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)

Ray Kurzweil pictures
www.KurzweilAI.net/pps/WorldHealthCongress/
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?

Performance [log]

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 😐

Instruction-Level Parallelism (ILP) in Superscalar Pipelines

START:
J = K + L
G = H + I
D = E + F
A = G + C
MEM[X] = MEM[X] + 14
X = X + 1
IF X < 1000 GOTO START:

K+L START:
Issue Logic
Microprocessors today: Whatever it takes to run one program fast.

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

Bad News #1:
We have already explored most ILP (instruction-level parallelism).

Bad News #2:
Looong wire delay \(\rightarrow\) slow CPUs

Quantitative data and trends according to V. Agarwal et al., ISCA 2000
Based on SIA (Semiconductor Industry Association) prediction, 1999

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

Sloow Memory

\[ A = B + C: \]
- Read B: 0.3 - 100 ns
- Read C: 0.3 - 100 ns
- Add B & C: 0.3 ns
- WriteA: 0.3 - 100 ns

Span -- Fraction of chip reachable in one cycle

<table>
<thead>
<tr>
<th>Year</th>
<th>Feature size (micron)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>.25</td>
</tr>
<tr>
<td>2005</td>
<td>.18</td>
</tr>
<tr>
<td>2010</td>
<td>.13</td>
</tr>
<tr>
<td>2015</td>
<td>.10</td>
</tr>
<tr>
<td></td>
<td>.070</td>
</tr>
<tr>
<td></td>
<td>.050</td>
</tr>
<tr>
<td></td>
<td>.0035</td>
</tr>
</tbody>
</table>

Latency for a 5mm trace (RC mode)
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
  ~ Frequency
  ~ 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

Sloow Memory
Bad News #1: Not enough ILP 2(2)

- feed one CPU with instr. from many threads

SMT: Simultaneous Multithreading

"Combine TLP & ILP to find independent instr."

Thread-interleaved

- Each thread executes every n:th cycle in a round-robin fashion
- Historical Examples (1984..)
  Denelcor, HEP, Tera Computers [B. Smith]
- Poor single-thread performance
- Expensive (due to early adoption)

Bad News #2: Wire delay

- Multiple small CPUs with private L1$
**CMP: Chip Multiprocessor**

- more TLP & geographical locality

**Bad News #3: memory latency/bandwidth**

- memory accesses from many threads (MLP)

**Bad News #4: Power consumption**

- Lower the frequency 
- lower voltage

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

<table>
<thead>
<tr>
<th>Frequency Scaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq (=f)</td>
</tr>
<tr>
<td>freq (=f/2)</td>
</tr>
<tr>
<td>freq (=f/2)</td>
</tr>
</tbody>
</table>

**Example: Freq. Scaling**

- 20% higher freq.
- 20% lower freq.
- 20% lower freq.

**TLP \(\Rightarrow\) MLP**

- Thread-Level Parallelism \(\Rightarrow\) Memory-Level Parallelism (MLP)
Now, everyone is doing it!

“Intel have 10 projects in the works that contain four or more computing cores per chip”
[Paul Otellini, Intel Chief Executive at IDF fall 2005]

“How to get parallelism?

“Today, processors with multiple CPUs and a large cache on a single chip are becoming common. Attempts to tease the parallelism out of a sequential program automatically haven’t worked out very well. We need better education, better languages, and better tools, since building concurrent programs is hard.”
[Andrew Herbert, Director of Microsoft Cambridge Research Lab, May 2005]
CMP: Chip Multiprocessor

Looks and Smells Like an SMP?

Well, how about:
- Cost of parallelism?
- Cache capacity per thread?
- Memory bandwidth per thread?
- Cost of thread communication? …

Trends (my guess!)

Design Issues for CMPs

Erik Hagersten
Uppsala University
Sweden
Niagara I

Sun Microsystems

Niagara Chip

Sun Microsystems

Niagara

4 x DDR-2 = 25GB/s (!)

TILERA Architecture

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

Shared L2!

Shared caches: Good or bad?
**AMD Barcelona, 65 nm**

- Hyper Transport
- DDR-2
- L3 2MB (→ 8MB)
- X-bar
- L2$ 512kB
- D$ 64kB
- Is 64kB
- CPU

**Intel Quad, 45 nm**

- South Bridge
- I/O
- North Bridge
- DRAM
- Front-side Bus (FSB)
- Die 1
  - L2$ 6MB
  - D$ 64kB
  - Is 64kB
  - CPU
- Die 2
  - L2$ 6MB
  - D$ 64kB
  - Is 64kB
  - CPU

**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?

How Acumem can Help

Example: 470.lbm

Original code executed in four instances

Modified according to Acumem’s advice
How Common is Poor Throughput?

![Quad Core Throughput Scaling](image)

More transistors → More Threads

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

A case for: off-chip memory controllers?

- CPU
- CPU
- CPU
- Mem

![SIA Prediction: #Transistors vs. pin bandwidth](image)

![Medium-speed links](image)
A case for: off-chip memory controllers?

A case for off chip L3 cache controllers

A case for multiple small CMPs

Capacity or Capability Computing?

Capacity? (≈several sequential jobs)
or
Capability? (≈one parallel job)

Issues:
- 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

Privat/Shared Caches?
Early Commercial CMPs

Cache Interference?

- 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
- SMT
- Run-ahead/Execute-ahead

Questions for the Future

- What applications?
- How to get parallelism and data locality?
- Will funky languages see a renascence?
- 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?

Major points for CMP designers:

- Start from a clean slate – this is not an SMP!
  - bandwidth & memory capacity is king
  - a case for memory compression [M. Ekman, ISCA 2005]
  - on-chip memory controllers is not a given
  - off-chip cache is not a given
- New rules for coherence
  - snooping is still in the ballgame
  - directory-based only if needed
  - cache-line size is not a given ...
- Consider cheap options for hiding memory latency
- Trade-off core-obesity/#cores/cache_size can be very application specific!
- Manage cache sharing
- Capacity/capability computing is very different
- ...

...