CS 258
Parallel Computer Architecture
Lecture 2

Convergence of Parallel Architectures

January 28, 2008

Prof John D. Kubiatowicz

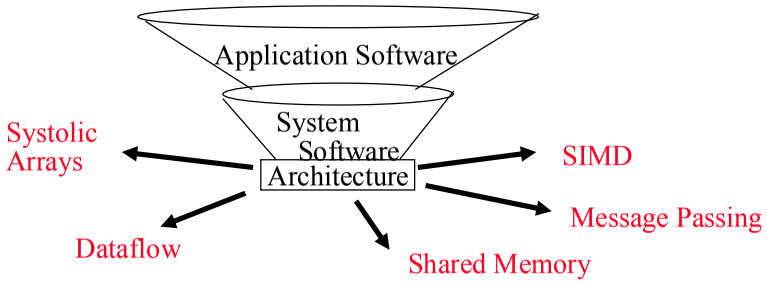
http://www.cs.berkeley.edu/~kubitron/cs258

Review

- Industry has decided that Multiprocessing is the future/best use of transistors
  - Every major chip manufacturer now making MultiCore chips
- History of microprocessor architecture is parallelism
  - translates area and density into performance
- The Future is higher levels of parallelism
  - Parallel Architecture concepts apply at many levels
  - Communication also on exponential curve
- Proper way to compute speedup
  - Incorrect way to measure:
    - » Compare parallel program on 1 processor to parallel program on p processors
  - Instead:
    - » Should compare uniprocessor program on 1 processor to parallel program on p processors

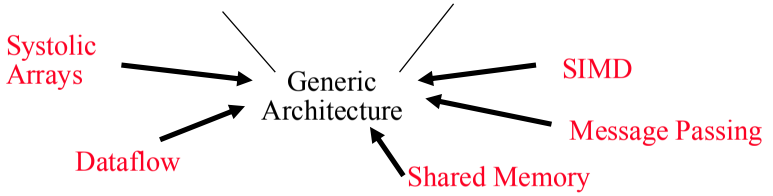
History

- Parallel architectures tied closely to programming models
  - Divergent architectures, with no predictable pattern of growth.
  - Mid 80s renaissance



Plan for Today

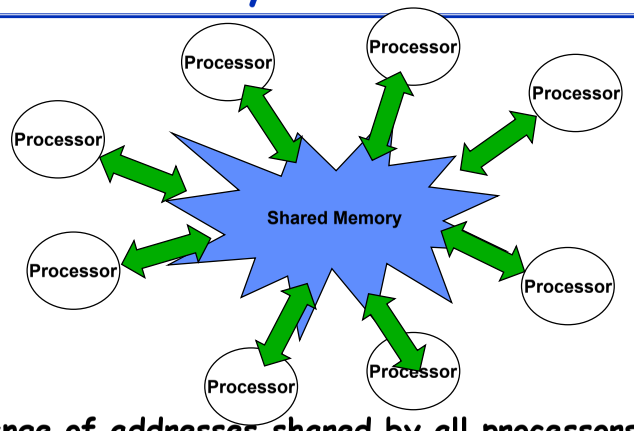
- Look at major programming models
  - where did they come from?
  - The 80s architectural rennaisance!
  - What do they provide?
  - How have they converged?
- Extract general structure and fundamental issues



Programming Model

- *Conceptualization of the machine that programmer uses in coding applications*
  - How parts cooperate and coordinate their activities
  - Specifies communication and synchronization operations
- Multiprogramming
  - no communication or synch. at program level
- *Shared address space*
  - like bulletin board
- *Message passing*
  - like letters or phone calls, explicit point to point
- *Data parallel*:
  - more regimented, global actions on data
  - Implemented with shared address space or message passing

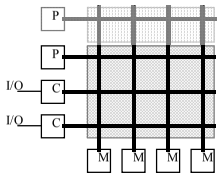
Shared Memory ⇒ Shared Addr. Space



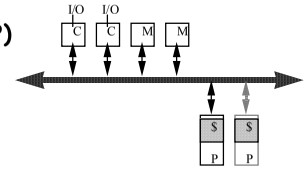
- Range of addresses shared by all processors
  - All communication is *Implicit* (Through memory)
  - Want to communicate a bunch of info? Pass pointer.
- Programming is "straightforward"
  - Generalization of multithreaded programming

Historical Development

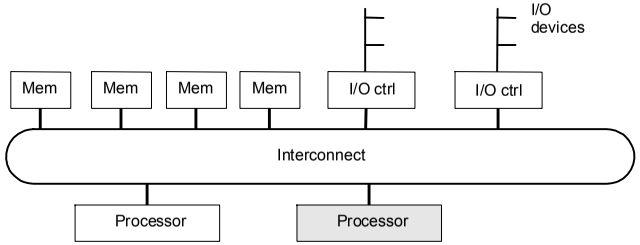
- "Mainframe" approach
  - Motivated by multiprogramming
  - Extends crossbar used for Mem and I/O
  - Processor cost-limited => crossbar
  - Bandwidth scales with *p*
  - High incremental cost
    - » use multistage instead



- "Minicomputer" approach
  - Almost all microprocessor systems have bus
  - Motivated by multiprogramming, TP
  - Used heavily for parallel computing
  - Called symmetric multiprocessor (SMP)
  - Latency larger than for uniprocessor
  - Bus is bandwidth bottleneck
    - » caching is key: *coherence* problem
  - Low incremental cost



Adding Processing Capacity



- Memory capacity increased by adding modules
- I/O by controllers and devices
- Add processors for processing!
  - For higher-throughput multiprogramming, or parallel programs

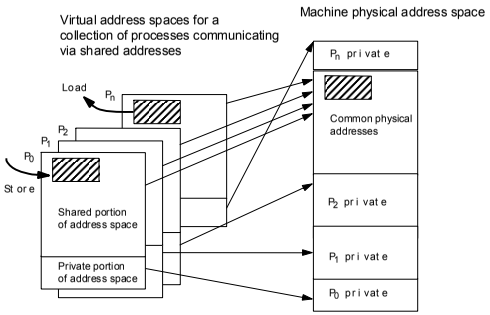
Shared Physical Memory

- Any processor can directly reference any location
  - Communication operation is load/store
  - Special operations for synchronization
- Any I/O controller - any memory
- Operating system can run on any processor, or all.
  - OS uses shared memory to coordinate
- What about application processes?

Shared Virtual Address Space

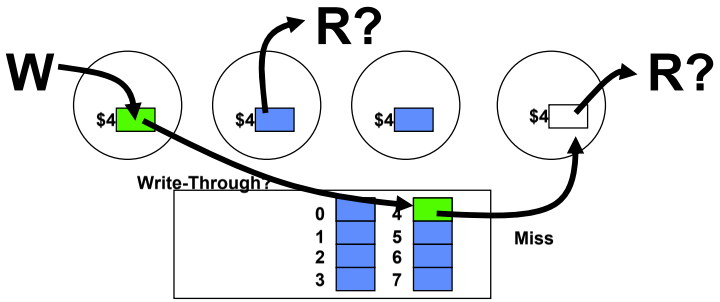
- Process = address space plus thread of control
- Virtual-to-physical mapping can be established so that processes shared portions of address space.
  - User-kernel or multiple processes
- Multiple threads of control on one address space.
  - Popular approach to structuring OS's
  - Now standard application capability (ex: POSIX threads)
- Writes to shared address visible to other threads
  - Natural extension of uniprocessors model
  - conventional memory operations for communication
  - special atomic operations for synchronization
    - » also load/stores

Structured Shared Address Space



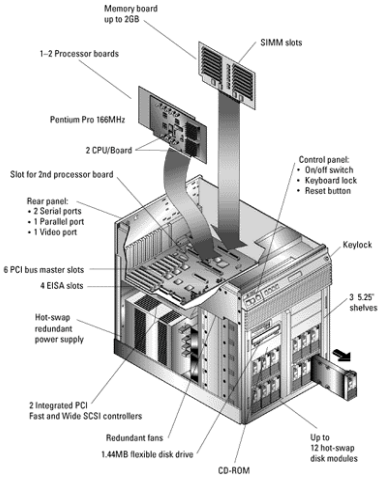
- Add hoc parallelism used in system code
- Most parallel applications have structured SAS
- Same program on each processor
  - shared variable X means the same thing to each thread

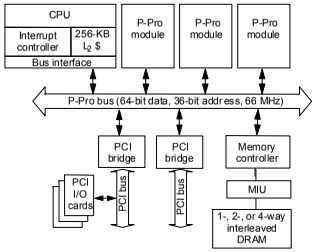
Cache Coherence Problem



- Caches are aliases for memory locations
- Does every processor eventually see new value?
- Tightly related: Cache Consistency
  - In what order do writes appear to other processors?
- Buses make this easy: every processor can snoop on every write
  - Essential feature: Broadcast

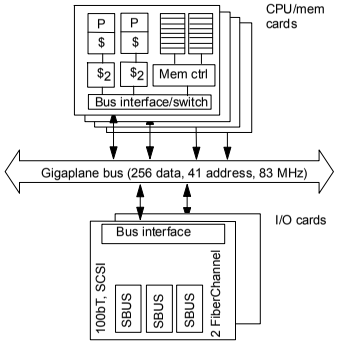
Engineering: Intel Pentium Pro Quad





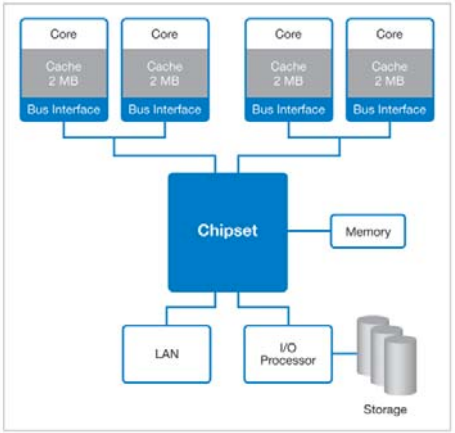
- All coherence and multiprocessing glue in processor module
- Highly integrated, targeted at high volume
- Low latency and bandwidth

Engineering: SUN Enterprise



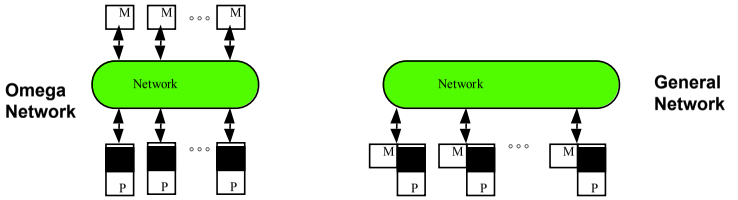
- Proc + mem card - I/O card
  - 16 cards of either type
  - All memory accessed over bus, so symmetric
  - Higher bandwidth, higher latency bus

Quad-Processor Xeon Architecture



- All sharing through pairs of front side busses (FSB)
  - Memory traffic/cache misses through single chipset to memory
  - Example "Blackford" chipset

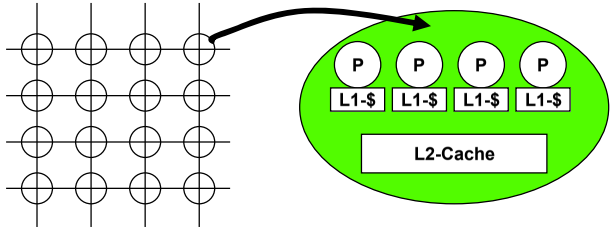
Scaling Up



- Problem is interconnect: cost (crossbar) or bandwidth (bus)
- Dance-hall: bandwidth still scalable, but lower cost than crossbar
  - » latencies to memory uniform, but uniformly large
- Distributed memory or non-uniform memory access (NUMA)
  - » Construct shared address space out of simple message transactions across a general-purpose network (e.g. read-request, read-response)
- Caching shared (particularly nonlocal) data?

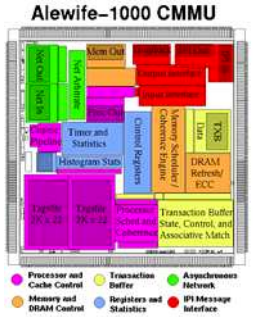
Stanford DASH

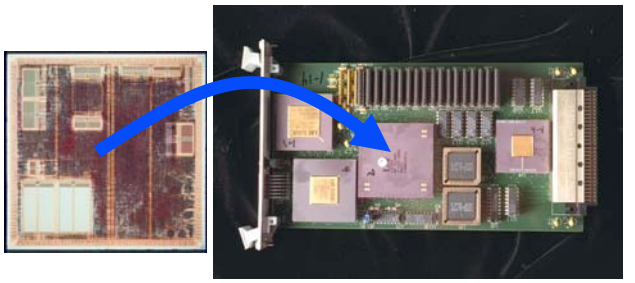
- Clusters of 4 processors share 2nd-level cache
- Up to 16 clusters tied together with 2-dim mesh
- 16-bit directory associated with every memory line
  - Each memory line has home cluster that contains DRAM
  - The 16-bit vector says which clusters (if any) have read copies
  - Only one writer permitted at a time



- Never got more than 12 clusters (48 processors) working at one time: Asynchronous network probs!

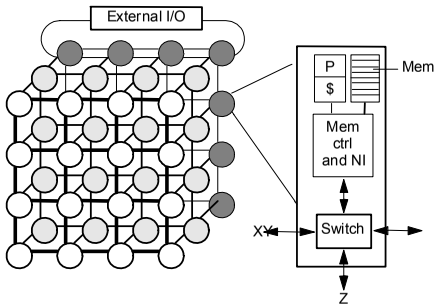
The MIT Alewife Multiprocessor





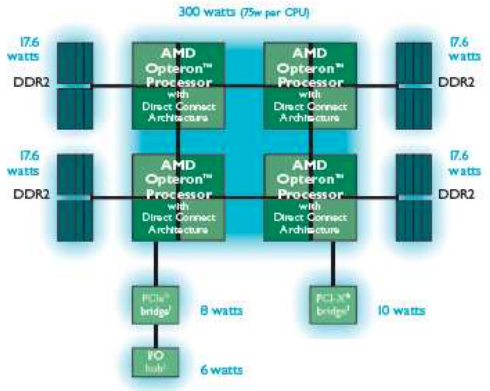
- Cache-coherence Shared Memory
  - Partially in Software!
  - Limited Directory + software overflow
- User-level Message-Passing
- Rapid Context-Switching
- 2-dimentional Asynchronous network
- One node/board
- Got 32-processors (+ I/O boards) working

Engineering: Cray T3E



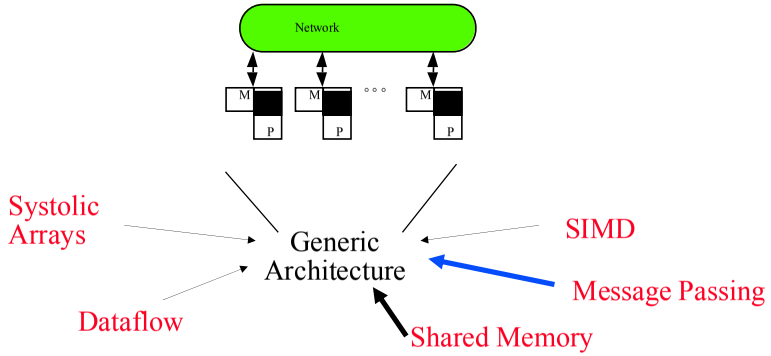
- Scale up to 1024 processors, 480MB/s links
- Memory controller generates request message for non-local references
- No hardware mechanism for coherence
  - » SGI Origin etc. provide this

AMD Direct Connect



- Communication over general interconnect
  - Shared memory/address space traffic over network
  - I/O traffic to memory over network
  - Multiple topology options (seems to scale to 8 or 16 processor chips)

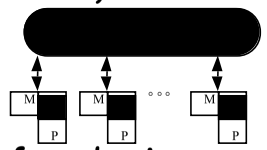
What is underlying Shared Memory??



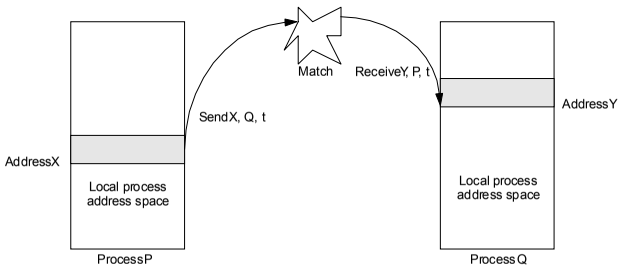
- Packet switched networks better utilize available link bandwidth than circuit switched networks
- So, network passes messages around!

Message Passing Architectures

- Complete computer as building block, including I/O
  - Communication via *Explicit* I/O operations
- Programming model
  - direct access only to private address space (local memory),
  - communication via explicit messages (send/receive)
- High-level block diagram
  - Communication integration?
    - » Mem, I/O, LAN, Cluster
  - Easier to build and scale than SAS
- Programming model more removed from basic hardware operations
  - Library or OS intervention



Message-Passing Abstraction

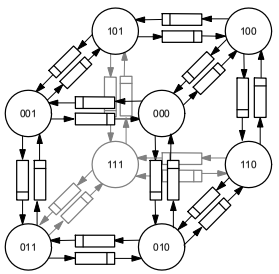


- Send specifies buffer to be transmitted and receiving process
- Recv specifies sending process and application storage to receive into
- Memory to memory copy, but need to name processes
- Optional tag on send and matching rule on receive
- User process names local data and entities in process/tag space too
- In simplest form, the send/recv match achieves pairwise synch event
  - » Other variants too
- Many overheads: copying, buffer management, protection

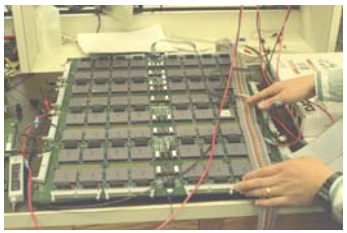
Evolution of Message-Passing Machines

- Early machines: FIFO on each link
  - HW close to prog. Model;
  - synchronous ops
  - topology central (hypercube algorithms)

CalTech Cosmic Cube (Seitz, CACM Jan 95)



MIT J-Machine (Jelly-bean machine)

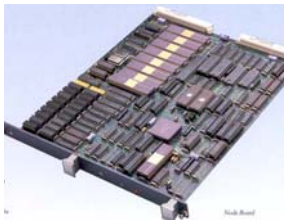


- 3-dimensional network topology
  - Non-adaptive, E-cubed routing
  - Hardware routing
  - Maximize density of communication
- 64-nodes/board, 1024 nodes total
- *Low-powered processors*
  - Message passing *instructions*
  - Associative array primitives to aid in synthesizing shared-address space
- *Extremely fine-grained communication*
  - *Hardware-supported Active Messages*

Diminishing Role of Topology?

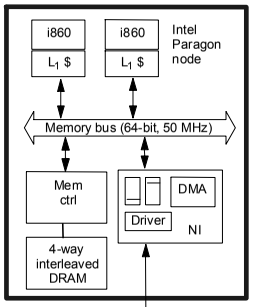
- Shift to general links
  - DMA, enabling non-blocking ops
    - » Buffered by system at destination until recv
  - Store&forward routing
- Fault-tolerant, multi-path routing:
- Diminishing role of topology
  - Any-to-any pipelined routing
  - node-network interface dominates communication time
    - » Network fast relative to overhead
    - » Will this change for ManyCore?
  - Simplifies programming
  - Allows richer design space
    - » grids vs hypercubes

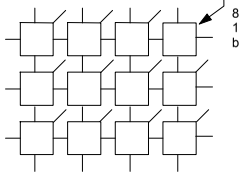
Intel iPSC/1 -> iPSC/2 -> iPSC/860



Example Intel Paragon

Sandia' s Intel Paragon XP/S-based Supercomputer

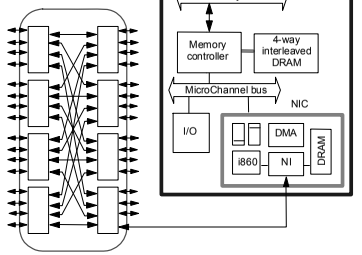


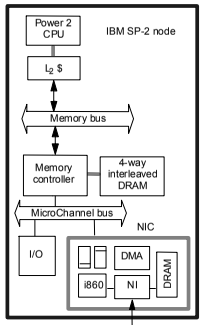


Building on the mainstream: IBM SP-2

- Made out of essentially complete RS6000 workstations
- Network interface integrated in I/O bus (bw limited by I/O bus)

General interconnection network formed from 8-port switches



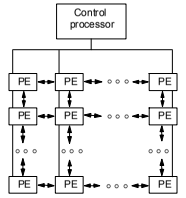


Berkeley NOW

- 100 Sun Ultra2 workstations
- Inteligent network interface
  - proc + mem
- Myrinet Network
  - 160 MB/s per link
  - 300 ns per hop

Data Parallel Systems

- Programming model
  - Operations performed in parallel on each element of data structure
  - Logically single thread of control, performs sequential or parallel steps
  - Conceptually, a processor associated with each data element
- Architectural model
  - Array of many simple, cheap processors with little memory each
    - » Processors don't sequence through instructions
  - Attached to a control processor that issues instructions
  - Specialized and general communication, cheap global synchronization
- Original motivations
  - Matches simple differential equation solvers
  - Centralize high cost of instruction fetch/sequencing



Application of Data Parallelism

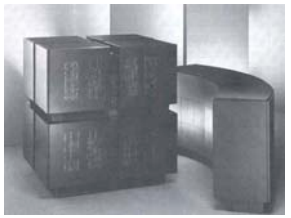
- Each PE contains an employee record with his/her salary

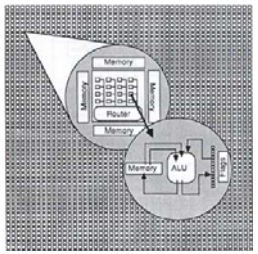
```
If salary > 100K then
   salary = salary *1.05
else
   salary = salary *1.10
```

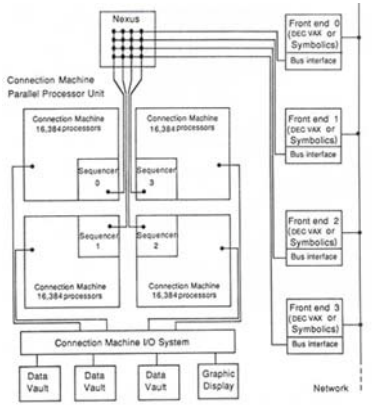
- Logically, the whole operation is a single step
- Some processors enabled for arithmetic operation, others disabled

- Other examples:
  - Finite differences, linear algebra, ...
  - Document searching, graphics, image processing, ...
- Some recent machines:
  - Thinking Machines CM-1, CM-2 (and CM-5)
  - Maspar MP-1 and MP-2,

Connection Machine

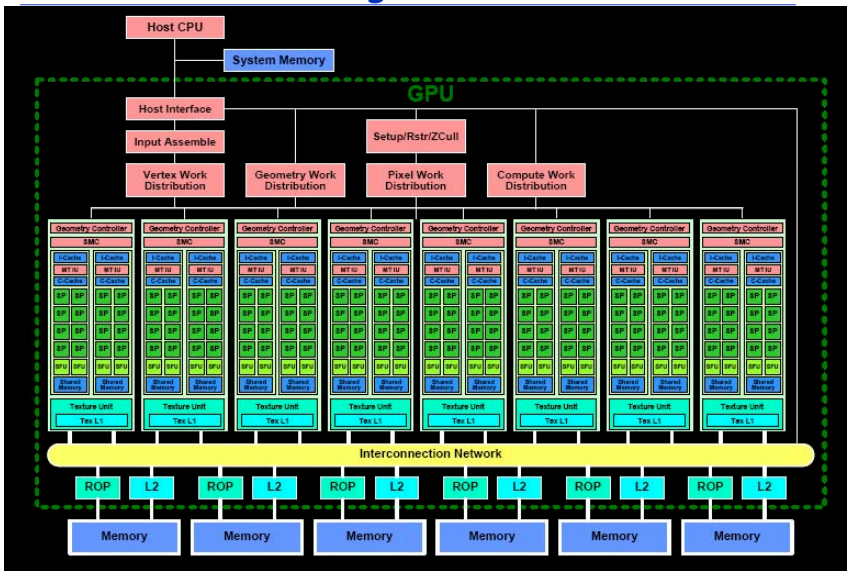




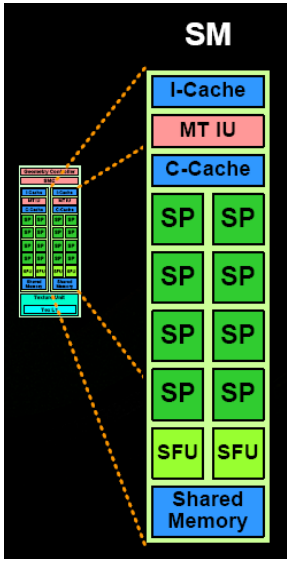


(Tucker, IEEE Computer, Aug. 1988)

NVidia Tesla Architecture
Combined GPU and general CPU



Components of NVidia Tesla architecture



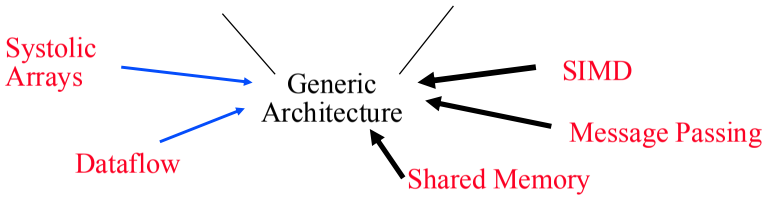
- SM has 8 SP thread processor cores
  - 32 GFLOPS peak at 1.35 GHz
  - IEEE 754 32-bit floating point
  - 32-bit, 64-bit integer
  - 2 SFU special function units
- Scalar ISA
  - Memory load/store/atomic
  - Texture fetch
  - Branch, call, return
  - Barrier synchronization instruction
- Multithreaded Instruction Unit
  - 768 independent threads per SM
  - HW multithreading & scheduling
- 16KB Shared Memory
  - Concurrent threads share data
  - Low latency load/store
- Full GPU
  - Total performance > 500GOps

Evolution and Convergence

- SIMD Popular when cost savings of centralized sequencer high
  - 60s when CPU was a cabinet
  - Replaced by vectors in mid-70s
    - » More flexible w.r.t. memory layout and easier to manage
  - Revived in mid-80s when 32-bit datapath slices just fit on chip
- Simple, regular applications have good locality
- Programming model converges with SPMD (single program multiple data)
  - need fast global synchronization
  - Structured global address space, implemented with either SAS or MP

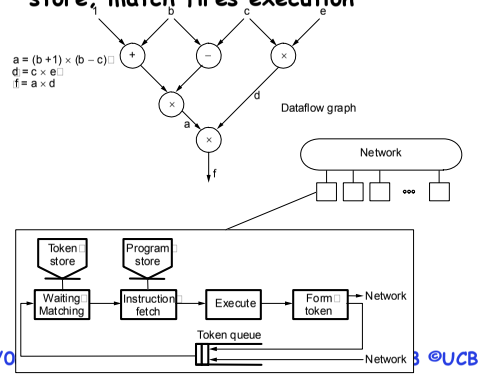
CM-5

- Repackaged SparcStation
  - 4 per board
- Fat-Tree network
- Control network for global synchronization



Dataflow Architectures

- Represent computation as a graph of essential dependences
  - Logical processor at each node, activated by availability of operands
  - Message (tokens) carrying tag of next instruction sent to next processor
  - Tag compared with others in matching store; match fires execution



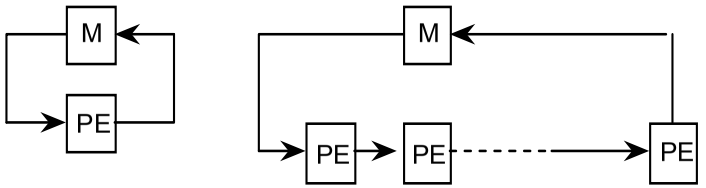
Monsoon (MIT)

Evolution and Convergence

- Key characteristics
  - Ability to name operations, synchronization, dynamic scheduling
- Problems
  - Operations have locality across them, useful to group together
  - Handling complex data structures like arrays
  - Complexity of matching store and memory units
  - Expose too much parallelism (?)
- Converged to use conventional processors and memory
  - Support for large, dynamic set of threads to map to processors
  - Typically shared address space as well
  - But separation of progr. model from hardware (like data-parallel)
- Lasting contributions:
  - Integration of communication with thread (handler) generation
  - Tightly integrated communication and fine-grained synchronization
  - Remained useful concept for software (compilers etc.)

Systolic Architectures

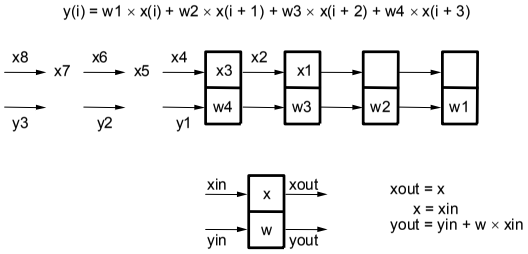
- VLSI enables inexpensive special-purpose chips
  - Represent algorithms directly by chips connected in regular pattern
  - Replace single processor with array of regular processing elements
  - Orchestrate data flow for high throughput with less memory access



- Different from pipelining
  - Nonlinear array structure, multidirection data flow, each PE may have (small) local instruction and data memory
- SIMD? : each PE may do something different

Systolic Arrays (contd.)

Example: Systolic array for 1-D convolution

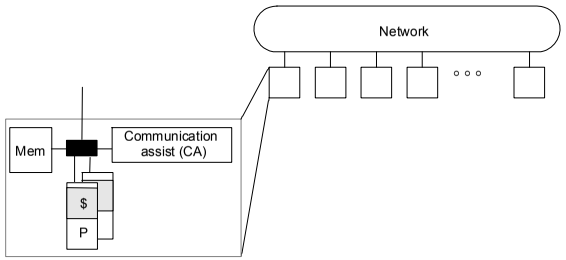


- Practical realizations (e.g. iWARP) use quite general processors
  - » Enable variety of algorithms on same hardware
- But dedicated interconnect channels
  - » Data transfer directly from register to register across channel
- Specialized, and same problems as SIMD
  - » General purpose systems work well for same algorithms (locality etc.)

Toward Architectural Convergence

- Evolution and role of software have blurred boundary
  - Send/recv supported on SAS machines via buffers
  - Can construct global address space on MP (GA -> P | LA)
  - Page-based (or finer-grained) shared virtual memory
- Hardware organization converging too
  - Tighter NI integration even for MP (low-latency, high-bandwidth)
  - Hardware SAS passes messages
- Even clusters of workstations/SMPs are parallel systems
  - Emergence of fast system area networks (SAN)
- Programming models distinct, but organizations converging
  - Nodes connected by general network and communication assists
  - Implementations also converging, at least in high-end machines

Convergence: Generic Parallel Architecture



- Node: processor(s), memory system, plus *communication assist*
  - Network interface and communication controller
- Scalable network
- Convergence allows lots of innovation, within framework
  - Integration of assist with node, what operations, how efficiently...

Flynn's Taxonomy

- # instruction x # Data
  - Single Instruction Single Data (SISD)
  - Single Instruction Multiple Data (SIMD)
  - Multiple Instruction Single Data
  - Multiple Instruction Multiple Data (MIMD)
- Everything is MIMD!
- However – Question is one of efficiency
  - How easily (and at what power!) can you do certain operations?
  - GPU solution from NVIDIA good at graphics – is it good in general?
- As (More?) Important: communication architecture
  - How do processors communicate with one another
  - How does the programmer build correct programs?

Any hope for us to do research in multiprocessing?

- Yes: FPGAs as New Research Platform
- As ~ 25 CPUs can fit in Field Programmable Gate Array (FPGA), 1000-CPU system from ~ 40 FPGAs?
  - 64-bit simple "soft core" RISC at 100MHz in 2004 (Virtex-II)
  - FPGA generations every 1.5 yrs; 2X CPUs, 2X clock rate
- HW research community does logic design ("gate shareware") to create out-of-the-box, Massively Parallel Processor runs standard binaries of OS, apps
  - Gateware: Processors, Caches, Coherency, Ethernet Interfaces, Switches, Routers, … (IBM, Sun have donated processors)
  - E.g., 1000 processor, IBM Power binary-compatible, cache-coherent supercomputer @ 200 MHz; fast enough for research

RAMP

- Since goal is to ramp up research in multiprocessing, called Research Accelerator for Multiple Processors
  - To learn more, read "RAMP: Research Accelerator for Multiple Processors - A Community Vision for a Shared Experimental Parallel HW/SW Platform," Technical Report UCB//CSD-05-1412, Sept 2005
  - Web page ramp.eecs.berkeley.edu
- Project Opportunities?
  - Many
  - Infrastructure development for research
  - Validation against simulators/real systems
  - Development of new communication features
  - Etc….

Why RAMP Good for Research?

	SMP	Cluster	Simulate	RAMP
Cost (1000 CPUs)	F ($40M)	C ($2M)	A+ ($0M)	A ($0.1M)
Cost of ownership	A	D	A	A
Scalability	C	A	A	A
Power/Space (kilowatts, racks)	D (120 kw, 12 racks)	D (120 kw, 12 racks)	A+ (.1 kw, 0.1 racks)	A (1.5 kw, 0.3 racks)
Community	D	A	A	A
Observability	D	C	A+	A+
Reproducibility	B	D	A+	A+
Flexibility	D	C	A+	A+
Credibility	A+	A+	F	A
Perform. (clock)	A (2 GHz)	A (3 GHz)	F (0 GHz)	C (0.2 GHz)
GPA	C	B-	B	A-

RAMP 1 Hardware

- Completed Dec. 2004 (14x17 inch 22-layer PCB)
- Module:
  - FPGAs, memory, 10GigE conn.
  - Compact Flash
  - Administration/ maintenance ports:
    - » 10/100 Enet
    - » HDMI/DVI
    - » USB
  - ~4K/module w/o FPGAs or DRAM

- Called "BEE2" for Berkeley Emulation Engine 2

RAMP Blue Prototype (1/07)

- 8 MicroBlaze cores / FPGA
- 8 BEE2 modules (32 "user" FPGAs) x 4 FPGAs/module = 256 cores @ 100MHz
- Full star-connection between modules
- It works; runs NAS benchmarks
- CPUs are softcore MicroBlazes (32-bit Xilinx RISC architecture)

Vision: Multiprocessing Watering Hole

Parallel file system Dataflow language/computer Data center in a box
Thread scheduling Security enhancements Internet in a box
Multiprocessor switch design Router design Compile to FPGA
Fault insertion to check dependability Parallel languages

- RAMP attracts many communities to shared artifact
  - ⇒ Cross-disciplinary interactions
  - ⇒ Accelerate innovation in multiprocessing
- RAMP as next Standard Research Platform? (e.g., VAX/BSD Unix in 1980s, x86/Linux in 1990s)

Conclusion

- Several major types of communication:
  - Shared Memory
  - Message Passing
  - Data-Parallel
  - Systolic
  - DataFlow
- Is communication "Turing-complete"?
  - Can simulate each of these on top of the other!
- Many tradeoffs in hardware support
- Communication is a first-class citizen!
  - How to perform communication is essential
    - » IS IT *IMPLICIT or EXPLICIT?*
  - What to do with communication errors?
  - Does locality matter???
  - How to synchronize?