Designing a Cluster for a Small Research Group

Jim Phillips, John Stone, Tim Skirvin

Low-cost Linux Clusters for Biomolecular Simulations Using NAMD

Outline

- Why and why not clusters?
- Consider your…
  - Users
  - Application
  - Budget
  - Environment
  - Hardware
  - System Software
- Case study: local NAMD cluster

Why Clusters?

- Cheap alternative to “big iron”
- Local development platform for “big iron” code
- Built to task (buy only what you need)
- Built from COTS components
- Runs COTS software (Linux/MPI/PVM)
- Lower yearly maintenance costs
- Single failure does not take down entire facility
- Re-deploy as desktops or “throw away”

Why Not Clusters?

- Non-parallelizable or tightly coupled application
- Cost of porting large existing codebase too high
- No source code for application
- No local expertise (don't know Unix)
- No vendor hand holding
- Massive I/O or memory requirements

Know Your Users

- Who are you building the cluster for?
  - Yourself and two grad students?
  - Yourself and twenty grad students?
  - Your entire department or university?
- Are they clueless, competitive, or malicious?
- How will you to allocate resources among them?
- Will they expect an existing infrastructure?
- How well will they tolerate system downtimes?

Your Users' Goals

- Do you want increased throughput?
  - Large number of queued serial jobs.
  - Standard applications, no changes needed.
- Or decreased turnaround time?
  - Small number of highly parallel jobs.
  - Parallelized applications, changes required.

Your Application

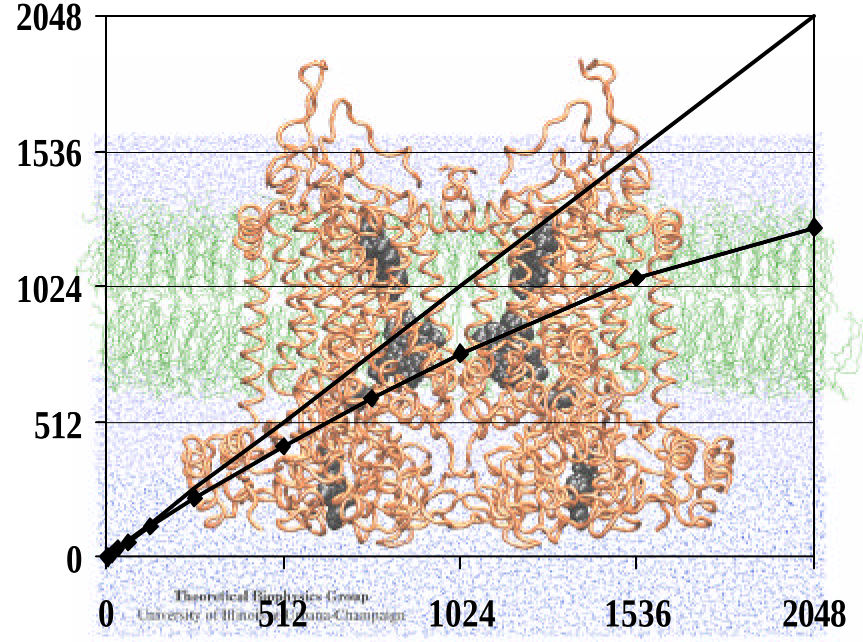
- The best benchmark for making decisions is your application running your dataset.
- Designing a cluster is about trade-offs.
  – Your application determines your choices.
  – No supercomputer runs everything well either.
- Never buy hardware until the application is parallelized, ported, tested, and debugged.

Your Application: Serial Performance

- How much memory do you need?
- Have you tried profiling and tuning?
- What does the program spend time doing?
  - Floating point or integer and logic operations?
  - Using data in cache or from main memory?
  - Many or few operations per memory access?
- Run benchmarks on many platforms.

Your Application: Parallel Performance

- How much memory per node?
- How would it scale on an ideal machine?
- How is scaling affected by:
  - Latency (time needed for small messages)?
  - Bandwidth (time per byte for large messages)?
  - Multiprocessor nodes?
- How fast do you need to run?



NAMD
Scalable Molecular Dynamics

Budget

- Figure out how much money you have to spend.
- Don’t spend money on problems you won’t have.
  – Design the system to just run your application.
- Never solve problems you can’t afford to have.
  – Fast network on 20 nodes or slower on 100?
- Don’t buy the hardware until…
  – The application is ported, tested, and debugged.
  – The science is ready to run.

Environment

- The cluster needs somewhere to live.
  - You won't want it in your office.
  - Not even in your grad student's office.
- Cluster needs:
  - Space (keep the fire martial happy).
  - Power
  - Cooling

Environment: Space

- Rack or shelve systems to save space
  - 36” x 18” shelves ($180 from C-Stores) will hold 16 PCs with typical cases
  - Multiprocessor systems save space
  - Rack mount cases are smaller and expensive

Environment: Power

- Make sure you have enough power.
  - 1.3Ghz Athlon draws 1.6A at 110 Volts = 176 Watts
  - Wall circuits typically supply about 20 Amps
    - Around 12 PCs max (8-10 for safety)

Environment: Uninterruptable Power Systems

- 5kVA UPS ($3,000)
  - Will need to work out building power to them
  - Holds 24 Athlon PCs (Safely)
  - Larger/Smaller UPS systems are available
  - May not need UPS for all systems, just root node

Environment: Cooling

- Building AC will only get you so far – large clusters will require dedicated cooling.
- Make sure you have enough cooling.
  - An Athlon PC puts out 600 BTU of heat.
  - 1 ton of AC = 12,000 BTUs = ~3500 Watts
  - Can run ~50 PCs per ton of AC (30-40 safely)

Hardware

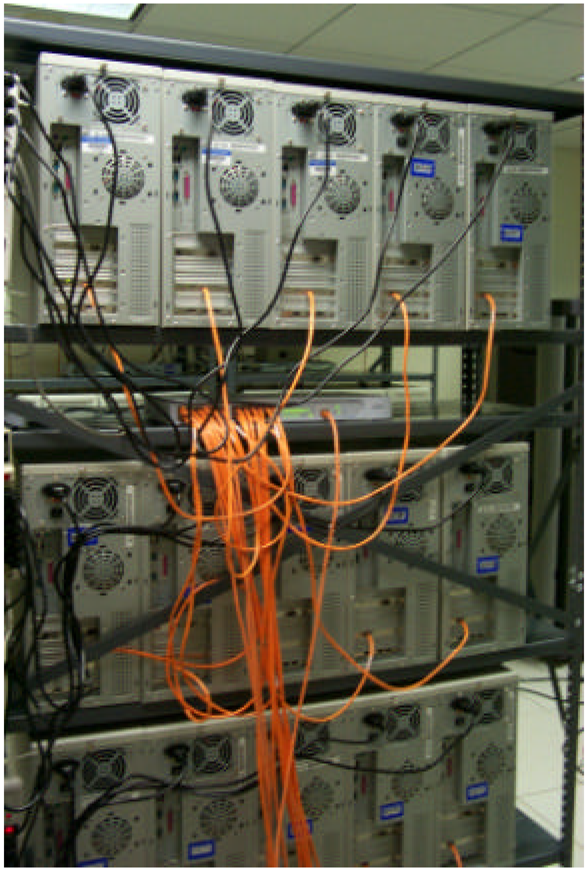
- Many important decisions to make
- Keep application performance, users, environment, local expertise, and budget in mind
- An exercise in systems integration, making many separate components work well as a unit
- A reliable but slightly slower cluster is better than a fast but non-functioning cluster

Hardware: Computers

- Benchmark a “demo” system first!
- Buy identical computers
- Can be recycled as desktops:
  - CD-ROMs and hard drives are still a good idea.
  - Don’t bother with a good video card, by the time you recycle them you’ll want something better anyway.

Hardware: Networking (1)

- Latency
- Bandwidth
- Bisection bandwidth of finished cluster
- SMP performance and compatibility?



Hardware: Networking (2)

- User-space message passing
  - Virtual interface architecture
  - Avoids per-message context switching between kernel mode and user mode, can reduce cache thrashing, etc.

Hardware: Networking (3)

- Three main options:
  - 100Mbps Ethernet – cheap ($150/node), highly supported, good for clusters up to ~32-64.
  - Gigabit Ethernet – expensive ($1000/node), less well supported or tested.
  - Special interconnects:
    - Giganet – expensive ($1500/node), low latency, VI architecture
    - Myrinet – very expensive ($2500/node), very low latency, best for huge clusters

Hardware: 100Mbps Ethernet (1)

- Performance: Underpopulate switches or buy non-blocking switches
- Large clusters: switches must interlink at high-speeds (gigabit or better, low latency)

Hardware: 100Mbps Ethernet (2)

- Channel bonding:
  - Multiple network interfaces per machine
  - Provide better performance than 100Mbps Ethernet
  - Cheaper than Gigabit or special interconnects.
  - More complex to setup and maintain
  - May not work with your message passing library of choice

Hardware: 100Mbps Ethernet (3)

- Sample prices (from cdwg.com):
  - 3Com 3300XM (24-ports + 1x matrix [3Gbps]): $900
  - 3Com 3300TM (24-ports + 1x matrix + 1x gigabit uplink): $1300
  - 3Com 3300MM (24-ports + 3x matrix): $1400

Hardware: Other Components

- Filtered Power (Isobar, Data Shield, etc)
- Network Cables: Buy good ones, saves debugging time later.
- If a cable is at all questionable, throw it away.
- Power Cables
- Monitor
- Video/Keyboard Cables

System Software

- More choices: operating system, message passing libraries, numerical libraries, compilers, batch queueing, etc.
- Performance
- Stability
- System security
- Existing infrastructure considerations

System Software: Operating System (1)

- Clusters have special needs, use something appropriate for the application, hardware, and that is easily clusterable
- Security on a cluster can be nightmare if not planned for at the outset
- Any annoying management or reliability issues get hugely multiplied in a cluster environment

System Software: Operating System (2)

- SMP Nodes:
  - Does kernel TCP stack scale?
  - Is message passing system multithreaded?
  - Does kernel scale for system calls made by intended set of applications?
- Network Performance:
  - Optimized network drivers?
  - User-space message passing?
  - Eliminate unnecessary daemons, they destroy performance on large clusters (collective ops)

Scyld Beowulf

- Single front-end master node:
  – Fully operational normal Linux installation.
  – Bproc patches incorporate slave nodes.
- Severely restricted slave nodes:
  – Minimum installation, downloaded at boot.
  – No daemons, users, logins, scripts, etc.
  – No access to NFS servers except for master.
  – Highly secure slave nodes as a result

System Software: Compilers

- No point in buying fast hardware just to run poor performing executables
- Good compilers might provide 50-150% performance improvement
- May be cheaper to buy a $2,500 compiler license than to buy more compute nodes
- Benchmark real application with compiler, get an eval compiler license if necessary

System Software: Message Passing Libraries

- Usually dictated by application code
- Choose something that will work well with hardware, OS, and application
- User-space message passing?
- MPI: industry standard, many implementations by many vendors, as well as several free implementations
- PVM: typically low performance avoid if possible
- Others: Charm++, BIP, Fast Messages

System Software: Numerical Libraries

- Can provide a huge performance boost over "Numerical Recipes" or in-house routines
- Typically hand-optimized for each platform
- When applications spend a large fraction of runtime in library code, it pays to buy a license for a highly tuned library
- Examples: BLAS, FFTW, Interval libraries

System Software: Batch Queueing

- Clusters, although cheaper than “big iron” are still expensive, so should be efficiently utilized
- The use of a batch queueing system can keep a cluster running jobs 24/7
- Things to consider:
  – Allocation of sub-clusters?
  – 1-CPU jobs on SMP nodes?
- Examples: DQS, PBS, NQS, Load Leveler

Case Study

- Users:
  - Many researchers with MD simulations
  - Need to supplement time on supercomputers
- Application:
  - Not memory-bound, runs well on IA32
  - Scales to 32 CPUs with 100Mbps Ethernet
  - Scales to 100+ CPUs with Myrinet

Case Study 2

- Budget:
  – Initially $20K, eventually grew to $100K
- Environment:
  – Full machine room, slowly clear out space
  – Under-utilized 12kVA UPS, staff electrician
  – 3 ton chilled water air conditioner (Liebert)

Case Study 3

- Hardware:
  - Fastest AMD Athon CPUs available.
  - Fast CL2 SDRAM, but not DDR.
  - Switched 100Mbps Ethernet, Intel EEPro cards.
- System Software:
  - Scyld clusters of 32 machines, 1 job/cluster.
  - Existing DQS, NIS, NFS, etc. infrastructure.