Parallel Panorama

Parallel computation appears to be a straightforward idea, but it has not turned out to be as easy as everyone initially thinks. Today, an overview of successes and failures

Amdahl's Law

Parallel computation has limited benefit ...

- A computation taking time T, (x-1)/x of which can be parallelized, never runs faster than T/x
    - Let T be 24 hours, let x = 10
    - 9/10 can be parallelized, 1/10 cannot
    - Suppose the parallel part runs in 0 time: 2.4 hrs
- Amdahl's Law predates most parallel efforts ... why pursue parallelism?
- New algorithms

Why would one want to preserve legacy code?

Early High Performance Machines

- High Performance computing has always implied the use of pipelining
    - IBM Stretch, S/360 Model 91, Cray 1, Cyber 205
- Pipelining breaks operations into small steps performed in "assembly line" fashion
    - The size t of the longest step determines rate
    - Operations are started every t time units
- The most common application of pipelining became "vector instructions" in which operations could be applied to all elements of a vector
- Pipelining is used extensively in processor design and parallelism is a more effective way to achieve high perf

Early Parallel Machines

- The first successful parallel computer was Illiac IV built at the Univeristy of Illinois
- 64 processors (1/4 of the original design) built
    - Constructed in the preLSI days, hardware was both expensive and large
- A SIMD computer with a shared memory few registers per node
- Though it was tough to program, NASA used it

Flynn's Taxonomy:
SIMD -- single instruction multiple data
MIMD -- multiple instructions multiple data

Related term:
SPMD -- single program multiple data

SIMD Is Simply Too Rigid

- SIMD architectures have two advantages over MIMD architectures
    - There is no program memory -- smaller footprint
    - It is possible to synchronize very fast ... like on the next instruction
- SIMD has liabilities, too ...
    - Performance: `if a>0 then ... else ...`
    - Processor model is the virtual processor model, and though there are more processors than on a typical MIMD machine there is never 1pt/proc
    - Ancillary: instr. distribution limits clock, hard to share, single pt of failure, etc
- SIMD not a contender

VLSI Revolution Aided Parallel Computing

- Price/density advances in Si => multiprocessor computers were feasible
- SIMD computers continued to reign for technical reasons
    - Memory was still relatively expensive
    - Logic was still not dense enough for a high performance node
    - It's how most architects were thinking
- Ken Batcher developed the Massively Parallel Procesor (MPP) for NASA with 16K procs
    - Danny Hillis built two machines CM-1,-2 scaling to 64K
    - MASPAR also sold a successful SIMD machine

Denelcor HEP

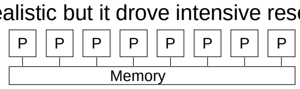
- Designed and built by Burton Smith
- Allowed multiple instructions to “be in the air” at one time
  - Fetch next instruction, update PC
  - Suspend, check to see if others need attention
  - Decode instruction, computing EA of mem ref(s), issue mem ref(s)
  - Suspend, check to see if others need attention
  - Evaluate instruction
- Great for multithreading, or for good ILP

Effects of VLSI on Early MIMD Machines

- Single chip processors spawned flury of design in nonshared memory domain
  - ZMOB, ring of 30, 8-bit procs, Maryland
  - Pringle, 8x8 config mesh, 8-bit procs, Purdue CS
  - Cosmic Cube, 6-cube, 16-bit procs, Caltech -- Intel commercialized this as iPSC, iPSC/2
- Quickly, 32-bit MIMD machines arrived
  - Sequent sold an elegant shared bus machine
  - BBN Butterfly was a large shared memory machine
  - Two different approaches to choherency

PRAM Machine Model

- Parallel Random Access Machine ... simplest generalization to a parallel machine
- Synchronous, “unit-cost” shared memory -- it is unrealistic but it drove intensive research

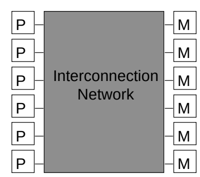


- Theoretically interesting as a way to discover the limits of parallelism

Varieties: CRCW
EREW, CREW, ...

Research Parallel Machines

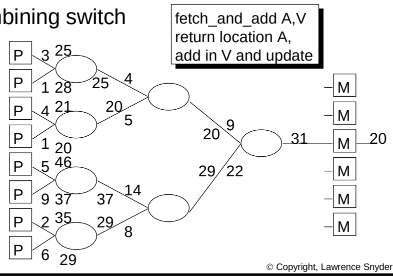
- University of Illinois developed the Cedar machine, a 32 processor shared memory -- a dance hall architecture



Ultra Computer and RP-3

- The Ultracomputer developed at NYU had a cute idea in the interconnection network: a combining switch

fetch_and_add A,V
return location A,
add in V and update



Combining Switch

- Combining switch design solves problems like “bank conflicts”, busy waiting on semaphores...
- The theory was that rather than avoiding bad operations, do them with impunity ... there was even a programming methodology based on it

Unfortunately, combining doesn’t work beyond 64 processors based on both analysis and experimentation

Early Programming Approaches

- Every machine has powerful “features” that the programmer should exploit
- Program “close” to the machine
- Parallelizing FORTRAN compilers will be along shortly

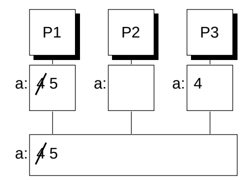
This viewpoint was nearly disasterous ...

Memory and MIMD Computers

- It was easier to invision an MIMD machine than to build one ... the problem is memory
- We are accustomed to a flat, global memory

Memory Coherency

P1 reads a into cache
P3 reads a into cache
P1 updates a to 5, wt
-- P3 has stale data --
Writeback is worse



Nonshared Memory Avoids Problem

- No shared memory implies no coherency prob
- Put a greater burden on the programmer, sw

Conventional Wisdom: Its harder to program nonshared memory

- A mid-point design is non-coherent shared address space
  - If every processor has the same amount of memory, the x^{th} power of 2, then interpret address bits left of x as processor number
  - P0 has the low addresses, P1 next lowest ...

Shared vs Nonshared Memory

- The memory organization is crucial in parallel computing. That is an incontrovertible fact
- Has engendered many wars, but one wonders why
  - Clearly, there must be a program, accessible locally to the processor, i.e. MIMD
  - There must be local data, stack, etc.
  - These are referenced frequently and must be fast
- Programmers should not have to know or care because a decent programming language can give a global view of a computation without any mention of memory organization
  - Many languages give a global view
  - ZPL’s accomplishment is it maps easily onto any memory organization

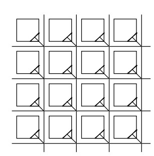
Another Round of Architecture

- The late 80’s was a rich period of machine design
  - Driving force was the VLSI revolution
  - Back pressure came from the “killer micros”
- Architecture design focused on “problems”
  - The problem the architect thought was the most pressing varied by the background of the architect
  - Examples were low latency communication, fast synchronization, coherent shared memory, scalability, ...

iWARP

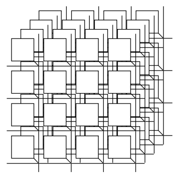
- iWARP (HT Kung, CMU/Intel) -- fast communication and synchronization; ideal for Cannon’s algorithm

Culmination of several research projects: PSC and WARP



J-Machine

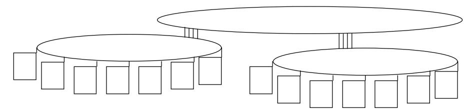
- J-Machine (Bill Dally, MIT/Intel) -- fast communication, large scalability, 3D mesh



3D design scales in physical space with minimum path lengths

KSR

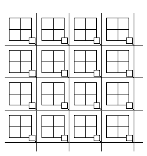
- Kendall Square Research -- cache-only machine allowing data to move to where it is needed; ring communication structure



Raised many intresting technical questions ...

DASH

- DASH (John Hennessy, Stanford/SGI) -- true distributed cache coherence



Significant research project with impact in many areas ... will study ideas of design

Recent Commercial Machines

A variety of large machines have been deployed in recent years
  - IBM SP-2, nonshared memory
  - CM-5, nonshared memory
  - Cray T3D and T3E, shared address space
  - SGI Origin 2000, coherent shared memory
- Will the rich architectural diversity of the past continue, or will all parallel machine finally look alike?

Programming Approaches

- “Automatic Parallelization” of Fortran
  - Preserves the investment in the “legacy codes”
  - Replaces programmer effort with compiler effort
  - Should be as successful as vectorization
  - Has been demonstrated to achieve 0.9-10 fold speedup, with 2.5 being typical -- Amdahl’s Law
- Alternative Languages
  - Functional -- SISAL, Id Nouveau, etc.
  - Logic -- Prolog
- These approaches failed ... the only successful programmers have programmed close to their machine; vectorized Cray Fortran continued to reign

Next Strategy to Save Legacy Codes ...

- As automatic parallelization foundered, adding “directives” or extending existing languages with new constructs came into fashion
  - Annotating a program can take as much (more?) effort than rewritting
  - HPF, HPC, HPC++, CC++, pC++, Split C, etc
- Extending an existing language requires that its semantics be preserved; parallel extensions are at odds with sequential semantics
- Approach has failed ... programs are not easily transformed: parallelism => paradigm shift from sequential

The prevailing technique

Message Passing

- Message passing libraries provide standardized facilties for programmers to define a parallel implementation of their computation ...
  - Uses existing languages, Fortran, C, etc. => save legacy
  - Interface is standard across machines
  - Lowest common denominator ... works on shared and distributed memory machines
- MP programming is difficult, subtle, error-prone; programmer implements paradigm shift
- Message passing embeds machine assumptions in code; not very portable

As many as a dozen mp libs proposed, PVM, MPI are only contenders

State of Parallel Computing

- Many companies thought parallel computing was easy
  They're gone now ...
- SGI/Cray, IBM, HP, Sun make serious parallel computers
- Seattle's Tera Computer Inc struggles to introduce a new parallel architecture, MTA
- The basic reality of large computers has changed: Servers drive the market, not speed-freaks
- The DoE's ASCI program pushes the envelope
- SMP's are ubiquitous

Budget Parallelism

- "Rolling your own" parallel computer with workstations on a network (Beowulf) is popular
- This is simple and cost effective, and the machines can be used as workstations during the business hours
- What are the impediments?
  - Nonshared memory, nonshared address space
  - Must be programmed w/ msg passing or ZPL
  - As incubator for new applications, Beowulfs do not promote the use of shared memory

Everything in parallelism seems to come down to programming

Applications

- Traditionally, NASA, DoD, DoE labs and their contractors have been big iron users
  - CFD
  - Structural
  - "Bomb Codes"

The ability to run legacy code, dusty decks, can be significant

- A huge early success was Shell Oil's seismic modelling code developed by Don Heller -- parallelism that made money
- IBM did circuit simulation on a custom SIMD
- Many claims were made but actual *working* parallelism was rare in 80s

Government Programs

Over the years numerous efforts by funding agencies have tried to jump-start high performance parallel computing

- DARPA, NSF, ONR, AFOSR, DoE, NASA, ...
- Some have been criticized as corporate welfare
- Initial thrust was on hardware

Companies have invested heavily, too

The most significant federal effort was the High Performance Computer and Communication Initiative (HPCC) in early '90s

HPCC

Took on the "whole" problem by considering hw, sw and applications involving "real" users

- Compared to predecessors, it was well planned
- Attempt at interagency coordination
- "Real" users with science or engineeering apps had to collaborate with "real" computer scientists

- Introduced the concept of a "grand challenge" problem, a computation which if it received real performance would cause better science to be done

Grand Challenge Problems

Booklets with snappy graphics enumerate these

Example classes

- Galaxy simulation
- Molecular dynamics
- Climate modelling
- Protein folding
- CFD
- Circuit simulation
- Structural analysis
- Seismic modelling

... and many, many variations

HPCC's main error -- It promised success in 5 years

HPCC Legacy

Ironically, the HPCC initiative left many "real" scientists and engineers with view that computation is a third piller in research, along with theory and experimentation

It also convinced most people of the obvious:

Its the ~~Economy~~, Stupid
SOFTWARE

As things stand, most users are writing message passing programs at considerable effort