Revisiting the Future 😊

Erik Hagersten
Uppsala University
Sweden

DARK2 in a nutshell
1. Memory Systems (caches, VM, DRAM, microbenchmarks, ...)
2. Multiprocessors (TLP, coherence, interconnects, scalability, clusters, ...)
3. CPUs (pipelines, ILP, scheduling, Superscalars, VLIWs, embedded, ...)
4. Future: (physical limitations, TLP+ILP in the CPU,...)

How do we get good performance?
- Creating and exploring:
  1) Locality
     a) Spatial locality
     b) Temporal locality
     c) Geographical locality
  2) Parallelism
     a) Instruction level (ILP)
     b) Thread level (TLP)
     c) Memory level (MLP)
Doubling (or Halving) times

- Dynamic RAM Memory (bits per dollar): 1.5 years
- Average Transistor Price: 1.6 years
- Microprocessor Cost per Transistor Cycle: 1.1 years
- Total Bits Shipped: 1.1 years
- Processor Performance in MIPS: 1.8 years
- Transistors in Intel Microprocessors: 2.0 years
- Microprocessor Clock Speed: 2.7 years

Old Trend 1: Deeper pipelines
Exploring ILP (instruction-level parallelism)

Old Trend 2: Wider pipelines
Exploring more ILP

Old Trend 3: Deeper memory hierarchy
Exploring access locality

Are we hitting the wall now?

Pop: Can the transistors be made even smaller and faster?

Possible path, but requires a paradigm shift

Business as usual...

Well uhm ... The transistors can be made smaller and faster, but there are other problems 😊
Microprocessors today: Whatever it takes to run one program fast.

Exploring ILP (instruction-level parallelism):
- Faster clocks
- Deep pipelines
- Superscalar Pipelines
- Branch Prediction
- Out-of-Order Execution
- Trace Cache
- Speculation
- Predicate Execution
- Advanced Load A1
- Return Address Stack
- ...

Bad News #1:
We have already explored most ILP

Bad News #2:
Long wire delay → slow CPUs

Bad News #3:
Memory latency/bandwidth is the bottleneck...

Bad News #4:
Power is the limit

Power consumption is the bottleneck
- Cooling servers is hard
- Battery lifetime for mobile computers
- Energy is money

Dynamic effect is proportional to:
\[ P_{\text{dyn}} \sim \text{Capacitance} \times \text{Frequency} \times \text{Voltage}^2 \]

Now What?
#1: Running out of ILP
#2: Wire delay is starting to hurt
#3: Memory is the bottleneck
#4: Power is the limit
Solving all the problems: exploring threads parallelism

#1: Running out of ILP
- feed one CPU with instr. from many threads

#2: Wire delay is starting to hurt
- Multiple small CPUs with private L1$

#3: Memory is the bottleneck
- memory accesses from many threads (MLP)

#4: Power is the limit
- Lower the frequency -> lower voltage

Bad News #1: Not enough ILP 1(2)
- feed one CPU with instr. from many threads

Bad News #2: Wire delay
- Multiple small CPUs with private L1$

SMT: Simultaneous Multithreading
"Combine TLP&ILP to find independent instr."

Thread-interleaved

Historical Examples:
- Denelcor, HEP, Tera Computers [B. Smith] 1984
  Each thread executes every n:th cycle in a round-robin fashion
  -- Poor single-thread performance
  -- Expensive (due to early adoption)
- Intel’s Hyperthreading (2002??)
  -- Poor implementation
**Bad News #4: Power consumption**

- Lower the frequency → lower voltage

\[ P_{\text{dyn}} = C \cdot f \cdot V^2 \approx \text{area} \cdot \text{freq} \cdot \text{voltage}^2 \]

- \( P_{\text{dyn}}(C, f, V) = CFV^2 \)

<table>
<thead>
<tr>
<th>CPU freq=f</th>
<th>VS.</th>
<th>CPU freq=f/2</th>
<th>CPU freq=f/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{\text{dyn}}(C, f, V) = CFV^2 )</td>
<td>( P_{\text{dyn}}(2C, f/2, V) &lt; CFV^2 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( P_{\text{dyn}}(C, f, V) = CFV^2 \)

\[ \frac{P_{\text{dyn}}(C, f/2, V)}{P_{\text{dyn}}(C, f, V)} < \frac{1}{2} \]

**Example: Freq. Scaling**

- 20% higher freq.
- 20% lower freq.
- 20% lower freq. Two cores

**High-performance single-core CPUs:**

- Athlon 64, Athlon 64 FX and Athlon 64 X2 family, dual-core desktop processors.
- Opteron, dual- and quad-core server/workstation processors.
- Phenom, triple- and quad-core desktop processors.
- Core Duo, a dual-core processor.
- Core 2 Duo, a dual-core processor.
- Core 2 Quad, a quad-core processor.
- Core i7, a quad-core processor, the successor of the Core 2 Duo and the Core 2 Quad.
- Itanium 2, a dual-core processor.
- Pentium D, a dual-core processor.
- Teraflops Research Chip (Polaris), an 3.16 GHz, 80-core processor prototype, which the company says will be released within the next five years.
- Xeon dual-, quad- and hexa-core processors.

**Multicore CPUs:**

- PowerPC 970MP, a dual-core processor, used in the Apple Power Mac G5.
- Core Duo, a dual-core processor.
- Core 2 Duo, a dual-core processor.
- Core 2 Quad, a quad-core processor.
- Core i7, a quad-core processor.
- Itanium 2, a dual-core processor.
- Pentium D, a dual-core processor.
- Teraflops Research Chip (Polaris), an 3.16 GHz, 80-core processor prototype, which the company says will be released within the next five years.
- Xeon dual-, quad- and hexa-core processors.

**Example: Freq. Scaling**

- 20% higher freq.
- 20% lower freq.
- 20% lower freq. Two cores

**High-performance single-core CPUs:**

- Athlon 64, Athlon 64 FX and Athlon 64 X2 family, dual-core desktop processors.
- Opteron, dual- and quad-core server/workstation processors.
- Phenom, triple- and quad-core desktop processors.
- Core Duo, a dual-core processor.
- Core 2 Duo, a dual-core processor.
- Core 2 Quad, a quad-core processor.
- Core i7, a quad-core processor, the successor of the Core 2 Duo and the Core 2 Quad.
- Itanium 2, a dual-core processor.
- Pentium D, a dual-core processor.
- Teraflops Research Chip (Polaris), an 3.16 GHz, 80-core processor prototype, which the company says will be released within the next five years.
- Xeon dual-, quad- and hexa-core processors.
Many shapes and forms...

Darling, I shrunk the computer

How to create threads?

Design Examples CMPs

Intel Core2 Quad, 45 nm
**Intel Dunnington, 45 nm**

Hyper Transport

- L3 2MB
- X-bar
- L2 512kB
- L2 512kB
- L2 512kB
- L2 512kB
- CPU
- CPU
- CPU
- CPU

**AMD Barcelona, 65 nm**

Hyper Transport

- L3 8MB
- X-bar
- L2 512kB
- L2 512kB
- L2 512kB
- L2 512kB
- CPU
- CPU
- CPU
- CPU

**AMD MC System Architecture**

**AMD Shanghai, 45 nm**

Hyper Transport

- L3 8MB
- X-bar
- L2 512kB
- L2 512kB
- L2 512kB
- L2 512kB
- CPU
- CPU
- CPU
- CPU

**AMD Istanbul, 6 cores**

**Intel: Nehalem, Core i7**

- L3 8MB
- X-bar
- L2 256kB
- L2 256kB
- L2 256kB
- L2 256kB
- CPU, 2 thr.
- CPU, 2 thr.
- CPU, 2 thr.
- CPU, 2 thr.
Nehalem “Core i7”, 45nm

**AMD Magny-Cours, 2010 (??)**

**Intel: Core i7 2010 (??)**

**Sun Niagara, 2005!!**

**Niagara 2**

**Niagara Chip**
TILERA Architecture

- 64 cores connected in a mesh
- Local L1 + L2 caches
- Shared distributed L3 cache
- Linux + ANSI C
- New IDE
- New 2DE
- Stream computing

IBM Cell Processor

- IBM Cell
- Mem
- Front end

So-called accelerators

- Sits on the IO bus (!!!)
- GP Graphics processors, aka GPGPU? [e.g. NVIDIA, AMD/ATI]
- Specialized accelerators? [e.g., FPGA/Mitrionics, ASIC/ClearSpeed]
- Specialized languages for the above? [CUDA, Ct, Rapid Mind, Open-CL, ...]

Intel Larrabee 2010 (??)

- 32 single in-order cores
- 4 threads /core
- 16 wide SIMD (single precision)/8 wide (double)

Rumers on the net. Not based of actual facts.

Multicore CPUs:

- Intel Larrabee 2010 (??)
- AMD Phenom, a quad-core microprocessor with integrated graphics
- IBM Power5, a 32-bit microprocessor, 16 powerPC cores
- AMD Cell, a nine-core processor with one general purpose PowerPC core and eight synergistic processing units
- Intel Larrabee 2010 (??)
- NVIDIA Tegra Series, a seven-core processor

Design Issues for Multicores

Erik Hagersten
Uppsala University
Sweden

CMP bottlenecks/points of optimization
- Performance per Watt?
- Performance per memory byte?
- Performance per bandwidth?
- Performance per $?
- ...
- How large fraction of a CMP system cost is the CPU chip?
- Should the execution (MIPS/FLOPS) be viewed as a scarce resource?

DRAM issues
"Rock will have more than 1000 memory chips per Rock chip" [M. Trembley, Sun Fellow at ICS 2006]
- Memory will dominate cost?
  Fewer "open pages" accessed due to interleaving of several threads
  Far memory ➔ long latency & low per-pin BW
  Pushing dense memory technology/packaging?

Bandwidth issues
- #pins is a scarce resource!
- every pin should run at maximum speed
  ➔ external memory controllers?
  ➔ off-chip cache?
  ➔ is there room for multi-CMP?
  ➔ is this maybe a case for multi-CMP?

Shared Bottlenecks

Thread Interaction
- Coherence traffic
- Communication utilization
- Load imbalance
- Synchronization
- False sharing

Thread Interaction

Thread Interaction
**Example: Thread Interaction (True Communication)**

More cache sharing means more misses?

**Multicore Challenges: Non-uniformity Communication**

L2 sharing (here: pair-wise)

**Multicore Challenges (Multisocket) Non-uniformity Communication**

L2 sharing (here: pair-wise)

**Multicore Challenges (Multisocket) Non-uniformity Memory**

**Example: Poor Throughput Scaling**

Example: 470.1bm

Throughput (as defined by SPEC): Amount of work performed per time unit when several instances of the application is executed simultaneously.

**Throughput Scaling, More Apps**

SPEC CPU 20006 FP Throughput improvements on 4 cores
Nerd Curve: 470.lbm

- Miss rate (excluding HW prefetch effects)
- Utilization, i.e., fraction cache data used (scale to the right)
- Possible miss rate if utilization problem was fixed

Running one thread
Running four threads

Less amount of work per memory byte moved

Better Memory Usage!
Example: 470.lbm
Modified to promote better cache utilization

Cache size
Miss rate
Utilization
Possible miss rate

Nerd Curve (again)

- Miss rate (excluding HW prefetch effects)
- Utilization, i.e., fraction cache data used (scale to the right)
- Possible miss rate if utilization problem was fixed

Running four threads
Running one thread

Twice the amount of work per memory byte moved

More transistors ➔ More Threads

- Warning:
  - # transistors grows exponentially ➔ # threads can grow exponentially
  - Can memory BW keep up?

BW in the Future?

Computation vs Bandwidth

Source: International Technology Roadmap for Semiconductors (ITRS)

Capacity or Capability Computing?

Capacity? (=several sequential jobs)

- Memory requirement?
- Sharing in cache?
- Memory bandwidth requirement?

Memory: the major cost of a CMP system!
How do we utilize it the best?
- Once the workingset is in memory, work like crazy!

- Capability computing suits CMPs the best (in general)
Fat or narrow cores?

- Fat:
  - Fewer cores but...
  - wide issue?
  - O-O-O?

- Narrow: More cores but...
  - narrow issue?
  - in-order?
  - have you ever heard of Amdahl?
  - SMT, run-ahead, execute-ahead ... to cure shortcomings?

Read:
Maximizing CMP Throughput with Mediocre Cores
Davis, Laudon and Olukotun, PACT 2006

Cores vs. caches

- Depends on your target applications...
- Niagara’s answer: go for cores
  - In-order 5-stage pipeline
  - 8 cores a’ 4 SMT threads each ➔ 32 threads,
  - 3MB shared L2 cache (96 kb/thread)
  - SMT to hide memory latency
  - Memory bandwidth: 25 GB/s
  - Will this approach scale with technology?
- Others: go for cache
  - 2-4 cores for now

Cache Interference in Shared Cache

- Cache sharing strategies:
  1. Fight it out!
  2. Fair share: 50% of the cache each
  3. Maximize throughput: who will benefit the most?

Read:
STATSHARE: A Statistical Model for Managing Cache Share via Decay
Pavlos Petoumenos et al in MOBS workshop ISCA 2006
Predicting the inter-thread cache contention on a CMP
Chandra et al in HPCA 2005

Hiding Memory Latency

- O-O-O
- HW prefetching
- SMT
- Run-ahead/Execute-ahead

Questions for the Future

- What applications?
- How to get parallelism and data locality?
- Will funky languages see a renaissance?
- Will automatic parallelizing return?
- Are we buying:
  - compute power,
  - memory capacity, or
  - memory bandwidth?
- Will the CPU market diverge into desktop/capacity CPUs again?
- How to debug? ...
- A non-question: will it happen?