Multiprocessors

Today's topics:

Discuss midterm & course interaction level

Discuss HW4

Parallelism

via threads, cores, and/or processors

Flynn's taxonomy

basic organizational issues

Application Parallelism

some simple examples

The Midterm

- "A lot of theory" say some
  - Al's view – not really these basic concepts are what you'll retain
    - » equations you can always look up if you don't remember
    - » but conceptual issues will mark you as architecturally savy or not
      - if you have to look these up it will be embarrassing
- Surprised at some questions
  - after HW3 the branch prediction question should have been a cake walk
- We need to change how we interact
  - I need to talk less and you need to talk more
  - it will take effort from both sides
    - » ask questions if you don't understand – make me explain
    - » I ask questions to get a pulse – but often there are no takers
      - we need to fix this
- A brief review of the solutions

HW4

- Will employ a tool called "CACTI 6.5"
  - released last week form HPL and installed yesterday on the CADE machines
- 4 questions – none are trivial
  - you'll need to formulate experiments to run using CACTI
  - you'll need to interpret the data and draw conclusions which answer the question
- Not your typical homework – start NOW!
  - more similar to a research endeavor
    - » as grad students this should be your future
- Focus
  - introduce you to a valuable research tool
  - give you some scope on a critical future area
    - » e.g. register, cache, and memory organization
    - » introduce real delay and power/energy
      - mostly underplayed in your text

The Greed for Speed

- It's always been about parallelism
  - earlier – hardware & hidden from programmer
  - today – parallel cores, multiple sockets
    - » and multiple threads per core
- Change in usage
  - mobile "average user"
    - » use small dweeby light thing – cell phone, laptop, whatever
    - » grad students in CS or CE aren't part of this
  - tons of data
    - » sensors and Google camera cars are everywhere
  - heavy weight computing is done elsewhere
    - » data-center
    - » the "Cloud" – SETI@home gets a new name – whatever
    - » supercomputers
      - check out www.top500.org
        - – IBM Roadrunner
        - – Cray Jaguar

What Changes

- **Arlo**
  - "it's an organization"
    - » organizational problems
      - what to share vs. keep private
      - how to communicate
      - management overhead
- **3 basic components**
  - core – it's getting simpler
    - » primarily due to power issues & there are lots of them/socket
  - memory
    - » cache
      - shared on socket at L2 or L3 level
    - » main
      - also shared in a couple of options
  - interconnect
    - » specialized in supercomputer/data-center/HPC land
    - » commodity (a.k.a. fast ethernet) in cluster land

Today's Similarities

- **Microprocessor based**
  - today's uP's: multi-threaded and multi-core
- **Interconnect and memory system varies**
  - but it's all about communication
    - » memory accesses may be distant or local
    - » communication may be
      - via shared memory (implicit)
      - or based on message passing (explicit)
      - or both
  - power is a dominant concern
    - » all those long wires frequently used
    - » becoming a concern in the national energy footprint
- **Lots of options**
  - today we'll look at the high level
  - & decode some of the tower of Babel acronyms that are in common use

Application Parallelism

- **Multi-processing**
  - processes each run in their own protected virtual address space
    - » lots of overhead to provide that protection
    - » communicate via explicit mechanisms
      - pipes, sockets, etc.
- **Multi-threading**
  - share virtual address space
  - Wax hazard avoidance
    - » via synchronization mechanisms
      - barrier, semaphore, etc.
- **Confusion**
  - both may be inter-twined into the thread or processor term
    - » 1 core 2 threads
      - run two processes or two threads
  - add multiple sockets and life gets even more fuzzy

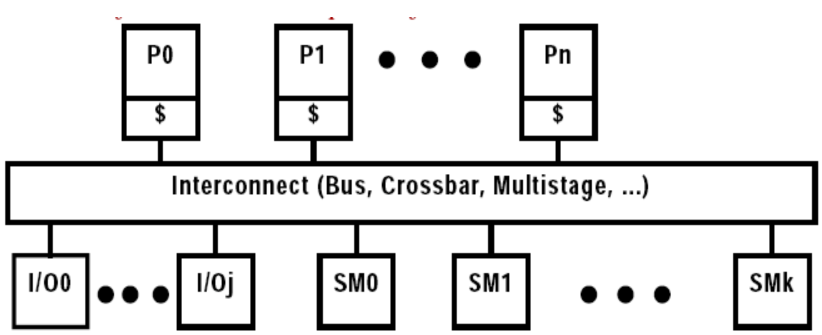
Flynn's Taxonomy (1972)

- **Too simple but the only one that moderately works**
  - taxonomy of parallel machines is a bit of a red herring
    - » doesn't work as well as in the plant and animal worlds
    - » change in computer structures isn't that "genetic"
- **(Single, Multiple) X (Data stream, Instruction stream)**
  - SISD – the killer uP of old
    - » gone in the mainstream segment
  - SIMD
    - » Illiac IV – the original supercomputer
      - broadcast too expensive and resource utilization problem
    - » today alive and well (exploits data parallelism)
      - vector processing, media instructions (SSEn, Altivec)
      - wide SIMD is the theme for GPGPU's
  - MISD
    - » nothing commercial – closest was HT Kung's iWARP @ CMU
  - MIMD
    - » exploits TLP – hence the focus of much of the industry

SMP Memory Organization

- **Main memory shared by all cores**
  - **private caches**
- **UMA – uniform memory access**
  - **all processors see the same memory org.**
    - » **hence the SMP moniker**
- **How well does it scale**
  - **for small core counts – not too bad**
    - » **banking and a good interconnect helps**
    - » **large caches should reduce contention on the interconnect**
  - **for large core count – unlikely win**
    - » **power consumed in interconnect will be prohibitive**
      - **common interconnect becomes the bottleneck**
      - **only option is add complexity and power to mitigate**
        - – **unacceptable option with high core counts**
    - » **delay and area costs on chip will constrain performance**
      - **area is a semi-zero-sum game**

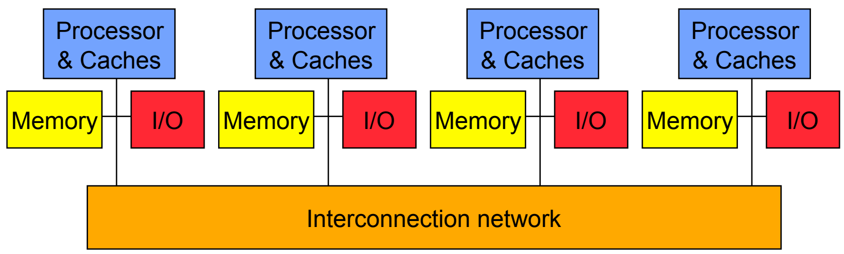
SMP/UMA Example



Early examples: Burroughs BSP, Sequent Symmetry S-81

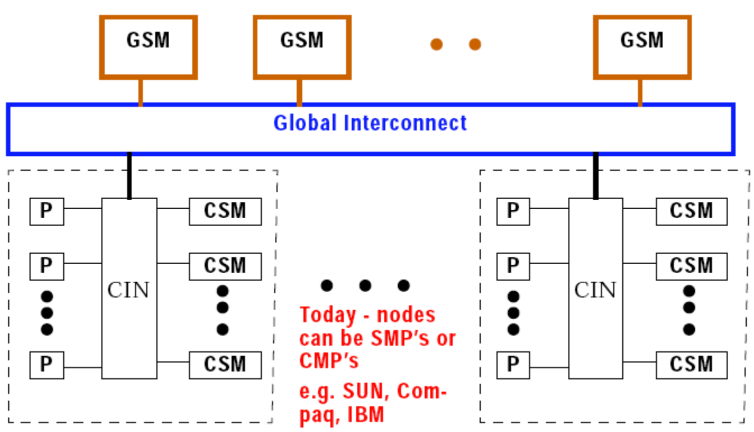
DSM/NUMA Organization

- **Higher core counts**
  - **distribute memory but shared access → DSM**
    - » **now have local vs. non-local references**
      - **NUMA**
    - » **compatible with multiple sockets and multiple MC's/socket**
  - **new problem**
    - » **memory chunks now local to some socket or MC**
      - **messages must be sent over interconnect for remote accesses**



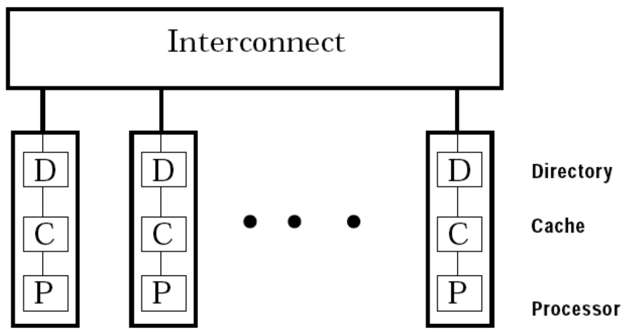
Extending the Hierarchy

- **NUMA opus 2**
  - **e.g. UIUc Cedar, CMU's CM* & C.mmp**



COMA – the Lunatic Fringe

- Treat all memory as cache
  - e.g. KSR-1 (which died at 1)

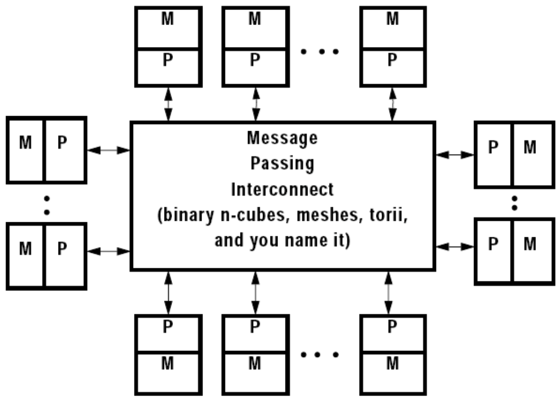


Cache Organizations

- Possible that caches can be shared as well
  - issue of coherence
    - » CC-NUMA vs. NCC-NUMA
  - CC-NUMA SMP
    - » snooping protocol and bus to maintain coherence
      - cache to cache transfers
        - – details next lecture
  - CC-NUMA DSM
    - » notion of home node
    - » global vs. local state of the line
      - MESI, MOESI, MSI variants
        - – details next week
  - NCC-NUMA
    - » no cache impact just DSM/NUMA
- More acronyms
  - UCA – banked cache
  - NUCA – distributed cache, possibly banked

NORMA

- **Message-Passing**
  - e.g. FAIM-1, Mayfly, Cosmic Cube, Ncube, iPSC, ...
  - Beowulf clusters, easy to make



Programming Models

- **Shared Memory**
  - CC-NUMA/NUCA
  - familiar model w/ implicit communication
    - » downside – easy to obtain horrible performance
    - » upside is no OS involvement
    - » communication is happening and it takes time
    - » hardware handles protection
    - » programmer handles synchronization when necessary
- **Message passing**
  - no cache coherence → simpler hardware
  - explicit communication
    - » +: programmer designs it into a good algorithm
      - visible in restructuring code
    - » -: increased programmer burden
      - OS tends to want to be in the way
  - model of choice for todays supercomputers
    - » MPI, Open-MP, MCAPI

Duality

- Shared memory on top of message passing
  - no problem
    - » lots of software packages have done this
    - » e.g. MUNIN and descendants at RICE
  - IBM SP-2 had a library
- Message passing on top of shared memory
  - also no problem
  - SGI Origin 2000 actually beat the SP-2 doing just this
    - » why?
    - » remember that OS overhead issue

Parallel Performance Scalability

- Amdahl's law in action
  - enhanced = parallel component
  - example 1: code centric
    - » 80% of your code is parallel
      - best you can do is 5x speedup if parallel part goes to 0
  - example 2: speedup centric
    - » want 80x speedup on 100 processors
      - $fraction_{enhanced}$ = .9975
      - this will be hard
- Linear speed up is hard
  - unless "embarrassingly parallel" threads
  - no dependence or cooperation
- Superlinear speed up is easier
  - lots more memory so no paging
  - beware of these claims in the literature

Parallel Workloads

- **Highly varying**
  - resource utilization: cores, threads, memory, interconnect
  - slight architecture change → big performance change
- **3 workload examples**
  - commercial
    - » OLTP – TPC-B
    - » DSS – TPC-D
    - » Web index search (AltaVista and a 200GB database)
  - multiprogrammed & OS
    - » 2 independent compiles of Andrew file system
    - » phases: compute bound (compile) & I/O bound (install and remove files)
  - scientific
    - » FFT, LU, Ocean, Barnes

Effort Variation

- **Commercial workload on a 4 processor server**

Benchmark	% time in user mode	% time in kernal mode	% time CPU idle
OLTP	71	18	11
DSS range for all 6 Queries	82-94	3-5	4-13
DSS average	87	3.7	9.3
AltaVista	>98	<1	<1

- **Multiprogrammed & OS on 8 processors**

	User	Kernel	Synch Wait	CPU idle (I/O wait)
% instructions xeq'd	27	3	1	69
% xeq time	27	7	2	64

FFT

- 1 D complex numbers
  - 3 data structures: in, out, and read-only roots matrix
  - steps
    - » transpose input data matrix
    - » 1D FFT on each row
    - » roots x data matrix
    - » transpose data matrix
    - » 1D FFT on each row of data matrix
    - » transpose data matrix
  - communication
    - » all to all communication in the three transpose phases
      - each processor transposes one local block and sends it to each other processor
- Synopsis
  - communication bound & tends to scale badly

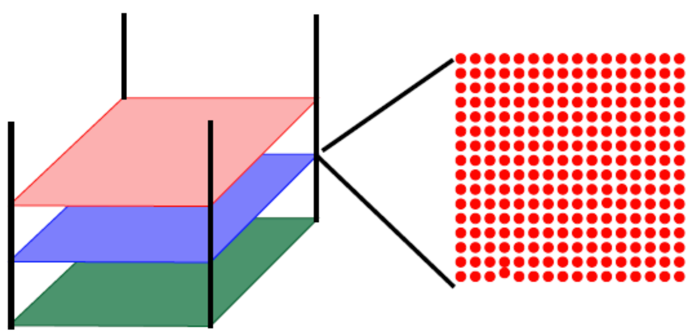
LU

- Typical dens matrix factorization
  - used in a variety of solvers & eigenvalue computations
- Turn matrix into upper diagonal matrix
  - blocking helps code to be cache friendly
- Block size
  - small enough to keep cache miss rate low
  - large enough to maximize parallel phase
- Synopsis
  - this one scales well

Ocean

- **3D weather modeling**
  - **75% of earth's surface is ocean**
    - » major weather impact
    - » eddy effect is significant
  - **4D problem**
    - » 3D physical space + the time dimension
  - **model**
    - » discrete set of equally space points
    - » simplify into a set of 2D planes
      - more difficult convergence but simpler communication
        - both take time
        - illustrative of the basic issues

Ocean Model



Rectangular basin = 3D

simplify = 2d plane set

separate 2d array for each variable

equal spaced points

continuous ==> discrete

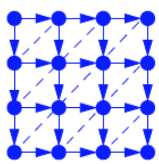
Ocean Benchmark

- **Data**
  - **2D arrays for each variable**
  - **all arrays model each plane**
- **Time**
  - **solve set of motion equations**
  - **sweep through all points per time step**
  - **then move to next time step**
- **Granularity**
  - **big influence on compute time**
    - » **2M miles x 2M miles = Atlantic Ocean**
    - » **points @ 1 km spacing & 5 years of 1 minute time steps**
      - **2.6 M steps x 4M points – intractable now but maybe not in the future**

Ocean Decomposition

❑ Model the weighted nearest neighbor average

- A[i,j] = 0.2 x (A[i,j] + A[i,j-1] + A[i-1,j] + A[i,j+1] + A[i+1,j])

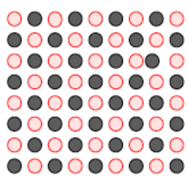


Evolve the sequential algorithm

bogus once again - little parallelism

Note the anti-diagonal option (orthogonal to resultant dependence vector)

Control and Load Imbalance Issues??

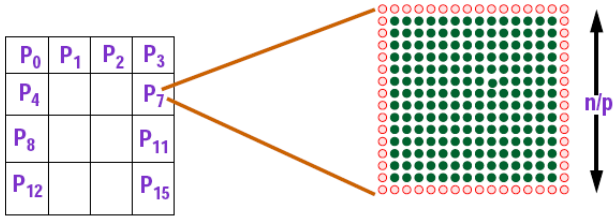


Red Black Decomposition

Dependencies?
Parallelism?
Convergence properties?

Ocean Decomposition

- **side effect of grid based solver**
  - **perimeter vs. area**

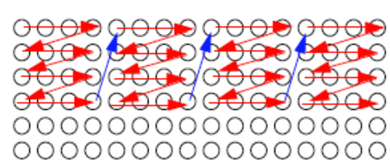


Local Work $\alpha \frac{n^2}{p}$

Remote Communication $\alpha \frac{4n}{\sqrt{p}}$

Blocking for Cache Locality

mindless 2D version

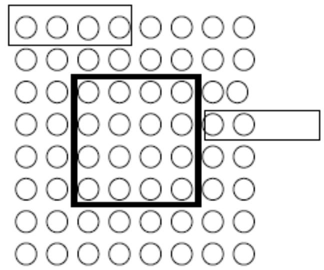


2D inside 2D = 4D arrays

- consider cache effects
  - spatial and temporal locality
- other effects
  - blocks can also be influenced by processor partition
  - particularly useful if address space is shared as in a DSM machine
- boundary problems?

Boundary Issues

- **Assume row-major order (think C) allocation**
  - column lines will have poor spatial locality



Barnes-Hut

- **Simulates evolution of galaxies**
  - class N-body gravitation problem
- **Characteristics**
  - no spatial regularity so computation is particle based
  - every particle influences every other particle
    - » $O(n^2)$ complexity – UGHly
    - » cluster distant star groups into one particle
      - based on center of mass since

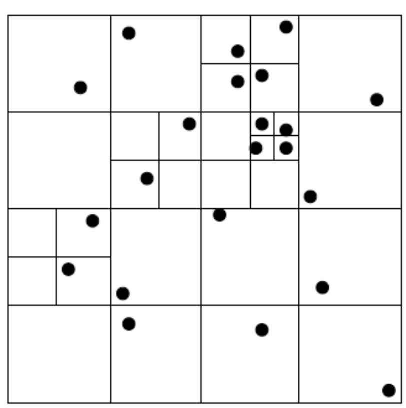
$$\text{Gravitational Force} = G\frac{M_1 M_2}{r^2}$$

      - simplifies complexity to O(n log n)
      - close stars must be handled individually

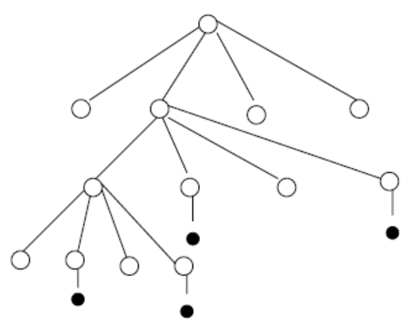
Oct-tree Hierarchy

- 3D galaxy representation
  - 8 equally sized children
    - » based on equal space volumes
  - tree traversed once per body to determine force
  - bodies move so rebuild tree on every step
- Group optimization
  - if cell is far enough away
    - » l/d < x
      - l = cell side length, d = distance from cell center
      - x is accuracy parameter – typically between .t and 1.2
    - » then treat as single body
    - » otherwise open cell and proceed

2D Quadtree Illustration



2D Spatial Decomposition

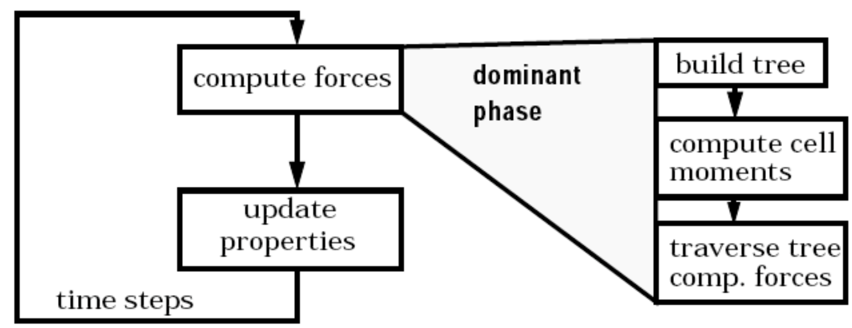


QuadTree Equivalent

Each non-leaf has center of mass for the group

Each leaf has mass, velocity, etc.

Algorithm Flow



Scientific Workload Scaling

Application	Computation Scaling per processor	Communication Scaling	Compute/ Communicate Scaling
FFT	(n log n)/p	n/p	log n
LU	n/p	$\frac{\sqrt{n}}{\sqrt{p}}$	$\frac{\sqrt{n}}{\sqrt{p}}$
Barnes	(n log n)/p	approximately $\frac{\sqrt{n}\log n}{\sqrt{p}}$	approximately $\frac{\sqrt{n}}{\sqrt{p}}$
Ocean	n/p	$\frac{\sqrt{n}}{\sqrt{p}}$	$\frac{\sqrt{n}}{\sqrt{p}}$

Concluding Remarks

- **Lots of diversity in parallel systems**
  - **architecture style**
    - » memory, interconnect, and processor XU's
  - **application space**
    - » any huge problem has lots of parallelism
      - but what type data vs. control
    - » programming model
      - message passing vs. shared memory
  - **mapping**
    - » who does it
      - programmer, compiler, OS, hardware
      - all are hard
  - **result**
    - » big difference in how resources are used
    - » there's always a bottleneck
      - trick is to figure out how to reduce it