Wimpy Nodes Make Light of Hefty Data

Dave Andersen
Low-Power Computing
Carnegie Mellon University

Running efficiently

- Last few lectures were about idling well
- Let's get back to running efficiently when you're going full-bore

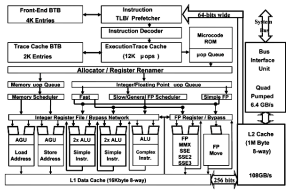
Remember The Memory Wall

- DRAM access latency increased from ~20 to ~400 clock cycles (92-2003)
- Memory bandwidth improved faster, but not entirely commensurate
- Processors aggressively use bandwidth to hide latency: deep pipelines, fetch before use, prefetch, etc.

But...

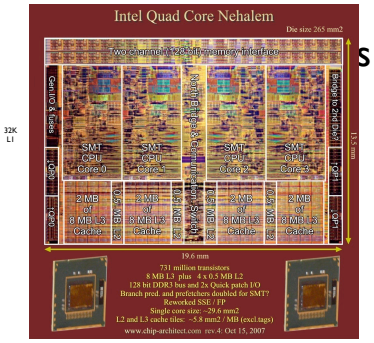
- Deep pipelines are hard if you have branches
- So you add branch prediction
- And out-of-order execution so that you can wait longer for fetches
- And really, really (really) big caches

p4 31-stage pipeline



~99% branch prediction.
3.4Ghz (!)

Turned out bad.



If you can't measure...

- you can't improve it
- Think about how much play benchmarks such as TPC-C get -- they provide a basis for comparison across time and different systems
- Need the same measures for energy efficiency (and PUE isn't it! PUE just measures non-computer overhead)

How to pick?

- Scale-forward: Benchmarks must remain valid for a while. Tuning parameters help -- in this case, size.
- Simple, but comprehensive -- yes, it's hard to have both.
- Hard to game -- eg, CPU vendors have been known to tune compilers (or cores!) specifically for Spec benchmarks. Recognize spec code, output hand-optiized code for entire solution.
- Sort is actually pretty nice. It's also the core of MapReduce.

Real Energy Metrics

- At the wall - excludes PUE, but PUE can often be improved independent of computer architecture... seems like right decision.

Conventional Systems

- Server: 299 sec, 1206 SRecs/J
- Laptop: 727 sec, 3270 SRecs/J
- Server is fastest, but is bottlenecked by its disk controller.
- Laptop uses 1/6th the power for ~1/2 the speed. And its CPU is still bored...
- Increase I/O? Shrink CPU?

How to win

- The key is *balance*: Don't waste power on components that are bored
  - Thinking back: scaling is less effective than not going there in the first place
- Very efficient code / algorithms (only matters if you're running near the limit of the hw -- see "balance")
- Efficient h/w

Try #1

- Server with low-ish power Xeon, boot from low-power laptop drive (not really used), and give it lots of disks.
- Good I/O controller; use both channels.
- Result: 3863 SRecs/J. -> Beats the unbalanced laptop. But can get better.

Their winner

- 2.33 Ghz laptop CPU: Core 2 duo (34W)
  - Pinned to 1660 Mhz..
- 2GB of lower-power DRAM (~1.9W)
- 13 laptop disks (5400RPM, 1-1.8W)
- Two disk controllers + one on motherboard
- 11,300 SRecs/J
  - 24% faster than GPUTeraSort, 3x less power

Playing with efficiency

- Efficiency *increases* going from 996Mhz to 1660 -- storage/IO/Mem is idle, wasting power
- Efficiency same or decreases after -- CPU becomes overpowered (or lowers its frequency anyway)

But these guys didn't go far enough...

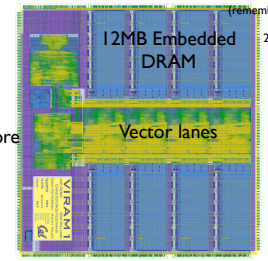
It's about Data

- DRAM has very high bandwidth internally
- Hard drives have very high b/w internally
- Flash can be enormously parallel
- Getting all of this in & out of the CPU is painful, high latency, and power-hungry

So mix them up!

iRAM

200Mhz
(remember, this was research...)



2W power draw!

scalar core

Vector?

- iRAM chip has a *lot* of bandwidth... how to use it?
- Back to the future: vector supercomputers massive data parallelism, explicitly process 4 64-bit data items with 1 instruction
- 3.2 GOPS for 32-bit integer ops. not bad.
- Key: After they built it, they needed explicit parallelism
- Key2: Embedded DRAM is hard...

Today's iRAM - GPUs?

On to Disks

- If the memory wall is bad - the IO wall is horrible!
- So: mix them up!
- Active Disks

Data Mining Appliances

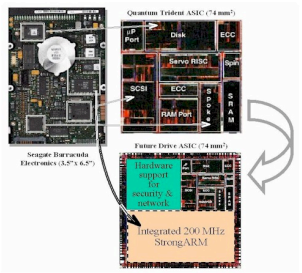
- Going crazy
- IBM Power 595 9119-FHA ($17M)
- 10,992 15KRPM drives
- 4TB DRAM ($6M worth)
- 128 5.0Ghz Power6 CPUs

Historical

- MS TerraServer - 324 disks, 8x440Mhz alpha CPUs
- Compact ProLiant TPC-C 4x400 Mhz pentiums, 141 disks
- Digital AlphaServer TPC-D, 12 612Mhz Alpha CPUs, 521 disks

That's a lot of disks...

- Unstructured data-mining (or semi-structured): *scan-intensive* workloads
- From simple - text mining with hadoop/etc
- to complex - image mining with Diamond
- Video searching



How to partition functionality?

- Common approach: data restriction at disks
  - Simple processing that can eliminate most of the data
  - Then do the complex stuff at the server
- Diamond, Netezza, etc., all do this now. Works really well *if* workload is agreeable

Gordon