Grid/MetaComputing Lecture-7

Spring 2003

Dr Graham Fagg

with support material from
Mark Baker and Michael Resch

Lecture 7 Overview

- Making it all work
  - Resource Management and Scheduling
    - From Cluster/Batch schedulers to MetaComputing
  - Scheduling MPI jobs
    - All we really need to do

Contents - Resource Management and Scheduling

- Clusters, MetaComputing and Super Computers
- Motivation for CMS
- Motivation for using Clusters
- Cycle Stealing
- Management Software
- Problems with Distributed Computing Resource Management
- Cluster Management Software Systems
- From Clusters to MetaComputing

Cluster, Super Computers and Metacomputing

- A main goal of distributed computing research is to provide users with simple and transparent access to a distributed set of heterogeneous computing resources.
- This is often called Metacomputing - the user submits jobs to a virtual distributed computer, rather than specifying particular computers.
- Super Computing on the other hand is designed to provide massive computational power to users. Performance is more important than usability.
- MetaComputing promises both.

Cluster and Metacomputing

- Metacomputing is still a research area - current implementations are limited, mostly applying to LANs rather than WANs.
  - Except for when SuperComputing Confferences are taking place!
  - Situation is getting better all the time.
- LAN implementations are Cluster Management Software (*CMS*) or Cluster Computing Environments (*CCE*).
- MetaComputing systems can be built by extending LAN systems, or by using many LAN systems together.

Cluster and Metacomputing

- Cluster Management Software
  - Managing clusters of (mostly) workstations as a distributed compute resource.
  - Built on top of existing OS.
- Cluster Computing Environments
  - Software to allow the cluster to be used as an applications environment, similar to Distributed Shared Memory (DSM) systems.
  - Built into OS kernel for improved performance.

Cluster and Metacomputing

- The World Wide Web is now so ubiquitous that it is becoming the platform of choice for distributed computing (Internet or Intranet) and Metacomputing.
  - See the Webflow project later.
  - Mostly used as the server to client binding structure.
    - I.e. submit a web form rather than a job request ticket.

Motivation for Resource Management

- Users want to be able to submit their jobs without having to worry about where they run - i.e. submit jobs to a metacomputer (virtual computer) rather than search for spare cycles on a real computer.
  - Ease of use. Requires both distributed code as well as data!
- Large organisations (companies, universities, national labs, etc.) typically have hundreds or thousands of powerful workstations for use by employees, which is a major under-utilised compute resource.
  - Check lecture 3 notes on Resource Management.
    - What is Spare Cycles and do we want to use them?

Motivation for using Clusters

- Surveys show utilisation of CPU cycles of desktop workstations is typically <10%.
- Performance of workstations and PCs is rapidly improving (my Laptop > 60 Mflops/s on Fortran 77 code).
- As performance grows, percent utilisation will decrease even further!
- Organisations are reluctant to buy large supercomputers, due to the large expense and short useful life span.

Usage

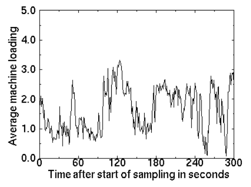


Figure: 4.14 Loading on a Meteorology Department machine sssweac2.

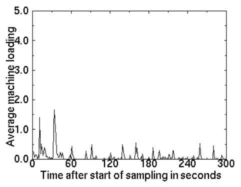


Figure: 4.15 Loading on a Psychology Department machine sssxcacl .

Usage depends on the class of the users..

As shown here; Meteorology verse Psychology.

Motivations for Clusters

- The communications bandwidth between workstations is increasing as new networking technologies and protocols are implemented in LANs and WANs.
- Workstation clusters are easier to integrate into existing networks than special parallel computers.
  - Install Linux from the local NFS copy… (look at NASA Beowulf)
  - MPPs require special HiPPi switches and interface hookups.
- Many MetaComputers will be made from Clusters although most of the larger research efforts prefer to integrate different MPPs rather than clusters.
  - I.e. more output for less effort from the MetaComputing System itself (I.e getting the Bell Award).

Motivation for using Clusters

- The development tools for workstations are more mature than the contrasting proprietary solutions for parallel computers - mainly due to the non-standard nature of many parallel systems.
- Workstation clusters are a cheap and readily available alternative to specialised High Performance Computing (HPC) platforms.
- Use of clusters of workstations as a distributed compute resource is very cost effective - incremental growth of system!!!
  - CPUs and disks are a lot cheaper, but some of the better InterConnection cards like Myranet, Gigabit etc are expensive.
  - Well almost… try upgrading all your systems to PIIIs after you just brought all the PIIs… with the wrong motherboards.
    - Do a few a week maybe?

Cycle Stealing

- Usually a workstation will be *owned* by an individual, group, department, or organisation - they are dedicated to the exclusive use by the *owners*.
- This brings problems when attempting to form a cluster of workstations for running distributed applications.
  - – Unless it is a dedicated cluster like the TORC cluster. (If it is managed correctly that is).
    - TORC runs too many different tests/configuration
      - – I.e. not a stable platform?

Cycle Stealing

- Typically, there are three types of owner, who use their workstations mostly for:
  1. Sending and receiving mail and preparing documents.
  2. Software development - edit, compile, debug and test cycle.
  3. Running compute-intensive applications.

Cycle Stealing

- Cluster computing aims to steal spare cycles from `(1)` and `(2)` to provide resources for `(3)`.
- However, this requires overcoming the *ownership hurdle* - people are very protective of *their* workstations.
- Usually requires an organizational mandate that computers are to be used in this way.

Cycle Stealing

- Stealing cycles outside standard work hours (e.g. overnight) is easy, stealing idle cycles during work hours without impacting interactive users (both CPU and memory) is much harder.

Management Software

- Software for managing clusters or metacomputers must handle many complex issues:
  - – Heterogeneous environments (computer and network hardware, software, OS, protocols, etc.).
  - – Resource Management.
    - CPUs, disk arrays, and sometimes long haul network connections.
  - – Job scheduling.
    - Handling multiple schedulers at the same time.
  - – Job allocation policy (prioritisation).
  - – Security and authentication.
  - – Cycle stealing from desktop computers without impacting interactive use by owner.

Management Software

- – Fault tolerance.
- – Support for batch and interactive jobs.
- – Should support all programming paradigms (sequential, data parallel, message passing, shared memory, threads, etc.).
- – Should support legacy applications.
- – User interface and job specification mechanism.

Problems with Distributed Computing

- Relatively immature technology.
- Few standards (particularly for Metacomputing).
- Lack of expertise of users, developers, and systems support staff.
- High cost of good commercial software.
- Relatively immature software development tools (debugger/profiler/etc).
- Applications difficult and time consuming to develop.

Problems with Distributed Computing

- Portability problems in heterogeneous environment.
  - Use Corba? But then which ORB are you using?
  - Use Java, maybe the native compilers will be available soon?
- Difficult to tune and optimise across all platforms.
  - On MPPs very true.. Sometime you have to use threads and sometimes just different compiler flags.
- Scheduling and load balancing.
- Trade off between performance hit in using remotely mounted files, and diskspace requirements for using local disk.
  - Local disks or local disk arrays (I.e. PARALLEL IO)
    - How do you get the data there and back? All 15 TerraBytes of it?
- How to handle legacy systems - rewrite, add interface, or use wrappers?

Problems with Distributed Computing

- Management becomes much more difficult when going from:

  LAN to WAN
  - Administration, access permissions and authentication issues are much harder across site boundaries.
  - Maintenance of management software is harder - administration...
  - Latency increased and bandwidth decreased.
  - Need wide-area file system access, temporary disk space.
    - Moving data again. What about IBP?

Problems with Distributed Computing

- One motivation for WAN-based systems is access to remote data repositories, which involves complex issues with storage and migration of large data sets.

Sequential to Parallel

- Latency and bandwidth more problematic on distributed computers.
- Load balancing is very difficult on multi-user systems, particularly for cycle-stealing of desktop systems (what happens if someone runs a job on a machine that is currently running a parallel program?).
- Fault tolerance and process migration are issues.

Existing Cluster Management Software

- Many research and commercial packages exist - almost all originate from academic research projects.
- Widespread use in large organisations, National labs., supercomputer centres.
- Mostly designed for Unix - some support for PCs via Linux, support for NT sparse...
- Not much use of Web as yet - some packages can use Web browser for alternate GUI.
- Targeted at LANs rather than WANs.

Cluster Computing - Research Projects

- Batch - UCSF, USA
- CCS (Computing Centre Software) - Paderborn, Germany
- Condor - Wisconsin State University, USA
- DJM (Distributed Job Manager) - Minnesota Supercomputing Center
- DQS (Distributed Queuing System) - Florida State University, USA
- EASY - Argonne National Lab, USA
- *far* - University of Liverpool, UK
- Generic NQS (Network Queuing System) - University of Sheffield, UK
- MDQS - ARL, USA
- NOW (Network of Workstations) - Berkeley, USA
- NIMROD - Monash University, Australia
- PBS (Portable Batch System) - NASA Ames and LLNL, USA
- PRM (Prospero Resource Manager) - University of S. California, USA
- QBATCH - Vita Services Ltd., USA

Cluster Computing - Commercial Software

- Codine (Computing in Distributed Network Environment) - GENIAS GmbH, Germany
- Connect:Queue - Sterling Corp., USA
- CS1/JP1 - Hitachi and Cummings Group, USA
- Load Balancer - Unison Software, USA
- LoadLeveler - IBM Corp., USA
- LSF (Load Sharing Facility) - Platform Computing, Canada
- NQE (Network Queuing Environment) - Craysoft Corp., USA

Using Cluster Management Software

1. Produce a job description file.
   - Text file produced manually or via a GUI.
   - Information such as program name, maximum runtime, desired platform, etc..
2. Submit the job.
   - Job is submitted to the metacomputer manager.
   - Job description file is passed to a master scheduler.

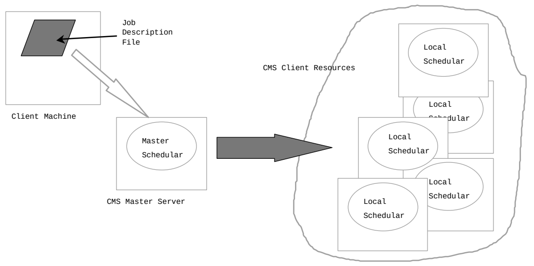
Using Cluster Management Software

3. Job scheduling.
   - Each computer in the cluster runs a resource daemon that communicates state to the master scheduler.
   - Scheduler checks system configuration, loads, queues, etc.
   - Multiple queues may be used for different types of jobs - sequential, parallel, batch, those needing fast turnaround, etc..
   - Scheduler matches job request to available resources while attempting to balance computational load.

Using Cluster Management Software

4. Job completion and fault tolerance.
   - Scheduler monitors each job to check that it finishes successfully - if not, it will reschedule it.
     - But not across different schedulers as of yet!
     - Can be configured to send you email when finished, but output will probably end up in file out.job.XXXX .
     - Make sure you get your large data file results out before your files are removed/purged.

Using Cluster Management Software



Desirable Features of CMS - I

- Resource configuration file for all computers - CPU type, performance, memory, times of availability (e.g. 5pm - 9am), authorised users, etc.
  - Look back at the GRM scheduler...
- Resource administration - control over access permissions to resources, authentication, accounting, configurable queue types, system loads, available disk space, etc.
- Resource load balancing is a key requirement.

Desirable Features of CMS - II

- Checkpointing (save state at regular intervals during run).
  - Not just the job, but the schedulers as well!
- Periodic checkpointing is useful (though costly) for fault tolerance of long jobs - job can be restarted at last checkpointed position.
- Process migration...
- Scalability, preferably across administrative boundaries to medium-area or wide-area networks.

Desirable Features of CMS - III

- Robustness - system can handle network problems and dynamic addition and removal of compute resources.
- No single points of failure - e.g. does system need to be restarted if master scheduler crashes?
- GUI (preferably from Web browser) as well as command-line interface.
- Configuration - easy to set up, maintain, and administer.

Desirable Features of CMS - IV

- Easy to use - simple job specification.
- User Information - user can query status of a job and get execution information.
- Minimise impact on workstation owner:
  - Change priority (*nice*).
  - Suspend a job or migrate to another machine (but this uses CPU/memory/disk/network resources).
- Security - provide at least standard Unix level of security and authentication.

Problems with Existing CMS

- LAN-based - not scalable.
- Limited platform and OS support (not truly heterogeneous), very little support for PC clusters, which are becoming a major compute resource.
- NT support will probably be available in future - MS Wolfpack, Codine, LSF...
- Do not support all programming paradigms - weak support for parallel computing.

Problems with Existing CMS

- Load balancing is generally rudimentary (e.g. based on `rup`) and not dynamic.
- Cycle stealing is not used effectively (ownership hurdle).
- Limited fault tolerance and checkpointing.
- Single points of failure - particularly master scheduler.

Problems with Existing CMS

- Current R&D work involves creating distributed schedulers with fault-tolerant distributed database to store system configuration information.
  - Globus uses the Matadata Directory Service (MDS) which is based on Netscapes LDAP server.
  - SNIPE uses RCDS.

Recommended CMS Packages

- See review at …
  `http://nhse.cs.rice.edu/NHSEreview/CMS/`
- Research packages
  - DQS is probably the most comprehensive, functional and commonly used research offering.
  - GNQS is robust and widely used (e.g. CERN), but does not support parallel jobs.
  - NOW is an impressive project that aims to address problem areas with current CMS packages.

Recommended CMS Packages

Commercial Packages

- These are generally more robust but can be very expensive:
  - Codine
  - LFS
  - LoadLeveler
  - NQE
  - Connect:Queue

Other Packages

Other Packages (mainly MPP ones)

- Maui Scheduler (NMSC)
- Portable Batch System (NASA)
- Parallel Operating Environment POE (IBM)

Scheduling Message Passing

(what Meta-Computing systems need to do)

CS-594

Dr Graham E. Fagg

Spring 2003

Overview

- Scheduling Message Passing
  (what Meta-Computing systems need to do)

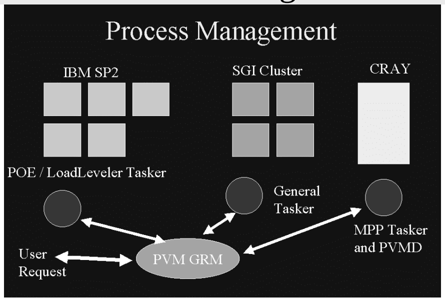
Closely coupled vs Cluster Computing

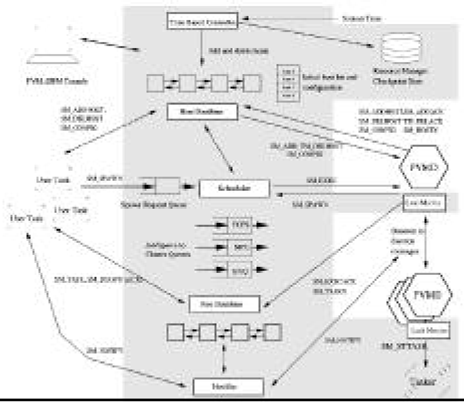
- Bottom line
  - MPI is better at message passing than PVM
  - More complex
  - Less flexible at anything else
    - I.e. it’s a message passing system not a distributed environment

Scheduling

- Not built into MPI as it has no process control
  - But maybe an option under MPIRUN
  - Yep PVM has it all
    - user controllable pvm_spawn()
    - pvm_rm interface
      - also
        - pvm_tasker interface
        - pvm_hoster interface

Scheduling





Scheduling

- Two types to worry about
  - At startup time
    - static allocation based on the environment
  - During run time
    - I.e. migration of tasks
      - system level migration
        - Special support needed (Condor)
      - User level
        - check points / restarts
    - Change work load allocated (bag of tasks)

Task allocation in PVM I.e. pvm_spawn()

- Before improving on it, had to figure out how it worked as it wasn't random but round-robin
- Aimed at using spare capacity
  - what spare capacity??

  - MPI_Spawn same problems

What is spare (what is even machine load?)

- Condor people claimed 10% utilisation for their systems
  - At Reading was more like 40-60% all the time.
- Load
  - machine average is not a good metric but without more specific help from the kernel it would have to do.
- Defined user classes and loading based on observations on the Reading University (UK) system over a year...

Typical loading

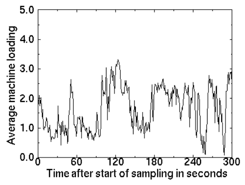


Figure: 4.14 Loading on a Meteorology Department machine sssvcsc2.

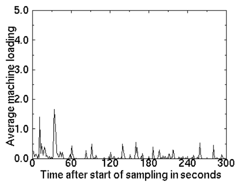


Figure: 4.15 Loading on a Psychology Department machine sssxcscl.

Better Spawn

- Added checks for load before starting.
  - Based on two methods, central RPC and distributed monitor daemons
  - Checked effects of this system on startup performance, and accuracy of placement.

Slaves	1 (or 2*)	70	Intercept	Slope
System	seconds	seconds	seconds	seconds/ slave
pvm_spawn	1.246	4.4	1.2	0.046
central server loadsvr_heavy	(*)[a] 14.884	32.6	14.4	0.261
distributed loadserver_d and mon_d	3.669	17.1	3.5	0.195

a. Even though the test was via a set of mean averaged results, the first point in this particular was completely out of character with the rest of the data and was discarded to allow for a more representative line fit.

Application Performance

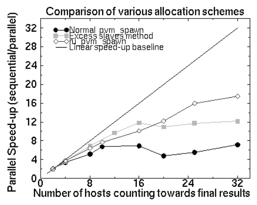
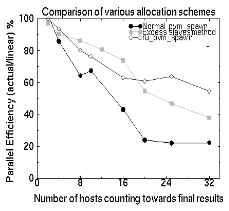
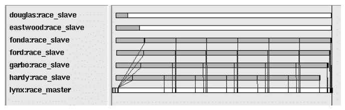
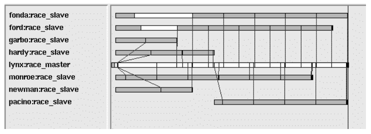


Figure: 4.21 Parallel speed-up of PNG Monte-Carlo simulations.



User level migration

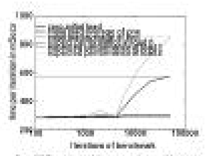




User level migration

Back to load or not

- The load was not always a good measure and soon the network was very heterogeneous…
- Could you use a benchmark to find the real load?



Modified spawn using effective speed-up

- Takes into account benchmark and loading
  - Helps on heterogeneous networks

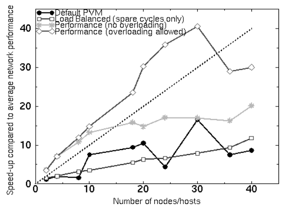


Figure: 4.31 Monte-Carlo speed-up for final version of ru_pvm_spawn.

Very advanced Schedulers

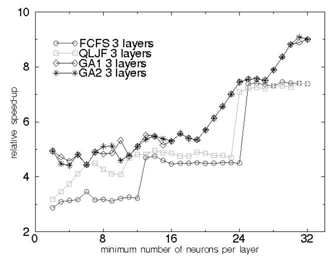


Figure: 2.39 Different scheduling algorithms compared.

Fault Tolerance

- Multiple methods
  - System level
    - from checkpoint file
  - User level
    - from data check point
  - How do we detect the failure?
    - Notify Message and/or time out

Meta-computing/GRID Projects

- Legion - University of Virginia
  - Object based
- Wide Area Metacomputer Manager (WAMM) -- Italy
  - Grow from experiments with PVM
- Wide Area Network Environment (WANE) -Florida State University
- WebFlow and MetaWeb - Syracuse University
- DISCWorld - University of Adelaide
- Globus - Argonne National Lab. And ISI
  - Tool kit
- SNIPE - UTK/ORNL
- HARNESS - UTK/ORNL/Emory
- Globe - Virje University in Amsterdam.
  - See http://www.cs.vu.nl/~steen/globe/