Experiences in Managing Resources on a Large Origin3000 cluster

CUG Summit 2002,
Manchester, May 20 2002,
Mark van de Sanden & Huub Stoffers
http://www.sara.nl

A Coarse Outline of this Presentation

- Overview Origin3000 cluster called ‘Teras’
  - General Information about `Teras’
  - `Teras’ Cluster Configuration
  - Key Software Components
  - Batch Environment
- User Experiences Managing Resources on `Teras’
  - Why Managing Resources?
  - User Experiences on Managing CPU, Memory and I/O resources
- Conclusions

General Information about Teras (1/2)

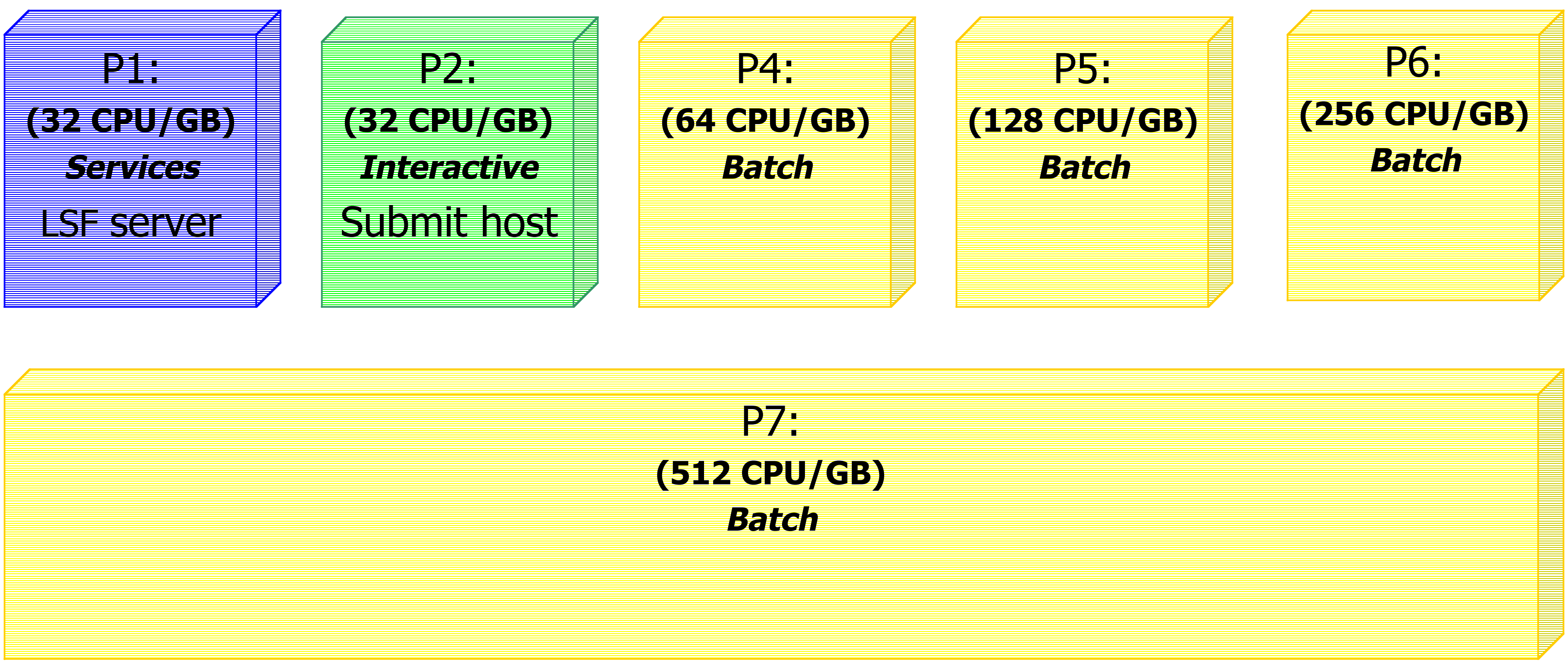
- Dutch national supercomputer
- Funded by the *Netherlands National Computing Facilities Foundation* (NCF)
- Available to the academic community of the Netherlands
- Currently about 250 active users

General Information Teras (2/2)

Basic Teras Hardware Specifications

- 1024 x 500 MHz MIPS R14,000 CPUs
- 1 Teraflops/sec. theoretical peak performance
- 1 Terabyte of memory (1Gigabyte/cpu)
- 10 Terabytes of net on-line RAID5 storage (FC RAID array, 10,000 RPM disks)
- 100 terabytes of near-line storage (tapes for data migration, archiving and backup)

Teras Cluster Configuration



Key Software Components

- Operating System
  - IRIX 6.5.14f
  - Cpusets
  - Joblimits
- Batch environment
  - LSF 4.1 (with cpusets, joblimits support)
- Programming environment
  - MPT 1.4.0.2 (MPI, shmem, PVM)
  - MIPSpro 7.3.1.2 (OpenMP)

Batch Environment

- Most jobs are single or MPI/OpenMP parallel
- All jobs are scheduled on a single host
- CPU and Memory requests are adapted to the 1 Gigabyte/ 1 CPU ratio
- No scheduled overcommitment on CPUs or memory in batch environment
- Largest regularly scheduled job can have 256 CPUs or Gigabyte

Why Managing Resources? (1/2)

- To guarantee that the requested resources by a job are available for the duration of the job
- To achieve reproduceable timings for jobs
- To achieve the highest possible system utilization
- Prevent monopolization of the system by 1 or a type of job

Why Managing Resources? (2/2)

- LSF configured for the use of CPUsets
- Wrapper around LSF submit command to enforce 1 Gigabyte/CPU ratio
- Multiple queues in LSF
- Limiting the number simultaneous runable jobs for certain type of jobs

System resources

- CPU resources
  - The requested number of CPUs for the duration of the job
- Memory resources
  - Maximum memory usage within a job for the duration of the job
- I/O resources
  - Cannot be requested by the user or job
  - Strongly depends on type of program
  - Prevent monopolization by 1 job or a type of job

Available System Tools (1/3)

- CPUsets
  - Makes groups of CPUs and memory
  - Processes are attached to a CPUset, child process are automatically attached
  - Regulates access to resources outside the CPUset for processes bound to a CPUset
  - Regulates access to resources within the CPUset for processes not attached to the CPUset

Available System Tools (2/3)

- Joblimits
  - Group processes into a job container
  - Set resource limits on groups of process within job container, similar to userlimits
  - When 1 process exceeds a limit, it effects all processes with the job container
  - Not all limits are destructive, processes are not killed
  - Parent process creates job container, child processes belongs automatically to job container

Available System Tools (3/3)

- LSF batch scheduler
  - With CPUsets and joblimits support

Managing CPU resources (1/3)

- LSF settings
  - 1 jobslot per CPU defined, PJOB_LIMIT=1.000
  - 4 CPUs are reserved for system processes, number of jobslots available per host = number CPUs - 4
  - LSF creates a CPUset per job
    - LSF runs special daemons to determine number of free CPUs
    - LSF knows topology of Origin3000 architecture, CPUs are select via best-fit algorithm

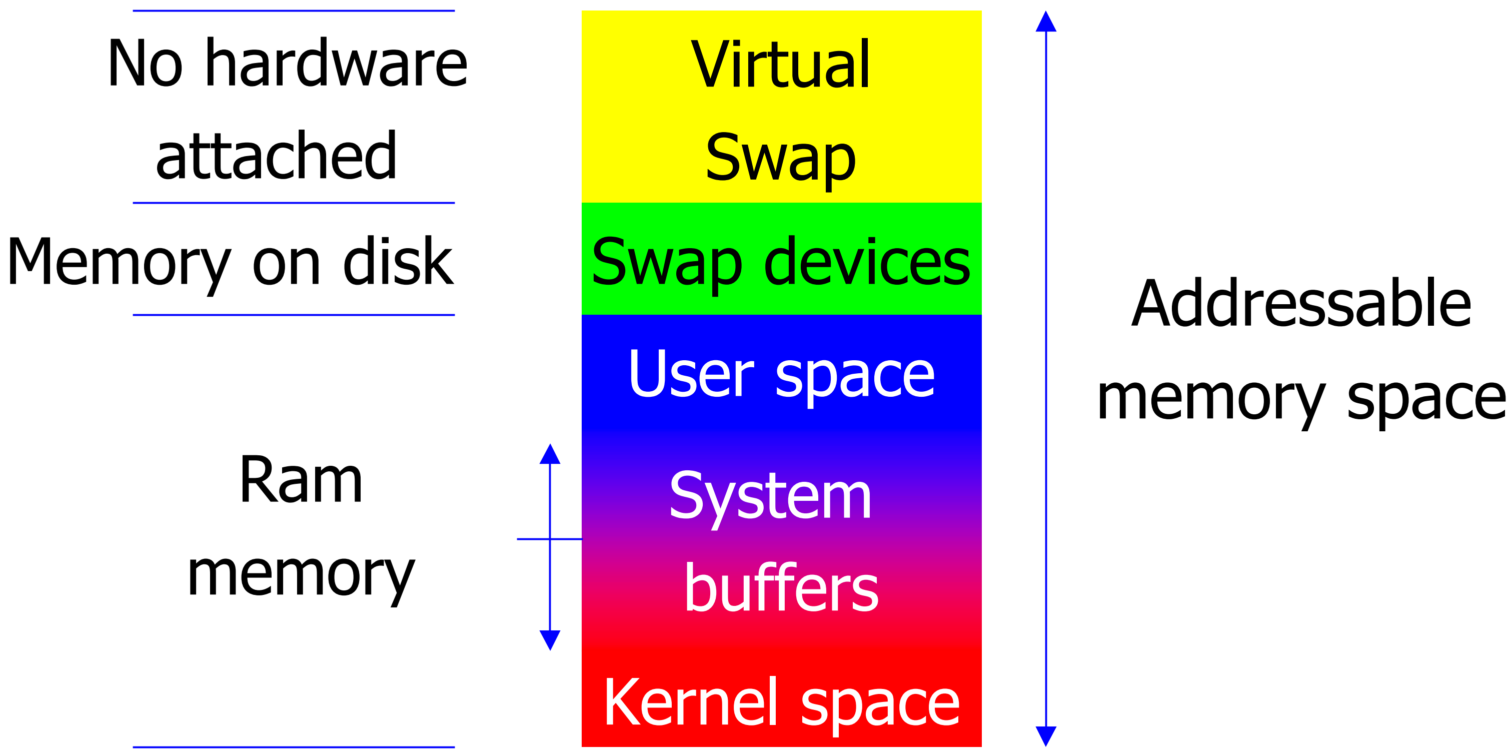
Managing CPU resources (2/3)

- CPUset tokens are set via LSF_DEFAULT_EXTSCHED
  - CPUSET_CPU_EXCLUSIVE defines a restricted CPUset
    - Attached processes run only on CPUs allocated to the CPUset
    - Non-attached processes are not allowed to run on allocated CPUs
- LSF creates per job a job container
  - LSF defines own ULDB domain
  - sets current and maximum CPU time limits on defined queue PROCLIMIT*RUNLIMIT settings
  - monitors wall-clock (RUNLIMIT) time of jobs and kills when exceeded

Managing CPU resources (3/3)

- MPI with CPUsets
  - Normally arrayd is parent of MPI child processes
  - Needs at least MPT version 1.3
    - MPI master process is parent of MPI child processes
- PVM with CPUsets
  - Normally 1 pvmd per user per host
  - Every job must have it's own pvmd
    - Use PVM_VMID environment variable

Managing Memory resources (1/7)



Simple memory overview

Managing Memory resources (2/7)

- Memory on IRIX
  - malloc() reserves logical memory, this means that only memory counters are recalculated
  - Physical memory is allocated on first touch
  - System can be out of logical memory even with physical memory available
  - Virtual memory (swap) is needed to solve this
    - Defined swap space without physical hardware attached to it
    - IRIX has a maximum of 1 Terabyte of virtual swap

Managing Memory resources (3/7)

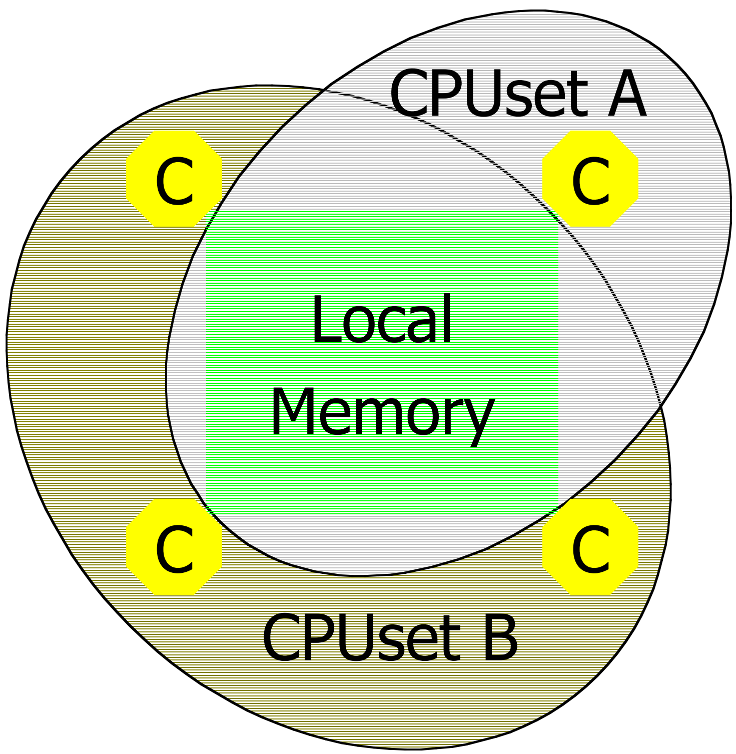
- IRIX cannot calculate memory usage
  - Shared memory usage is not calculated (at this moment)
  - Memory usage is calculated on process basis
- LSF
  - Process Information Manager (pim) calculates memory usage
    - Job memory usage on per process basis
    - Has facilities to calculate shared memory usage on a per job basis
    - IRIX has limited tools to determine memory usage
  - Sets current resident set size (RSS) limit in IRIX job container, this limit does not kill jobs
  - Does not kill jobs when memory limits are exceeded

Managing Memory resources (4/7)

- CPUsets
  - Managing memory resources using CPUsets is limited
  - Via CPUset tokens (MEMORY_MANDATORY + POLICY_KILL) jobs can be killed
    - Jobs are killed when far memory is accessed
    - Far memory is memory outside the defined CPUset
  - 1 Gigabyte/CPU ≠ 4 Gigabytes / 4 CPUs
    - C-brick has 4 CPUs and 4 gigabytes of memory

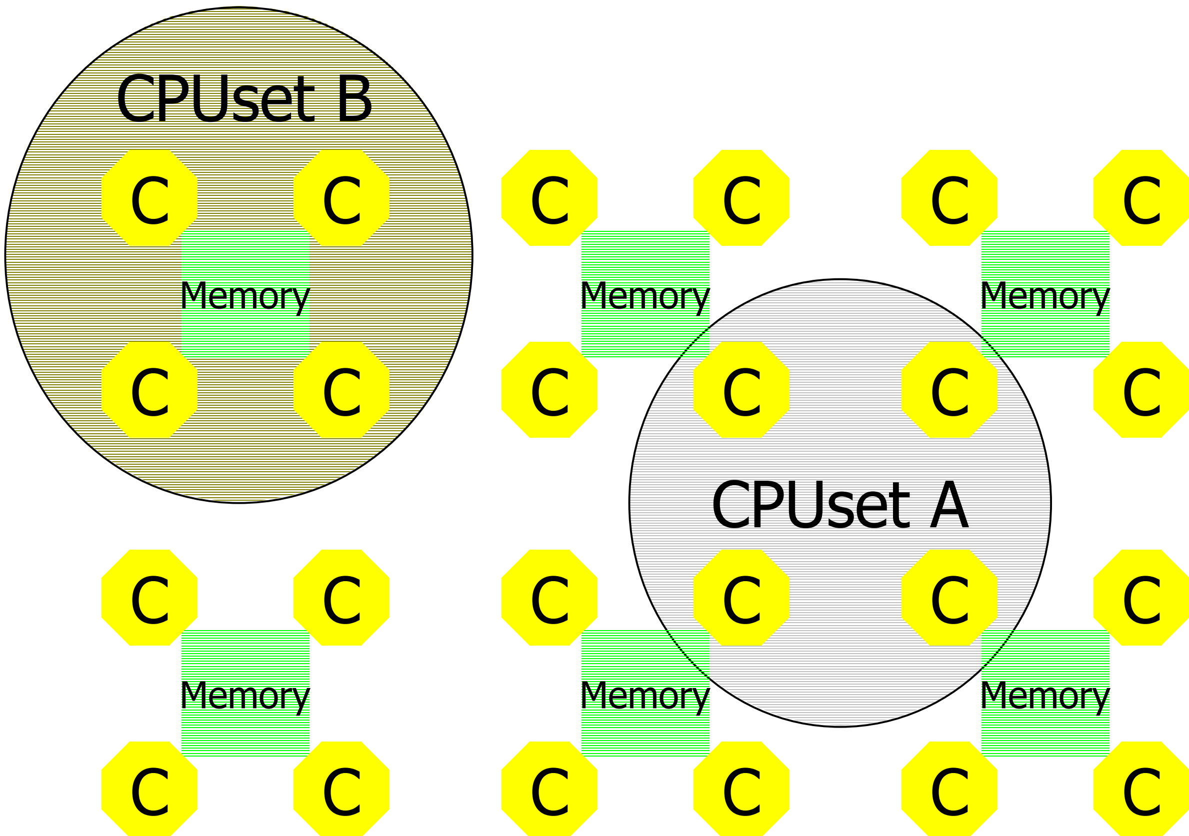
Managing Memory resources (5/7)

CPUset A:
1 CPU,
Max. 4GB memory



CPUset B:
3 CPUs,
Max. 4GB memory

Managing Memory resources (6/7)



CPUset B:
4 CPUs,
Max. 4GB memory

CPUset A:
4 CPUs,
Max. 16GB memory

Managing Memory resources (7/7)

- Joblimits
  - Because of the SGI MPI implementation virtual memory can not be limited
    - MPI allocates with mmap() per process-to-process communication channel a memory block (~1 Gigabyte per communication channel)
      - 4 processes ~ 20 Gigabytes virtual memory
      - 8 processes ~ 75 Gigabytes virtual memory
      - 32 processes ~ 1 Terabyte virtual memory
  - Interactive physical memory usage is limited

Managing I/O resources (1/2)

- I/O resources can be split into 2 separate parts
  - Hardware I/O channels
  - Kernel processes handling I/O requests
- IRIX uses dynamic algorithm for allocating and releasing system buffers (memory)
- System buffers are used for caching of file system data
  - To reduce physical reads by reuse of cached data
  - Optimize physical writes (delayed write)
    - Optimize data blocks to reduce head placements
    - Discard physical writes

Managing I/O resources (2/2)

  - IRIX has a maximum number of system buffers (kernel parameter nbuf $\leq$ 600,000), limited scalability
- I/O intensive jobs
  - Jobs high rate of read()'s and/or write()'s
  - Example: job with 8 processes with a high number of open files (~ 60 per process), data (~500 Megabyte per file) and on average ~50 Gigabyte of file system data
    - Job makes 99% use of cached data
    - Single job uses ~90% of the system buffer entries
    - Maximum of 2 jobs on a single host, uses ~100% of the system buffer entries
  - Large indirect memory usage of cached data (60-70% of memory on small nodes)

Conclusions

- Memory and I/O resources are not really manageable
  - SGI is developing a solution for calculating shared memory usage
  - The IRIX kernel is limited scalable for handling I/O processes
- CPUsets and joblimits are good developments for managing resources but the functionality and the integration within IRIX should be extended
- The integration of the current tools (LSF, CPUset, joblimits) is functional but has limited possibilities for managing system resources

Questions?

Mark van de Sanden
Senior Systems Programmer
sanden@sara.nl