenPBS, PBS Pro, Maui, LSF, LoadLeveller
- Shortcomings:
  - Difficult (or impossible) to modify large monolithic systems
  - Interfaces between components poorly documented
- Cobalt: smaller and simpler is better
  - Cobalt guts admittedly feature-poor
    - “RISC approach to resource and job management”
    - ~4K lines of component code (mostly Python)
    - ~1K lines of BG/L specific code
  - However, its agility makes it the perfect research platform
    - Rapid reconfiguration of components permits exploration of many interlinked system management issues
    - Porting and adapting code to new platforms and system models is relatively easy
    - Small codebase size makes the system easy to modify, when needed
  - If you don't like a particular component implementation, write another one

Cobalt Architecture on Blue Gene



Cobalt on BlueGene/L

```
%man cqsub
cqsub - submit jobs to the queue
   manager for execution

cqsub  [-d]  [-q  queue]  [-p project]
   [-t time] [-n number of nodes] [-m
   mode] [-c process count]
   <executable> <options>
[…]
-k kernel profile
Run the job with the specified kernel
   profile.
```

- Dynamic Kernel Selection
  - Combined with ZeptoOS, jobs can use *different* I/O node kernels and tuning parameters
  - This extremely important user-level feature is *ONLY available via Cobalt*
  - Provides unique flexibility to explore new territory
  - Testing and benchmarking of experimental kernels
    - Application-dependent kernel tuning
  - Required for system software research
- Small partition support
  - Cobalt's component-based design allowed rapid support for 32-node partitions
  - Absolutely critical for application porting workshops
  - The most requested Cobalt feature for BG/L!

Submitting Jobs: cqsub

- 32 nodes
- 30 minutes

```
$ cqsub -n 32 -t 30 cpi
2702
```

- virtual node mode
- 64 processes
- default queue

```
$ cqsub -n 32 -c 64 -m vn -t 30 -q default cpi
2703
```

Other Options

- Output Prefix: -O <output path prefix>
- Kernel Profile: -k <profile>
- Environent Variables: -e “var1=val2:var2=val2”
- Working Directory: -C <path>

Job Status: cqstat

```
cqstat -f
JobID      User   WallTime   Nodes     State        Location   Mode   Procs     Queue           StartTime
==========================================================================================================
16674    cpsosa   03:40:00    1024    queued             N/A     co    1024   default                 N/A
16743   runesha   02:00:00    1024    queued             N/A     co    1024   default                 N/A
16941    linsay   24:00:00     128   running  R000_J102-128     vn     256   default  10/03/05 11:59:05
16982   fischer   24:00:00     512   running        ANL_R001     vn    1024   default  10/04/05 03:40:24
```

Killing Jobs: cqdel

```
cqsub -n 32 -c 1 -t 10 -k ZeptoOS-1.1 ./test
14
Cqstat
JobID User WallTime Nodes State Location
=========================================================
14 beckman 00:10:00 32 running UE_R001_32B

cqdel 14
    Deleted Jobs
JobID User
================
14 beckman
```

Combined with ZeptoOS, jobs can use *different* I/O node kernels and tuning parameters

Partition Status: partlist

```
$ partlist
<Partition admin="online" name="R000_J102-32" queue="short" state="idle" />
<Partition admin="online" name="ANL_R001" queue="default" state="busy" />
<Partition admin="online" name="R000_J102-64" queue="short" state="idle" />
<Partition admin="online" name="R000_J106-64" queue="short" state="idle" />
<Partition admin="online" name="R000_J111-64" queue="short" state="idle" />
<Partition admin="online" name="R000_J115-64" queue="default" state="idle" />
<Partition admin="online" name="R000_J203-64" queue="default" state="idle" />
<Partition admin="online" name="R000_J207-64" queue="default" state="idle" />
<Partition admin="online" name="R000_J210-64" queue="default" state="idle" />
<Partition admin="online" name="R000_J214-64" queue="default" state="idle" />
<Partition admin="online" name="R000_J102-128" queue="default" state="busy" />
<Partition admin="online" name="R000_J111-128" queue="default" state="idle" />
<Partition admin="online" name="R000_J203-128" queue="default" state="idle" />
<Partition admin="online" name="R000_J210-128" queue="default" state="idle" />
<Partition admin="online" name="R000_J104-32" queue="short" state="idle" />
<Partition admin="online" name="R000_J106-32" queue="admin" state="idle" />
```

Scheduling

- Partitions are defined for scheduling purposes
  - Includes size, queue, etc
- One partition definition per location for user jobs
  - Partitions can overlap, but dependencies need to be definied
- The scheduler can effectively pack jobs onto the machine
- Greedy backfill is implemented
- Reservations
- Per-Queue policies
  - default (fifo + backfill)
  - short queue (< 30 minute jobs)
  - easy to implement more

Dynamic Kernel Selection

- User-setup kernel profiles
  - includes CNK, ION kernel, ION ramdisk, and loader
- Each partition configured with a partition specific boot location
- User jobs include a kernel profile
  - with a default profile of “default”
- The partition specific boot location is a symlink
- Cobalt modifies this link during each job, once execution location has been established
- The partition boots the specified kernel upon job startup

Active Development Areas

- Support for user specified ZeptoOS kernel and ramdisk options
- Direct process manager interface via the Bridge APIs (replacement for mpirun interactions)
- Python bindings for the Bridge APIs
  - Will allow easy implementation of BG/L specific management and status scripts
- Scheduler improvements
  - Multi-rack allocation policies
  - More efficient backfill
  - Investigate rule-based scheduling policies
- Explore coordination of ZeptoOS node startup and shutdown features
  - Better user interface for system software development
  - Perhaps remove the partition reboot requirement
- Improved installation process

Cobalt Code Status

- In production at Argonne since 08/2003
  - on BG/L since 02/2005
- In production at NCAR since 05/2005
- Available at EPCC since Tuesday (10/2005)
- Under evaluation at several other sites
- Development process is open
  - Suggestions and patches are both welcome
  - NCAR has helped with both code and documentation improvements
- Available from http://www.mcs.anl.gov/cobalt

Cobalt Results

- Small codebase allows easy modifications
  - Usability improvements
  - New features (~3 minutes is the current record)
  - Site-specific customizations
  - Porting to new systems is quite easy
- Providing a uniform interface for clusters and BG/L systems
- Properly arbitrating between system software developers and computational science users
  - Allows on-the-fly system configuration changes
  - Ensures that computational jobs get non-development versions of system software
  - Able to protect each user group from the other and its software requirements
- Theraputic effect on sysadmin blood pressure
  - System software small enough to be readily understood and modified

The End

- Questions?

http://www.mcs.anl.gov/cobalt