# Computing and Programming @SERC

Supercomputer Education & Research Centre
Indian Institute of Science
Bangalore

# Overview

- Introduction to Scientific Programming – Facets and typical development cycle.
- Computing Environment @ SERC.
  - PCs and Printing Facility.
  - Access and High-end Visualization Stations.
  - Computational Servers:
    - Serial servers.
    - Parallel servers.

# Introduction to Scientific Computing

# Different  Faces of Computing

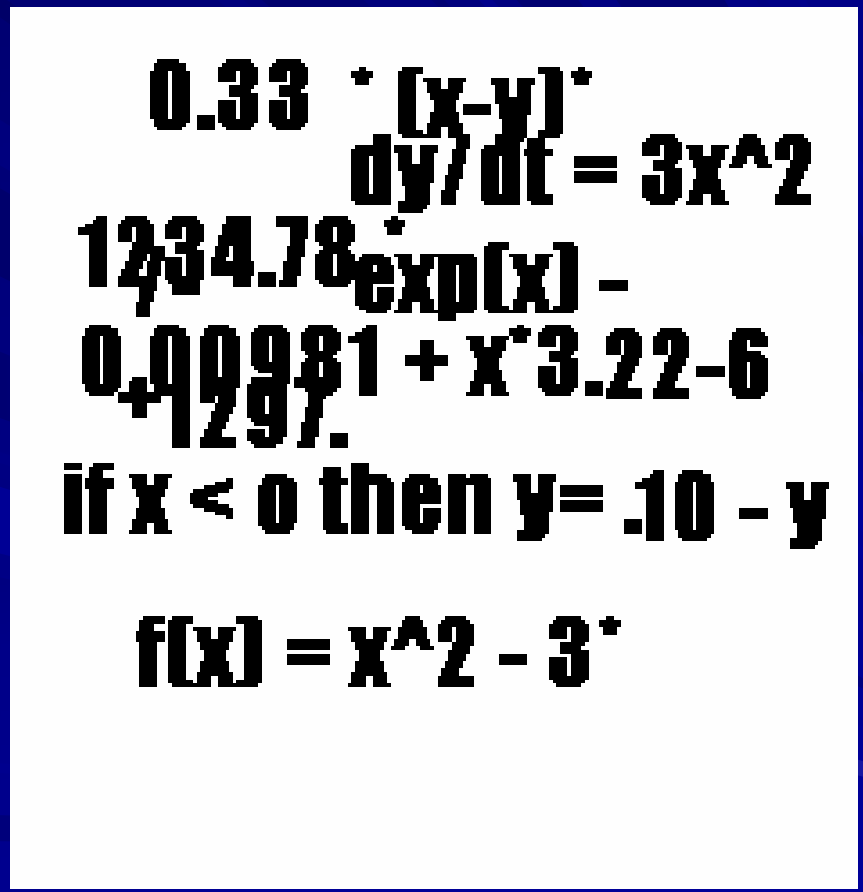
- Document Processing
- Graphics & Imaging
- Commercial & Database Applications
- Process Engineering
- Design
- Network Computing
- Scientific Computing (Number Crunching)

# Scientific Computing Features

- Few weeks to few months of coding for a single developer (most cases).
- Developer runs the code himself/herself.
- Codes are typically less fault-tolerant, less cosmetic.
- Aims to extract maximum throughput from the system (hardware & software).

# What does it involve ?

- Deals with numbers - integer, floating-point and complex as well.
- Involves complicated operations on a lot of variables and arrays.
- Often requires large resources (cpu, memory, disk) - to handle the complexity of problem.



# Development Process Cycle

- Developing the Algorithm.
- Developing the Code.
- Compiling and Debugging the code to make it error-free.
- Optimizing & Tuning the Code for efficiency.
- Production Runs & Result Analysis.

# Things to Know About

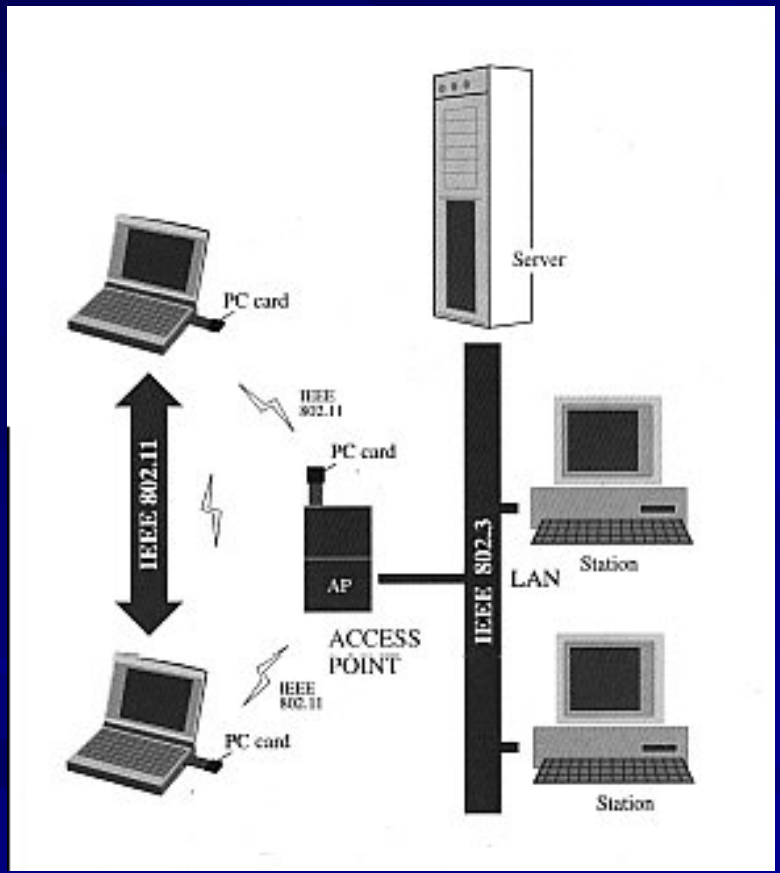
- The Science and the Algorithm.
- Basic Numerical Analysis.
- Any High-level Language (Fortran, C, C++, Java)
- Packages and Libraries.
- Programming Environment : Compilers, Debuggers, Profilers, Development Tools..
- Optimization Techniques.

# Pre-requisite : UNIX

- Most systems at SERC run Unix.
- Need a working knowledge on :
  - File and Directory manipulation commands. (***ls, cp, mv, rm, cd, grep, ..***)
  - Common Desktop Environment (CDE).
  - Text editor (preferably **vi** or **emacs**) to write your codes.
- Advanced Unix concepts (eventually).

# Computing Environment @ SERC

# Compute Systems @ SERC




- PC’s
- Unix Access Stations
- Servers
- Clusters
- Supercomputer

# Where to Start ?

- Use the PCs only if you need packages that are available in MS-Windows only.
- Otherwise, use an Unix Access Station. Build and test-run your codes here.
- For Production Runs go to a Server.
  - If your code is sequential, choose a serial computational server depending on how much cpu-time and/or runtime memory size your job needs and the necessary software you may need for your execution.
  - If your code is a parallel code, choose a suitable parallel machine.

# Unix Access Stations @ SERC

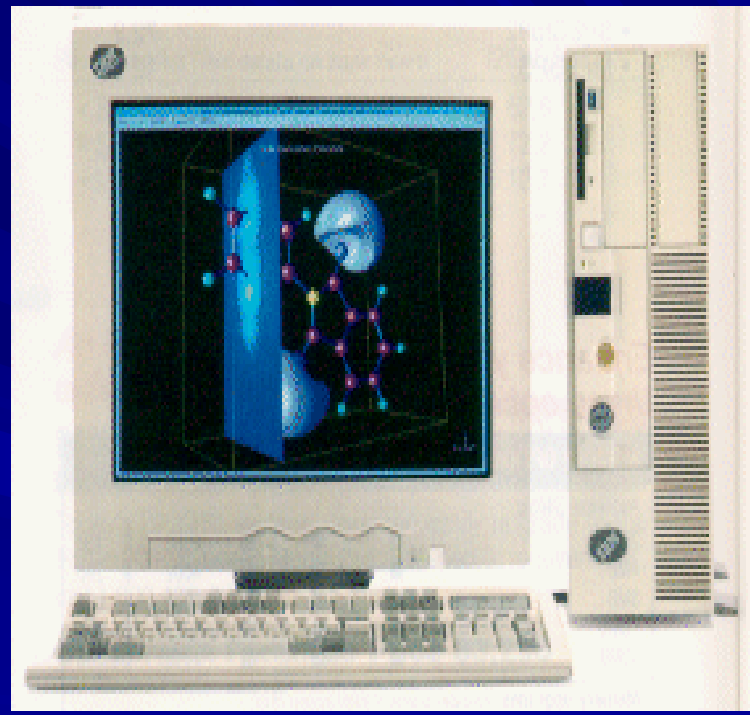
(when you apply for SERC account these are the machines that you get access to)

- IBM Pentium based Linux Workstations.
- SUN Ultra 5, Ultra 10, Ultra 60
- SGI O2, Octane, Fuel, Tezro
- HP C200
- Compaq XP1000

# Using the Access Stations

- For code development
- Running small jobs (< 30 mins) in background.

- Do not lock display
- Migrate to a Server for production runs.



# Development Environment

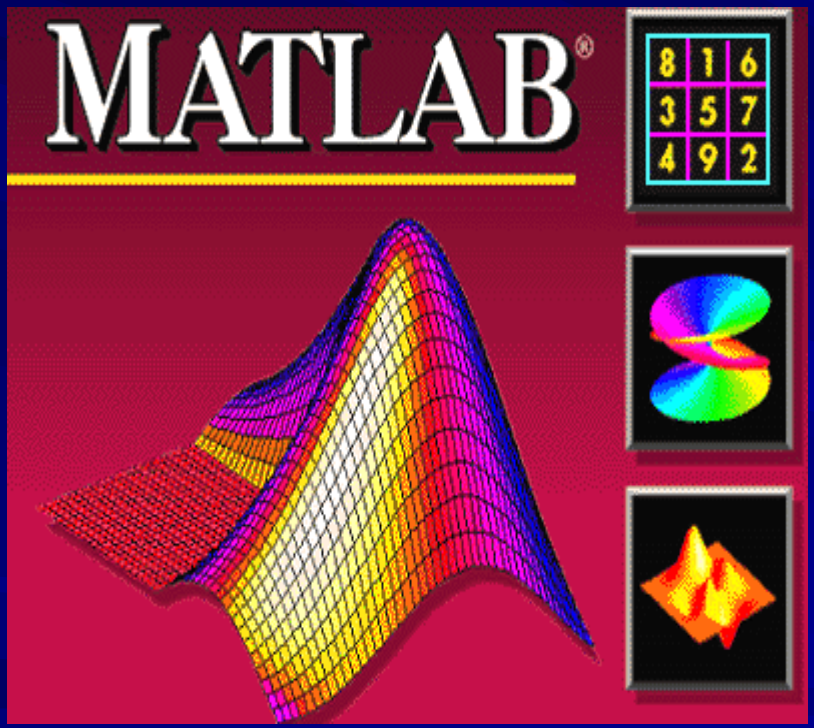
- Numerical Packages - MATLAB, MAPLE, MATHEMATICA
- Software Packages –Accelrys, MOE, Schrodinger, ADF, Gaussian, etc.
- Fortran/C/C++ Development Environment:
  - Compilers, Debuggers & Profilers
    or
  - Integrated Environment (SoftBench, Workshop VisualAge, DECFuse)
- Numerical Libraries: Vendor specific math libraries DXML, MLIB, etc.

# Numerical Packages

- Interactive packages can do a variety of numerical & symbolic manipulation jobs.
- Easy to use with GUI, on-line help and tutorials.

# MATLAB

http://www.mathworks.com/

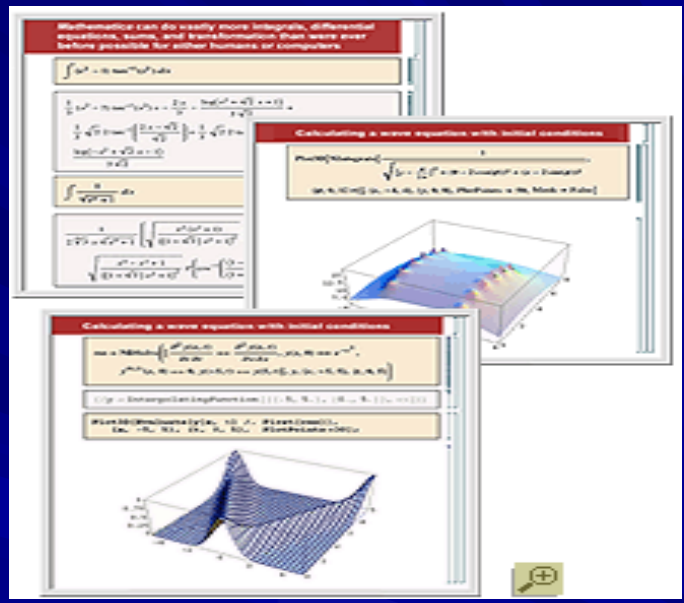



- Available on almost all Unix systems in SERC.
- Useful for matrix manipulations, solving equations, differential eqns, 2D and 3D data visualization and ...
- You can install on your machine in your lab.

# MATHEMATICA

http://www.wolfram.com/

- Strong in Symbolic Manipulation.
- Can also do Numeric Computation and Visualization.
- Is also available on most systems in SERC.
- You can also install in your lab machine.



***Mathematica Seminar @ SERC***

***12 September 2001***

# Numerical Libraries

- Canned routines for standard tasks (matrix digitalization, FFT …)
- Tuned for specific hardware to extract maximum speed.
- Using the libraries saves coding time and often improves performance also.

**Libraries @ SERC**

- ESSL (on IBM)
- SUNMATH (on SUN)
- DXML (on Compaq)
- IMSL (on IBM 59x)
- MLIB (on HP)
- PESSL (on IBM SP)
- SCSL (on SGI)
- Free stuff such as Scalapack, Lapack, BLAS (on IBM SP)

# Software Packages

- Tripos, Accelrys-Insight/Cerius – Molecular Modelling and analysis
- Schrodinger – Drug Design
- Gaussian – Electronic Structure Programs.
- MOE - Bioinformatics, Cheminformatics and Molecular Modeling

Lookup SERC homepage under Computing Facilities/Software or contact helpdesk@serc for more information.

# Building an Application

# The Compiler

- Compiler translates your program to a machine-understandable code.
- Fortran/C/C++/Java compilers are available on all Unix systems at SERC.
- Example :

```
f77 my_prog.f -o my_prog
```

creates an executable called `my_prog` from `my_prog.f`

# Compiler Flags (options)

For specific needs :

- `-O, -O2, -O3` for optimization.
- `-c` for creating object files.
- `-g` for debugging.
- `-l` for including a library.

# Run-time Errors

- A perfectly compiled code may exit while running saying ***core dumped***
- Such run-time errors may be caused by various reasons, such as :
  - Divide by zero
  - Accessing an array element beyond its declared dimension
  - Number overflow or underflow

# Debuggers

- Debuggers will be the saviour if you get run-time errors on a complex code.
- Compile with `-g` flag.
- Run the executable through debugger:

```
dbx exe_filename
```

- This will bring you to `(dbx)` prompt
- Check system-specific manuals.

# Integrated Development Environments.

- IDEs are integrated environments that assist code development.
- One single GUI for making, compiling, executing, and debugging your codes.
- Enables faster code development cycles.
- Variety of IDEs available – like Fuse, Workshop, Softbench and VisualAge.

# Timing your Code

- Use the *timex* command to know how much CPU time your code takes.

```
% timex a.out
   real 10.06
   user 3.01
   sys  0.03
```

**The *time* command in the *cshell* gives a more detailed output.**

# Profilers

- The profilers will give you detailed time statistics of your code.
- Compile with `-p` and run.
- Post-Mortem profile with:

```
prof  > prof.dat
```

- Complete profiler information is stored in `prof.dat`

# Alternative to Profilers

- Instead of profilers, you can also manually time code segments.
- E.g. in IBM systems :

```
integer t1,t2,time_in_sec
t1 = mclock()
call subprog(x,y,z)
t2 = mclock()
time_in_sec = (t2-t1)/100
```

# How big is the code?

- The size of the code is a very important parameter. A code which fits into the memory runs faster as it avoids paging.
- Too large a code also might result in insufficient memory errors.
- *size* command gives the executable program size.

```
% size a.out
```

# What to do in case of size problem?

- Check for user limits using *ulimit* command. If not sufficient increase them to corresponding maximum.
- Try running the code on a system with larger memory.
- If everything fails you might have to reduce problem size.

# Thriving for Speed

- Need to improve the performance of your code if you are going to run it again and again.
- This can be achieved in two ways :
  - Through compiler  flags for optimization
  - By modifying your code appropriately
- Modify code to  save resources (CPU time, memory, disk space …)

# Optimization through Compiler

- Use `-O[n]` compiler flag for optimization; `n` signifies optimization level (different in each architecture).
- Use architecture-specific tuning flags.

  E.g. for IBM 59x Servers

  `-qarch=pwr2 -qtune=pwr`
- High levels of optimization may sometime change the order of expression evaluation, and cause wrong results.
- Check the Compiler man pages for details on the optimization flags.

# Optimization by Code Tuning

A few General Rules :

- Break up complex expressions into simpler sub-expressions.
- Pull out constant operations from a loop.
- Access array elements in proper order.
- Maintain Locality of Reference.
- Cut down redundant arrays.

# Accessing Array Elements

- Fortran stores arrays in column-wise, C stores in row-wise format.
- Your access pattern should conform to this.
- Access consecutive elements of an array within a close proximity in your code.

# Access : Fortran Example

Wrong Access :

```
DO I = 1, N
  DO J = 1,N
    A(I,J)=2*A(I,J)
  ENDDO
ENDDO
```

Right Access :

```
DO J = 1, N
  DO I = 1,N
    A(I,J)=2*A(I,J)
  ENDDO
ENDDO
```

**For a 4000 x 4000 array, the Right access pattern improves the speed by a factor of 40 !!!**

# More Help on Optimization

- *Optimization and Tuning Guide* -- IBM Manual
- Man pages of compilers
- *High Performance Computing*, by Kevin Dowd (O'Reilly & Assoc.)

# Running an Application

# How to use Servers ?

(Some require special access.)

- About 6 Compaq Servers, few IBM Servers, 2 SGI Tezro visualization servers and Serial Queues of IBM SP3 and SGI Altix 350 in SERC.
- Servers run multiple background jobs.
- Difficult to figure out which server is least loaded at a given point of time.
- Do not submit jobs on servers manually.
- Use LoadSharingFacility(LSF) or LoadLeveler or PortableBatchSystem(PBS) utility to submit jobs on servers.

# The Batch Scheduling utility.

- Batch Scheduling Utilities are used on SERC compute servers to distribute job load evenly to achieve effective utilisation of the resources in a distributed environment.
- LoadLeveler, LoadSharingFacility and Portable Batch system are the three such utilities available at SERC.
- They are available on IBM, Compaq, Sun and SGI Access Stations and Servers.
- Predefined job classes (queues) are available depending on requested resource (cpu time, memory ..)

# Load Sharing Facility (LSF)

- LSF is a batch processing tool.
- Available on Ultra 60, XP-1000 & and ES40 servers.

Common LSF commands :

- ***bsub*** : submits a job
- ***bkill*** : cancels a job
- ***bjobs*** : lists job status
- ***bqueues*** : lists available queues
- ***bhosts*** : lists available hosts
- ***xbsub*** : GUI to do all the above

# LoadLeveler

- LoadLeveler is yet another batch scheduling utility.
- Include */home/loadl/bin* in your search path (for IBM systems)
- Create a file (say, *myjob*) with the following lines :

```
#@ executable = a.out
#@ error = err.log
#@ output = out.log
#@ class = q120s
#@ queue
```

- Submit as : *llsubmit myjob*

# More LoadLeveler Commands

- `llq` lists job status
- `llcancel` cancels a job
- `llclass` lists available queues
- `llstatus` shows status of all machines
- `xloadl` A graphical interface (GUI) to do all the above

# Portable Batch System

- The portable batch system(PBS) is one more batch scheduling facility.
- It is available on the SGI-Altix and Tezro machines.
- HP Woodcrust

- Include ***/usr/pbs/bin*** in your search path (for Altix system)
- Create a file (say, ***myjob***) with the following lines :

```
#!/bin/sh
#PBS -l ncpus=4
#PBS -l pcput=00:30:00
#PBS -o
  /home/phd/secaj/job.out
./job1
```

- Submit as : ***qsub myjob***

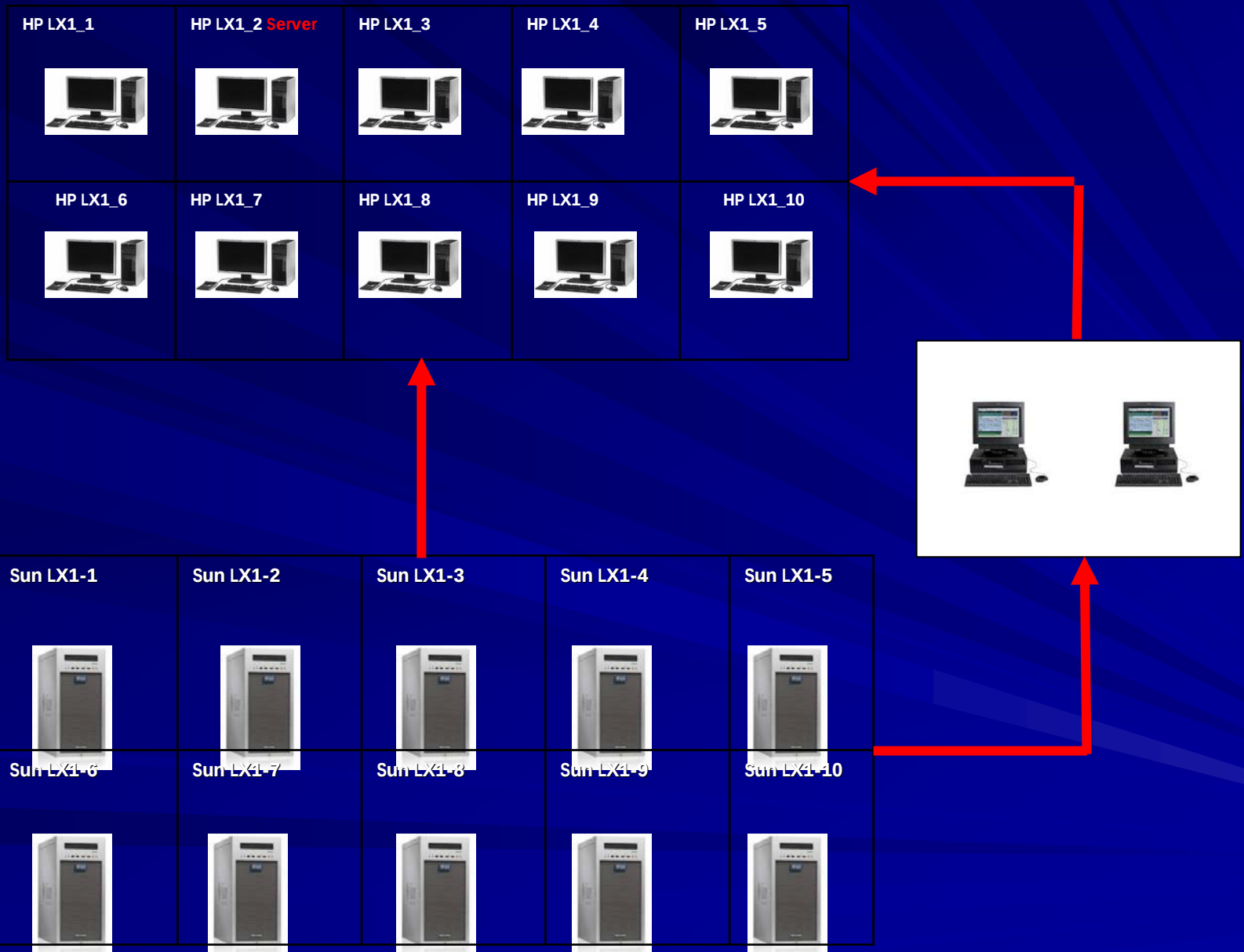
# What is a cluster

- **Cluster is a group of two or more systems that work together. It generally refers to the multiple servers that are linked together in order to handle variable workloads or to provide continued Operation in the event of one fails. Each computer may be a multiprocessor itself.**

# SERC Cluster

- We have a Cluster with PBSPro-8.0 (Portable Batch System) Here, the cluster of 10 woodcrest H.P W/S, each two CPUs would provide a 20 CPUs + each CPU from 10 SUNULTRA W/S would provide a total of 30 CPUs and two IBM Intellistation E-pro are only submission hosts.

- The cluster of HP Woodcrest w/s housed in cpu room, where PBSPro-8.0 is installed where hplx1_2 is the PBSPro-server and other 09 hplx machines, 10 Sunlx machines and 2 ibmintelli stations are clients. Sunlx and ibmintelli stations are housed in first floor of SERC. Sunlx systems can be used for both interactive and batch Job processing and ibmlx3_2 & ibm2_4 are the two submission hosts, reserved for submitting jobs remotely.




1 CPU of each Sunlx System used as client in the cluster

## Hardware configuration:

HPXW6400 DualCore Xeon 5130

- Front Side Bus : 1333 Mhz
- 160 GB hard disk
- 4 GB DDR2 667 Mhz  ECC Memory

## SUN ULTRA-20 workstations

### Hardware configuration:

- Dual Core opteron 4GHz CPU
- Ultra 20 1 GB ECC Kt (Memory)
- 250 GB 7.2K RPM SATA HDD

## IBM intellistation E-pro

### Hardware configuration:

- Pentium IV 2.4 Ghz
- Standard memory : 256 MB DDR  SDRAM (ECC)
- Hard Disk : Single 36 GB Ultra160 SCSI non hot swap HDD (10k rpm

# Queues to submit jobs on HP Cluster

- **Qh8** for 8 hours
- **Qh16** for 16 hours
- **Qh32** for 32 hours
- **Qh64** for 64 hours
- **Qh256** for 256 hours

- **Users can submit their serial jobs to any queue according to their requirement, but each user can submit only two jobs to run on first 4 queues but in the fifth queue only one job will be running. even if the user submits more than two, only two jobs will be running other jobs will be in queue.**

# How to submit a job on the Cluster

- %**qsub** job-name

  (where job-name is the shellscript)

- %**qsub –q** quename job-name

  (where queuename can be Qh8, Qh16,Qh32, Qh256 mentioned depending on the time requirement )

- %**qsub –q @server** job-name

  (where server is pbs-server)

# Other Useful Commands:

- %**qstat –f** job-number
  (to get the details of the job)

- %**qstat –u** username
  ( to list user-specific jobs)

- %**qstat –n**
  ( to know the node assigned to job)

- %**qstat –s**
  (displays the job comments, in addition to the other information)

## System Administrator:

- Mr.T.A.Chandrappa @Room #218

Email: chandru@serc.iisc.ernet.in

## Helpdesk@SERC:

- In general, for any queries/problems regarding SERC-computing facility, you can contact @ 444 in serc or from other departments @22932737 - 444 or email: helpdesk@serc.iisc.ernet.in

# Supercomputing @ SERC

# Supercomputers
## *(Parallel Machines)*

- For a large problem, the capability of a single CPU becomes insufficient.
- Such problems could be handled by harnessing the power of many CPU's.
- Supercomputer has many CPU's across which an application is distributed.
- Different parts of the code run on different CPU's simultaneously.

# Classes of Supercomputers

**Shared Memory (SMP)**

Many CPU's share a single bank of memory.

*Systems @ SERC*

- IBM SP(3) High Node and Regatta.
- SGI Altix

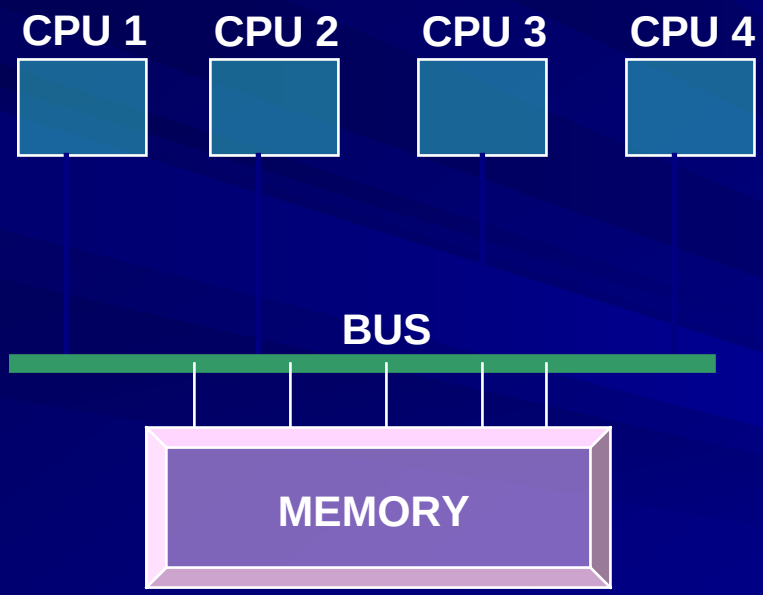
**Distributed Memory (DMP)**

Each CPU has it's own memory, and accesses other cpu's memory over network.

*Systems @ SERC*

- IBM SP(2)

# Shared Memory Paradigm




- Single memory, single address space.
- Easy to program.
- Auto-parallelizers render serial code to parallel.
- Program performance does not scale beyond 8-16 CPU's (most cases).
- Still suitable for medium-sized problems.

# IBM RS/6000 SP(3) High Node

- 16-CPU node
- Power3-II 375 MHz CPU
- 32 GB Main Memory
- AIX Parallelizing Compilers.
- Configured to accept batch jobs only.

# IBM RS/6000 SP(3) Thin Node

- 4-CPU node
- Power3-II 375 MHz CPU
- 8 GB Main Memory
- AIX Parallelizing Compilers
- For test runs & development of codes to be run on the high node.

# IBM RS/6000 SP(3) : Shared Memory Parallel Libraries

- **SP(3) has both serial and parallel versions of scientific and linear algebra libraries ESSL and PESSL available.**
- **To use the any particular library correct path has to be specified while linking.**
- **For example if shared memory parallelized version of ESSL is to be used add *–lessl* while linking (or compiling)**

# IBM RS/6000 SP(3) : Distributed Memory Parallelization

- **SP(3) can also be used in distributed memory mode with each processor having 2GB memory of its own.**
- **For the program to be run in the distributed memory mode user must add specific calls to message passing library like MPI.**
- **Details of MPI subroutines and how to use them is available with documentation for SP.**
- **User has to use –lmpi option while linking the program which has mpi subroutine calls.**
- **Users can take help of POE when running the distributed memory parallel code.**

# IBM RS/6000 SP(3) : Serial Code Optimizations

- **It is power3 architecture computer so the optimization option to be used are**

  **-qarch=pwr3 and –qtune=pwr3**
- **All other optimizations are like IBM-590 servers.**
- **Options like –bmaxdata and –bmaxstack might have to be specified for large sized codes.**

# IBM Regatta P690

- 3 machines, 2 with 32-CPUs 256 Gb RAM and 1 with 16 CPU-128 Gb RAM.
- Power4+ processors @1.9MHz
- AIX parallelising compilers.
- Support for OpenMP and MPI directives.
- Parallel ESSL.

# IBM P690 – Shared Memory Parallel Libraries.

- **P690 has both serial and parallel versions of scientific and linear algebra libraries ESSL and PESSL available.**
- **To use the any particular library use –lessl while linking (or compiling).**
- For parallelising use compiler xlf_r with directives -qsmp=auto for auto-parallelising and -qsmp=noauto for using openMP directives.

# IBM P690 – Distributed Memory Parallelisation.

- Same as for IBM SP(3).

# IBM P690 – Serial Code Optimisations.

- **It is power4 architecture computer so the optimization option to be used are**

  **-qarch=pwr4 and –qtune=pwr4**
- **All other optimizations are like IBM-590 servers.**
- **Options like –bmaxdata and –bmaxstack might have to be specified for large sized codes.**

# Tools on IBM P690

- IDEBUG is a general purpose debugger that supports remote heterogeneous debugging of distributed applications.
- Common GUI that can connect to distributed debug engines on a variety of platforms.
- Lookup manuals for more information.

# SGI Altix 3700

- 32 CPUs with 256 Gb RAM.
- Intel Itanium 1300 MHz Processors.
- Intel parallelising compilers.
- Parallel SCSL library.
- Support for OpenMP and MPI directives.

# SGI Altix – Shared Memory Parallelisation

- **Altix has both serial and parallel versions of scientific and math library SCSL available.**
- **The SCSL routines can be loaded by using -lscs option or the -lscs_mp option.**
- **For example if serial version of SCSL is to be used add**

  ***–lscs*** **while linking (or compiling)**
- **For parallelising your codes use explicit multi-threading constructs or OpenMP directives or use auto-parallelising compiler directives.**

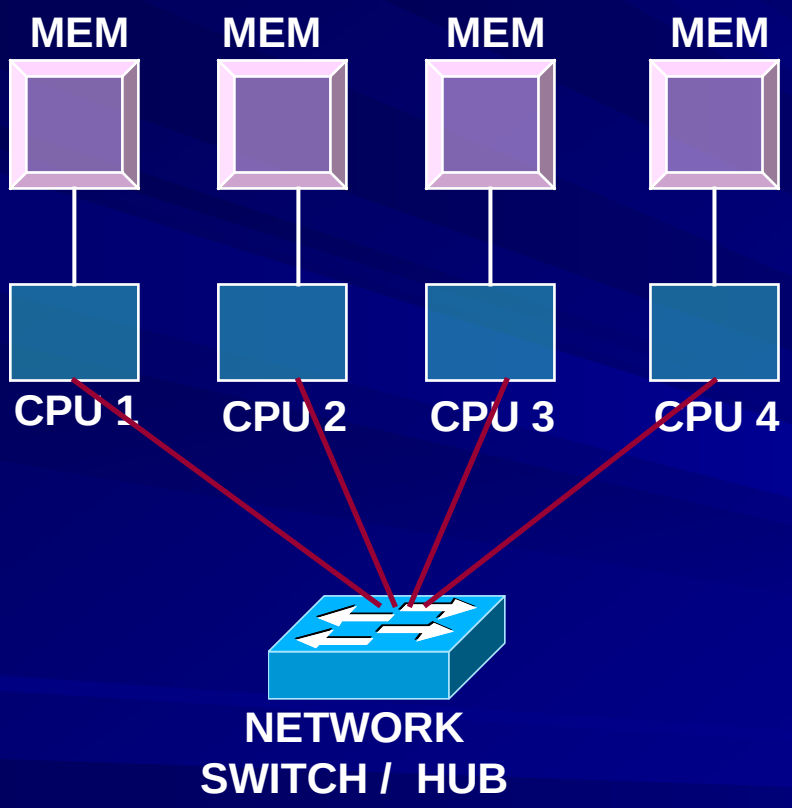
# SGI Altix – Distributed Memory Parallelisation.

- **Altix can also be used in distributed memory mode with each processor having 8GB memory of its own.**
- **For the program to be run in the distributed memory mode user must add specific calls of message passing library like MPI into his code.**
- **Details of MPI subroutines and how to use them is available with documentation for Altix.**
- **User has to use –lmpi option while linking the program which has mpi subroutine calls.**

# SGI Altix – Tools to aid parallel code development.

- ParaWise, the Computer Aided Parallelisation Toolkit.Takes a serial code as input, analyses and generates either a parallel code containing Message Passing library calls or OpenMP Shared Memory directives.
- Vampir and Vampirtrace - An interactive visualization tool designed to analyse and debug parallel programs, in particular message-passing programs using the MPI.

# Distributed Memory Paradigm




- New programming technique : need to use message-passing calls (MPI) in Fortran/C code to access remote memory
- Scalable : No bottleneck on memory, performance improves by adding more and more CPU’s
- Only choice for really large-scale problems

# IBM p720 Cluster

- 64 nodes, each node having 4 Power5 processors@1.65GHz., 4GB memory, and 73.4GB * 2 numbers hard disk.
- Nodes interconnected by high-speed gigabit Nortel Switches.

# IBM p720 Cluster.

- Operating System: Suse linux enterprise 9.0
- ESSL Mathematical Libraries ver.4.2.1
- Compilers: fortran compiler (xlf) 9.1.0 c compiler (xlc) 7.0
- Batch submission system: LoadLeveler 3.2.1.0 4)
- Parallel operating environment: mpich-1.2.6-1 6)

# Troubleshooting

# Problems : Search Path

Q. How do I add a directory in my search path?

A. For C-shell users :

```
set path = ($path /directory)
```

For Bourne- and K-shell users :

```
export PATH=$PATH:/directory
```

# Problems : Disk Space

Q. I need more disk space than what was allotted. What to do ?

A. Try the following :

- See if you can cut down some I/O statements and save space.
- Use the `/tmp` directory.
- Transfer some unused files to tapes or cdroms to reclaim space. Try deleting unwanted files.
- If all these fail, apply for disk quota increase (form available in Room 103)

# Usage of /tmp Space

- Each system has its own */tmp* directory.
- Always create a directory of your own and work within this directory.
- */tmp* is purged regularly. Anything more than 7 days old can get deleted without notice.
- Use */tmp* for data files only. Do not store your software repositories, installations and thesis files, they may get deleted without notice.

# Problems : Memory

Q. My problem requires large memory.

- Find out how much memory (bytes) are needed by : `size exe_filename`
- Try to reduce memory by cutting some unnecessary arrays. Do not over-dimension arrays.
- Use the servers with larger memory.
- Migrate to a parallel systems.

# Problems: CPU time.

- I need to run a job that executes for several days?
- Analyze why this is so? Can the job be parallelised?
- As a first step attempt shared memory machines to auto-parallelise and see if it gives a performance boost. Fine tune by hand parallelization by using OpenMP or MPI libraries.
- Precaution: Any job that can execute beyond 2 hours of CPU-time should be written such that it can restart from the point of failure.

# Sources of Help

- Man pages on the systems.
- On-line Hypertext (through help icon in CDE) in the machines.
- Hardcopy manuals in SERC library.
- Through SERC web pages.
- Many other tutorials and e-books generally available on Net.

# Information on the Web

- WWW is the  best way to keep you updated on the latest.
- Check out SERC web site :

  http://www.serc.iisc.ernet.in/

  (click on ***Computing Facilities*** )
- Many of the lecture notes in this series will also be made available under ***User Information*** in SERC web pages.

**SERC Wishes you a Happy Computing Experience over the years to Come!!**

**SERC Wishes you a Happy Computing Experience over the years to Come!!**