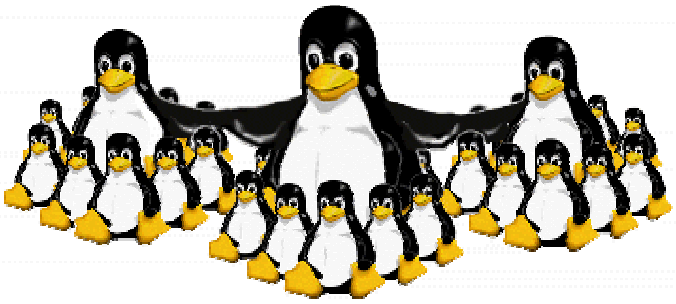
PENNSTATE
Aerospace Engineering

Beowulf Clusters and Parallel Computing using Linux

Anirudh Modi
(http://www.anirudh.net)
1/29/2002



OUTLINE

PENNSTATE
Aerospace Engineering

- What is Beowulf?
- Our cluster: COCOA
- How COCOA was made?
- Parallelization strategies
- Sample MPI code
- Parallel Benchmarks
- Applications
- Concluding remarks

What is Beowulf?

- Beowulf is the earliest surviving epic poem written in English. It is a story about a hero of great strength and courage who defeats a monster.
- Beowulf is a multi-computer architecture which can be used for parallel computations. It usually consists of one server node, and one or more client nodes connected together via some network
- The first Beowulf cluster was built by NASA (UTK??) in 1994!!
  - Consisted of 16 486DX4-100 Mhz machines each with 16 MB of memory.
  - Ran Linux kernel v1.1 and PVM.

What is Beowulf?

PENNSTATE
Aerospace Engineering

- It is a system built using commodity hardware components, like any PC capable of running Linux, standard Ethernet adapters, and switches.
- It does not contain any custom hardware components and is trivially reproducible.
- Beowulf also uses commodity software like the Linux operating system, Parallel Virtual Machine (PVM) and Message Passing Interface (MPI), and other widely available open-source software.
- A Beowulf system behaves more like a single machine rather than many workstations as the server node controls the client nodes transparently.

COCOA

PENNSTATE Aerospace Engineering

COst effective COmputing Array (COCOA)

25 Dual PII 400 MHz
512 MB ECC RAM each (12+ GB!!)
100 GB Ultra2W-SCSI Disk on server
100 Mb/s Fast Ethernet cards
Baynetworks 450T 27-way switch
(backplane bandwidth of 2.5 Gbps)
Monitor/keyboard switches
RedHat Linux with MPI

http://cocoa.ihpca.psu.edu

Cost just $100,000!! (1998 dollars)

COCOA-2

PENNSTATE Aerospace Engineering

***COst effective COmputing Array-2* (COCOA-2)**

21 Dual PIII 800 MHz (1U each)
1 GB PC133 ECC each (21 GB!!)
100 GB Ultra160-SCSI Disk on server
Dual 100 Mb/s Fast Ethernet cards
Two 1U HP-Procurve 2324 24-way switches (b/p bandwidth of 9.6 Gbps)
Monitor/keyboard switches
RedHat Linux 7.0 with MPI
Two 4U 3000 VA APC SmartUPS

http://cocoa2.ihpca.psu.edu

Cost just $58,000!! (2000 dollars)

COCOA: Motivation

- To get even 50,000 hrs of CPU time in a supercomputing center is difficult. COCOA can offer more that 400,000 CPU hrs annually!
- One often has to wait for days in queues before the job can run.
- Commodity PCs are getting extremely cheap. Today, it just costs $1K to get a dual AMD 2000XP computer with 512 MB RAM (30-50% more from reliable vendors like Dell/IBM/HP).
- Advent of Fast Ethernet (100 Mbps) networking has made a reasonably large PC cluster feasible (at a very low cost; 100 Mbps ethernet adaptor ~ $10). Myrinet and Gigabit networking are soon getting popular.
- Price/performace (or $/Mflop) for these cheap clusters is way better than for a IBM SP/SGI/Cray supercomputer (atleast factor of 10 better!)
- Maintenance for such a PC cluster is less cumbersome than the big computers. A new node can be added to COCOA in just 10 minutes!

COCOA

PENNSTATE Aerospace Engineering

- COCOA runs on commodity PCs using commodity software (RedHat Linux).
- Cost of software: negligible. The only commercial software installed are Portland Group Fortran 90 compiler and TECPLOT.
- Free version of MPI from ANL (MPICH) and Pentium GNU C compiler (generates highly optimized code for Pentium class chips) are installed.
- Distributed Queueing System (DQS) has been setup to submit the parallel/serial jobs. Several minor enhancements have been incorporated to make it extremely easy to use. Live status of the jobs and the nodes is available on the web:

  http://cocoa.ihpca.psu.edu

- Details on how COCOA was built can be found in the COCOA HOWTO:

  http://bart.ihpca.psu.edu/cocoa/HOWTO/

COCOA: Hardware

PENNSTATE Aerospace Engineering

Setting up the hardware was fairly straight-forward.

Here are the main steps:

- Unpacked the machines, mounted them on the rack and numbered them.
- Set up the 24-port network switch and connected one of the 100 Mbit ports to the second ethernet adapter of the server which was meant for the private network. The rest of the 23 ports were connected to the ethernet adapters of the clients. Then an expansion card with 2 additional ports was added on the switch to connect the remaining 2 clients.
- Stacked the two 16-way keyboard-video-mouse (KVM) switches and connected the video-out and the keyboard cables of each of the 25 machines and the server to it. A single monitor and keyboard were then hooked to the switch which controlled the entire cluster.
- Connected the power cords to the four UPS (Uninterruptible Power Supply).

COCOA: Server setup

PENNSTATE Aerospace Engineering

Setting up the software is where the real effort came in!

Here are the main steps:

1. The ***server*** was the first to be set up.
   i. Installed RedHat Linux (then 5.1) from the CD-ROM.
   ii. Configured all relevant hardwared which were all automatically detected.
   iii. Partitioned the 54 GB drives into /, /usr, /var, /home, /tmp, and chose the relevant packages to be installed. Two 128 MB swap partitions were also created.
2. Latest stable Linux kernel was installed:
   1. Latest kernel was downloaded from kernel.org (then v2.0.36, now v2.2.16).
   2. It was compiled with SMP support using the Pentium Optimized GNU CC compiler pgcc (now a part of *egcs*) http://www.goof.com/pcg/, which generates highly optimised code specifically for the Pentium II chipset [`pgcc -mpentiumpro -O6 -fno-inline-functions`]. Turning on SMP support was just a matter of clicking on a button in the *Processor type and features* menu of the kernel configurator (started by running `make xconfig`).

COCOA: Networking

PENNSTATE Aerospace Engineering

3. Secure-shell was downloaded from http://www.cs.hut.fi/ssh/, compiled and installed for secure access from the outside world. Nowadays, RedHat RPMs for **ssh** are available at http://www.rpmfind.net/ and several other RPM repositories, which make it much easier to install.
4. Both the fast-ethernet adapters (3Com 3C905B) were then configured:
   - **eth1** to the outside world with the real IP address 128.118.170.11
   - **eth0** to the private network using a dummy IP address 10.0.0.1.
   - Latest drivers for the adapters were downloaded and compiled into the kernel to ensure 100 Mbit/sec Full-duplex connectivity.
   - For the network configuration, the following files were modified: **/etc/sysconfig/network, /etc/sysconfig/network-scripts/ifcfg-eth0** and **/etc/sysconfig/network-scripts/ifcfg-eth1**.

COCOA: Networking

/etc/sysconfig/network:

HOSTNAME=cocoa.ihpca.psu.edu
DOMAINNAME=ihpca.psu.edu
GATEWAY=128.118.170.1
GATEWAYDEV=eth1

/etc/sysconfig/network-scripts/ifcfg-eth0:

DEVICE=eth0
IPADDR=10.0.0.1
NETMASK=255.255.255.0
NETWORK=10.0.0.0
BROADCAST=10.0.0.255

/etc/sysconfig/network-scripts/ifcfg-eth1:

DEVICE=eth1
IPADDR=128.118.170.11
NETMASK=255.255.255.0
NETWORK=128.118.170.0
BROADCAST=128.118.170.255

COCOA: Networking

Aerospace Engineering

5. For easy and automated install, BOOT protocol was used to assign IP addresses to the client nodes. The BOOTP server was enabled by uncommenting the following line in **/etc/inetd.conf**:

   bootps dgram udp wait root /usr/sbin/tcpd bootpd.

   A linux boot floppy was prepared with the kernel support for 3c905B network adapter which was used to boot each of the client nodes to note down their unique 48-bit network hardware address (eg. *00C04F6BC052*, also known as **MAC** or Media Access Control address). Using these address, the /etc/bootptab was edited to look like:

```
.default:\
        :hd=/boot:bf=install.ks:\
        :vm=auto:\ :dn=hpc.ihpca.psu.edu:\
        :gw=10.0.0.1:\
        :rp=/boot/client/root:
node1:ht=ethernet:ha=00C04F6BC0B8:ip=10.0.0.2:tc=.default
node2:ht=ethernet:ha=00C04F79AD76:ip=10.0.0.3:tc=.default
node3:ht=ethernet:ha=00C04F79B5DC:ip=10.0.0.4:tc=.default
. . .
node25:ht=ethernet:ha=00C04F79B30E:ip=10.0.0.26:tc=.default
```

COCOA: Networking

Aerospace Engineering

6. The **/etc/hosts** file was edited to look like:

```
127.0.0.1       localhost       localhost.localdomain
# Server [COCOA]
128.118.170.11 cocoa.ihpca.psu.edu cocoa.aero.psu.edu cocoa

# IP address <--> NAME mappings for the individual nodes of the cluster
10.0.0.1        node0.hpc.ihpca.psu.edu node0          # Server itself!
10.0.0.2        node1.hpc.ihpca.psu.edu node1
10.0.0.3        node2.hpc.ihpca.psu.edu node2
…
10.0.0.26       node25.hpc.ihpca.psu.edu node25
```

The **/etc/host.conf** was modified to contain the line:

```
order hosts, bind
```

This was to force the lookup of the IP address in the **/etc/hosts** file before requesting information from the DNS server.

COCOA: NFS

Aerospace Engineering

7. The filesystems to be exported were added to **/etc/exports** file which looked like:

```
/boot        node*.hpc.ihpca.psu.edu (ro,link_absolute)
/mnt/cdrom   node*.hpc.ihpca.psu.edu (ro,link_absolute)
/usr/local   node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
/home        node*.hpc.ihpca.psu.edu (rw,no_all_squash,no_root_squash)
```

COCOA: Client setup

Aerospace Engineering

8. For rapid, uniform and unattended installation on each of the client nodes, RedHat KickStart installation was ideal. Here is how my kickstart file **/boot/install.ks** looked like:

```
lang en
network --bootproto bootp
nfs --server 10.0.0.1 --dir /mnt/cdrom
keyboard us
zerombr yes
clearpart --all
part / --size 1600
part /local --size 2048
part /tmp --size 400 --grow
part swap --size 127
install mouse ps/2
timezone --utc US/Eastern
rootpw --iscrypted kQvti0Ysw4r1c
lilo --append "mem=512M" --location mbr
```

COCOA: Client setup


```
%packages
@ Networked Workstation
%post
rpm -i ftp://10.0.0.1/pub/CLUSTER/RPMS/wget-1.5.0-2.i386.rpm
rpm -i ftp://10.0.0.1/pub/CLUSTER/RPMS/xntp3-5.93-2.i386.rpm
/usr/bin/wget ftp://10.0.0.1/pub/CLUSTER/kernel/vmlinuz -O/boot/vmlinuz
/usr/bin/wget ftp://10.0.0.1/pub/CLUSTER/conf/lilo.conf -O/etc/lilo.conf
/sbin/lilo
/usr/bin/wget ftp://10.0.0.1/pub/CLUSTER/conf/hosts.equiv -O/etc/hosts.equiv
sed "s/required\(.*securetty\)/optional\1/g" /etc/pam.d/rlogin > /tmp/rlogin
mv /tmp/rlogin /etc/pam.d/rlogin
```

COCOA: Client setup

In one of the post installation commands above, the first line of the **/etc/pam.d/rlogin** file is modified to contain:

```
auth       optional     /lib/security/pam_securetty.so
```

This is to required enable rlogin/rsh access from the server to the client without password which is very useful for the software maintenance of the clients. Also, the **/etc/hosts.equiv** file mentioned above looks like this:

```
node0
node1
node2
…
node25
```

The RedHat Linux CD-ROM was then mounted as **/mnt/cdrom** on the server which was exported to the client nodes using NFS. A new kernel with SMP and BOOTP support was compiled for the client nodes.

COCOA: Client setup

9. Clients were rebooted after installation, and the cluster was up and running! Some useful utilities like **brsh** (http://www.beowulf.org/software/RPMS/beobase-2.0-1.i386.rpm) were installed to enable **rsh** a single identical command to each of the client nodes. This was then used to make any fine changes to the installation. NIS could have been installed to manage the user logins on every client node, but instead a simple shell script was written to distribute a common **/etc/passwd, /etc/shadow** and **/etc/group** file from the server.
10. All the services were disabled in **/etc/inetd.conf** (except for **in.rshd**) for each of the client nodes as they were unnecessary.
11. The Portland Group Fortran 77/90 and HPF compilers (commercial) were then installed on the server.

COCOA: MPI

12. Installing MPI:
    - Source code for freeware implementation of **MPI** library, **MPI-CH** was downloaded from http://www.mcs.anl.gov/mpi/
    - Compiled using **egcs** for optimum performance.
    - Installed on **/usr/local** partition of the server which was NFS mounted across all the client nodes for easy access.
    - The **mpif77/f90** script was modified to suit our needs. The **/usr/local/mpi/util/machines/machines.LINUX** was then modified to add two entries for each client node (as they were dual-processor SMP nodes). Jobs could now be run on the cluster using interactive **mpirun** commands!

COCOA: Queueing

Aerospace Engineering

13. Queueing system:
    - **DQS v3.2** was downloaded from Florida State University (http://www.scri.fsu.edu/~pasko/dqs.html).
    - It was compiled and installed as **/usr/local/DQS/** on the server making it available to all the client nodes through NFS.
    - Appropriate server and client changes were then made to get it functional (i.e. adding the relevant services in **/etc/services**, starting **qmaster** on the server and **dqs_execd** on the clients).
    - Wrapper shell scripts were then written for **qsub, qstat** and **qdel** which not only beautified the original **DQS** output (which was *ugh* to begin with!), but also added a few enhancements. For example, **qstat** was modified to show the number of nodes requested by each pending job in the queue.

COCOA: Queueing

14. **COCOA** was now fully-functional, up and running and ready for benchmarking and serious parallel jobs! As with the kernel, use of **pgcc/egcs** compiler was recommended for all the C/C++ codes. In particular, using **pgcc** with options:

    -mpentiumpro -O6 -funroll-all-loops

    for typical FPU intensive number crunching codes resulted in ***30%*** increase in execution speed over the conventional optmization flags.

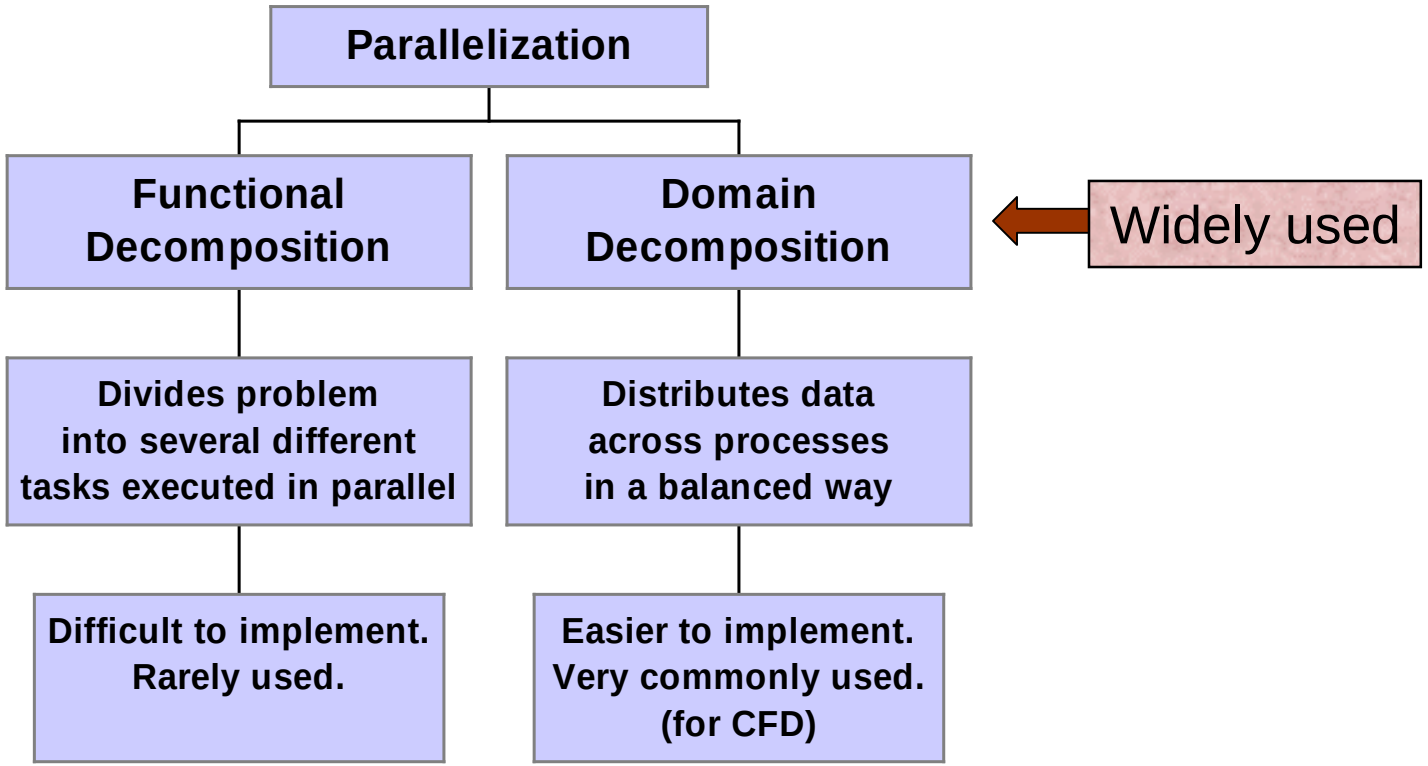
COCOA: Advantages

Aerospace Engineering

- Easy to administer since the whole system behaves like one single machine.
  - All modifications to the server/clients are done remotely.
- Due to the kickstart installation process, adding a new node is a very simple process which barely takes 10 minutes!!
- Due to the uniformity in the software installed on all the clients, upgrading them is very easy.
- Submitting and deleting jobs is very easy.
  - A regular queue is present for regular jobs which take hours or days to complete.
  - An 8-proc debug queue is present for small jobs.

Parallelization strategies

Aerospace Engineering



Most massively parallel codes use *Single Program Multiple Data (SPMD)* parallelism, i.e., same code is replicated to each process.

Parallelization strategies

PENNSTATE Aerospace Engineering

Domain decomposition:

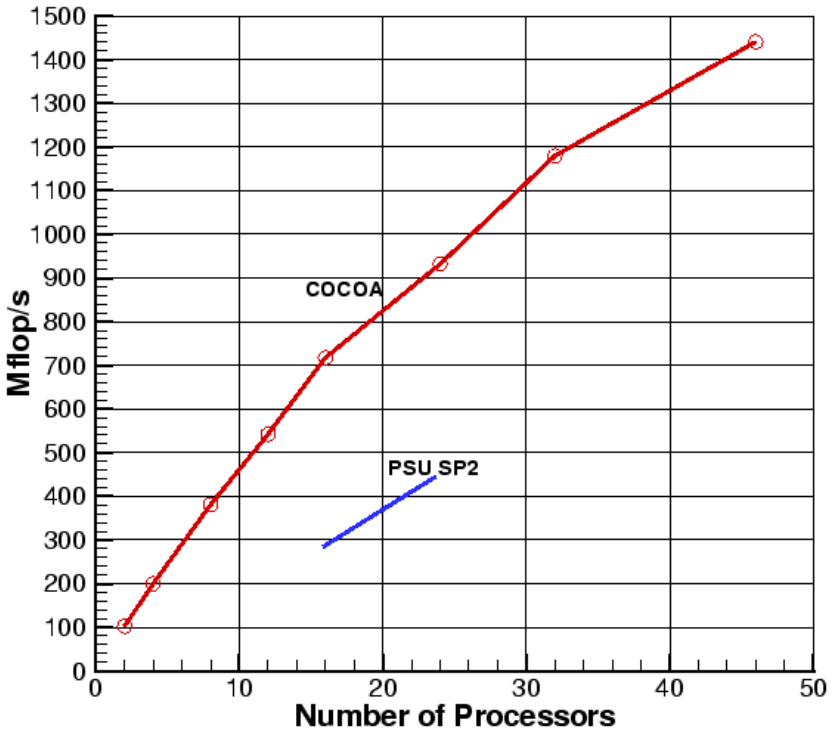
Grid around RAE 2822 a/f

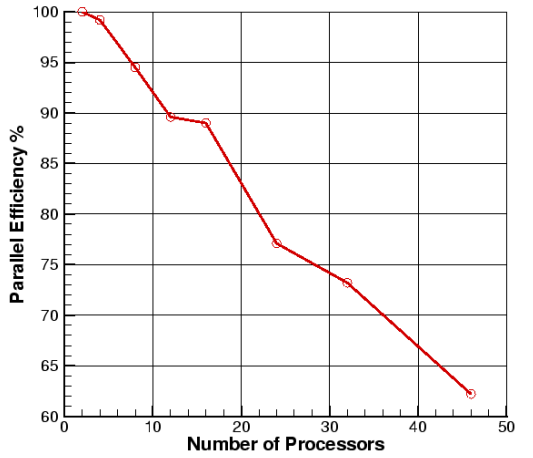
MPI: Sample program


```
#include "mpi.h"

main(int argc, char **argv ) {
char message[20];
int i,rank, size, type=99;
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD,&size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0) {          /* Master Node */
            strcpy(message, "Hello, world");
            for (i=1; i<size; i++)
            MPI_Send(message, 13, MPI_CHAR, i, type, MPI_COMM_WORLD);
      }
else                      /* Slave Node */
       MPI_Recv(message, 20, MPI_CHAR, 0, type, MPI_COMM_WORLD, &status);

printf( "Message from node =%d : %.13s\n", rank,message);
MPI_Finalize();
}
```

COCOA: Benchmarks

Aerospace Engineering



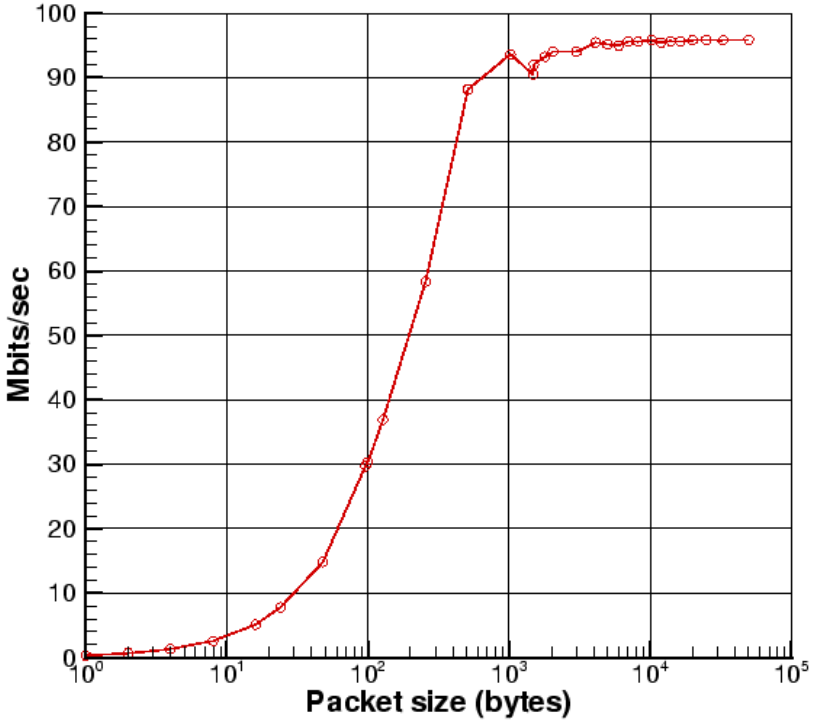


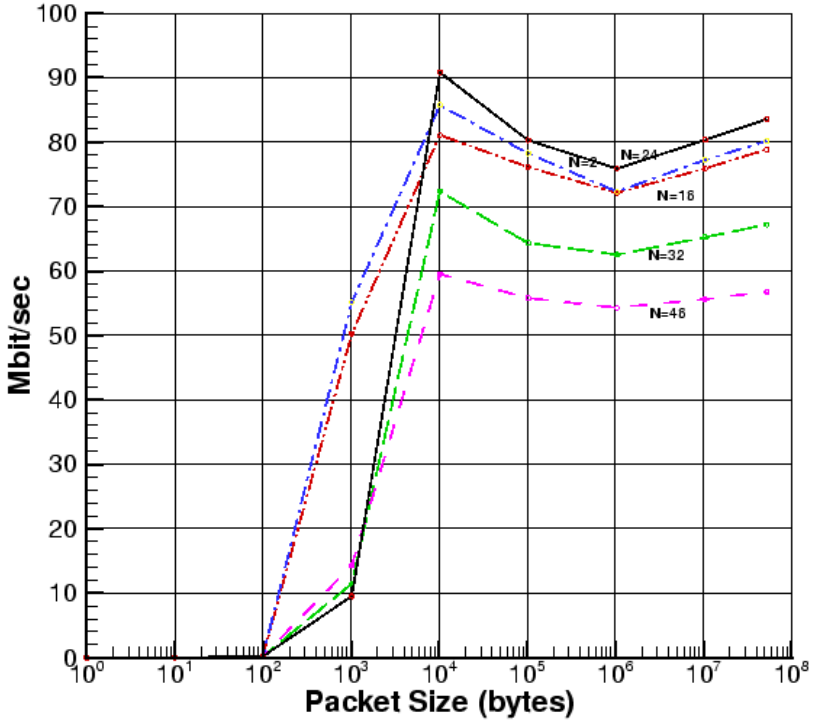
Performance of "*Modified PUMA*"

COCOA: Benchmarks

Aerospace Engineering

Network Performance





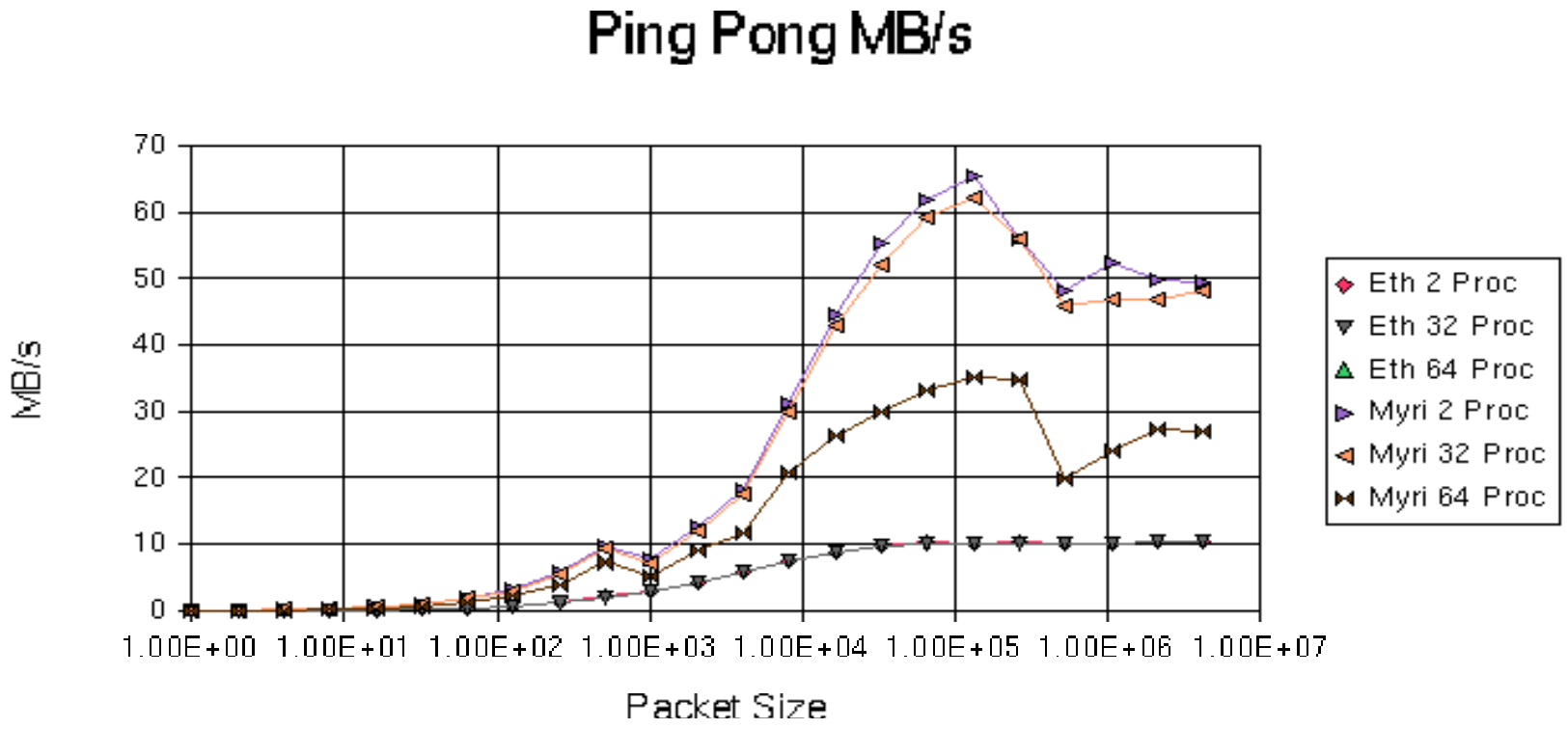
netperf test between any two nodes

MPI_Send/Recv test

➡ Ideal message size >= 10 Kbytes

Notes on Networking

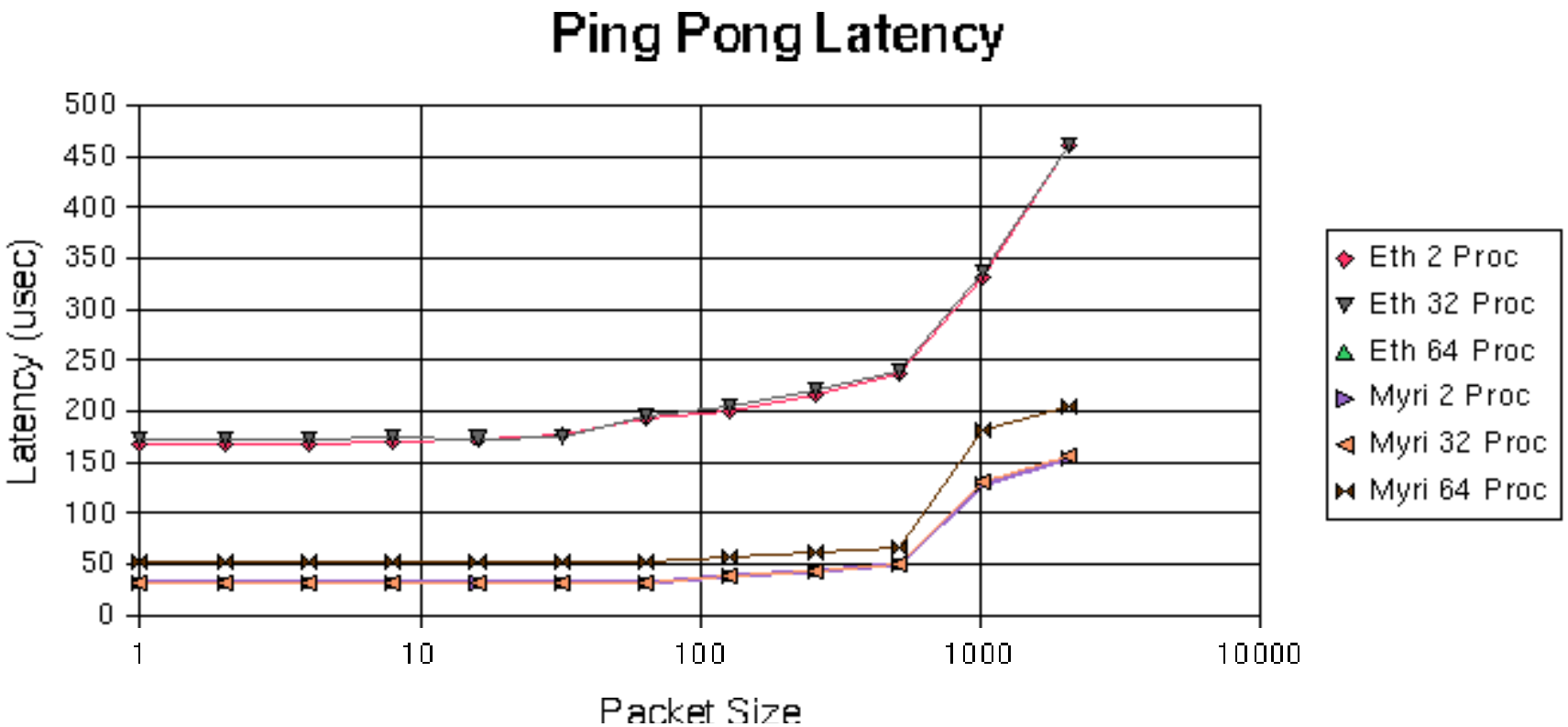
Aerospace Engineering



Comparison of Ethernet vs Myrinet on LION-X (CAC cluster)

Notes on Networking

Aerospace Engineering

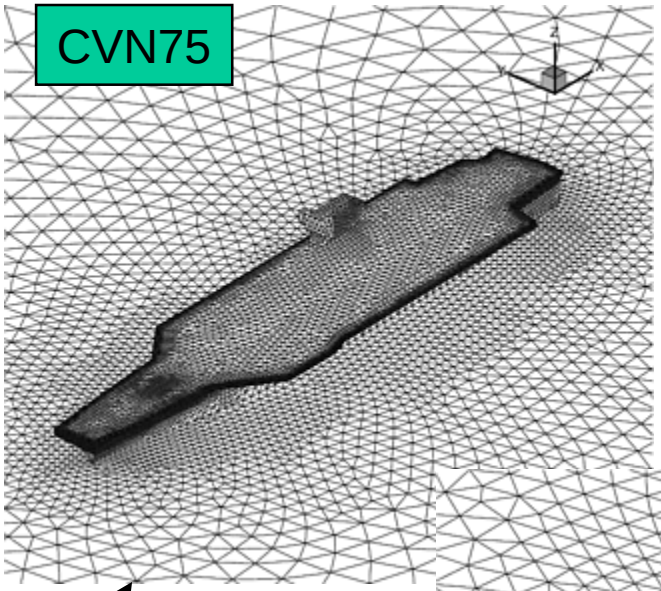


Comparison of Ethernet vs Myrinet on LION-X (CAC cluster)

Sample Applications: Grids

Aerospace Engineering

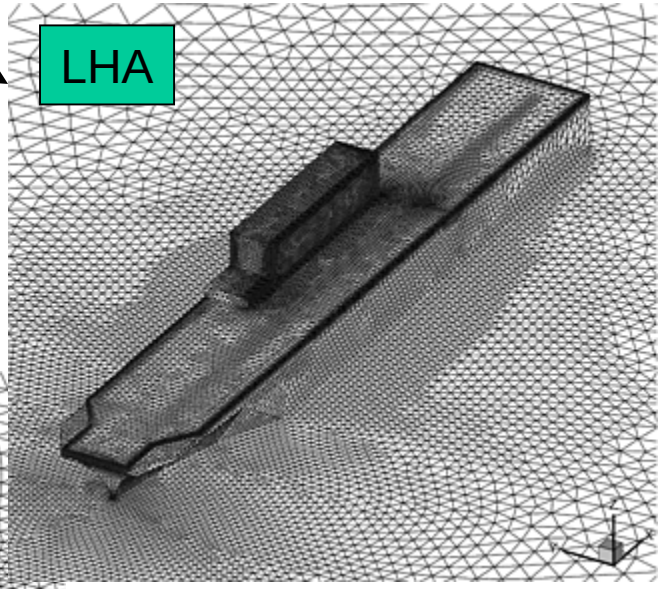
CVN75



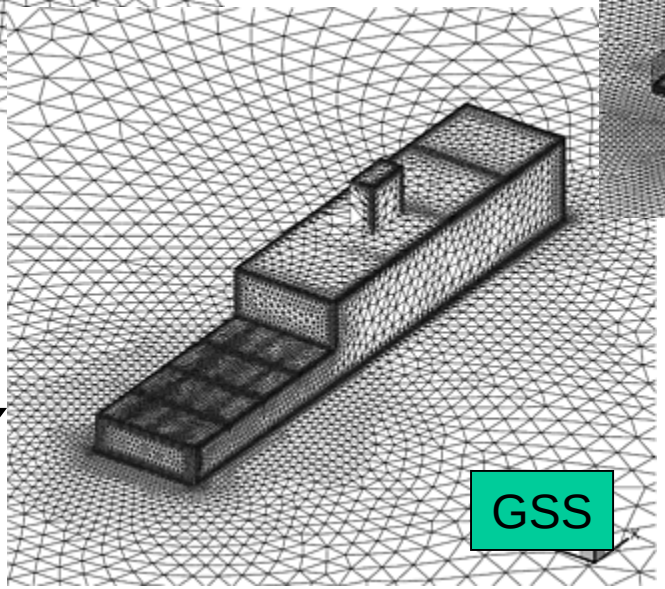
478,506 cells
974,150 faces

1,216,709 cells
2,460,303 faces

LHA



483,565 cells
984,024 faces



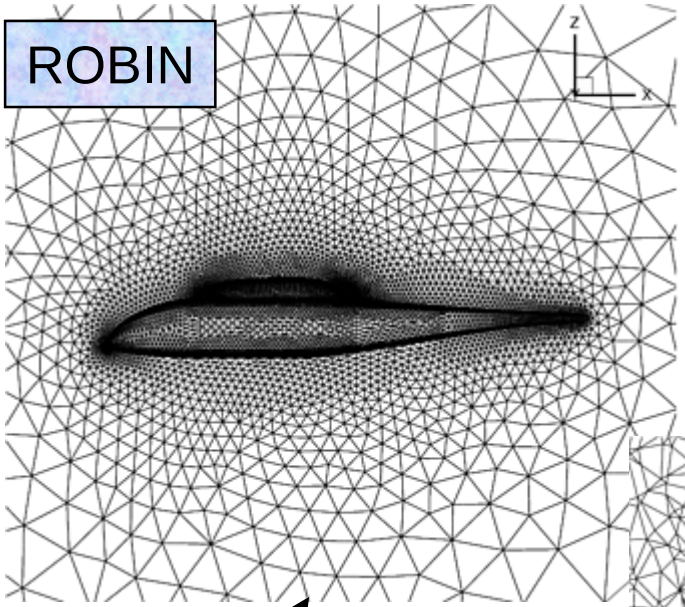
GSS

Ship Configurations

Sample Applications: Grids

Aerospace Engineering

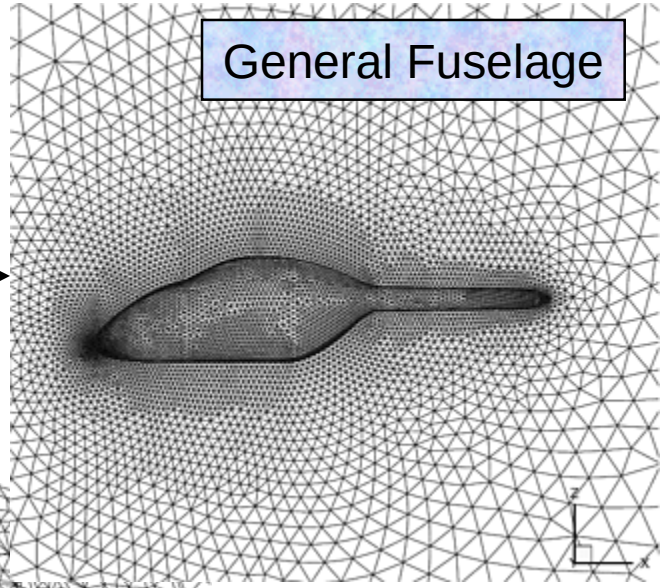
ROBIN



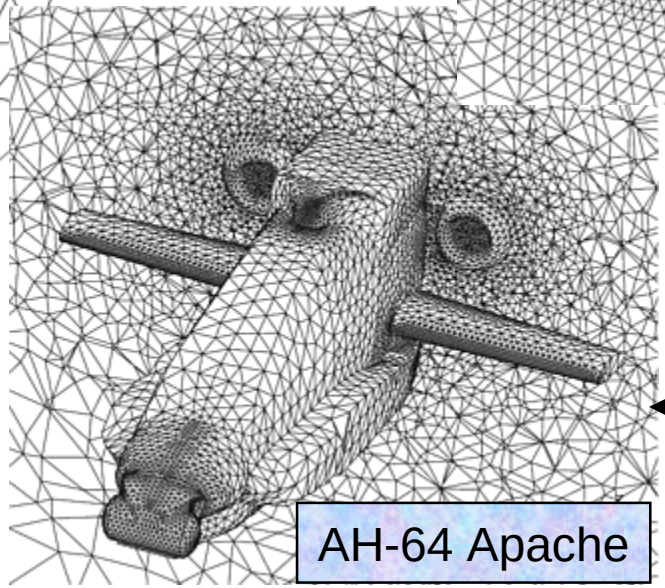
260,858 cells
532,492 faces

General Fuselage

380,089 cells
769,240 faces



Helicopter Configurations



555,772 cells
1,125,596 faces

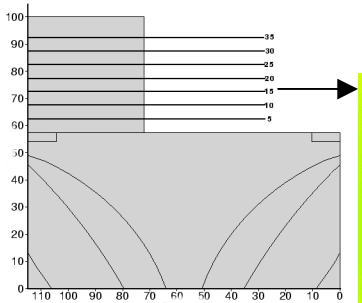
AH-64 Apache

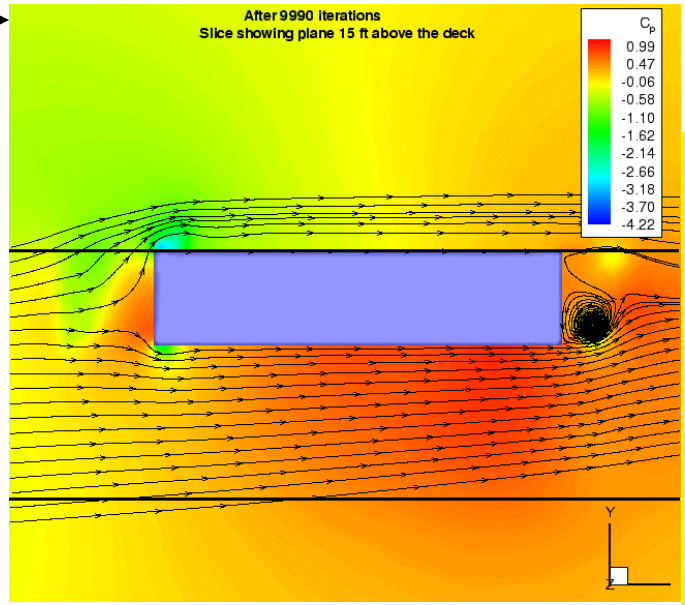
Results: Ship Configurations

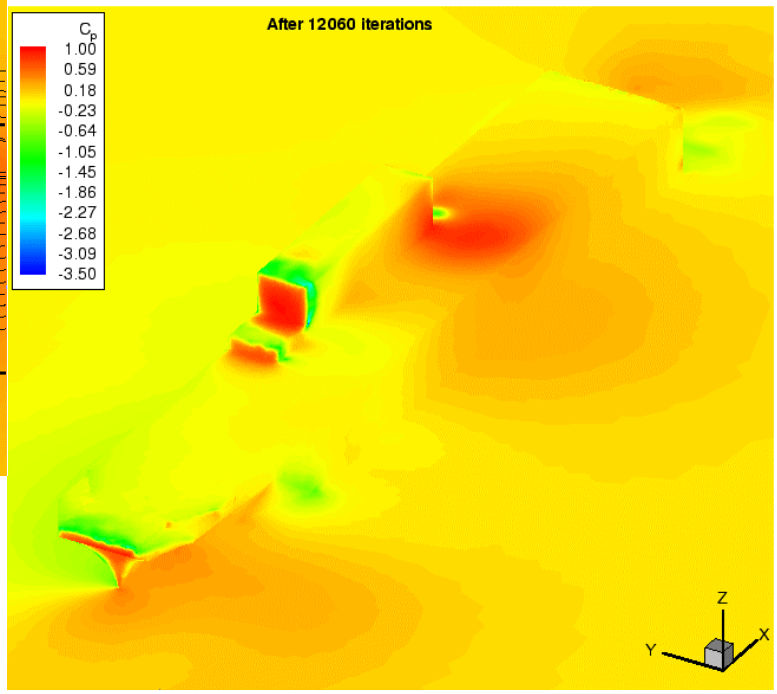
Aerospace Engineering

Flow conditions:
U=25 knots
β=5 deg

1,216,709 cells
2,460,303 faces
3.7 GB RAM



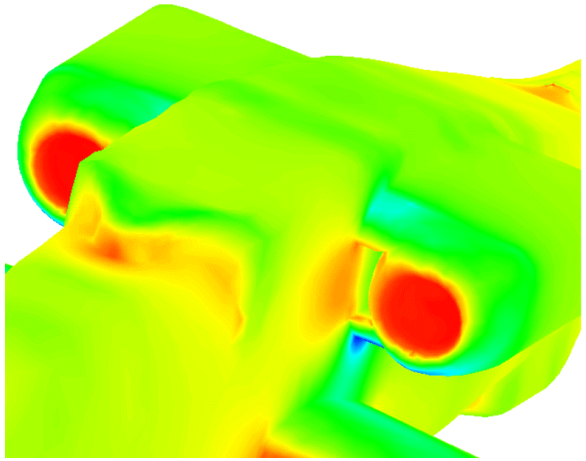




Landing Helicopter Aide (LHA)

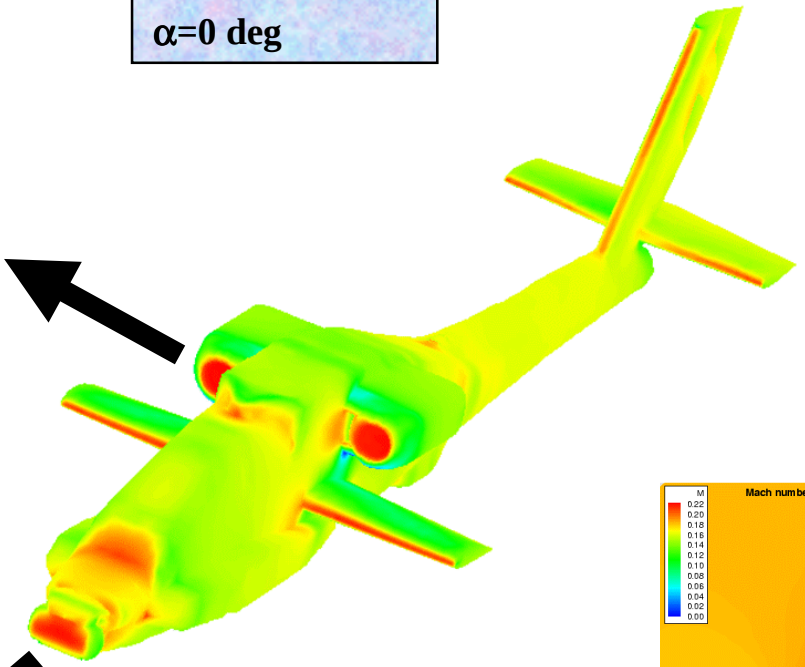
Results: Helicopter Configurations

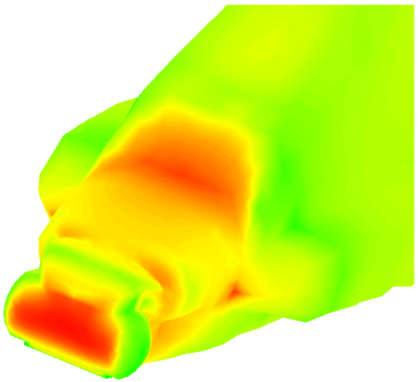
Aerospace Engineering



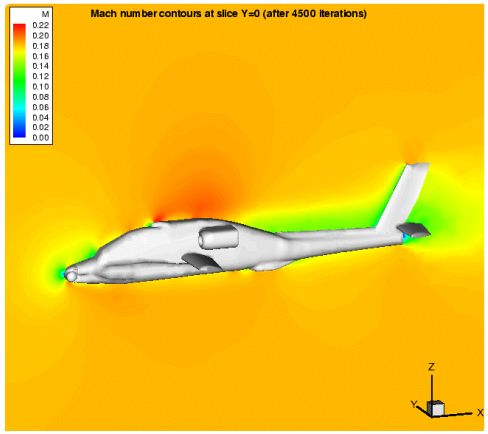
Flow conditions:
U=114 knots
α=0 deg

555,772 cells
1,125,596 faces
1.9 GB RAM



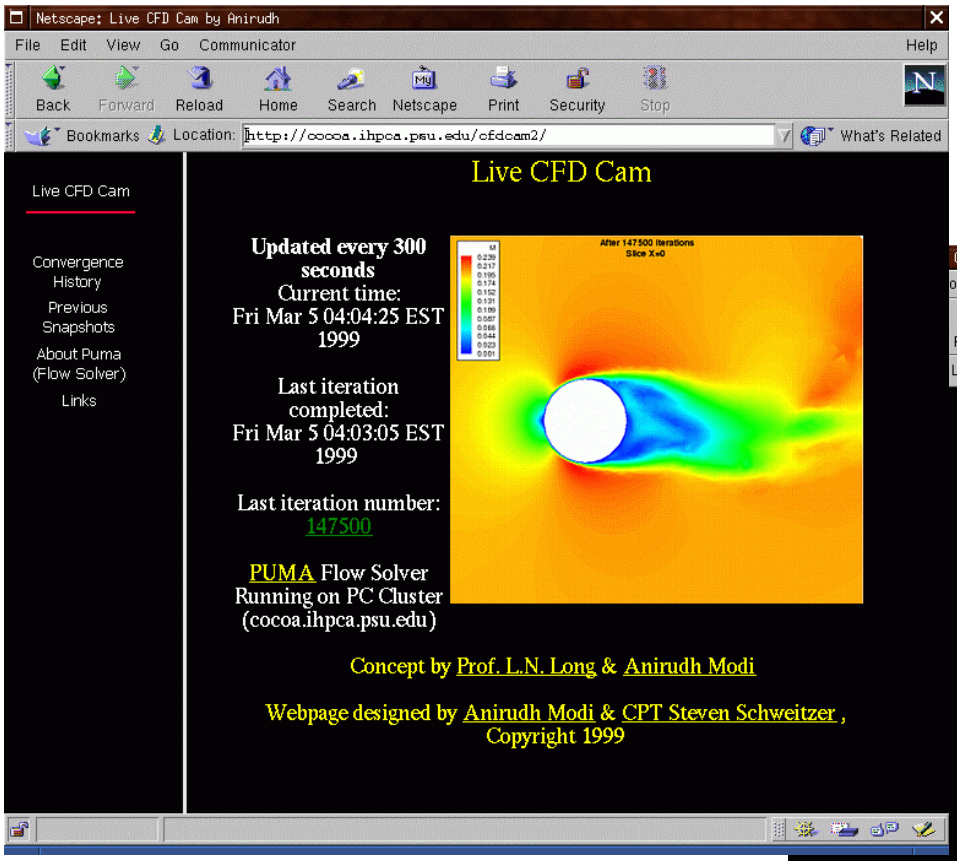


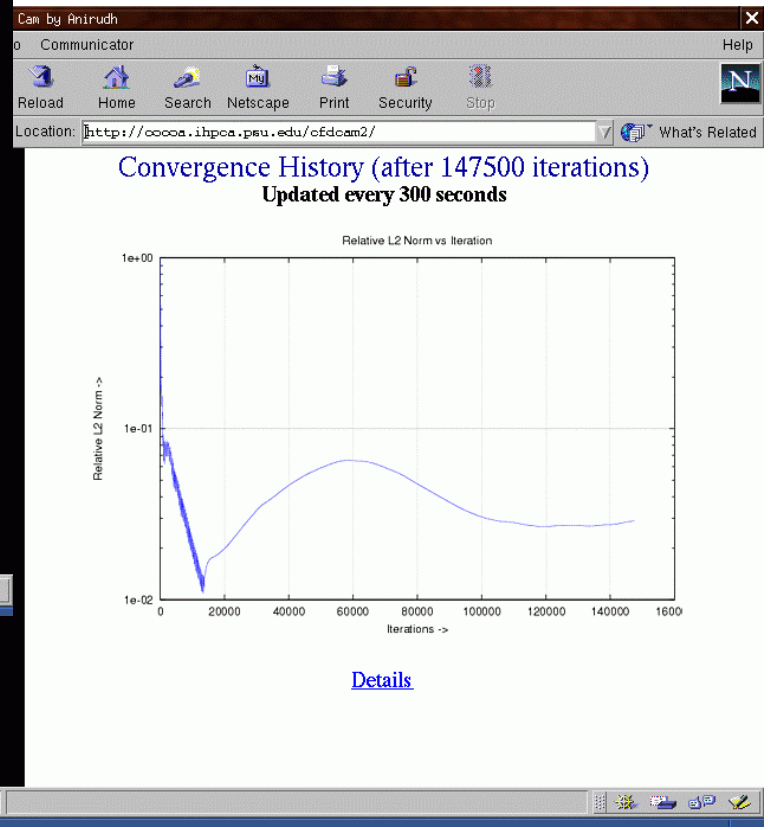
AH-64 Apache



Live CFD-Cam

PENNSTATE Aerospace Engineering





Beowulf Facts

- The fastest known Beowulf cluster to date (Jan 2001) is the LosLoBoS (Lots of Boxes on Shelves) built by IBM for University of New Mexico and NCSA in March 2000.
  - It consists of 256 dual-processor PIII-733 IBM Netfinity servers (500 processors total) running Linux.
  - Uses Myrinet for networking
  - Has peak theoretical performance rating of 375 Gflops/sec!! (24^{th} in the Top-500 list of Supercomputers)
  - Costs approx $1.2 million.
  - Even if it gets only 10-11% of the theoretical performance, i.e., 40 Gflops, this gives the cost per Gflop to be $30K (cheap)!!
- Several clusters using Alpha chips were popular till mid-2001 but are now disappearing as development of Alpha chips has stopped.
- Check out http://top500.org/list/2001/11/

Beowulf limitations

- Beowulf clusters are limited by the networking infrastructure available on the PCs today. Although Myrinet and Gigabit Ethernet are available, they are still very expensive to be cost effective. Systems using fast ethernet typically do not scale well beyond 30-40 nodes.
- Applications requiring a frequent inter-node communications (or very low-latency) are not well suited to run on these systems.
- Beowulf clusters are not meant to entirely replace the modern Supercomputers like IBM SP2, Crays, SGIs, Hitachi (although there are exceptions as discussed in the previous slide).
- They are best suited as low-cost supplements to traditional supercomputing facilities which are still a lot better owing to the better networking and infrastructure, although Beowulf clusters have a factor of 10 cost advantage.

Make your own cluster!

PENNSTATE Aerospace Engineering

Let's make an 8-processor cluster with each node consisting of:

- Dual AMD Athlon 2000XP/ 1 GB PC2100 (2x512)/ 20 GB 7200rpm HDD/ FDD / Case / CPU & case Fan / Video card / Assembly / 10/100 ethernet card
  = $1100.00 (from micropro.com, configured 1/29/02)
- 4 such machines (4 GB memory total) w/CD-ROM/100 GB for server = $4600 [cost of memory = $120x8 = $960, or 21%]
- 8-port switch: $100, 4-port KVM switch w/cables: $100
- 1 monitor/keyboard/mouse+cables = $300
- Linux and other s/w = FREE

TOTAL cost for 8-proc Athlon 2000XP cluster w/ 4 GB memory = $5.0K!
(If you buy individual parts online and assemble it yourself, you should be able to save an additional 10-15%)

Peak Mflop: 2000x8 Mflops = 16.0 Gflop (theoretical), assuming 10% efficiency (conservative) = 1.6 Gflop.i.e., $5.0K/1.6 = $3.125K/Gflop!!!! Amazing!

For those that want some luxury:
Add more disk-space, good video card, sound card, great monitor = $400 at most!
(or else, get just 2 GB of memory and use the $480 saved on the above luxuries)