

What are Clusters?

- Our definition : A parallel machine built of commodity components and running commodity software
- Cluster consists of *nodes* with one or more processors (CPUs), memory that is shared by all processors in (and only in) the node, possibly peripherals such as disks, and are connected to other nodes by a *network*.
- Nodes with more than one processor is called an SMP (symmetric multiprocessor) node.
- Clusters can be do-it-yourself or packaged

Why Clusters? - a Short History

- Same reasons as supercomputers were needed
 - Need for increased computational power to solve larger more complex problems
 - large run times or real time constraints
 - large memory usage
 - high I/O usage
 - Fault tolerance (web or scientific computation)
- Interest in low cost alternative to expensive parallel machines

Taxonomy of Parallel Architectures

- Arranged by tightness of coupling i.e. latency
- Systolic – special hardware implementation of algorithms, signal processors, FPGA
- Vector – pipelining of arithmetic operations (ALU) and memory bank accesses (Cray)
- SIMD (Associative) – Single Instruction Multiple Data, same instruction applied to data on each of many processors (CM-1, MPP, Staran, Aspro, Wavetracer)
- Dataflow – fine grained asynchronous flow control depending on data precedence constraints
- PIM (processor-in-memory) – combine memory and ALU on one circuit die. Gives high memory bandwidth and low latency

Taxonomy of Parallel Architectures

- MIMD (Multiple Instruction Multiple Data) – execute different instruction on different data
- MPP (Massively Parallel Processors)
 - Distributed memory (Intel Paragon)
 - Shared Memory w/o coherent caches (BBN Butterfly, T3E)
 - CC-NUMA [cache coherent non-uniform memory architecture] (HP Exemplar, SGI Origin 2000)
- Clusters – ensemble of commodity components connected by an interconnection network within a single administrative domain and usually in one room
- (Geographically) Distributed Systems – exploit available cycles (Grid, DSI, Entropia, SETI@home)

Taxonomy of Clusters

- Workstation Clusters (Networks of Workstations, NOW) (Sun, SGI)
- Beowulf class clusters – ensembles of PCs integrated with COTS (commercial off the shelf e.g. ethernet) or SAN (system area e.g. Myrinet) networks
- Cluster farms – existing LANs and PCs which when idle can be used to perform work
- Superclusters – clusters of clusters, within a LAN or a campus, perhaps using the campus network for interconnection.

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 5

Clusters - Deployment

- Originally research groups within National Labs and Universities
 - used for computational science not just Math & CS research
- Now being deployed in supercomputer centers & used commercially
 - OSC recently replaced IBM SP2 by a Compaq cluster
- Smaller ones can be afforded by departments and research groups

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 6

From Cray to Beowulf

- Vector computers
- Parallel Computers
 - shared memory, bus based (SGI Origin 2000)
 - distributed memory, interconnection network based (IBM SP2)
- Network of Workstations (Sun, HP, IBM, DEC) - possibly shared use
 - NOW (Berkeley), COW (Wisconsin)
- PC (Beowulf) Cluster – originally dedicated use
 - Beowulf (CESDIS, Goddard Flight Center, 1994)
 - Possibly SMP nodes

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 7

Berkeley NOW



- 100 Sun UltraSparcs
 - 200 disks
- Myrinet SAN
 - 160 MB/s
- Fast comm.
 - AM, MPI, ...
- Ether/ATM switched external net
- Global OS
- Self Config

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 8

Evolution of Parallel Machines

- Originally Parallel Machines had
 - custom chips (CPU), custom bus/interconnection network, custom I/O system
 - proprietary compiler or library
- More recently parallel machines have
 - custom bus/interconnection network and possibly I/O system
 - standard chips
 - standard compilers (f90) or library (MPI)

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 9

Cluster Advantages

- Capability Scaling
- Convergence Architecture - standard
- Price/Performance
- Flexibility of Configuration and Upgrade
- Technology Tracking
- High Availability – redundancy
- Personal Empowerment
- Development Cost and Time for Manufacturers

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 10

Application Requirements

- Computational Requirements
 - Number of floating point operations
 - Compare with *attainable performance* for types of operations in calculation **not** with peak Mflops rating of manufacturer
 - Mflops is often only speed in cache (more later)
- Memory
 - Cache / Main memory / Virtual Memory (on disk)
 - Virtual memory not used if high performance needed
- I/O
 - Often need large data output
 - NFS issues (low performance and concurrency issues)

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 11

Application Requirements - Parallelism

- *Embarassingly* (or pleasingly) parallel : Can be broken into multiple tasks that need little or no communication
 - Parameter study, web server
- Applications requiring explicit parallelism
 - Need to evaluate whether can be run effectively on cluster
 - Metrics used:
 - Latency (minimum time to send a message)
 - Overhead (time CPU spends in sending message)
 - Bandwidth
 - Contention (for shared resource e.g. network/memory bus)

KENT STATE DiSCoV

Fall 2008

Paul A. Farrell
Cluster Computing 12

Simplest Model

- $T = s + r n$
 - T time to transfer n bytes
 - s is latency
 - $r = 1/B$ where B is bandwidth
- Typical numbers for clusters
 - s from 5 to 100 microsecs
 - r from 0.01 to 0.1 microsecs/byte
- Note 2GHz processor can do a new fp instruction every 0.00005 microsecs
- Easy to see that need significant processing between communications
- More sophisticated LogP model separates overhead from latency

Estimating Application Requirements

- 3D PDE in cube, N points per side = N^3 points
- Time marching 6 floating points operations (flops) per mesh point
- Each mesh point given by 4 values (x, y, z) and f
- Let $N = 1024$, need 2 time steps in memory
 - Data size = $2 \times 4 \times (1024)^3 = 8$ Gwords = 64 Gbytes
 - Work per step = $6 \times (1024)^3 = 6$ Gflops
- Need for cluster
 - Memory size exceeds availability on single node
 - Memory size exceeds 4 Gbytes addressable by 32-bits
 - Work seems reasonable for CPU (now approx 6 Gflops but that is peak rate !!)

Real Performance

- Suppose 2GHz is clock rate
- Consider code
 - for ($i=0; i<n; i++$) $c[i] = a[i] * b[i]$
- 2 loads and 1 store of double
- 2 billion per sec requires moving $3 \times 8 \times 2 \times 10^9 = 48$ GB/sec from memory
- Not available in commodity nodes – more like 0.2 to 1.0 GB/sec
- Achievable performance only 0.5 to 2% of peak
- Dominated by memory performance rather than CPU
- More about this later

Domain Decomposition

- p processors
- Break into sub-domains (subcubes) of $N/p^{1/3}$ points per side
- Values from sides of 6 neighboring pieces needed – $(N/p^{1/3})^2$ per side. Now have communication overhead of $6(s + r N^2/p^{2/3})$
- Time T now
 - $T = N^3 f/p + 6(s + r N^2/p^{2/3})$
 - f is flops per mesh point
- Even with infinite p have $T = 6s$

Choosing number of nodes

- Based on $T = N^3 f/p + 6(s + r N^2/p^{2/3})$
- Minimize cost
 - Fit subdomain on node
 - 2GB memory per node -> at least 32 nodes
- Real time constraint – given T solve for p
 - Floating point work must be large compared with communication implies $N^3 f/p > 6s$ or $p < N^3 f/6s$
 - For typical s/f and N = 1024 limits p to a few thousand nodes
 - But for N = 128 and fast ethernet means $p < 10$
- Moral : Be careful – don't try to use too many nodes

Performance Conclusions

- Cluster useful – could not do on single processor due to memory size needed
- Expected performance small % of peak
- If enough nodes, computation might fit in cache and get *superlinear* speedup (not likely)
- Latency key here, but bandwidth may be in others
- If we record each time step, need to do 64 GB of output, so need high performance I/O system
- Note problem was 3D – rarely worth doing 2D ones in parallel

Choosing Cluster

1. Understand application needs
2. Decide number and type of nodes
 - Uni-processor or SMP
 - Processor type (64 or 32 bit, fp or integer performance)
 - Memory system
3. Decide on Network
 - Low latency or high bandwidth needs
4. Determine physical needs
 - Floor space, power, cooling
5. Determine Operating System

Choosing Cluster

- Cost Tradeoffs
 - Newest fastest nodes more expensive per flop
- Cost
 - Irrelevant : buy fastest -> less overhead
 - Total computing power goal: mid to low-end nodes but replace frequently (18 months)
 - Specific computational power needed for application (e.g. to achieve computation rate) then analyze trade-offs

Reading

- *Parallel Supercomputing with Commodity Components* , by Michael S. Warren, Donald J. Becker, M. Patrick Goda, John K. Salmon, and Thomas Sterling, PDPTA97.
 - Original Beowulf paper?
- *A Case for NOW (Networks of Workstations)* , by Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team, IEEE Micro, Feb, 1995.