SSS Update 5/05
Infrastructure and Blue Gene

Narayan Desai

desai@mcs.anl.gov

Overview

- SSS Infrastructure changes
- Cobalt
- Blue Gene deployment
- Validation of interfaces
- Differences between cluster and BG/L Implementations

SSS BCM Updates

- Quite stable system
- Only one bugfix in the last 4 months.
- SSS Infrastructure in use in many locations at ANL
  - Clusters
  - BG/L
  - ia32 and ppc64
- Better documentation now exists (finally)

LRS Syntax

- Language specification is done
- SDK implementation is complete
  - Message processing library
  - Server class integration
- SSSlib integration remains to be done
  - needed for a LRS service directory
  - will be finished this summer
- Components can be written with LRS right now

Blue Gene/L

- Arrived in January
- Online with Cobalt in February
- Substantial feature requests starting in March
- 1 rack system
  - 1024 compute nodes (2048 processors)
  - 32 I/O nodes (system call nodes; 1 per 32 compute nodes)
  - 16 storage nodes
  - 1 service node
  - 4 login nodes
- Complex allocation requirements
  - allocated by I/O node + compute node chunks
  - requires network partitioning
- DB2 used for everything

Cobalt

- Same software as on Chiba City
- Like SSS, somewhat cluster-centric
- Complete system implementation of most SSS components
  - System Management
  - Resource Management
  - Process Management
- All python components
  - implemented using the SSS-SDK
- Several major extensions required for Blue Gene/L

Job Execution Architecture

- Similar to Chiba City
- Four main components involved
  - Scheduler
  - Queue manager
  - Process manager
  - Allocation manager

Scheduler (bgsched)

- New implementation
  - Cluster scheduler not really appropriate
  - Needs to match system topology for allocations
  - Needs to coordinate with DB2
- Topology aware
  - Understands system partitioning
  - Can detect conflicts
- DB2
  - Can detect job failures
  - Works around broken IBM tools

Queue Manager (cqm)

- Same software as on Chiba City
- Minor modifications to accommodate system oddities
  - Addition of execution modes
- Used existing support for on-the-fly operating system changes
  - Changes on clusters far more heavyweight
  - Blue Gene/L nodes can be rebooted before every job, so OS changes are simple

Process Manager (bgpm)

- New implementation
- Compute nodes don't run a full OS, so MPD is not an option
- Native process management system uses private interfaces to boot nodes and load applications
- mpirun is extraordinarily complicated
  - Reboots nodes
  - Boots I/O nodes
  - Loads application
  - Talks to DB2
  - Not quite perfect

Allocation Manager (am)

- Same as on Chiba City
- Quite simple minded
  - Project verification
  - Usage tracking
- So far, we haven't needed much more functionality, but this will be changing

mcs

Experiences

- Small codebase
  - Easy to port
  - Simple to find and fix bugs
- Simple approach makes the system easy to understand
- Agility has been absolutely required on BG/L
  - Frequently changing (IBM supplied) system software
  - Odd bugs frequently crop up
  - All software is under constant development
- Comprehensive interfaces expose actionable information
  - Administrators can access internal state
  - Makes component behavior less mysterious
  - Extracting new information is easy

What Have We Learned?

- The basic component interfaces are right
  - However, modifications will be required to accommodate esoteric platforms and unusual hardware features
- Component interfaces simplify the porting process
  - Appropriate components can be reused
  - New components easily integrated as needed
- Component architectures are pretty manageable
  - Abstract across heterogeneity
  - Administrators get familiar software across architectures
  - Administrators like systems composed of little pieces