Zen and the Art of Beowulf Clusters

Robert G. Brown
Duke University Physics Department
rgb@phy.duke.edu

November 8, 2006

Principles of Zen

- **Zen is a blend of Taoism and Buddhism with an emphasis on *meditation* (sitting zazen).**
- **The purpose of pursuing a zen path is to seek *Enlightenment*.**
- **Enlightenment comes from quieting the mind to live in the *now*.**
- **With a quiet mind focussed on the present, one can obtain great insight.**

Principles of Beowulf Clusters

- **A Beowulf Cluster is a blend of Commodity Off The Shelf (COTS) computer hardware and a high quality network.**
- **The purpose of building a Beowulf Cluster is to perform *parallel computations*.**
- **Parallel computations can complete work in much less time than serial computations.**
- **With the rapid results obtained from parallel computations, the quiet mind can obtain great insight.**

(an obvious connection, right...? :-)

How to build a Parallel Cluster

- **Get a pile of PC's.**
- **Install Linux**
  - **Fedora Core via PXE from a server (VERY efficient, see fedora.redhat.com).**
  - **Warewulf (www.warewulf.org).**
  - **Debian (www.debian.org).**
  - **Scyld (www.scyld.com)**
  - **It is possible and sometimes advantageous to run linux diskless in a small cluster.**
- **Install parallel computation support as needed – network, PVM, MPI, libraries.**
- **Program and run parallel programs.**

There are a few details, of course. To design an *optimal* cluster for any given task one has to understand parallel computations and how to match them to cluster design.

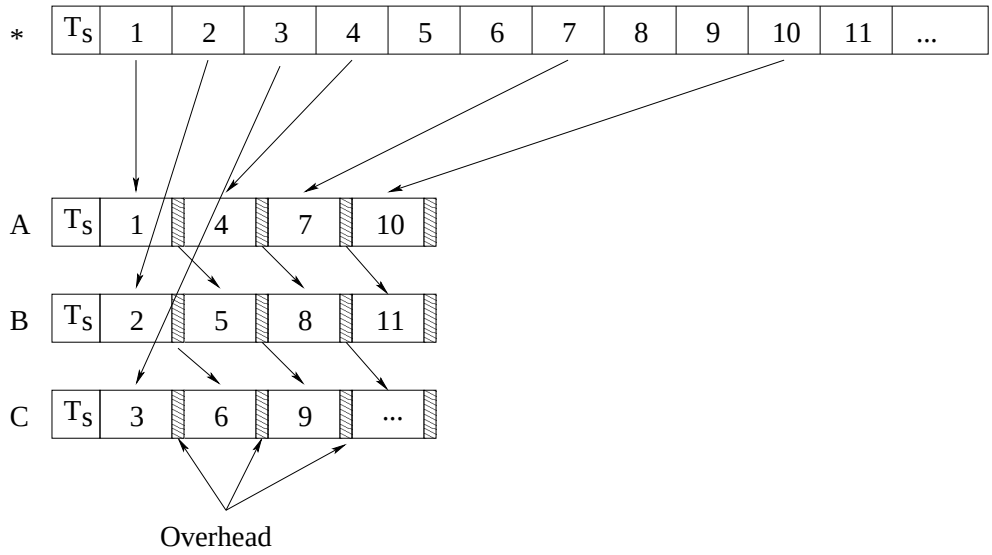


Figure 1: Parallelization of a Task

So what ARE Parallel Computations?

- **"Tasks" typically have both serial and parallel components.**
- **Parallel subtask completion time under ideal circumstances scales like $1/N$ where N is the number of parallel tasks undertaken (on e.g. different processors) at the same time. "Many hands make light work".**
- **Parallel subtasks often (but not always) require interprocessors communications (IPCs) between the subtasks. This communication time adds to to the total and can take more or less time than the work itself.**
- **All this is made formal in Amdahl's Law and quantitatively corrected in books on parallel computation.**

Amdahl's Law

The speedup S experienced running a task on P processors is less than or equal to:

$$S \leq \frac{(\mathrm{T_s} + \mathrm{T_p})}{(\mathrm{T_s} + (\mathrm{T_p}/\mathrm{P}))} \tag{1}$$

where T_s is the time program spends doing "serial work" and T_p is the time spend doing "parallelizable work" split up on P processors.

Limiting result, not horribly useful quantitatively except to tell you when there is *no point* in parallelizing something. Can do much better.

For example, we can account for the time spent communicating between processors, the time spent setting things up, and changes in the times to perform various tasks with different algorithms. Defining things like:

T_s The original single-processor serial time.

T_{is} The (average) additional *serial* time spent doing things like IPC's, setup, and so forth, per processor, in all parallelized tasks.

T_p The original single-processor parallizable time.

T_{ip} The (average) additional time spent by each processor doing just the setup and work that it does in parallel. This may well include idle time, which is often important enough to be accounted for separately.

we can obtain improved estimates of the speedup:

$$T_{\rm tot}(P) = T_s + P * T_{is} + T_p/P + T_{ip}. \tag{2}$$

or

$$S \approx \frac{T_s + T_p}{T_s + P * T_{is} + T_p/P + T_{ip}}. \tag{3}$$

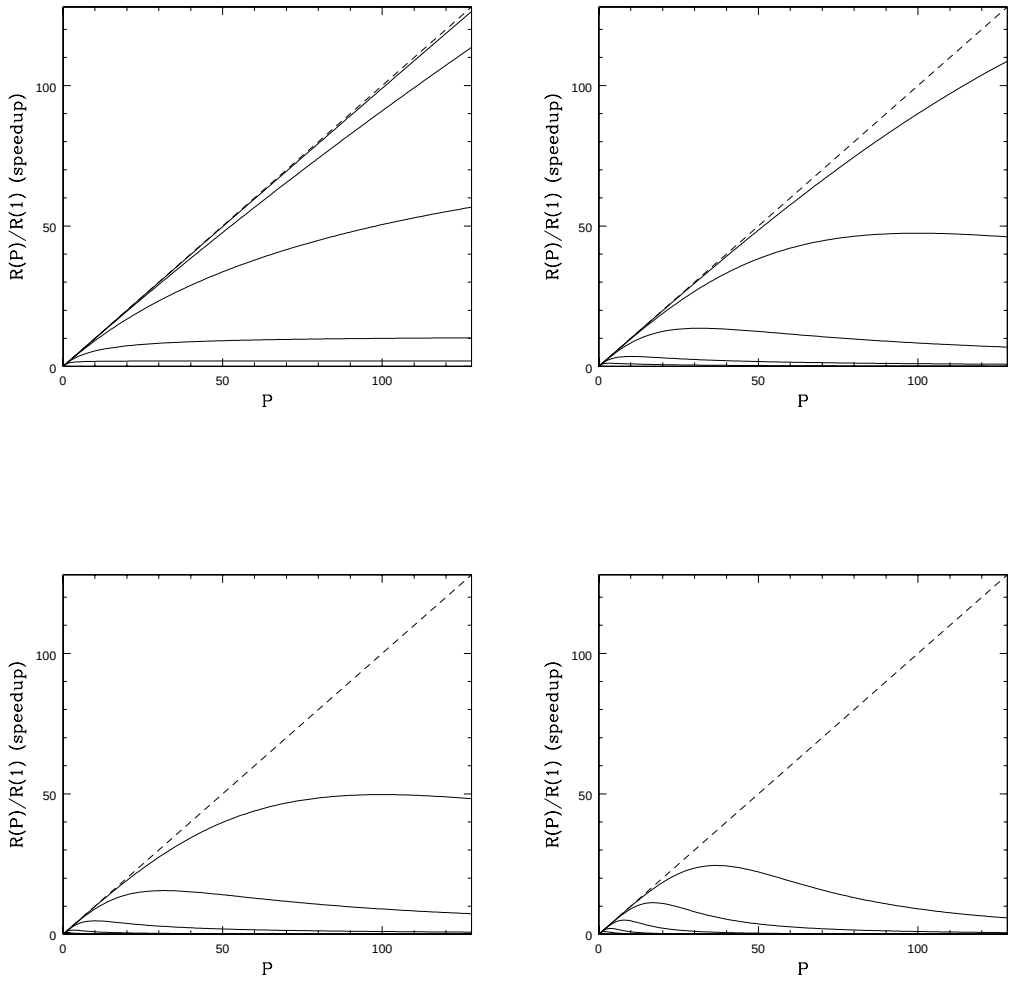
All You Need to Know About Code Granularity

- **If $t_{\text{computation}} \gg t_{\text{communication}}$, (lots of work, little communication) coarse grained. P completely independent jobs are "embarrassingly parallel" (EP). (e.g. Monte Carlo, data field explorations.)**
- **If $t_{\text{computation}} > t_{\text{communication}}$ (but not tremendously so) medium grained. (e.g. problems on a lattice (where the lattice is partitioned among nodes with short range communications), lattice gauge theory.)**
- **If $t_{\text{computation}} = t_{\text{communication}}$ or less fine grained. (e.g. – Cosmology, molecular dynamics with long range interactions, hydrodynamics, computational fluid dynamics.)**

Granularity typically is somewhat controllable. Network speed and latency, scaling of computation to communication as a function of problem size, CPU/memory speed, program organization all control variables.

Fine grained tasks are "bad" for scaling to many nodes N. Coarse grained tasks are "good".

Beware nonlinearities! CPU/cache/memory/disk bottlenecks can create "superlinear speedup" and violate Amdahl's Law!



Huh? Whaddideesay?

In all the figures below, $T_s = 10$ (which sets our basic scale, if you like) and $T_p = 10, 100, 1000, 10000, 100000$. In the first three figures we just vary $T_{is} = 0, 1, 10$ for $T_{ip} = 1$ (fixed). Finally, the last figure is $T_{is} = 1$, but this time with a *quadratic* dependence $P^2 * T_{is}$.

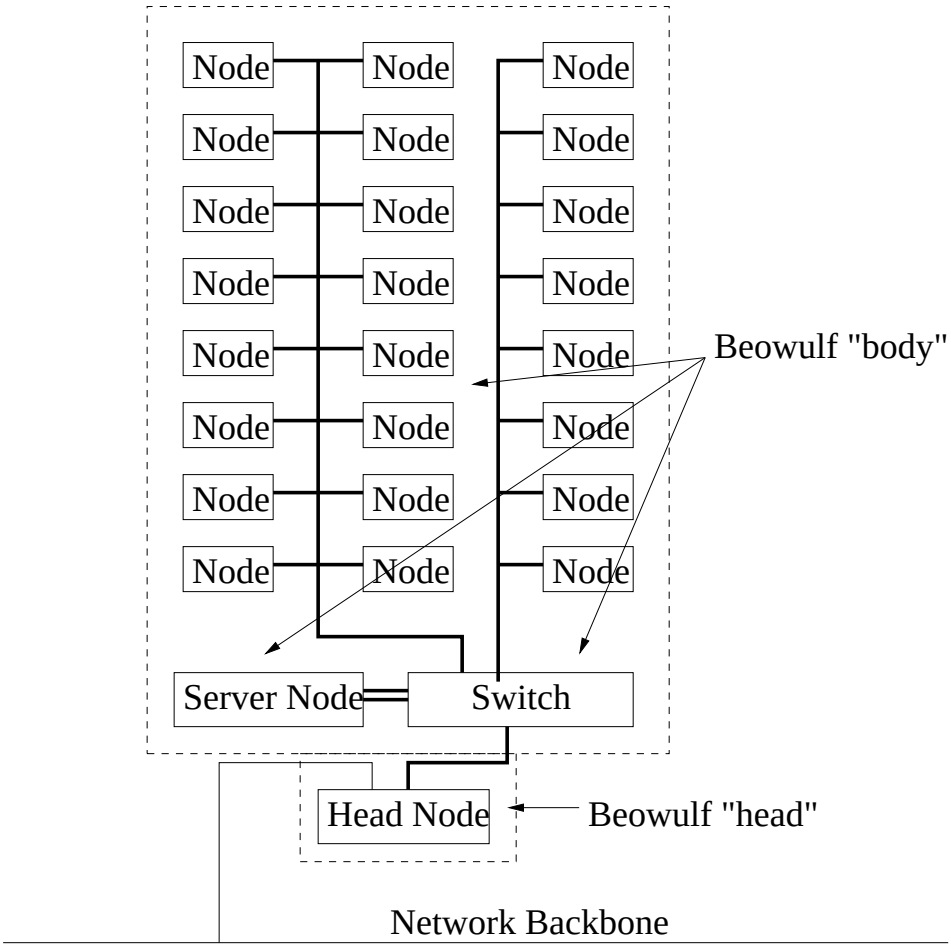
Designs: NOW/COW/Beowulf

Goal is optimizing overall performance per dollar. The following are appropriate for increasing fineness of program granularity:

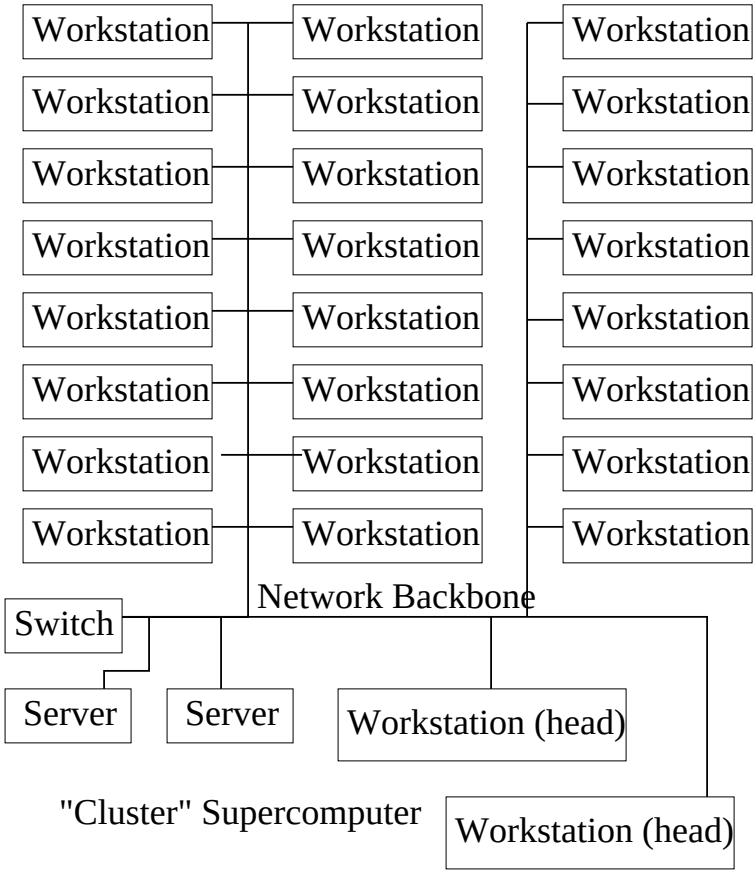
- **GRID (Network of clusters, supercluster). SGE or shell-level tools. EP tasks, primarily.**
- **NOW (Network of Workstations) + e.g. Mosix, master/slave PVM, MPI, shell-level tools or perl scripts permit double usage of all CPUs.**
- **COW (Cluster of Workstations) same as NOW but protects the network a bit and isolates the compute resource from interactive humans and GUIs. Most common Duke design?**
- **Beowulf (dedicated, single headed COW) + Scyld/clustermatic and PVM/MPI. A totally isolated COW with (usually) a private network, custom OS, and a single head.**

These are suitable for increasingly fine granularity, at increasing cost and decreasing general purpose utility.

Schematics for the general designs follow, first a "true beowulf" and then a workstation cluster.



A True Beowulf



A Workstation Cluster

Node Design and Cost

The following are some possible node configurations and prices:

- **Dimension C521 AMD Athlon 64 X2 Dual-Core 5000+ (2GB) with 10/100 ethernet and 3 year onsite service = $900**
- **PowerEdge SC1435 Dual Dual Core AMD Opteron 2210; 1.8GHz,2X1MB Cache,1Ghz HyperTransport, 4 GB, (1U form factor), dual Gigabit ethernet, 3 year Standard Support = $2900**
- **High end network interconnects range in price from $800 to $1500 per PC, plus switch hardware (where gigabit ethernet is not a high end interconnect and for small clusters is actually cheap).**
- **In addition, a cluster is likely to need: Shelving or a rack, network switch, cabling, a KVM arrangement, a system configured as a "server". A pro-grade cluster is likely to cost around $1000/CPU (and up).**

The cheapest barebones clusters for learning and experimentation can cost surprisingly little. On www.clustermonkey.net, for example, you can find an article on a "value cluster" – an 8 node cluster that cost $2500 *total* in 2005. This is easily within the reach of individuals, clubs, or small schools.

Turnkey Clusters

Turnkey clusters can make sense if you are building a very specialized cluster and need help designing and installing it. A turnkey integrator will typically resell the hardware components to you pretty much at standard retail marked up to cover their "integration fee" for designing the cluster, installing the clusterware on it, and so forth. This ends up being anywhere from a 20% markup of OTC prices on up.

Cluster Networks

- **Switched 100BT (old standard, nearly obsolete).**
- **Switched 1000BT (emerging standard). Good bandwidth. Relatively poor latency. Cheap.**
- **Infiniband. Excellent bandwidth and latency. Expensive**
- **10 GB Ethernet. Excellent bandwidth and latency. Expensive.**
- **Dolphinics. Excellent bandwidth and latency. Expensive.**
- **Etc. (You get the idea – bandwidth and latency beyond ethernet are expensive).**

Parallel Program Support

- **MPI (Message Passing Interface). API + library for writing portable parallel programs with a message passing interface for IPC's. Several versions available, LAM in Red Hat and on repository.**
- **PVM (Parallel Virtual Machine). API + library for writing portable parallel programs that run across networks. My personal favorite API (written as open source effort from beginning, not by a consortium of massively parallel supercomputer vendors under governmental threat).**
- **Raw Sockets (yeah!)**
- **Mosix**
- **Remote Shells (e.g. rsh, ssh)**
- **Miscellaneous: Monitors, batch/queue systems, GUI's, scripts, bproc, scyld, cod, more...**

Simple Example: xep (PVM Mandelbrot Set)

- **Mandelbrot set is iterated map that either "escapes" or doesn't.**
- **Colors mapped to steps until escape makes pretty pictures.**
- **Self-referential, fractal, infinitely fine structure as we rubber-band down into set.**
- **Easily parallelizeable (coarse grained parallel).**

On a good day, this will work as a demo...

Physical Infrastructure Requirements

- **Space:** **Shelfmount > 1 ft^2/node, Rackmount ≈ 0.5 ft^2/node, blades "different". 1-2 CPUs/node, maybe UPS. Heavy! Strong floors!**
- **Power:** **Guestimate 100W/CPU, better to measure. Special wiring requirements for switching power supplies! Overwire!**
- **A/C:** **All power IN turns to heat and must come OUT. 1 Ton of A/C removes ~ 3500 W. Again, need surplus to keep room COOL, plus specific delivery/circulation/return design. Thermal kill?**
- **Network:** **Cable trays, patch panels, backbone ports on copper or fiber. BOTH local network(s) for cluster AND connection to departmental LAN/WAN.**
- **Etc:** **Decent lighting. Work bench and tools! Chairs and carts. Monitor, keyboard. KVM switch? Jackets and ear protectors or noise-reduction headphones plus music. Phone. X10/temp/humidity/intrusion monitoring?**

Physical Infrastructure Costs

- **Anywhere from $400 to $5000 per node straight compute hardware cost. Typically $1000/CPU "reasonable" memory non-bleeding edge clock config.**
- **Anywhere from $30 to $1000 (or more?) per node for network. In some designs network will cost more than CPU!**
- **Amortized renovation costs. For example, $100,000 for space to hold 100 nodes, over 10 years, is ballpark $150/node/year (including cost of money).**
- **Recurring costs. $1 W/year for power/cooling, maybe rent or physical space maintenance. 100 nodes at 100 W each cost at least $10,000 year to run 24x7 for the year!**

Note well that recurring costs for operating a node can compete with the cost of the node! This favors getting relatively expensive nodes and dumping nodes quickly when obsolete!

Administrative Infrastructure

- **Installation:** Min: 15 min TOTAL/node (unpacking it and racking it plus e.g. kickstart. Max: Any nightmarish thing you can imagine (prototype)!
- **Operational maintenance:** Min: 1 hour per node per year (OS upgrades, fixing "rare" hardware failures, new software). Presumes automation of nearly everything (yum) and preexisting LAN (with accounts, fileservers, etc.). Max: Any nightmarish thing you can imagine.
- LARGE Monitoring: Min: 20-30 minutes/day per cluster Presumes syslog-ng, monitoring tools like ganglia or xmlsysd/wulfstat, alert users. Max: A couple hours a day.
- LARGE User support: Min: 0 minutes a day if you have smart users and a sucker rod handy to school the lazy. Max: Arrrrrggghh! (*whack!* *whack!*)

In summary, Min: $\sim$ 1 hour a day, on average, for a "good" 100+ node cluster; Max: full time job and then some for a "bad" cluster (depending on luck, hardware reliability, your general admin skills, your cluster admin skills, user support requirements, and the availability of cluster expertise in a distributed support environment).

Conclusions

Total Cost of Ownership (TCO) can range from:

- **$1000 (node) + $300 (power and A/C) + $100 (3 hours sysadmin time) = $1400 per node for a three year expected lifetime; to**
- **$3000 (node) + $600 (power and A/C) + $450 (amortized share of expensive renovation) + $800 (24 hours sysadmin time) $150 (amortized share of four post smoked glass rack, UPS, = $5000 for the same three year lifetime.**

Wide range, provokes TCO fistfights in bars.

Still, beowulfish clusters often yield staggering productivity efficiency. Generally 3-10x more cost/benefit than comparable power "big iron". SO, literally everybody is buying or building them.

References and Resources

- **http://www.beowulf.org**
- **http://www.clustermonkey.net**
- **http://www.phy.duke.edu/∼rgb/Beowulf/beowulf.php (see especially my book on cluster engineering).**
- **http://www.phy.duke.edu/∼rgb/Beowulf/zen_of_clust (this talk).**
- **http://www.phy.duke.edu/brahma/ (lots of resources)**
- **"How To Build a Beowulf", by Sterling, Becker, et. al.**
- **Online book on designing parallel programs by Ian Foster at Argonne National Labs, http://www-unix.mcs.anl.gov/dbpp/**
- **"Highly Parallel Computing", by Amalsi and Gottlieb.**

Conclusion and Example

- **Cluster computing is *easy* to understand.**
- **Cluster computers are *cheap* (compared to nearly all HPC alternatives).**
- **Cluster computers allow one to perform supercomputing at home, at your college, anywhere.**
- **In the best tradition of a Zen master, I offer you the following Koan to mediate upon that can lead you to enlightment...**

Koan: Zen Wulf

Computers are linked
Fleeting packets join their power
Data Satori!