MONASH University
Information Technology

FIT3142 Distributed Computing

Topic 4: Clusters, Cluster Performance, MPI

Dr Carlo Kopp
Dr Asad I. Khan
Faculty of Information Technology
Monash University

Why Study Clusters, Performance and MPI?

- Clusters are the most common "building block" used to form distributed systems, therefore must be understood;
- Foundation knowledge: Because the performance of clusters determines many critical aspects of distributed system performance, understanding clusters is essential;
- Foundation knowledge: The limitations of a cluster can limit what a distributed application can or cannot do;
- Practical skills: When coding distributed applications you will have to run the code on a cluster, or a cluster in a grid/cloud;
- Practical skills: You may have to benchmark an application on a cluster, so understanding cluster performance matters;
- Practical skills: MPI is one of the most commonly used APIs for distributed parallel programming, and was developed initially for use on clusters;

Clusters

- Clusters emerged during the 1990s as an alternative to traditional "supercomputers", which were usually optimised for vector processing, and architected to exploit *Instruction Level Parallelism*;
- A "cluster" is a term applied to a group of general purpose processors, connected by a high speed "fabric" of links, that are running software to execute usually large parallel processing jobs;
- Parallelism in clusters occurs at the level of the process, unlike traditional supercomputers;
- In principle, the limits to the number of cores in a cluster are determined by the performance limits of the "fabric" interconnecting the machines forming the cluster;
- Most contemporary supercomputers are built as clusters.

Clusters vs. Grids vs. Clouds (I)

- **The term "Cloud" is now very widely employed to describe cloud computing environments, but also any large distributed computing system, even if it is not running genuine "Cloud" middleware and runtime environment!**
- **Clusters are typically confined to one computer room, and run middleware and programming environments optimised for parallel jobs, especially "supercomputing" tasks;**
- **A "Grid" is usually formed by aggregating multiple clusters over a Wide Area Network (WAN), to increase aggregate performance, using "Grid Middleware";**
- ***A "Cloud" like a "Grid" may aggregate vast numbers of cores over multiple sites, but "Cloud middleware" is usually built to support a disparate mix of different users, and provide "elastic" allocation of computing resources.***

Clusters vs. Grids vs. Clouds (II)

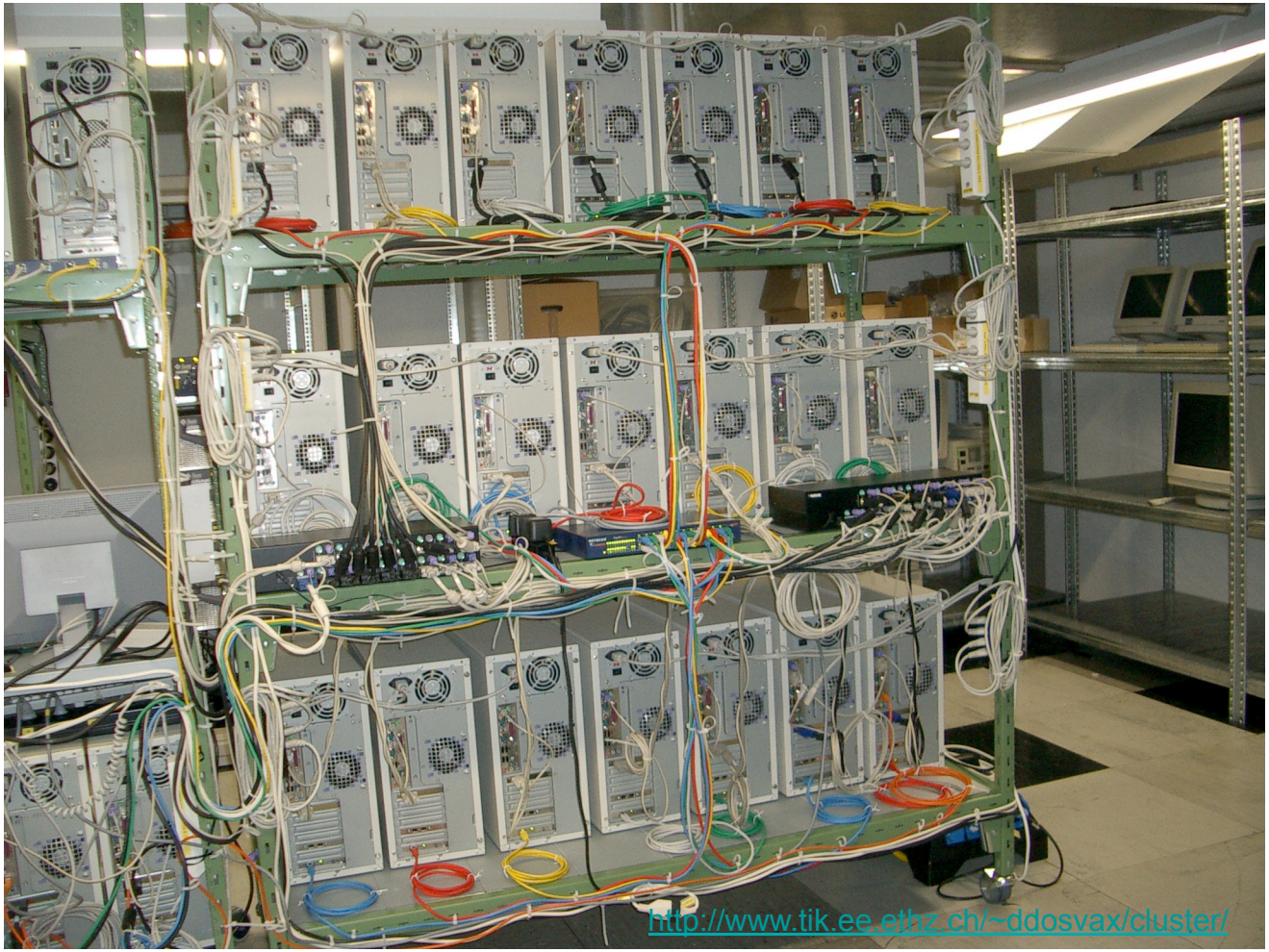
- A cluster or multiple clusters may be running cluster middleware, or Grid middleware, or Cloud middleware;
- In some instances, such middleware may be run concurrently, such as cluster that is used for local jobs but also participates in a Grid;
- An ongoing problem with distributed computing is the imprecision of language and labels used to describe systems, especially in industry;
- The best indication of what category a distributed system falls under is the type of middleware and programming environment being run;
- In current usage, the term “cluster” is often only used to describe the hardware, reflecting the fact that middleware may support various models;

Clusters - Integration

- The simplest way to form a cluster is to interconnect a large number of racked general purpose processors, using a high speed network;
- The principal challenge is in providing a way of managing jobs, distributing the computing load across the cores, and providing seamless IPC between processes in jobs;
- These tasks are typically performed by "middleware", and in many instances, "runtime environment" software for managing jobs, down to individual processes;
- In constructing clusters, which are the basic building block in Grids and clouds, the performance of the interconnecting fabric is critically important;
- Two parameters are of interest, these are *latency* and *bandwidth* in interconnecting processes running on cores.

HARDWARE AND FABRICS

TIK Experimental Cluster “Scylla” – ETH Zurich, Switzerland



http://www.tik.ee.ethz.ch/~ddosvax/cluster/

Ethernet Switches
22 x Athlon
Commodity PCs
Debian Linux

Juno Linux Cluster - Lawrence Livermore National Laboratory, USA

Cluster Design

- Earliest clusters formed by stacking PCs or Unix workstations on benches, and later 19 inch racks, and interconnecting them with 10 Mbits/s 802.3 Ethernet (e.g. Monash PPME);
- Interconnect performance was soon found to be important for computing tasks performing a lot of IPC, or demanding a lot of bandwidth;
- The general trend since then has been to use commodity Ethernet and Ethernet switches for small / cheap / low performance clusters, and much more expensive and elaborate interconnects for large / expensive / high performance clusters;
- Clusters are usually used as building blocks in larger data centres – hardware configuration reflecting performance demands of the application;

“Economy” Cluster Interconnects

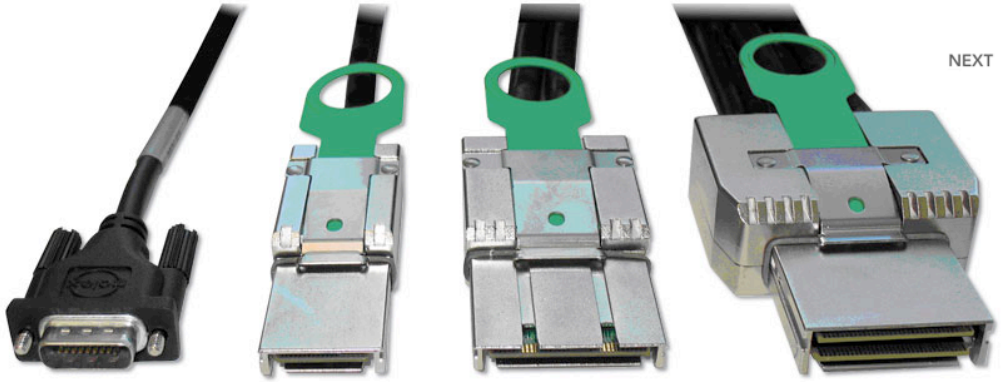
- Initially 10 and later 100 Mbit/s “Fast Ethernet”;
- Currently “Gigabit Ethernet” (GbE) mostly used, with commodity “Gigabit Ethernet” switches;
- Gigabit Ethernet over TCP/IP doesn’t substantially reduce latency in comparison with “Fast Ethernet”; this is due to protocol processing delays in hosts which remain CPU bound;
- Network interconnects are queuing systems and behave accordingly;
- Emerging “economy” interconnect is based on PCIe (Express) switches, similar in performance to Thunderbolt, also based on PCIe – BSD Socket API via PCIe device driver or TCP/IP over PCIe.

External PCIe (PCI Express) Hardware

Cronologic Ndigo External PCIe Expander

	Gen 1	Gen 2	Gen 3
x1	2.5Gb/s	5Gb/s	8Gb/s
x4	10Gb/s	20Gb/s	32Gb/s
x8	20Gb/s	40Gb/s	64Gb/s
x16	40Gb/s	80Gb/s	128Gb/s

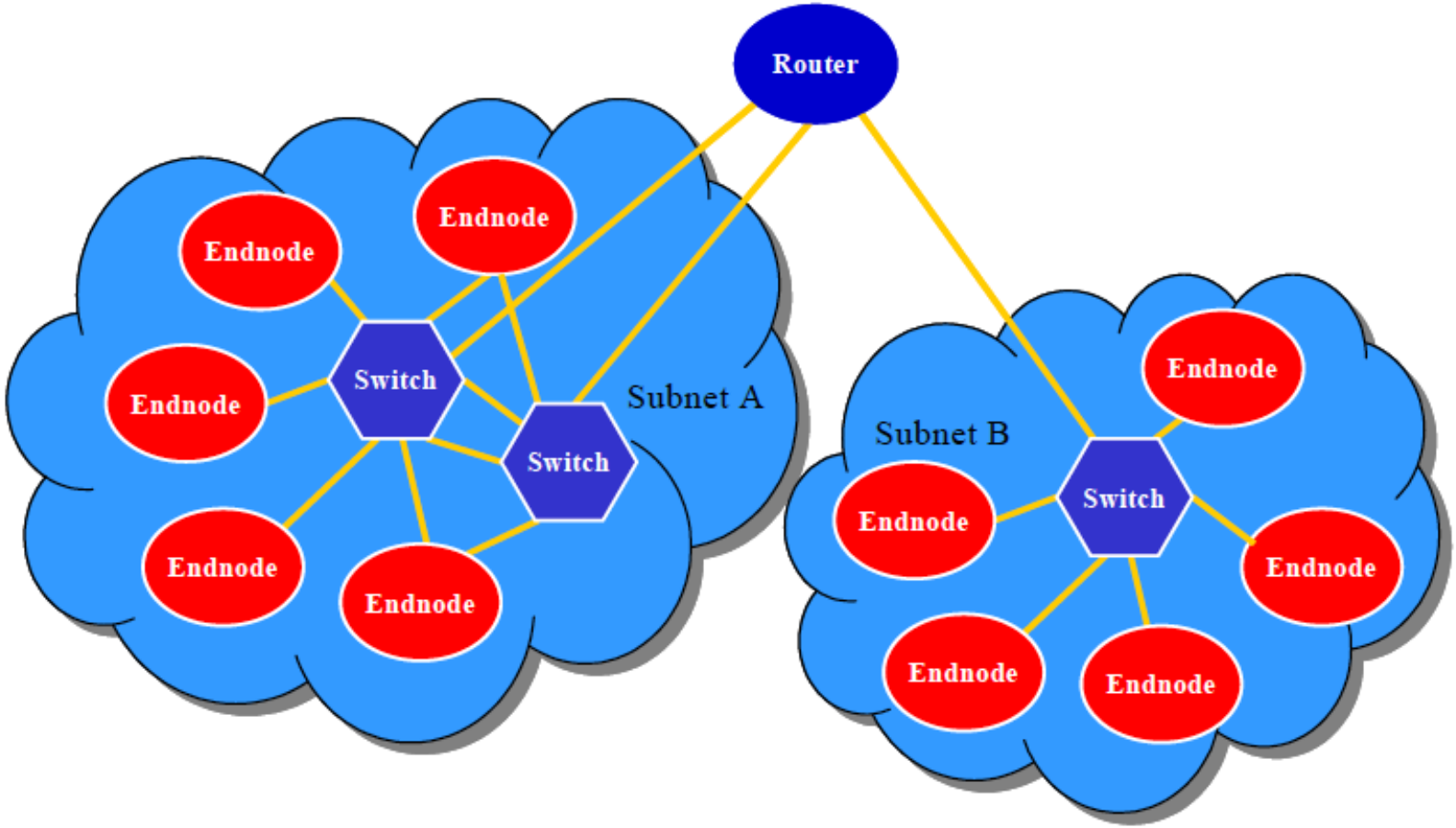
External PCIe Cables [http://www.onestopsystems.com]



“Performance” Cluster Interconnects

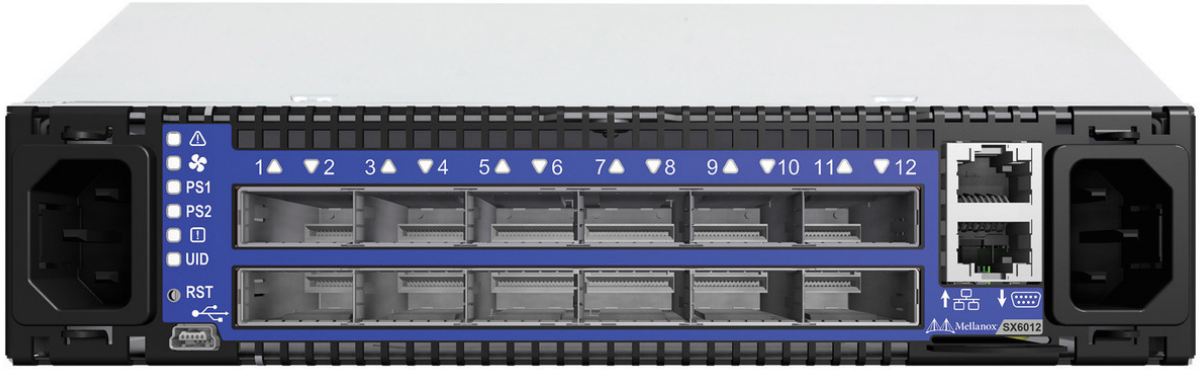
- cLAN VIA implementation. Very low latencies of 0.5µsec and high bandwidth. Vendor hardware may not be inter-operable;
- QsNet from Quadrics, similar to Myrinet, latency around 5µsec;
- ANSI/VITA 26-1998 Myrinet - simple low-latency switches, 640 Mbps to 10 Gbps;
- Infiniband – high performance mesh providing typically 56 – 100 Gbps;
- Infiniband is now widely used in large commercial and scientific supercomputing clusters;

InfiniBand Fabric

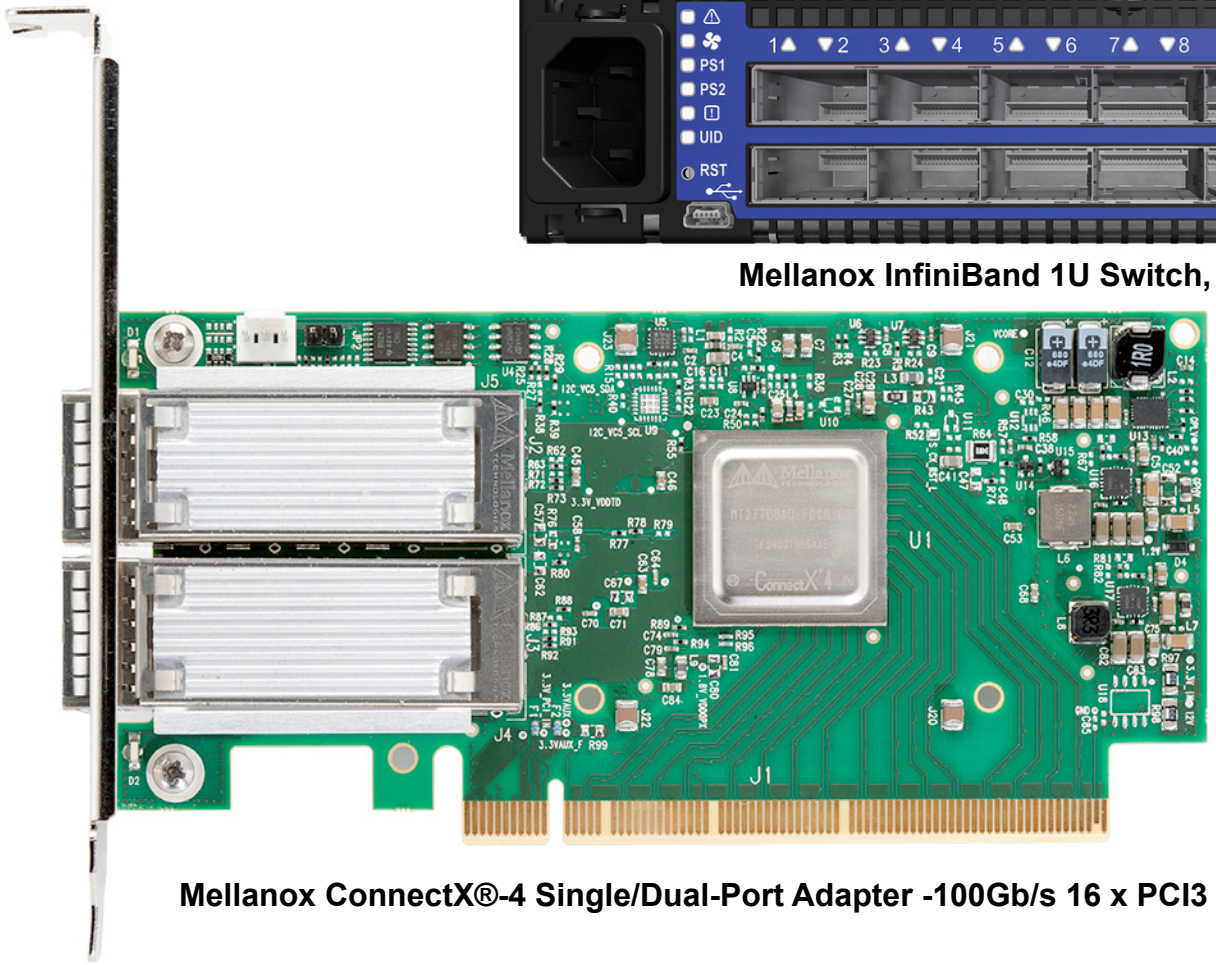


Mellanox - http://www.mellanox.com/pdf/whitepapers/IB_Intro_WP_190.pdf

InfiniBand Hardware



Mellanox InfiniBand 1U Switch, 12 QSFP+ ports



Mellanox ConnectX®-4 Single/Dual-Port Adapter -100Gb/s 16 x PCI3

CLUSTER PERFORMANCE

Benchmarking the Cluster

- How do we know how fast the cluster runs?
- Measurement of performance is an important factor while selecting a system design and architecture
- Numerous methods exist for benchmarking a cluster – none are ideal
- Important that the technique used provides a representative compute load compared to the intended application;
- Workloads can be homogenous – e.g. parallel identical jobs – *clusters, grids*;
- Workloads can be inhomogeneous/hetergenous – parallel but very different jobs – *clouds*;

Network Latency and Bandwidth

- Latency is easy to determine by measurement of elapsed time using common clocks;
- Bandwidth is easy to determine by benchmarking;
- Hard to rely on, as time variant network conditions dictate latency and usable bandwidth at any time;
- These only characterise the network and not the computational aspects of the system;
- Measurements usually done on lightly loaded systems to determine best achievable performance;

Performance Metrics

- Maximum aggregate performance of the system can be measured in terms of Maximum aggregate floating-point operations:

  $P = N*C*F*R$

- Where:
- *P* performance in flops,
- *N* number of nodes,
- *C* number of CPUs,
- *F* floating point ops per clock period,
- *R* clock rate.
- The other measure is for integer operations – using MOPS/MIPS

Application Performance

- Number of operations performed while executing the application, divided by the total runtime, MFLOPS or MOPS.
- Computed using a program similar to the actual program the user intends to run on the production system.
- More meaningful than theoretical peak performance.
- Need to correctly estimate the number of floating point (or integer) operations in the code.
- The algorithm must be optimised.
- Problems may arise if the code was tuned for a particular platform; specifically where code includes features written around machine specific performance accelerators.

Application Runtime Comparison

- The total wall-clock run time for an existing application and dataset.
- It frees us from counting the operations in the code – simply compare time to compute
- Also it removes the need to develop benchmarking code which may differ from the intended application.
- Performance tuning could distort results
- The same application must be run on all systems being benchmarked against one another

Scalability How Well Does It Scale?

- Scalability is computed thus:

  $S = T(1) / T(N)$
- Where T(1) is the wall clock time for a program to run on a single processor
- *T(N)* is the runtime over *N* processors
- A scalability figure close to *N* means the program scales well
- Stability metric helps estimate the optimal number of processors for an application
- An inefficient single CPU version of the application could falsify results
- Memory bottleneck may form for the single CPU version

Efficiency

- It is calculated thus:

  $E = P(N)/N$

- Values close to unity or 100% are ideally sought
- This metric suffers from the same problems as the scalability one

Percentage of Peak

- Application performance statistics are gathered in terms of the percentage of the theoretical peak performance
- Such statistics highlight the extent an application is making use of the computational power of the system
- It depends on the type of the applications

System Utilisation

- **System level effects include**
  - Competition between tasks executing on the system
  - I/O contention
  - Memory swapping
  - Job Scheduler inefficiencies
  - Job start-up delays
- **A system can be assessed on its long-term throughput through these statistics**
- **Statistics easily collected using `sar`, `vmstat`, `netstat`, `iostat` or other tools.**

Ping-Pong Test

- A widely used measure in clusters
- Tests the aggregated bandwidth and latency of the interprocessor communication network
- The API will bewritten in C and assumes the MPI libraries have been installed

LINPACK Benchmark

- High Performance LINPACK (HPL) Benchmark is widely used within clusters.
- These benchmarks execute the LINPACK codes available on Netlib.
- These benchmarks can overestimate performance.
- LINPACK is a library of functions written to solve linear equations and linear least-squares problems; written in FORTRAN; widely used in many applications.

NAS Parallel Benchmark Suite

- NBP designed by NASA Ames
- Consists of eight separate benchmarks
- These relate to five general computational kernels and;
- Three simulated computational fluid dynamics applications;
- As with LINPACK, NBP can be used for single or multi-node Beowulf clusters
- NBP suite MPI is recommended for use in these benchmarks

Reference:
David Bailey, How Fast is my Beowulf in Thomas Sterling (editor) Beowulf Cluster Computing with Windows

Observations

- **Different applications require different runtime environments, operating systems, libraries, parallel or distributed computing middleware, and hardware interconnection topologies;**
- **There are no panacea solutions – Grids, Clouds, conventional Clusters, HPC "supercomputers" all perform best for applications which fit their unique characteristics;**
- **Coarse assessments of performance may be unrealistically optimistic for many applications;**
- **Benchmarks can be highly accurate, but only for applications which are very close in behaviour to the benchmark;**
- ***The best performance benchmark is the intended application itself, using a representative dataset or parameters.***

MPI PROGRAMMING ENVIRONMENT

Message Passing Interface

- Computations in a cluster are done by dividing the work among the nodes;
- Easiest way to do this is for the processes to coordinate their activities through messages;
- MPI provides for this message passing model;
- MPI specifies a library interface, it is not a language;
- MPI suitable for some problems, but not others – it is not a panacea solution;
- All early variants of MPI were developed as a layer above the BSD `socket` API, although recently developed extensions in MPI 3.0 provide for messaging over shared memory;
- *MPI is very widely used in scientific computing, as the model used maps nicely onto mesh based simulations, such as finite element modelling, used in science and engineering.*

A Simple MPI Program

```
#include “mpi.h”
#include <stdio.h>
int main(int argc,char *argv[])
{
int rank, size
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
Printf(“Hello World from process %d of %d\n”,
   rank, size);
MPI_Finalize();
Return 0;
}
```

MPI Program Output

```
Hello World from process 0 of 4
Hello World from process 2 of 4
Hello World from process 3 of 4
Hello World from process 1 of 4
```

- The output is not ordered 0 to 3
- MPI does not specify the behaviour of other routines or language statements such as `printf`

Compiling and Running MPI

- MPI provides tools for compiling as linking using the native C compilers
- The following command line for MPICH will execute the previously shown helloworld program on four processors/processes:
- `mpirun –n 4 helloworld`

MPICH-G2

- **It is the grid enabled version of MPI - implements MPI v1.1 standard.**
- **Uses services from GTK**
  - Job startup
  - Security
- **It automatically converts data for different machine architectures and switches between networking protocols such as TCP/IP to vendor supplied MPI protocols for intra-machine communications.**

MPICH-G2 continued.

- **MPICH-G2 implements grid support through a special device *globus2***
- **Allows integration of heterogeneous architecture for applications**
  - Numerical computation architecture and visualisation architecture
- **Provides greater access to computer nodes available anywhere on the grid**
- **Can integrate high performance architectures with single CPU ones**

References/Reading

- http://www.csse.monash.edu.au/~carlo/SYSTEMS/Vector-CPU-0600.htm
- http://www.csse.monash.edu.au/~carlo/SYSTEMS/Cluster-Practical-1299.htm
- http://www.drdobbs.com/parallel/managing-cluster-computers/184404165
- Http://www.csse.monash.edu.au/~carlo/SYSTEMS/SCSI-SAN-0799.htm
- http://gridbus.org/papers/encyclopedia.pdf
- http://www.csse.monash.edu.au/~carlo/SYSTEMS/Gigabit-IP-LAN-1097.htm
- http://en.wikipedia.org/wiki/Gigabit_ethernet
- http://www.linuxvirtualserver.org/
- http://www.csse.monash.edu.au/~carlo/SYSTEMS/Infiniband-Intro-0901.htm

Cont ….

- William Gropp and Ewing Lusk, Parallel Programming with MPI in Thomas Sterling (editor) Beowulf Cluster Computing with Windows
- MPI-G2, http://www3.niu.edu/mpi/
- Blaise Barney, Linux Clusters Overview, Lawrence Livermore National Laboratory: https://computing.llnl.gov/tutorials/linux_clusters/
- Eric Hazen, Linux Cluster for Computational Physics Applications, Boston Uni: http://joule.bu.edu/~hazen/LinuxCluster/

READING / EXAMPLES [N/E]

Clusters and Grid Programming Environments

- Clustered hosts in one or another form remain as the core computing assets in Grids, Clouds, and large server systems e.g. W3

1. Beowulf,
2. Berkeley Now
3. LVS
4. Cluster of Clusters
5. PVM and MPI programming models
6. Grid MPI

Beowulf Cluster

- **Most demanding communications are with other nodes in the cluster**
- **In Beowulf every node may need to communicate with every other node**
- **Various types of data may be exchanged**
  - Large blocks of contiguous information
  - Small packets containing single values
  - Synchronisation signals
- **Bandwidth and latency requirements vary with data types**
  - Higher bandwidth for large contiguous blocks
  - Low latency for signals

Parallel Virtual Machine

- PVM is based on a dynamic computing model.
- Nodes can be added/deleted “on the fly”.
- Parallel task spawned/killed during the computation.
- Not as rich in message passing features as MPI.
- However being a virtual machine provides features for creating dynamic parallel programs.
- Provides fault tolerance and adaptability.

Architecture

- Each host participating in the virtual machine executes the *pvmd* daemon.
- The *pvmds* of all the hosts combine to form the virtual machine.
- Applications with PVM primitives can contact their local *pvmd* to interact and/or execute actions across the virtual machine.

Building Fault Tolerant Programs

- PVM provides a monitoring and notification system.
- Any or all tasks in an application can asked to be notified of specific events; these include exiting from a task within the application.
- Failure or deletion of a node in the cluster could also generate a notification.
- Tasks can watch their neighbouring tasks in a logical ring. Monitoring is actually done by PVM, the logical ring reduces the number of messages.

Adaptive Programs

- Cluster programs can be made to adapt to cluster metrics and other circumstance.
- A parallel application may dynamically adapt the size of the virtual machine through adding and releasing nodes based on the computational needs of the application.

Reference

Al Geist and Stephen Scott, Parallel Programming in PVM in Thomas Sterling (editor) Beowulf Cluster Computing with Windows

Linux Virtual Server

- LVS is also known as the “load balancing cluster server”.
- It is available under Linux.
- Allows a pool of Linux servers to appear as a single resource to the application user.
- All user requests are handled by the load-balancer, which is a separate server.
- The load-balancer passes requests to the cluster for processing.

LVS Goals

- **Build a high-performance and highly available server for Linux**
- **Uses clustering technology for**
  - good scalability,
  - reliability and
  - serviceability.

Load Balancing

- It is achieved by allocating service requests according to the cluster's resources
- Data to memory locations
- Files to disks
- Tasks to processors
- Packets to network interfaces
- Requests to servers

LVS forms of Load Balancing

- **Layer-2 load balancing**
  - Network traffic is aggregated over the interfaces by trunking multiple links into a single logical link
- **Layer-4 load balancing**
  - Distributing requests at the transport layer level according to the protocols among the cluster nodes
- **Layer-7 load balancing**
  - Parsing requests at the application layer level according to the applications requirements and distributing to the servers

LVS load balancing continued.

- Stateless applications can be load balanced by re-directing the connecting by varying the outputs from the DNS to different nodes in the cluster
- Link load balancing is to balance traffic among multiple links from different ISPs
- Database load balancing, balances the queries among the database servers
- Computational load balancing is achieved by programming tools such as MPI or PVM.

Myrinet

- It is a System Area Network defined by ANSI/VITA 26-1998
- Designed around simple low-latency switches
- Bandwidth ranges from 640 Mbps to 10 Gbps
- Latency is < 7 µsec
- A blocked port can lead to choking of the network - Problem can be avoided by choosing interconnect-rich topologies
- Myrinet can be optimised for MPICH by using dedicated processor for network traffic and accompanying open source driver.