Introduction to the BEOWULF Design

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

Parallel Computation

- "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:

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

or

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

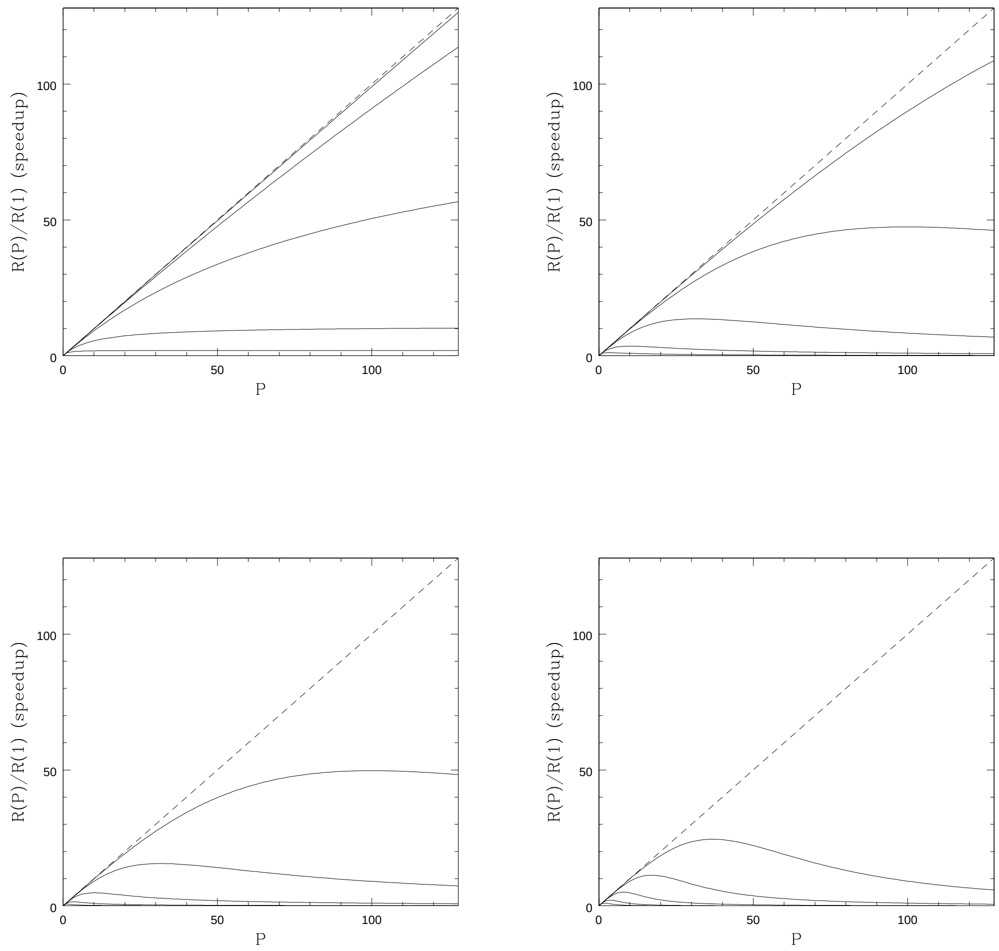
All You Need to Know About Code Granularity

- **If $t_{\rm computation} \gg t_{\rm communication}$, (lots of work, little communication) coarse grained. P completely independent jobs are "embarrassingly parallel" (EP). (e.g. Monte Carlo, data field explorations.)**
- **If $t_{\rm computation} > t_{\rm communication}$ (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 $t_{\rm computation} = t_{\rm communication}$ 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

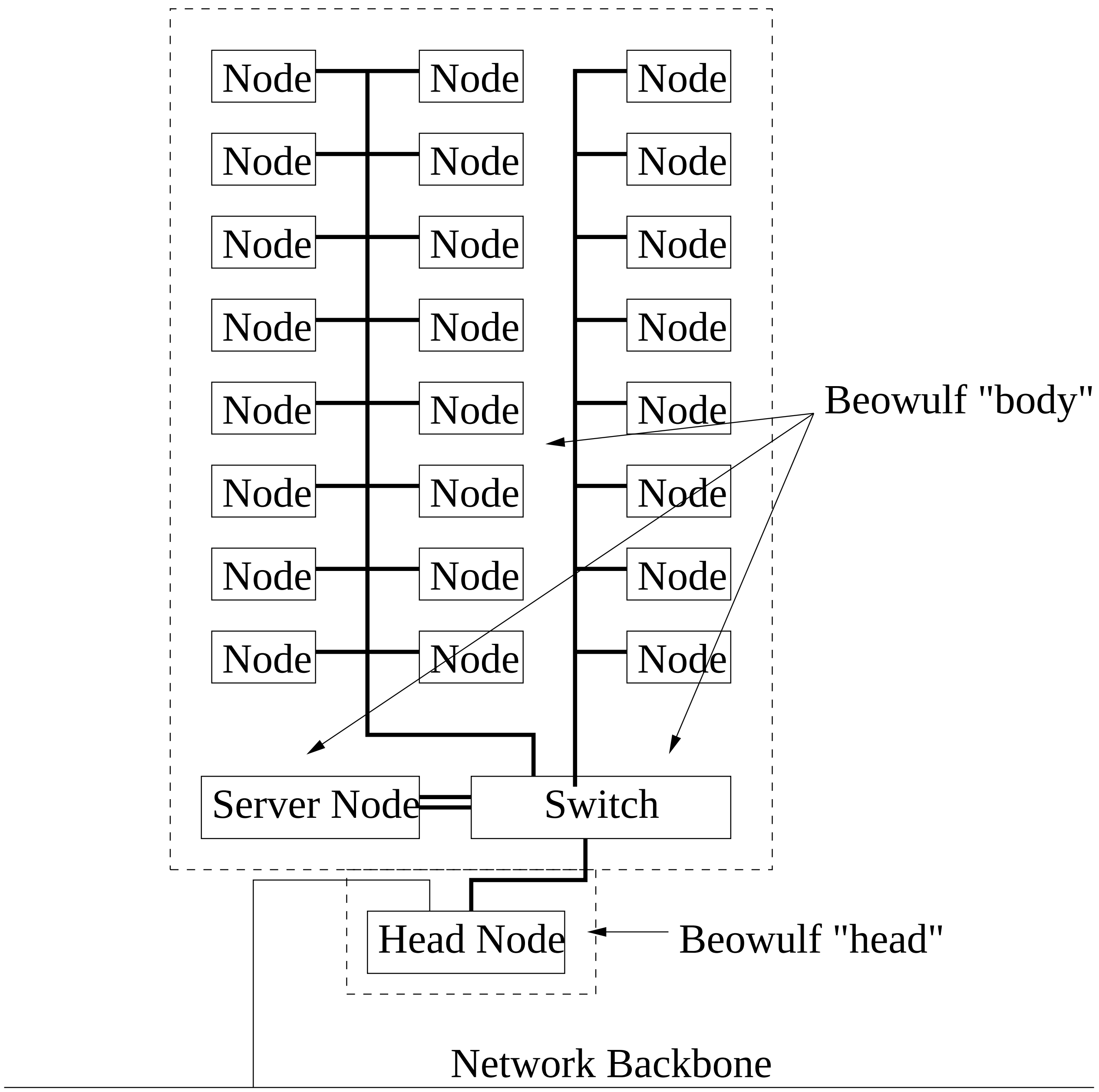
Goal is optimizing overall performance per dollar. The following are appropriate for increasing fineness of program granularity:

- 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.

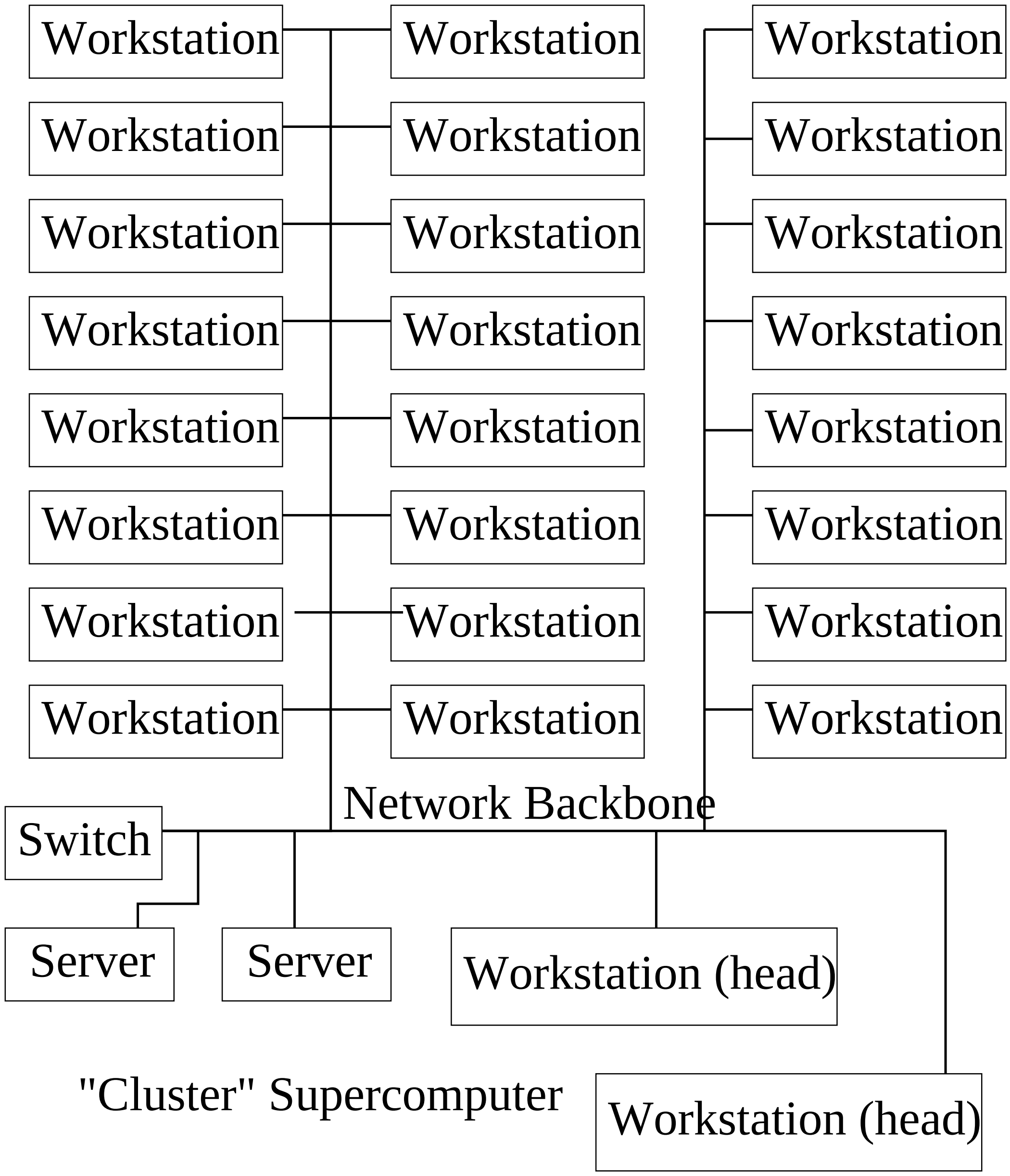
These are suitable for increasingly fine granularity, at increasing cost and decreasing general purpose utility.

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

Examples (Intrex-based estimate, YMMV):

- Tower case ($60), P4 motherboard with onboard PXE 100BT NIC and video($120), 2.4 GH P4 ($200), 1 GB expensive DDR ($230), "small" hard disk ($100) = $710.
- Same, in 1U or 2U rackmount case (add $200) = $910.
- Same in rackmount case, add 64 bit 1000BT NIC ($120) = $1030.
- Add $100 for three year service = $1130.
- Dual P4's in 2U rackmount case (add $330 for motherboard, $380 to go to 2 2.4 GHz P4 Xeons) = $1840.
- Same with Myrinet, 3 GB memory, large hard disk, fastest CPUs ≈ $4000 and rising...

Dells will run perhaps 10% higher plus shipping. Pricewatch min might save you 20%, BUT see later notes on "Administrative Infrastructure"! Per node pricing may have to absorb cost of rack(s) or heavy duty steel shelving from Home Depot, screws, cable ties, and so forth, so estimate 10% more than your base minimum.

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.

At Duke it will only VERY RARELY make sense to get a turnkey cluster. That is because:

- **We have our own, aggressively maintained, updated and automated linux repository thanks to the Sethbot (clap, cheer, whistle).**
- **We have enough local expertise that one can usually get "as good" a cluster setup for many EP to medium grained parallel tasks using this repository plus some on-campus (free) consultation.**
- **It isn't that hard...**

Cluster Networks

- Switched 100BT (standard, should be used ALSO in most configurations anyway).
- Switched 1000BT. Good bandwidth. Relatively poor latency. Relatively cheap in SMALL switches, more expensive for large switches.
- Myrinet. Excellent bandwidth. Excellent latency. Expensive as hell, but cheaper than 1000BT for large networks (where big 1000BT switches become VERY costly).
- Etc. There are more. I'm ignoring them out of sheer ignorance.

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.phy.duke.edu/~rgb/beowulf.php (see especially my book on cluster engineering).
- http://www.phy.duke.edu/~rgb/beowulf_intro_2003 (this talk).
- http://www.phy.duke.edu/brahma/ (lots of resources, including images of this talk)
- http://www.beowulf_underground.org/ (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.