# Beowulf clusters — an overview

Åsmund Ødegård

April 4, 2001

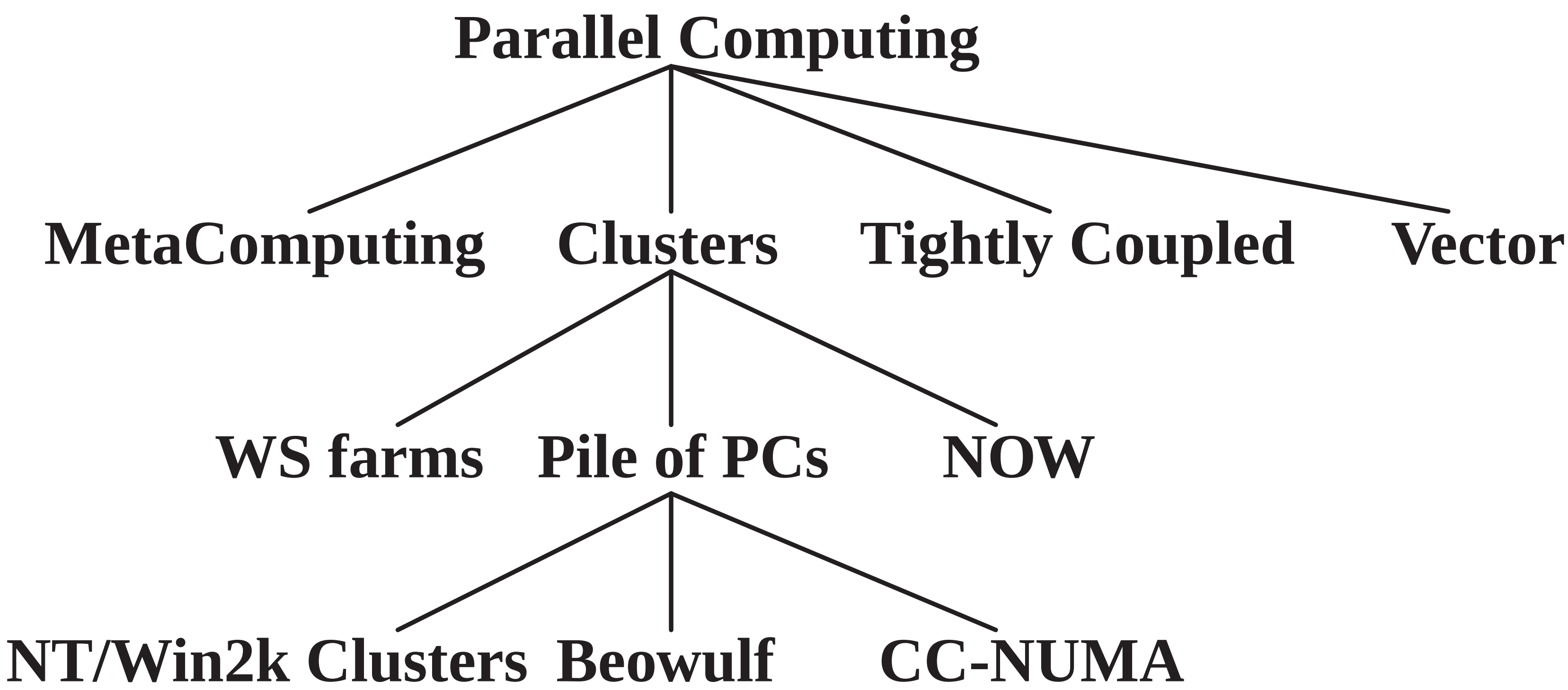
# Contents

# Introduction

**Why clusters ?**

- **“Work harder”**
  - **More CPU–power, more memory, more everything**
- **“Work smarter”**
  - **Better algorithms**
- **“Get help”**
  - **Let more boxes work together to solve the problem**
  - **Parallel processing**
- **by Greg Pfister**

- **Beowulfs in the Parallel Computing picture:**



# What is a Beowulf

- **Mass–market commodity off the shelf (COTS)**
- **Low cost local area network (LAN)**
- **Open Source UNIX like operating system (OS)**
- **Execute parallel application programmed with a message passing model (MPI)**
- **Anything from small systems to large, fast systems. The fastest rank as no.84 on todays Top500.**
- **The best price/performance system available for many applications**
- **Philosophy: The cheapest system available which solve your problem in reasonable time**

# The history of Beowulf

- **1993: Perfect conditions for the first Beowulf**
  - **Major CPU performance advance: 80286 $\longrightarrow$ 80386**
  - **DRAM of reasonable costs and densities (8MB)**
  - **Disk drives of several 100MBs available for PC**
  - **Ethernet (10Mbps) controllers and hubs cheap enough**
  - **Linux improved rapidly, and was in a usable state**
  - **PVM widely accepted as a cross–platform message passing model**
- **Clustering was done with commercial UNIX, but the cost was high.**

# History ...

- **NASA Project at Goddard Space Flight Center**
- **Required a single–user system for holding, manipulating and displaying large data sets**
- **Requirement: Cost < $50K, storage: 10GB, performance: 1Gflops**
- **Commercial systems at that time cost 10-20 times too much.**
- **Proposed solution: Use COTS, Linux, and develop some networking software. 1 Giga operations could be achieved.**
- **The Beowulf project started early 1994.**

# History . . .

- **Late 1994: The first Beowulf, named "Wiglaf"**
  - **16 Intel 80486 66MHz CPUs, quickly replaced with 100MHz DX4.**
  - **Performance: 4.6Mflops per node, sustained 74Mflops.**
- **End of 1995: The second Beowulf, "Hrothgar"**
  - **16 Intel Pentium CPUs and Fast Ethernet (100Mbps)**
  - **Performance: 17.5Mflops per node, sustained 180Mflops**
- **More information on these historical systems may be obtain on the original Beowulfs webpage [1]**

# History ...

- **End of 1996: The third generation, "Hyglac" and "Loki"**
  - **16 Intel Pentium pro CPUs, Fast Ethernet**
  - **Achieved performance: 1Gflops sustained**
- **During 1997: 10Gflops achieved**
- **During 1998, a Dec Alpha–based Beowulf, "Avalon", achieved 48Gflops, $\Rightarrow$ 113th on Top500.**
- **Current top500 list: An alpha–based Beowulf "Cplant" at SANDIA National Lab, rank as no. 84. This achieved 232.6Gflops.**

# Who can build a Beowulf

- **The hardware cost is low, so most institutions can afford one**
- **To put it together and run it, experience with Linux or a similar operating system is required**
- **The linux–knowledge you need is similar to what you need to administer linux workstations in a network**
- **Bear in mind that you need resources to manage the cluster in its entire life, not only to set it up**

# How to design a Beowulf

- **What kind of applications do you want to run**
- **What requirements there will be for:**
  - **CPU–performance**
  - **Memory size**
  - **Disk storage size, either temporary or permanent**
  - **Communication *bandwidth* and *latency***
- **What kind of migration or reuseability do you want for your hardware.**

# Beowulfs in more detail

- **To decide what kind of Beowulf you need, you need to know the options. We look briefly into the details of a system.**
- **The main parts of the system are:**
  - **The Beowulf node, a low–cost PC**
  - **The Network**
  - **The Operating System**
  - **The Software**

# The node — today

- **CPU: Intel Pentium, AMD Athlon, Alpha. Single or SMP**
- **Memory: The standard today is SDRAM, but depends on CPU choice. Pentium-4 use expensive RDRAM, Athlon may use DDR-SDRAM**
- **Hard Disk: The choice is cheap IDE or expensive SCSI. SCSI is more reliable, and the performance is better in multiuser environments (server)**
- **Floppy Drive: For initial installation and crash recovery**
- **CD-rom: For installation, one in the entire system is fine.**
- **Mainboard: Could be a bottleneck, so chose a good one! The above choices limit the options.**

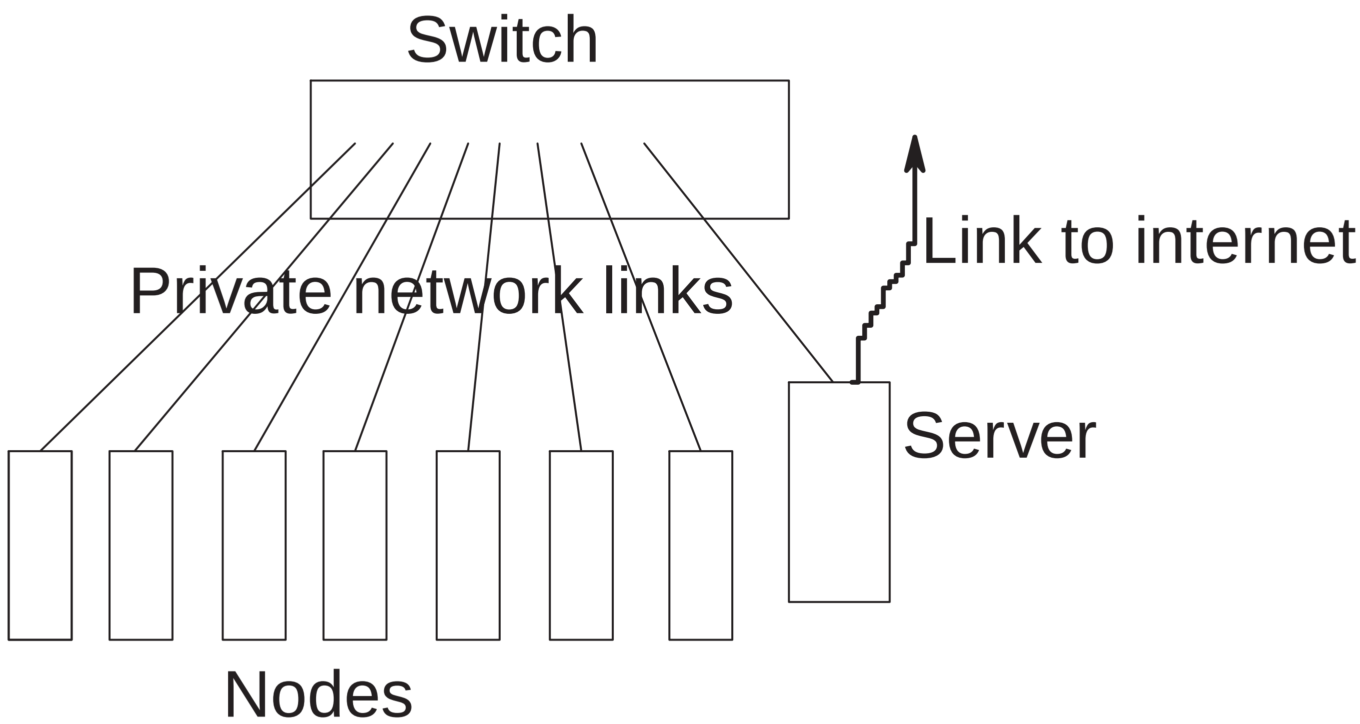
# The node . . .

- **Case: 1u/2u solutions for rack mount, or desktop/tower.**
- **The choice depends on: space, cost and migration/reuseability:**
  - **Rack–mount solutions saves space, but at a higher cost**
  - **Desktop or tower cases are cheap and big**
  - **If you go for the rack, you are stuck with the hardware for its entire life**
  - **If you go for standard cases, you can push the cluster hardware to office or lab use after a short period and update the cluster**
- **The next slide show both solution: Left is “Diplopodus”, our cluster, and right is a small part of “Chiba city” at Argonne National Lab.**

# The Network

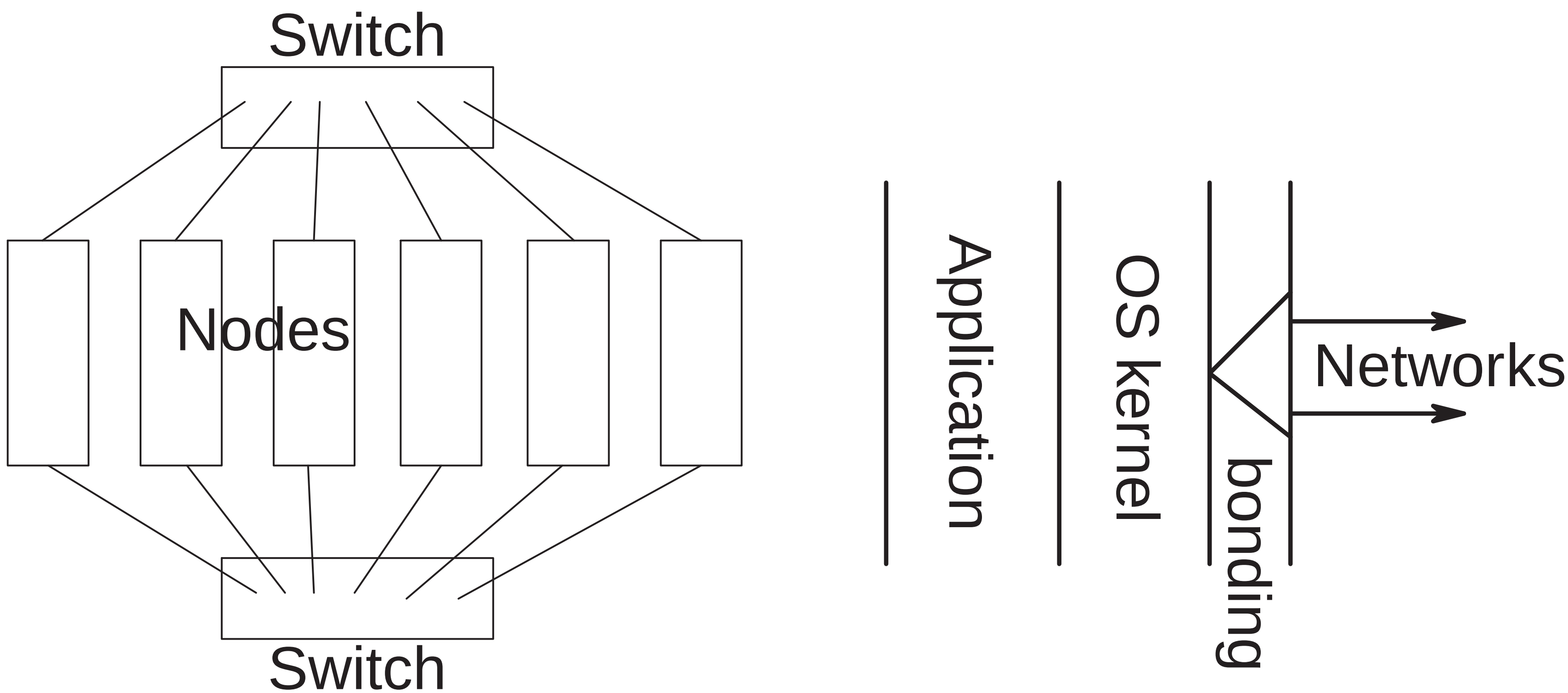
- **Basic: Fast ethernet (100Mbps bandwidth) and Switches. This is the cheapest solution. It is relatively slow and has high *latency*, but sufficient for many applications.**
- **You may have more than one ethernet card per node, and use *channel bonding* to increase the bandwidth.**
- **Other alternatives:**
  - **Myrinet. Up to 1.2Gbps bandwidth, low latency ( $< 9\mu s$). Single vendor. Expensive. See [7]**
  - **SCI. Up to 1.3Gbps bandwidth, low latency ( $< 5\mu s$ ). IEEE standard. Expensive. See e.g. [2]**
  - **Gigabit ethernet. Drop-in replacement for Fast Ethernet. Expensive but could drop in the future.**
- **Combinations of the above**

# A Fast Ethernet network




- **The link between the server and the switch may be one or more gigabit ethernet links.**
- **The internal network of the switch is fast enough to handle full duplex traffic on all ports simultaneously.**

# Channel Bonding




- **The channel bonding support is implemented as a kernel–layer. Multiple physical devices turns into one logical device.**

# Networking . . .

- Ethernet solutions
  - Use the TCP/IP protocol
  - Designed for reliability in wide–area networks (WAN).
  - High latency
  - Hard to get full bandwidth
- Other solutions
  - May use other protocols.
  - Design for high bandwidth, low latency in e.g. clusters
  - Actually not COTS, except Gigabit ethernet.
- The future: Maybe Infiniband [4]. Meant to replace ethernet as a general purpose interconnect.

# Other networking options

- How should a Beowulf relate to the rest of your network?
  - Stand alone system
  - Guarded system
  - Universally accessible system
- Depends on:
  - The need for external resources
  - What kind of user interaction you want
  - Guarded system: Consider Gbps from switch to worldly node and from worldly node to the world.
  - Universally accessible system: Consider Gbps from switch to the world.

# The operating system

**The main options are Linux and *BSDs. The original Beowulf project preferred Linux.**

- **Linux is free — good for the price/performance ratio**
- **It's open source, which is good if you want to fix thing**
- **All the necessary networking support is built in**
- **Support for a large range of devices**
- **Most of the above apply to *BSDs also, maybe except the device support**

# What is Linux

- **Strictly speaking, Linux is an operating system *kernel*:**
  - **Control all devices**
  - **Manages resources**
  - **Schedules user processes**
- **Runs on all Intel x86s, Alpha, PPC, Sparc, Motorola 68k, MIPS, ARM, HP-PA RISC, and more**
- **It is UNIX–like, but not UNIX. It is a rewrite based on published POSIX [10] standards**
- **To do any real work on Linux, you need tools. The kernel together with software from the GNU project and others, form a usable operating system.**

# Software

**Most software are included with Linux *distributions*:**

- **The Linux *kernel***
- **Precompiled and to some extent pre–configured applications and libraries**
- **Installation and management system**

## Software . . .

**Available packages include:**

- **Programming environment and debugging:**
  - **Compilers: g77, gcc, g++, java o.a**
  - **Debugger: Usually gdb, and several frontends for gdb.**
  - **Development tools: make, cvs, editors**
  - **Scripting/interpreted: Perl, Python, Tcl, Awk, Bash, lisp o.a**
- **MPI library and utilities: Tow main players, mpich [6] and lam [5].**
- **PVM: Available from netlib, usually pre–packaged.**
- **X11: Usually only necessary on server/login–node.**

# Software . . .

**Some tools are not included for various reasons. You may download or buy:**

- **Commercial Fortran and C/C++ compilers**
- **Parallel debuggers like Etnus *TotalView***
- **Queue system.**
  - **OpenPBS [8].**
  - **Commercial version; PBS Pro [9], in use on NOTUR resources**

# Rules of thumb

**Are there any general rules for how to design the Beowulf?**

1. **Being up–to–date with the market is a great advantage**
2. **Decide what kind of network you need**
3. **Use Intel–based SMP PCs**
4. **Choose the second last CPU generation**
5. **Choose a reasonable amount of harddisk space.**
6. **Buy RAM for the rest of your money**

- **Alpha CPUs may be considered. Implication on migration**
- **Compilers ?**

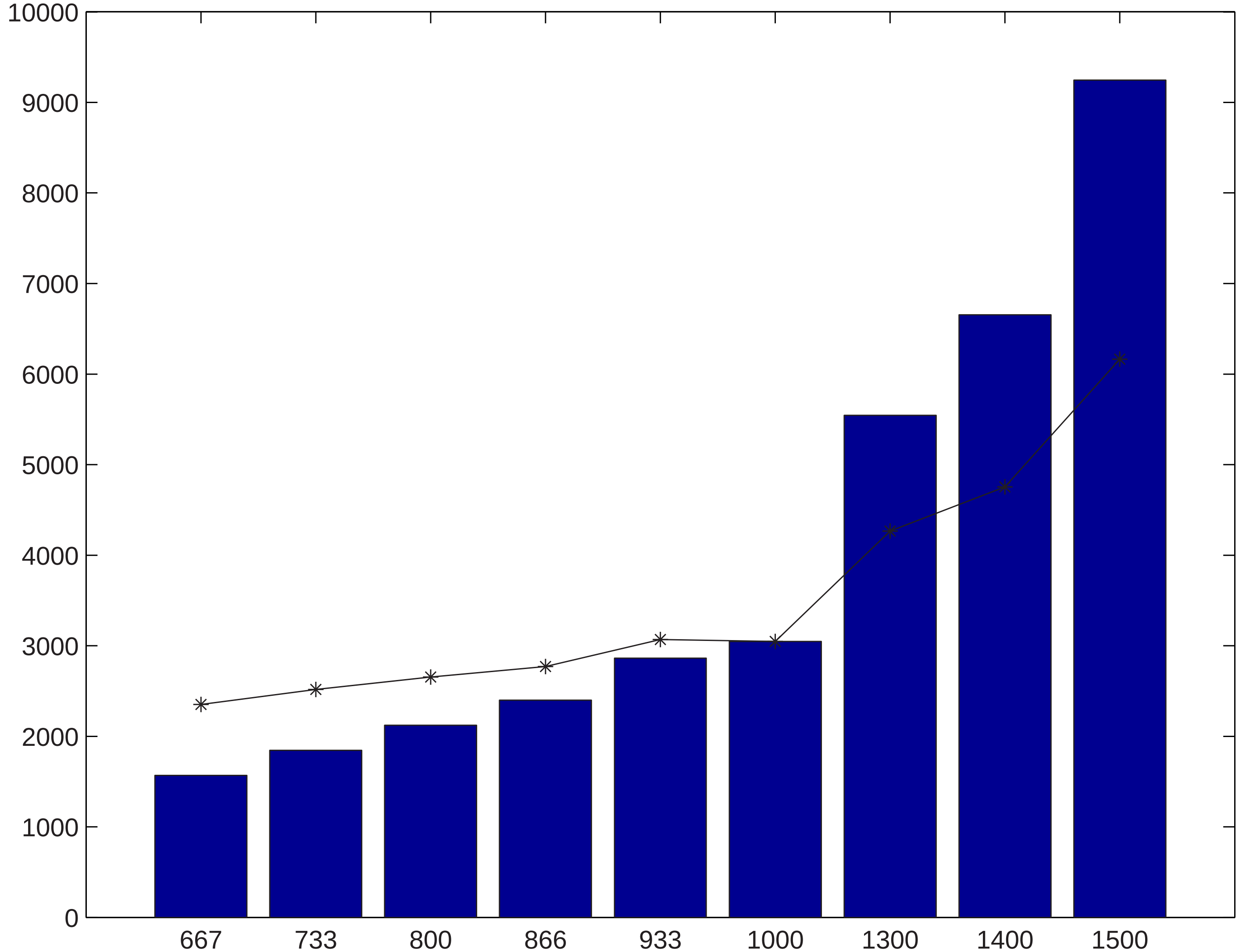



**Figure 1: Bars: price, line: price/GHz**

# Typical prices

- CPU, P-III 1GHz: NOK 3000
- Mainboard, Dual socket FCPGA: NOK 1600
- Harddisk, IDE, 45GB: NOK 1800
- Memory, 256MB ECC SDRAM: NOK 1700
- NIC, 100Mbps: Less than NOK 500
- Case, cables, etc: NOK 1000

# Typical prices . . .

**To complete the cluster, we must add:**

- **Switch: NOK 20.000 per 48 ports**
- **A server PC: NOK 40.000**
- **Maybe some Gigabit networking: NOK 50.000 (2 – 4 cards, 8 ports switch)**

# What Beowulfs are Good For

- **In general, Beowulfs are good for**
  - **High Performance Computing**
  - **High Throughput Computing**
  - **High Availability Computing**
  - **Data Intensive Computing**
- **We focus on Beowulfs with fast ethernet network**
- **Suitable for embarrassingly parallel to moderately coupled problems — that is much computing, lesser communication**
- **Our experiences are mostly from PDE problems solved with Finite Element Methods**

# Experiments

**Using our cluster, “Diplopodus” we have done some experiments:**

- **The “Top500” test and simple network tests**
- **3D nonlinear acoustic field simulation**
- **Incompressible Navier–Stokes simulation**
- **3D nonlinear water wave simulation**

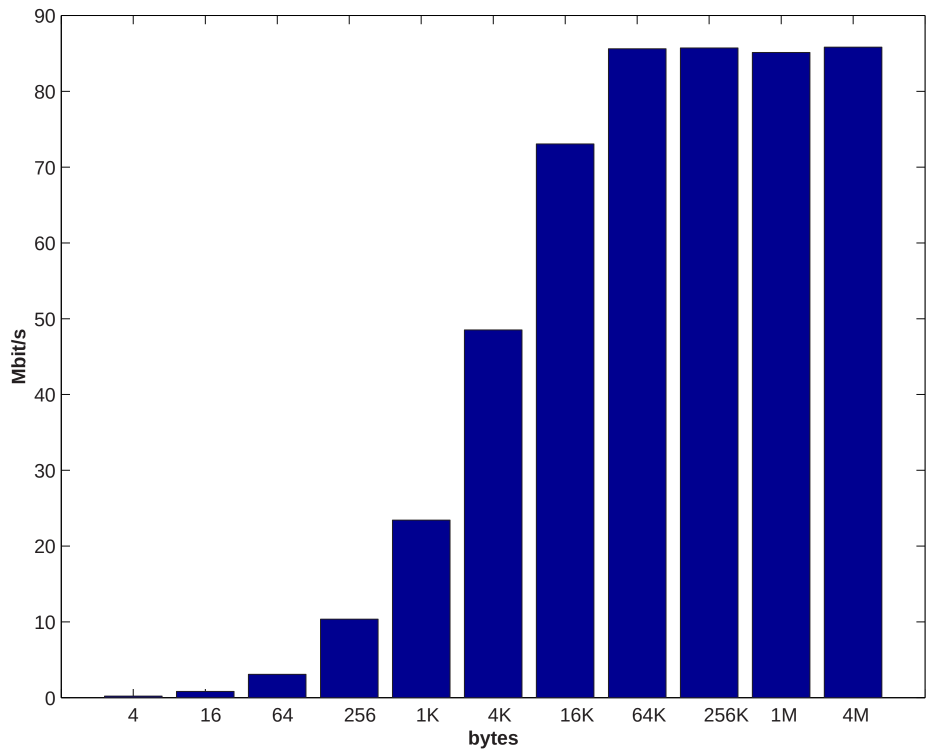
**We have also done a few comparisons with Parallabs Origin2000**

# Diplopodus - our Linux Cluster

- 24 dual 500MHz Pentium-III computing nodes
- Each node is equipped with 512MB of memory (12GB in total)
- Standard 100Mbit/s ethernet network is used as interconnect
- Cicso Catalyst 2926 switch, 3com905B nic's
- A separate single–CPU computer is used as front–end and file–server.
- Total cost: NOK 500,000 ( US$ 60,000) at the beginning of 2000
  - Debian GNU/Linux, kernel 2.2.14
  - PBS queue system, MPI library Mpich version 1.1.2

# Performance measurements

- **Diplopodus scored 9.27GFlops in the Top500–test from netlib.**
- **Performance numbers: Latency $\approx 150\mu s$, memory bandwidth 85Mbit/s**



# SGI–Cray Origin2000

- **128 R10000 195MHz MIPS processors**
- **Scored 40.25GFlops in the Top500–test**
  - **For comparison, we assume 48 CPUs will score at least 15.1GFlops**
- **Performance numbers: Latency $\approx 17\mu s$, memory bandwidth 860Mbit/s**
- **At the time these experiments where done, this was the best computer in Norway**

# 3D nonlinear acoustic fields

- **a nonlinear model:**

$$\nabla^2\varphi - \frac{1}{c^2}\frac{\partial^2\varphi}{\partial t^2} + \frac{1}{c^2}\frac{\partial}{\partial t}\left[(\nabla\varphi)^2 + \frac{B/A}{2c^2}\left(\frac{\partial\varphi}{\partial t}\right)^2 + b\nabla^2\varphi\right] = 0, \quad (1)$$

$$p - p_0 = \rho_0\frac{\partial\varphi}{\partial t}. \quad (2)$$

- **Initial condition**

$$\varphi(x,0) = \varphi_0, \quad x \in \Omega, \quad (3)$$

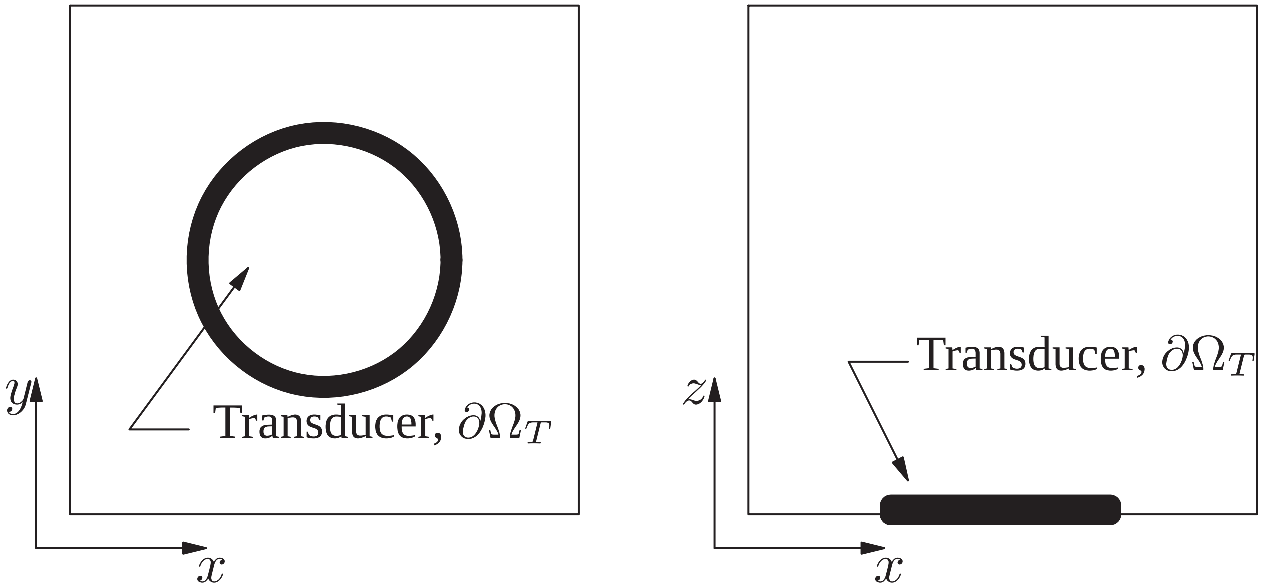
$$\frac{\partial\varphi}{\partial t}(x,0) = 0, \quad x \in \Omega. \quad (4)$$

- **Boundary: Partial prescribed and partial non–reflecting**

# Numerical method

- **Spatial domain: Galerkin finite element method**
- **Temporal derivatives: Finite difference approximations**
- **Nonlinear system: Solved with Newton iterations**
- **Starting guess: Solution of a linearised model**
- **Linear system: Solved with Krylov subspace method**
- **Implemented using Diffpack**

# Experiment




- $\Omega = [-0.004, 0.004]^2 \times [0, 0.008]$ **and transducer radius** $r = 0.002$

## Measurements

- We discretize the spatial domain by tri–quadratic elements
  - General mesh partition approach, METIS [3]
- Total number of degrees of freedom is 1.030,301
- Bi–Conjugate–Gradient method is used for linear systems

# Measurements for nonlinear model cont.

| | Origin2000 | | Linux-cluster | |
|---|---|---|---|---|
| $P$ | CPU | $\eta$ | CPU | $\eta$ |
| 2 | 8670.8 | N/A | 6681.5 | N/A |
| 4 | 4628.5 | 3.75 | 3545.9 | 3.77 |
| 8 | 2404.2 | 7.21 | 1881.1 | 7.10 |
| 16 | 1325.6 | 13.0 | 953.89 | 14.0 |
| 24 | 1043.7 | 16.6 | 681.77 | 19.6 |
| 32 | 725.23 | 23.9 | 563.54 | 23.7 |
| 48 | 557.61 | 31.1 | 673.77 | 19.8 |

# Measurements for nonlinear model cont.

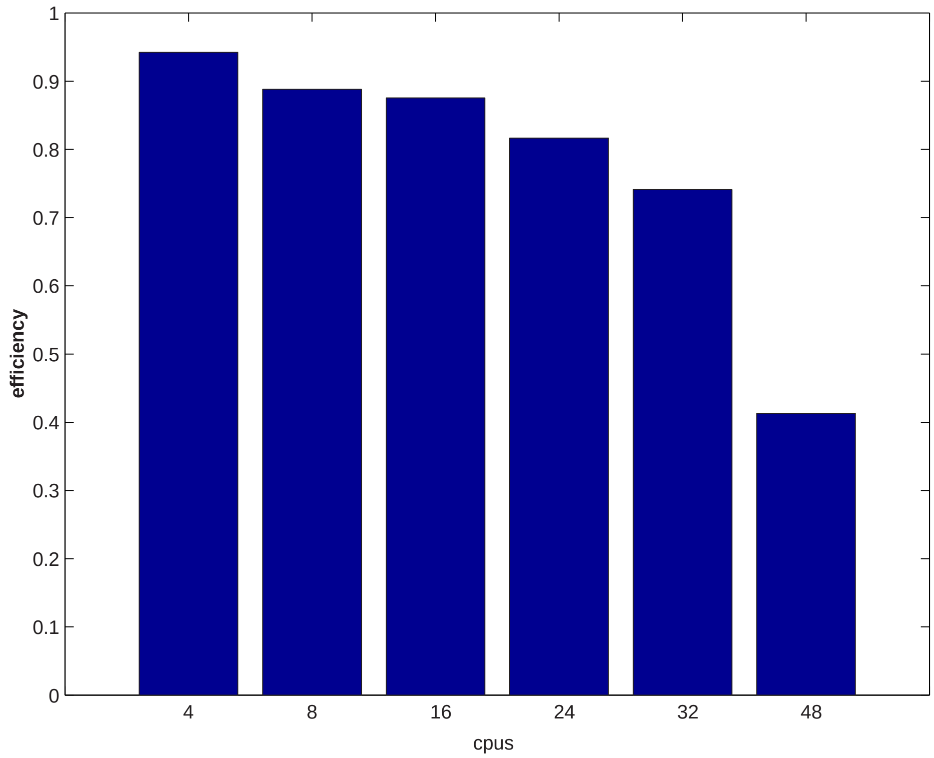



**Figure 2: Efficiency on Linux–cluster**

## Measurements for nonlinear model cont.

- **Implicit scheme: all-to–all communication required**
- **Node–overlap between neighbouring processes**
- **Load–imbalance:**
  - **Number of elements in each submesh may vary**
  - **Number of process–neighbours may vary**

# Incompressible Navier–Stokes

$$\varrho\left(\frac{\partial\mathbf{v}}{\partial t}+\mathbf{v}\cdot\nabla v\right)=-\nabla p+\mu\nabla^2\mathbf{v}+\varrho\mathbf{b}, \tag{5}$$

$$\nabla\cdot\mathbf{v}=0, \tag{6}$$

- **velocity field $\mathrm{v}$, pressure $p$, external body forces $\mathrm{b}$, density $\varrho$, and viscosity $\mu$**
- **Fast finite element method based on velocity–correction strategy**
- **Implemented in Diffpack.**
- **Refer to [11] for details on Parallel Navier–Stokes solvers**

# Measurements

- **2D simulation with 40,401 grid points**
- **Conjugate Gradient method for solving linear equations**

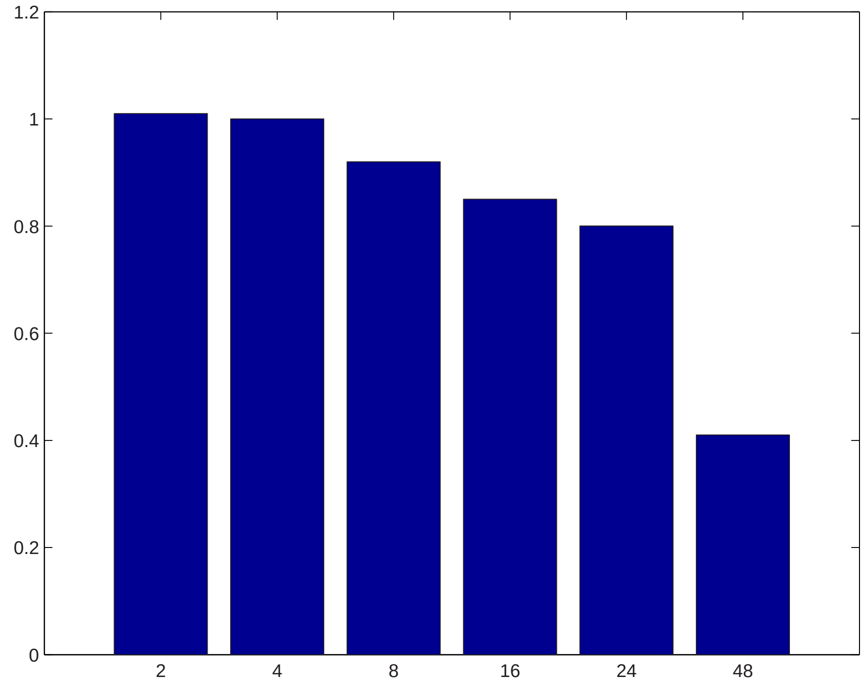



**Figure 3: Efficiency on Linux–cluster**

# 3D nonlinear water wave

$$-\nabla^2\varphi = 0 \quad \text{in the water volume} \tag{7}$$

$$\eta_t + \varphi_x\eta_x + \varphi_y\eta_y - \varphi_z = 0 \quad \text{on the free surface} \tag{8}$$

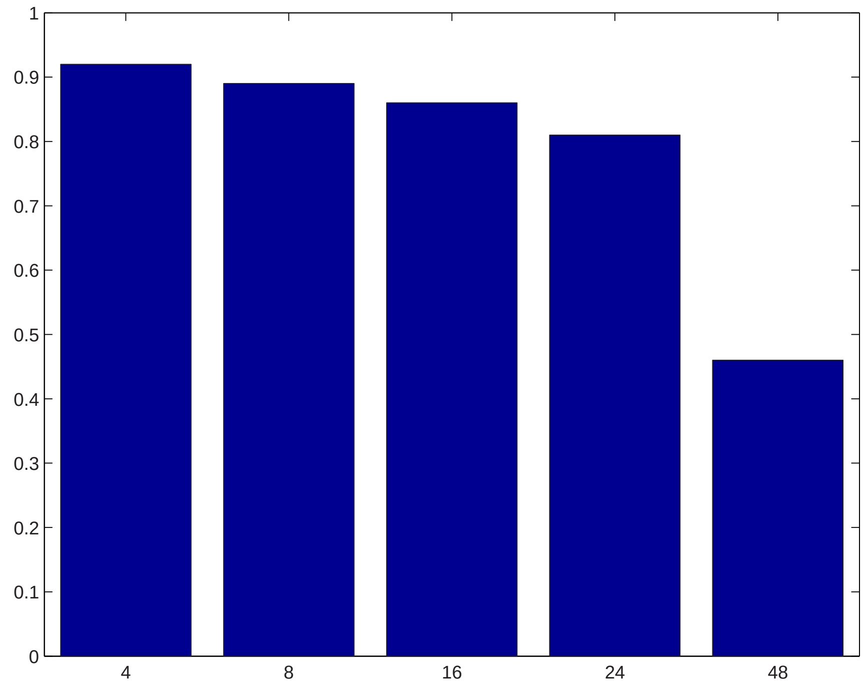
$$\varphi_t + \frac{1}{2}(\varphi_x^2 + \varphi_y^2 + \varphi_z^2) + g\eta = 0 \quad \text{on the free surface} \tag{9}$$

$$\frac{\partial\varphi}{\partial n} = 0 \quad \text{on solid boundaries} \tag{10}$$

- **Primary unknowns: velocity potential $\varphi$, free surface elevation $\eta$**
- **Numerical scheme: update $\eta$ explicitly, solving $\varphi$ by implicitly**

# Measurements

- **3D simulation on a $49 \times 49 \times 41$ grid**
- **Method: One parallel DD iteration as preconditioner for global CG**
- **Sub domains: One multigrid V–cycle**



# References

[1] The Original Beowulf's webpage. http://www.beowulf.org/gsfc-hw.html.

[2] The Dolphin Interconnect website. http://www.dolphinics.com.

[3] V. Kumar G. Karypis. Metis: Unstructured graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minnesota, Minneapolis/St. Paul, 1995.

[4] The Infiniband website. http://www.infinibandta.org.

[5] The LAM website. http://www.mpi.nd.edu/lam/.

[6] The MPICH website. http://www-unix.mcs.anl.gov/mpi/mpich/.

[7] The Myricom website. http://www.myricom.com.

[8] The OpenPBS website. http://pbs.mrj.com/.

[9] The PBS Pro website. http://www.pbspro.com/.

[10] The Portable Application Standards Committee. http://www.pasc.org/.

[11] Otto Munthe Xing Cal, Hans P. Langtangen. An Object-Oriented Software Framework for Building Parallel Navier-Stokes Solvers. In *Proceedings of the Prallel CFD'99 conference*, 1999.