CS252
Graduate Computer Architecture
Lecture 11

Prediction
Branches, Dependencies, and Data

October 6, 1999

Prof. John Kubiatowicz

Review: Vector Processing

- Vector Processing represents an alternative to complicated superscalar processors.
- Primitive operations on large vectors of data
- Load/store architecture:
  - Data loaded into vector registers; computation is register to register.
  - Memory system can take advantage of predictable access patterns:
    - » Unit stride, Non-unit stride, indexed
- Vector processors exploit large amounts of parallelism without data and control hazards:
  - Every element is handled independently and possibly in parallel
  - Same effect as scalar loop without the control hazards or complexity of tomasulo-style hardware
- Hardware parallelism can be varied across a wide range by changing number of *vector lanes* in each vector functional unit.

Review: Vector Processing #2

- Vector model accommodates long memory latency, doesn't rely on caches as does Out-Of-Order, superscalar/VLIW designs
- Much easier for hardware: more powerful instructions, more predictable memory accesses, fewer hazards, fewer branches, fewer mispredicted branches, ...
- What % of computation is vectorizable?
- Is vector a good match to new apps such as multimedia, DSP?

Prediction: Branches, Dependencies, Data New era in computing?

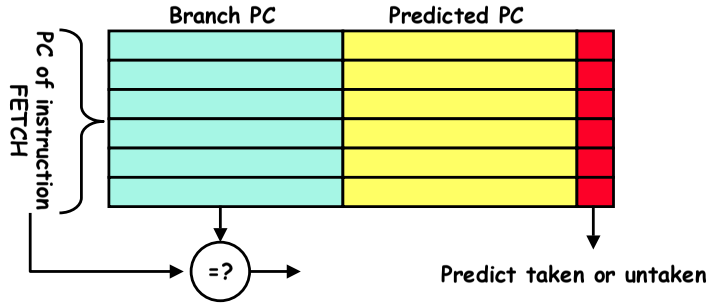
- Prediction has become essential to getting good performance from scalar instruction streams.
- We will discuss predicting branches, data dependencies, actual data, and results of groups of instructions:
  - At what point does computation become a probabilistic operation + verification?
  - We are pretty close with control hazards already…
- Why does prediction work?
  - Underlying algorithm has regularities.
  - Data that is being operated on has regularities.
  - Instruction sequence has redundancies that are artifacts of way that humans/compilers think about problems.
- Prediction $\Rightarrow$ Compressible information streams?

Dynamic Branch Prediction

- Is dynamic branch prediction better than static branch prediction?
  - Seems to be. Still some debate to this effect
  - Josh Fisher had good paper on "Predicting Conditional Branch Directions from Previous Runs of a Program." ASPLOS '92. In general, good results if allowed to run program for lots of data sets.
    - » How would this information be stored for later use?
    - » Still some difference between best possible static prediction (using a run to predict itself) and weighted average over many different data sets
  - Paper by Young et all, "A Comparative Analysis of Schemes for Correlated Branch Prediction" notices that there are a small number of important branches in programs which have dynamic behavior.

Need Address at Same Time as Prediction

- Branch Target Buffer (BTB): Address of branch index to get prediction AND branch address (if taken)
  - Note: must check for branch match now, since can't use wrong branch address (Figure 4.22, p. 273)



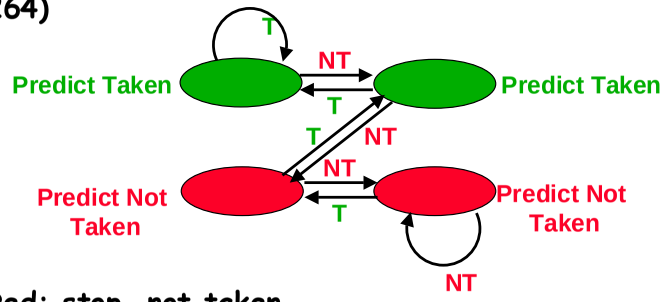
- Return instruction addresses predicted with stack
- Remember branch folding (Crisp processor)?

Dynamic Branch Prediction

- Performance = *f*(accuracy, cost of misprediction)
- Branch History Table: Lower bits of PC address index table of 1-bit values
  - Says whether or not branch taken last time
  - No address check
- Problem: in a loop, 1-bit BHT will cause two mispredictions (avg is 9 iteratios before exit):
  - End of loop case, when it exits instead of looping as before
  - First time through loop on *next* time through code, when it predicts exit instead of looping

Dynamic Branch Prediction (Jim Smith, 1981)

- Solution: 2-bit scheme where change prediction only if get misprediction *twice:* (Figure 4.13, p. 264)



- Red: stop, not taken
- Green: go, taken
- Adds *hysteresis* to decision making process

BHT Accuracy

- Mispredict because either:
  - Wrong guess for that branch
  - Got branch history of wrong branch when index the table
- 4096 entry table programs vary from 1% misprediction (nasa7, tomcatv) to 18% (eqntott), with spice at 9% and gcc at 12%
- 4096 about as good as infinite table (in Alpha 211164)

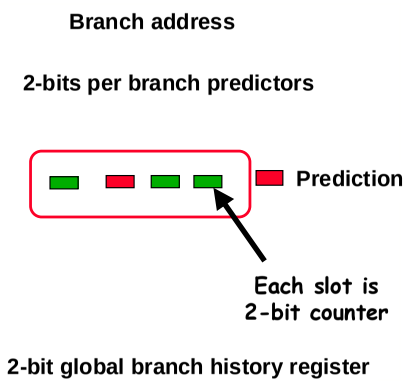
Correlating Branches

- Hypothesis: recent branches are correlated; that is, behavior of recently executed branches affects prediction of current branch
- Two possibilities; Current branch depends on:
  - Last m most recently executed branches anywhere in program Produces a "GA" (for "global adaptive") in the Yeh and Patt classification (e.g. GAg)
  - Last m most recent outcomes of same branch. Produces a "PA" (for "per-address adaptive") in same classification (e.g. PAg)
- Idea: record m most recently executed branches as taken or not taken, and use that pattern to select the proper branch history table entry
  - A single history table shared by all branches (appends a "g" at end), indexed by history value.
  - Address is used along with history to select table entry (appends a "p" at end of classification)
  - If only portion of address used, often appends an "s" to indicate "set-indexed" tables (I.e. GAs)

Correlating Branches

- For instance, consider global history, set-indexed BHT. That gives us a GAs history table.

(2,2) GAs predictor
- First 2 means that we keep two bits of history
- Second means that we have 2 bit counters in each slot.
- Then behavior of recent branches selects between, say, four predictions of next branch, updating just that prediction
- Note that the original two-bit counter solution would be a (0,2) GAs predictor
- Note also that aliasing is possible here...

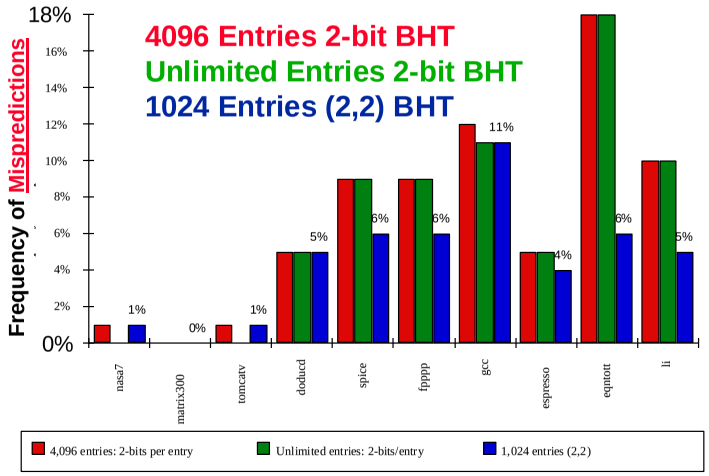


Discussion of Yeh and Patt classification

- Paper Discussion of Alternative Implementations of Two-Level Adaptive Branch Prediction

Accuracy of Different Schemes

(Figure 4.21, p. 272)



Re-evaluating Correlation

- Several of the SPEC benchmarks have less than a dozen branches responsible for 90% of taken branches:

program	branch %	static	# = 90%
compress	14%	236	13
eqntott	25%	494	5
gcc	15%	9531	2020
mpeg	10%	5598	532
real gcc	13%	17361	3214

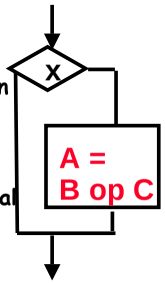
- Real programs + OS more like gcc
- Small benefits beyond benchmarks for correlation? problems with branch aliases?

Predicated Execution

- Avoid branch prediction by turning branches into conditionally executed instructions:

  if (x) then A = B op C else NOP

  - If false, then neither store result nor cause exception
  - Expanded ISA of Alpha, MIPS, PowerPC, SPARC have conditional move; PA-RISC can annul any following instr.
  - IA-64: 64 1-bit condition fields selected so conditional execution of any instruction
  - This transformation is called "if-conversion"



- Drawbacks to conditional instructions
  - Still takes a clock even if "annulled"
  - Stall if condition evaluated late
  - Complex conditions reduce effectiveness; condition becomes known late in pipeline

Dynamic Branch Prediction Summary

- Prediction becoming important part of scalar execution.
  - Prediction is exploiting "information compressibility" in execution
- Branch History Table: 2 bits for loop accuracy
- Correlation: Recently executed branches correlated with next branch.
  - Either different branches (GA)
  - Or different executions of same branches (PA).
- Branch Target Buffer: include branch address & prediction
- Predicated Execution can reduce number of branches, number of mispredicted branches

Discussion of Young/Smith paper

CS252 Administrivia

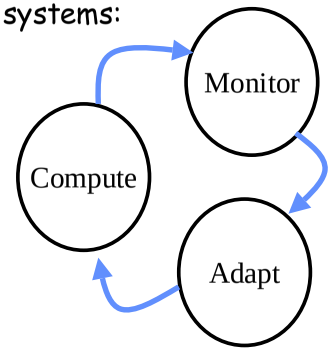
- Exam: Wednesday 10/18
  Location: 277 Cory
  TIME: 5:30 - 8:30
- Assignment due next Friday (Oct 13th)
  - Done in pairs. Put both names on papers.
- Select Project by Monday 10/23
  - Need to have a partner for this. News group/email list?
  - Web site will have a number of suggestions by tonight
  - I am certainly open to other suggestions
    - » make one project fit two classes?
    - » Something close to your research?

CS252 Projects

- DynaCOMP related (or Introspective Computing)
- OceanStore related
- Smart Dust
- ISTORE Related
- IRAM project related
- BRASS project related

DynaCOMP: Introspective Computing

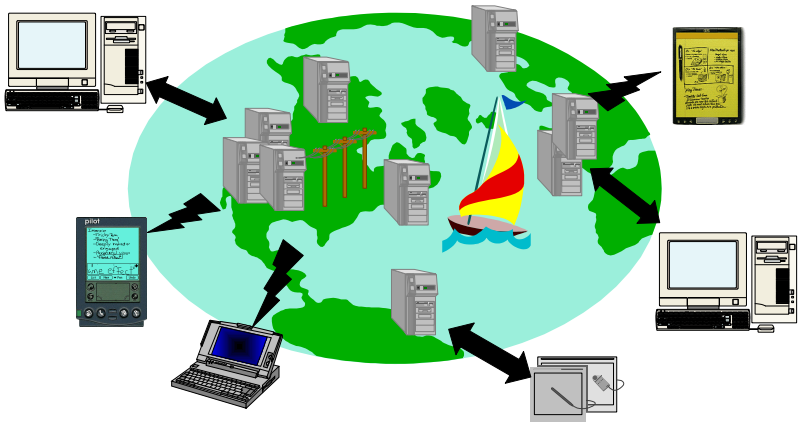
- Biological Analogs for computer systems:
  - Continuous adaptation
  - Insensitivity to design flaws
    - » Both hardware and software
    - » Necessary if can never be sure that all components are working properly…
- Examples:
  - ISTORE -- applies introspective computing to disk storage
  - DynaComp -- applies introspective computing at chip level
    - » Compiler always running and part of execution!



DynaCOMP Vision Statement

- Modern microprocessors gather profile information in hardware in order to generate predictions: Branches, dependencies, and values.
- Processors such as the Pentium-II employ a primitive form of "compilation" to translate x86 operations into internal RISC-like micro-ops.
- So, why not do all of this in software? Make use of a combination of explicit monitoring, dynamic compilation technology, and genetic algorithms to:
  - Simplify hardware, possibly using large on-chip multiprocessors built from simple processors.
  - Improve performance through feedback-driven optimization. Continuous: *Execution, Monitoring, Analysis, Recompilation*
  - Generate design complexity automatically so that designers are not required to. Use of explicit proof verification techniques to verify that code generation is correct.
- This is aptly called *Introspective Computing*
- Related idea: use of continuous observation to reduce power on buses!

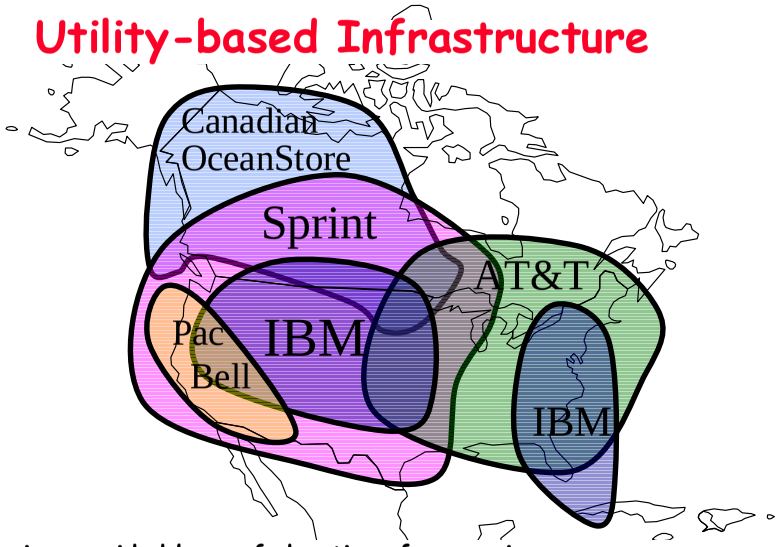
OceanStore Vision



Ubiquitous Devices ⇒ Ubiquitous Storage

- **Consumers of data move, change from one device to another, work in cafes, cars, airplanes, the office, etc.**
- **Properties *REQUIRED* for Endeavour storage substrate:**
  - **Strong Security: data must be encrypted whenever in the infrastructure; resistance to monitoring**
  - **Coherence: too much data for naïve users to keep coherent "by hand"**
  - **Automatic replica management and optimization: huge quantities of data cannot be managed manually**
  - **Simple and automatic recovery from disasters: probability of failure increases with size of system**
  - **Utility model: world-scale system requires cooperation across administrative boundaries**

Utility-based Infrastructure

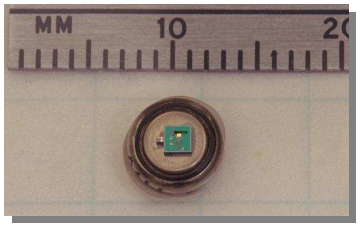


- **Service provided by confederation of companies**
  - **Monthly fee paid to one service provider**
  - **Companies buy and sell capacity from each other**

OceanStore Assumptions

- Untrusted Infrastructure:
  - Infrastructure is comprised of untrusted components
  - Only cyphertext within the infrastructure
  - Must be careful to avoid leaking information
- Mostly Well-Connected:
  - Data producers and consumers are connected to a high-bandwidth network most of the time
  - Exploit mechanism such as multicast for quicker consistency between replicas
- Promiscuous Caching:
  - Data may be cached anywhere, anytime
  - Global optimization through tacit information collection
- Operations Interface with Conflict Resolution:
  - Applications employ an operations-oriented interface, rather than a file-systems interface
  - Coherence is centered around conflict resolution

Preliminary Smart Dust Mote

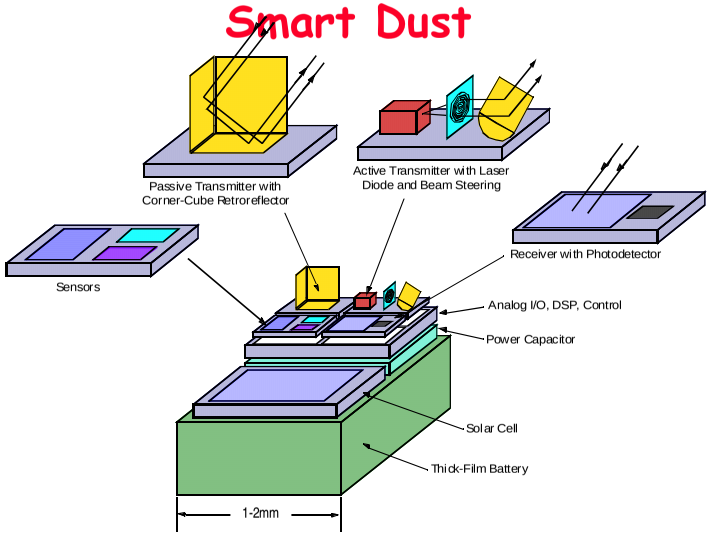


Brett Warneke, Bryan Atwood, Kristofer Pister

Berkeley Sensor and Actuator Center
Dept. of Electrical Engineering and Computer Sciences
University of California, Berkeley

HOT CHIPS 12 A Symposium on High-Performance Chips at Stanford University August 13-15, 2000

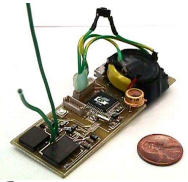
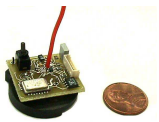
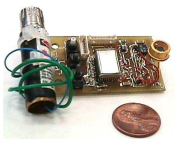
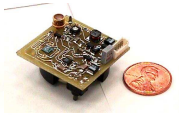
Smart Dust



COTS Dust

GOAL:
- Get our feet wet

RESULT:
- Cheap, easy, off-the-shelf RF systems
- Fantastic interest in cheap, easy, RF:
  - Industry
  - Berkeley Wireless Research Center
  - Center for the Built Environment (IUCRC)
  - PC Enabled Toys (Intel)
  - Endeavor Project (UCB)
- Optical proof of concept

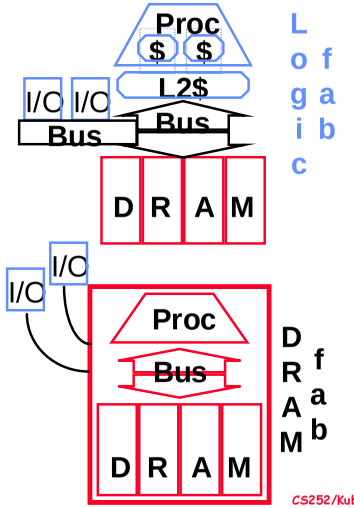
Smart Dust/Micro Server Projects

- David Culler and Kris Pister collaborating
- What is the proper operating system for devices of this nature?
  - Linux or Window is not appropriate!
  - State machine execution model is much simpler!
    - » Assume that little device is backed by servers in net.
    - » Questions of hardware/software tradeoffs
- What is the high-level organization of zillions of dust motes in the infrastructure???
- What type of computational/communication ability provides the right tradeoff between functionality and power consumption???

IRAM Vision Statement

Microprocessor & DRAM on a single chip:

- on-chip memory latency 5-10X, bandwidth 50-100X
- improve energy efficiency 2X-4X (no off-chip bus)
- serial I/O 5-10X v. buses
- smaller board area/volume
- adjustable memory size/width



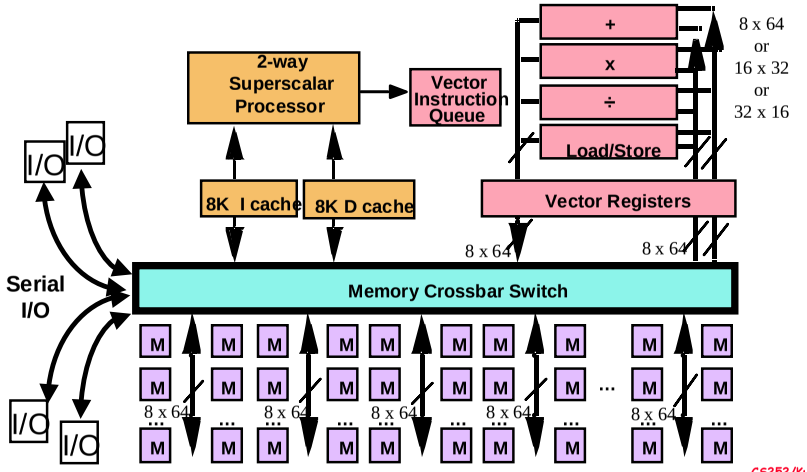
Intelligent PDA (2003?)

- Pilot PDA (todo,calendar, calculator, addresses,...)
- + Gameboy (Tetris, ...)
- + Nikon Coolpix (camera)
- + Cell Phone, Pager, GPS, tape recorder, TV remote, am/fm radio, garage door opener, ...
- + Wireless data (WWW)
- + Speech, vision recog.
- + Speech output for conversations

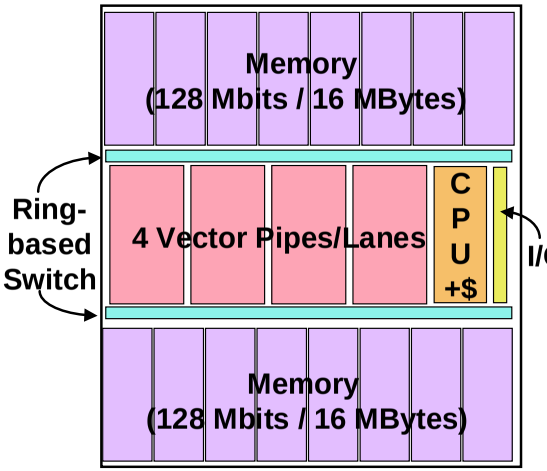
– Speech control of all devices

– Vision to see surroundings, scan documents, read bar codes, measure room

V-IRAM-2: 0.13 µm, Fast Logic, 1GHz 16 GFLOPS(64b)/64 GOPS(16b)/128MB



Tentative VIRAM-1 Floorplan



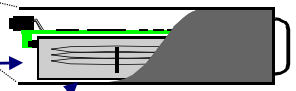
- 0.18 µm DRAM 32 MB in 16 banks x 256b, 128 subbanks
- 0.25 µm, 5 Metal Logic
- - 200 MHz MIPS, 16K I$, 16K D$
- - 4 200 MHz FP/int. vector units
- die: - 16x16 mm
- xtors: - 270M
- power: -2 Watts

ISTORE-1 hardware platform

- 80-node x86-based cluster, 1.4TB storage
  - cluster nodes are plug-and-play, intelligent, network-attached storage "bricks"
    - » a single field-replaceable unit to simplify maintenance
  - each node is a full x86 PC w/256MB DRAM, 18GB disk
  - more CPU than NAS; fewer disks/node than cluster

ISTORE Chassis
80 nodes, 8 per tray
2 levels of switches
•20 100 Mbit/s
•2 1 Gbit/s
Environment Monitoring:
UPS, redundant PS,
fans, heat and vibration
sensors...

Intelligent Disk "Brick"
Portable PC CPU: Pentium II/266 + DRAM
Redundant NICs (4 100 Mb/s links)
Diagnostic Processor



A glimpse into the future?

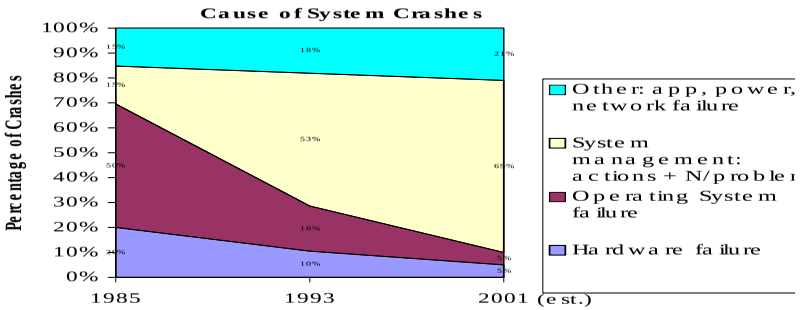
- System-on-a-chip enables computer, memory, redundant network interfaces without significantly increasing size of disk
- ISTORE HW in 5-7 years:
  - 2006 brick: System On a Chip integrated with MicroDrive
    - » 9GB disk, 50 MB/sec from disk
    - » connected via crossbar switch
    - » From brick to "domino"
  - If low power, 10,000 nodes fit into one rack!
- O(10,000) scale is our ultimate design point

ISTORE vision: Storage System of the Future

- Availability, Maintainability, and Evolutionary growth key challenges for storage systems
  - Maintenance Cost ~ >10X Purchase Cost per year,
  - Even 2X purchase cost for 1/2 maintenance cost wins
  - AME improvement enables even larger systems
- ISTORE has cost-performance advantages
  - Better space, power/cooling costs ($@colocation site)
  - More MIPS, cheaper MIPS, no bus bottlenecks
  - Compression reduces network $, encryption protects
  - Single interconnect, supports evolution of technology
- Match to future software storage services
  - Future storage service software target clusters

Is Maintenance the Key?

- Rule of Thumb: Maintenance 10X to 100X HW
  - so over 5 year product life, ~ 95% of cost is maintenance



- VAX crashes '85, '93 [Murp95]; extrap. to '01
- Sys. Man.: N crashes/problem, SysAdmin action
  - Actions: set params bad, bad config, bad app install
- HW/OS 70% in '85 to 28% in '93. In '01, 10%?

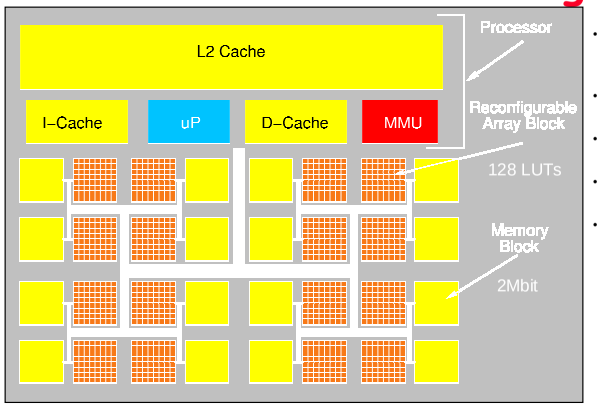
Availability benchmark methodology

- Goal: quantify variation in QoS metrics as events occur that affect system availability
- Leverage existing performance benchmarks
  - to generate fair workloads
  - to measure & trace quality of service metrics
- Use fault injection to compromise system
  - hardware faults (disk, memory, network, power)
  - software faults (corrupt input, driver error returns)
  - maintenance events (repairs, SW/HW upgrades)
- Examine *single-fault* and *multi-fault* workloads
  - the availability analogues of performance micro- and macro-benchmarks

Brass Vision Statement

- The emergence of high capacity reconfigurable devices is igniting a revolution in general-purpose processing. It is now becoming possible to tailor and dedicate functional units and interconnect to take advantage of application dependent dataflow. Early research in this area of reconfigurable computing has shown encouraging results in a number of spot areas including cryptography, signal processing, and searching --- achieving 10-100x computational density and reduced latency over more conventional processor solutions.
- BRASS: Microprocessor & FPGA on single chip:
  - use some of millions of transitors to customize HW dynamically to application

Architecture Target

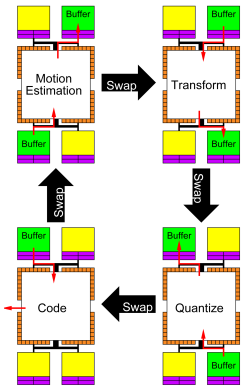


- Integrated RISC core + memory system + reconfigurable array.
- Combined RAM/Logic structure.
- Rapid reconfiguration with many contexts.
- Large local data memories and buffers.
- These capabilities enable:
  - hardware virtualization
  - on-the-fly specialization

SCORE: Stream-oriented computation model

Goal: Provide view of reconfigurable hardware which exposes strengths while abstracting physical resources.

- Computations are expressed as data-flow graphs.
- Graphs are broken up into compute pages.
- Compute pages are linked together in a data-flow manner with streams.
- A run-time manager allocates and schedules pages for computations and memory.



Summary #1 Dynamic Branch Prediction

- Prediction becoming important part of scalar execution.
  - Prediction is exploiting "information compressibility" in execution
- Branch History Table: 2 bits for loop accuracy
- Correlation: Recently executed branches correlated with next branch.
  - Either different branches (GA)
  - Or different executions of same branches (PA).
- Branch Target Buffer: include branch address & prediction
- Predicated Execution can reduce number of branches, number of mispredicted branches

Summary #2

- Prediction, prediction, prediction!
  - Over next couple of lectures, we will explore prediction of everything! Branches, Dependencies, Data
- The high prediction accuracies will cause us to ask:
  - Is the deterministic Von Neumann model the right one???