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/

Evolution of Computer Power/Cost

MIPS per $1000 (1998 Dollars)

Year 2000

Manual Calculation

Bacterium

1900
1920
1940
1960
1980
2000
2020

Babbage's Class 16
IBM Tabulator
Monroe Calculator
ASCC (Mark 1)

IBM 650
Burroughs 5000
IBM 1620

IBM 7040
IBM 560/75
DEC-10
IBM 1130
ENIAC
Colossus

VAX 11/780
DEC VAX 11/780
DEC-10
DG Nova
SDS 920

IBM 7090
IBM 704

Whirlwind

UNIVAC I

CDC 6600
DEC PDP-10

Mac II
Macintosh-128K
Commodore 64
IBM PC
Sun-2

Apple II
DQ Eclipse

Power Tower 1600s
AT&T Globalyst 6000
IBM PS/2 90
MPC 264
Gateway 486DX2/86
PowerMac 8100/80
Gateway G4-200
Mac II

1000

1

1 Million

1000

100

1 Billion

1 Million

100

10

1
Growth in Supercomputer Power

Flops (floating point operations)

- Blue Gene/P
- ASCI Purple
- ASCI White
- ASCI Red
- Sx-6
- Columbia
- Earth Simulator
- MCI/1024
- CM-5/1024
- CF-PACS/2049
- Num. Wind Tunnel
- Num. Wind Tunnel

Year

Doubling time = 1.2 years

Required for Human Brain Functional Simulation (2013)

Required for Human Brain Neural Simulation for Uploading (2025)

Logarithmic Plot

Trendlines

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

Processor Performance
(MIPS)

Logarithmic Plot

Year
MIPS

Doubling time: 1.8 years

8008 8050 8066 286 336 436 Xeon
Pentium 4
Pentium II
Pentium

© Erik Hagersten www.docs.uu.se/~eh
Dynamic RAM Price
Bits per Dollar at Production
(Packaged Dollars)

Logarithmic Plot

Note that DRAM speeds have increased during this period.
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

Thread 1

Issue logic

More pipelines + Deeper pipelines

Need more independent instructions

Regs

150 cycles

Mem

1 GB
Old Trend3: Deeper memory hierarchy
Exploring access locality

<table>
<thead>
<tr>
<th>Level</th>
<th>Access Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem</td>
<td>150 cycles</td>
</tr>
<tr>
<td>2MB</td>
<td>30 cycles</td>
</tr>
<tr>
<td>64kB</td>
<td>10 cycles</td>
</tr>
<tr>
<td>2kB</td>
<td>2 cycles</td>
</tr>
</tbody>
</table>

Thread 1

Issue logic

Regs

<table>
<thead>
<tr>
<th>Level</th>
<th>Access Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mem</td>
<td>150 cycles</td>
</tr>
<tr>
<td>2MB</td>
<td>30 cycles</td>
</tr>
<tr>
<td>64kB</td>
<td>10 cycles</td>
</tr>
<tr>
<td>2kB</td>
<td>2 cycles</td>
</tr>
</tbody>
</table>

kr

€

£
Are we hitting the wall now?

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

Performance $[\log]$

Now

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 = B + C
MEM[X] = MEM[X] + 14
X = X + 1
IF X < 1000 GOTO START:

K + L START:

Issue Logic
Instruction-Level Parallelism (ILP) in Superscalar Pipelines

START:
J = K + L
G = H + I
D = E + F
A = B + 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 ➔ 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 #2
Looong wire delay ➔ slow CPUs

Span -- Fraction of chip reachable in one cycle

Year
Feature size (micron)
.25  .18  .13  .10  .070  .050  .0035

SIA
100%

0.4 - 1.4 %
Bad News #3:
Memory latency/bandwidth is the bottleneck...

A = B + C:
- Read B: 0.3 - 100 ns
- Read C: 0.3 - 100 ns
- Add B & C: 0.3 ns
- Write A: 0.3 - 100 ns

Latency: 0.3 - 100 ns
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

Sloooow 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.”

Issue logic

Thread 1

PC...

Thread N

PC

Regs 1...

Regs N

Mem

Thread N

PC...

Regs 1...

Regs N

Thread 1

PC...

Regs 1...

Regs N

$EUROSEK

$EUROSEK

$EUROSEK
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)

Sloooow memory

TLP ⇒ MLP
Thread-Level Parallelism ⇒ Memory-Level Parallelism (MLP)
Bad News #4: Power consumption

- Lower the frequency → lower voltage

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

<table>
<thead>
<tr>
<th>CPU freq=f</th>
<th>VS.</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, &lt;V) &lt; CfV^2 ]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU freq=f</th>
<th>VS.</th>
<th>CPU CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq = f/2</td>
<td>[ P_{\text{dyn}}(C, f/2, &lt;V) &lt; \frac{1}{2} CfV^2 ]</td>
<td>Throughput &lt; 2x</td>
</tr>
</tbody>
</table>
Example: Freq. Scaling

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

(freq: 1.20, speed: 1.13, pow: 1.51)

(freq: 0.80, speed: 0.87, pow: 0.51)

(freq: 0.80, speed: 0.87, pow: 0.51)
Around the corner...
Chip Multiprocessors (CMP)
Simultaneous Multithreading (SMT)
Darling, I shrunk the computer

Mainframes

Super Minis:

Microprocessor:

Chip Multiprocessor (CMP): A multiprocessor on a chip!

Paradigm Shift

Sequential execution (≈ one program)

Parallel execution (TLP)
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]

"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]

How to get parallelism?
What thread parallelism?

**Capacity Computing**

- Prog1
- Prog2
- OS
  - Thread1
  - Thread2
  - Thread3

**Capability Computing**

- Seq. Prog.
- Parallelizing compiler
- Thread speculation
  - Thread1
  - Thread2
  - Thread3

- Parallel Prog
CMP: Chip Multiprocessor

- **Simple fast CPU core (w/ SMT?)**
- + local caches

**Diagram Details:**
- **Mem I/F**
- **External I/F**
- **L2$**
- **$1**
- **CPU**
- **$1**
- **$1**
- **$1**
- **$1**
- **t threads**
- **37**
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!)

Threads/Chip

Transistors/Thread

Memory/Chip

Cache/Thread

Bandwidth/Thread

Thr. Comm. Cost (temporal)
Design Issues for CMPs

Erik Hagersten
Uppsala University
Sweden
Niagara I

Sun Microsystems
Niagara Chip

Sun Microsystems
Niagara

$4 \times \text{DDR-2} = 25\text{GB/s (!)}$

- Memory ctrl
- Memory ctrl
- Memory ctrl
- Memory ctrl

- L2
- L2
- L2
- L2

- Xbar = 134 GB/s

- L1I
- L1D (wt)
- L1I
- L1D (wt)

- CPU
  - ... 8 ...
  - CPU

Shared L2!

Shared caches: Good or bad?
TILER A Architecture

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

The tile is the basic building block
AMD Barcelona, 65 nm

Hyper Transport

DDR-2

L3 2MB (→ 8MB)

X-bar

L2$ 512kB

L2$ 512kB

L2$ 512kB

L2$ 512kB

D$ 64kB  I$ 64kB

D$ 64kB  I$ 64kB

D$ 64kB  I$ 64kB

D$ 64kB  I$ 64kB

CPU

CPU

CPU

CPU
Intel Quad, 45 nm

South Bridge

North Bridge

North Bridge ↔ DRAM

South Bridge ↔ I/O

Front-side Bus (FSB)

Die 1

L2$ 6MB

Die 2

L2$ 6MB

CPU

CPU

CPU

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?
Yesterday...

Slooow memory

CPU
L1 cache
L2 Cache

A = B + C

1 cycle
1 - 200 c

200+ c
15 c
New Shared Bottlenecks

L2 Cache

Shared Resources

Bandwidth

CPU L1 CPU L1 CPU L1 CPU L1

CPU L1 CPU L1 CPU L1

CPU L1 CPU L1 CPU L1

CPU L1 CPU L1 CPU L1

CPU L1 CPU L1 CPU L1
How Acumem can Help

Quad Core Throughput Scaling

Example: 470.lbm

Modified according to Acumem’s advice

Original code executed in four instances
How Common is Poor Throughput?

Quad Core Throughput Scaling
More transistors  ➔ More Threads

**Warning:**

- # transistors grows exponentially
  ➔ # threads can grow exponentially
- Can memory BW keep up?
SIA Prediction: #Transistors vs. pin bandwidth
A case for: off-chip memory controllers?
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
- ...

67