Exploiting the ccNUMA Architecture for Application Performance

Chris Smith,
Integration Architect

Overview

- LSF Basics
- IRIX cpusets
- LSF and cpuset integration
  - Components of the integration
  - Component interactions
- Results

LSF Basics

- Used to build clusters
  - from NOW to single, capability SMP
- Centralized scheduling
  - queues are not host-based
  - services is available as long as one node in the cluster is available
- Rich set of scheduling algorithms
  - preemption
  - fairshare (host and queue level)
  - cpu reservation and backfill

IRIX cpusets

- Cpusets:
  - Group physical CPUs (& memory) as unit
  - Jobs execute in cpusets
  - Can set process and memory policies
- Static - can exist across reboots
- Dynamic - create/destroy before/after job runs
- Key benefit – control runtime variability by creating affinity between processes and local (node) memory

Cpuset integration with LSF

- Cpusets provide containment for jobs (open cpusets) as well as dedicated processors for the job (exclusive cpusets)
- Can request the supported cpuset options (e.g. MEMORY_MANDATORY, etc)
- A first-fit algorithm and a best-fit algorithm have been implemented
- Best-fit based on minimizing router hops between cpus in the cpuset

Components – topology daemon

- The topology daemon runs on each node in the cluster which supports cpusets
- It maintains a snapshot of the hardware graph, currently allocated cpusets, and understands how to create/destroy cpusets
- It supports requests to query the current availability of cpus and cpusets, and supports requests to create/destroy cpusets.

Components – external scheduler

- Mbatchd uses an external scheduler library in order to determine which node is best for a job based on topology requirements
- The external scheduler uses information from the topology daemon in order to make placement decisions
- This allows for a coordination of topology scheduling with LSF's current built-in scheduling policies

Components – job execution

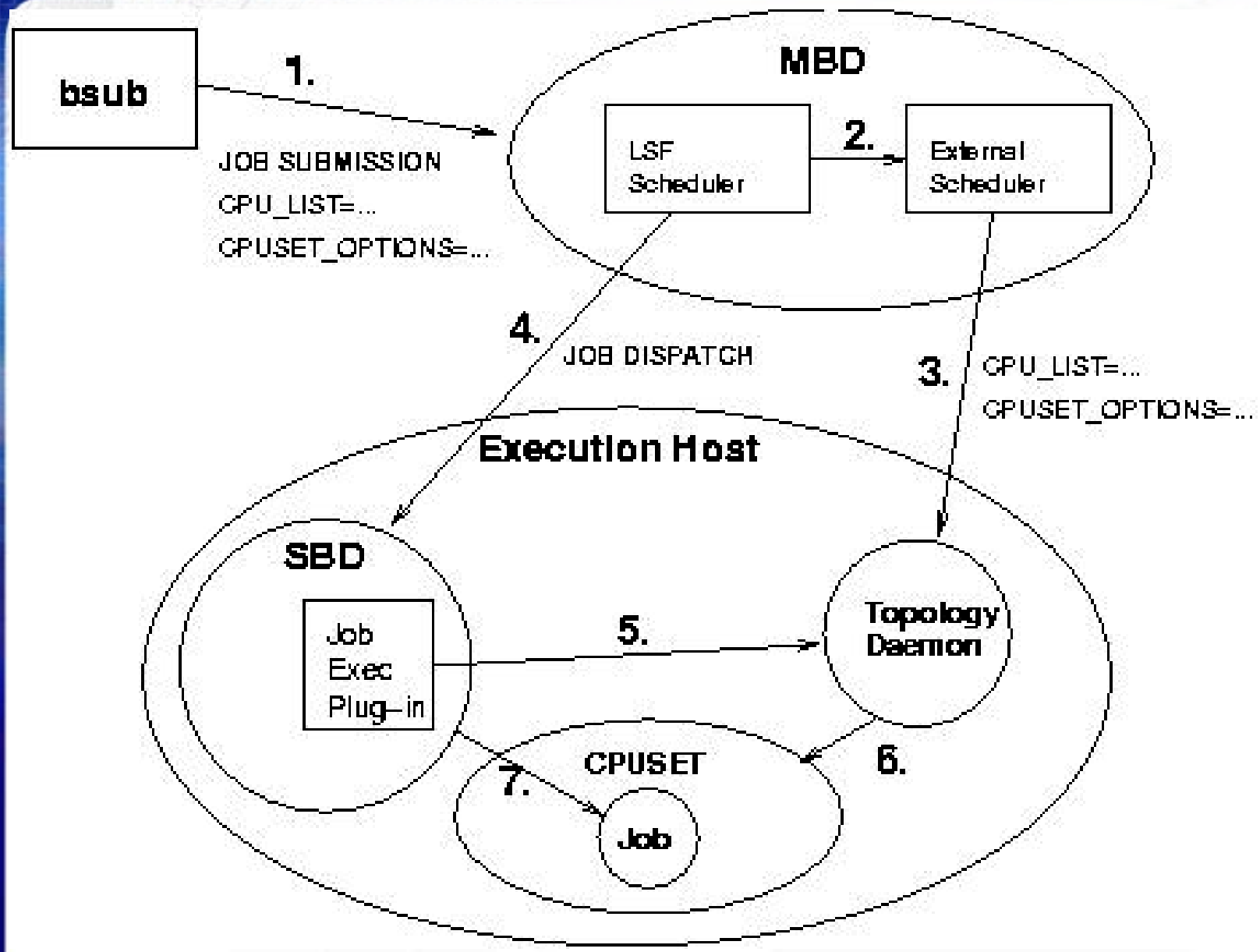
- Sbatchd loads a shared object (plugin) which knows how to bind the job to a cpuset
- Plugin contacts the topology daemon to create the cpuset before the job runs, and to destroy it when the job is done.
- Plugin places itself in the cpuset created for it by the topology daemon using cpusetAttach() API call

Components – job submission

- Cpuset options are passed through a new option to bsub – “-extsched”
- For example, the following submission runs the job within an exclusive cpuset of 8 cpus, with the MEMORY_MANDATORY option set:

```
Bsub –R “span[hosts=1]” –n 8 –extsched \
“CPUSET_OPTIONS=\
CPUSET_CPU_EXCLUSIVE,CPUSET_MEMORY_MANDATORY” \
myjob
```

Cpuset Integration Architecture



PLATFORM

Results: LSF + cpusets

- Customer throughput benchmark
- 50 jobs (10 codes) – all MPI, one hybrid
- No changes to number of processors, or order of job mix allowed
- Submission order included some 5 minute sleeps
- No tricks allowed
- Run on a 128p/128G Origin 3000

Results: LSF without cpusets

- **Total elapsed time = 4:19:01**
- **In particular, two identical pgm_02 jobs:**

```
038_pgm_02:real      39:34.36
039_pgm_02:real       8:17.02
```

Results: LSF with cpusets

- **Total elapsed time = 2:51:06**
- **In particular, two identical pgm_02 jobs:**

```
038_pgm_02:real      8:08:39
039_pgm_02:real      8:03.69
```

- **Ideal time (no sleeps) = 2:21:22**
- **Time without sleeps = 2:35:47**