

Chip-multiprocessing and Beyond

Keynote speech at HPCA-12, Feb 14, 2006, Austin, Texas

Per Stenström

Chalmers University of Technology

Sweden

Are We Going Up or Down?

Technology push

- Moore's law
- Economics

Sustainable?

Unclear, but all reasons to be optimistic



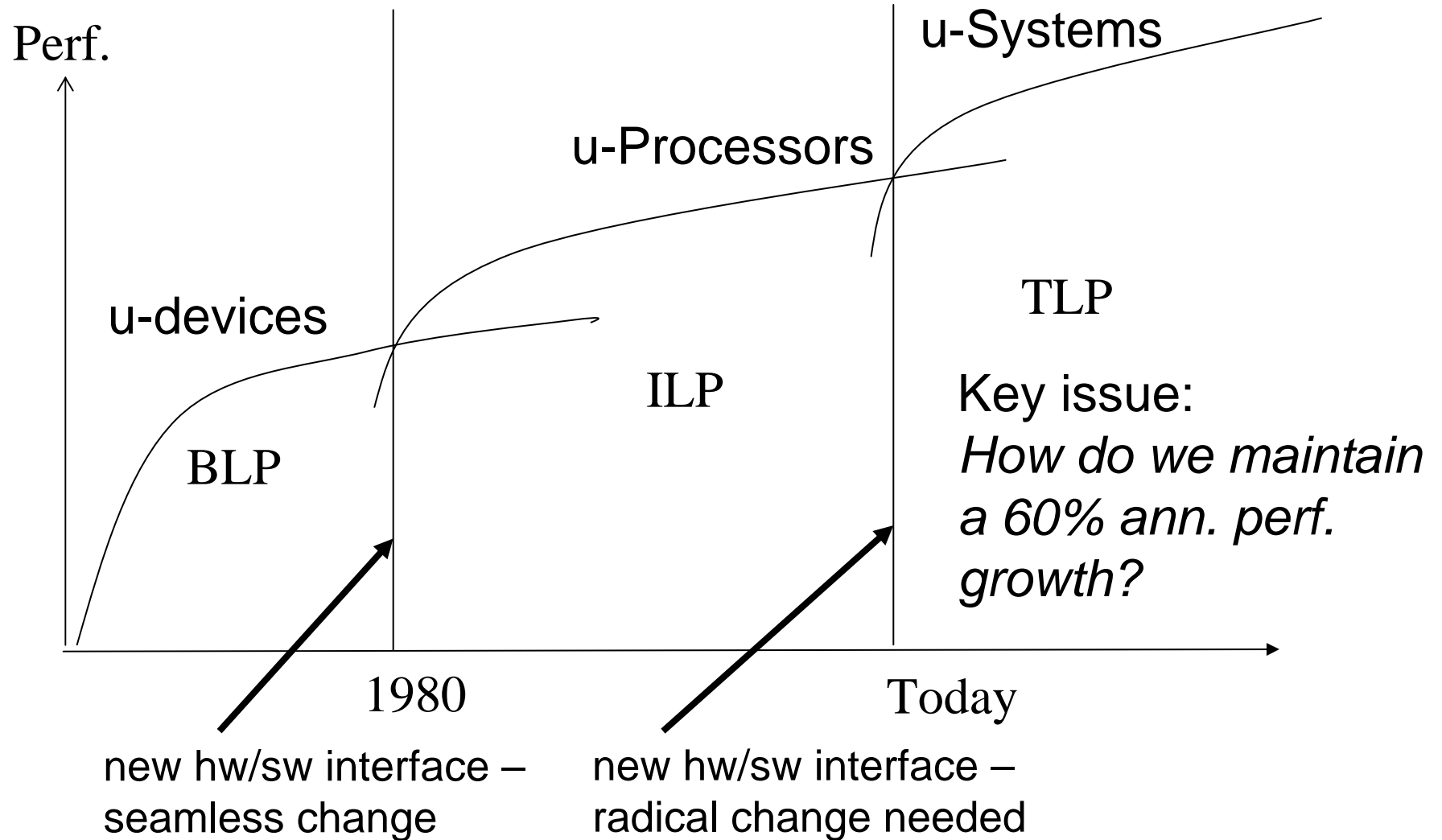
Source: M. C. Escher, 1960

Application pull

- Transaction processing
- HPC
- Embedded high-perf.

Sustainable? So it seems

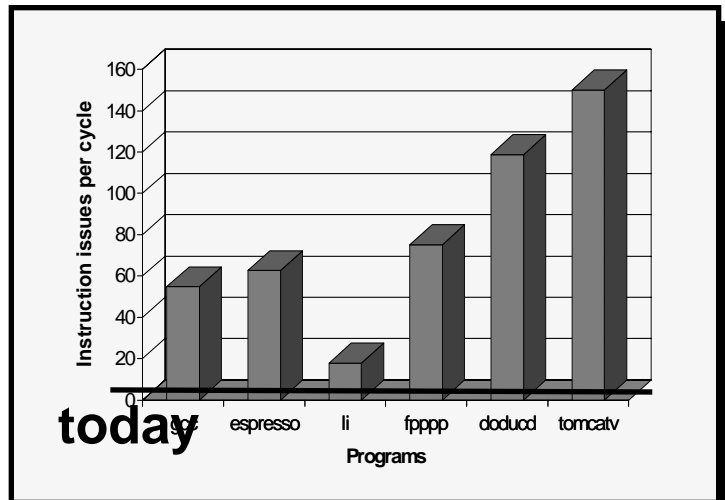
At the Edge of a New Transition



Three Challenges

- Parallelism exploitation
- Higher utilization of memory resources
- Higher bandwidth efficiency, esp. into and out of the processor chip

End of ILP Era?



Source: Hennessy &
Patterson: CA: AQA

Common arguments:

- Resource-inefficiencies
- Verification challenges
- Wire/gate delay gap

Challenging in the short term; long-term can be too late...

Simplicity-centric designs rather than performance at any price in complexity

The Multi-Core Ride just Began



Why?

- Replication scales up instruction window (parallelism opportunities and complexity reduction)
- Decentralized control (reduces gate/wire gap dilemma)

State of the art:

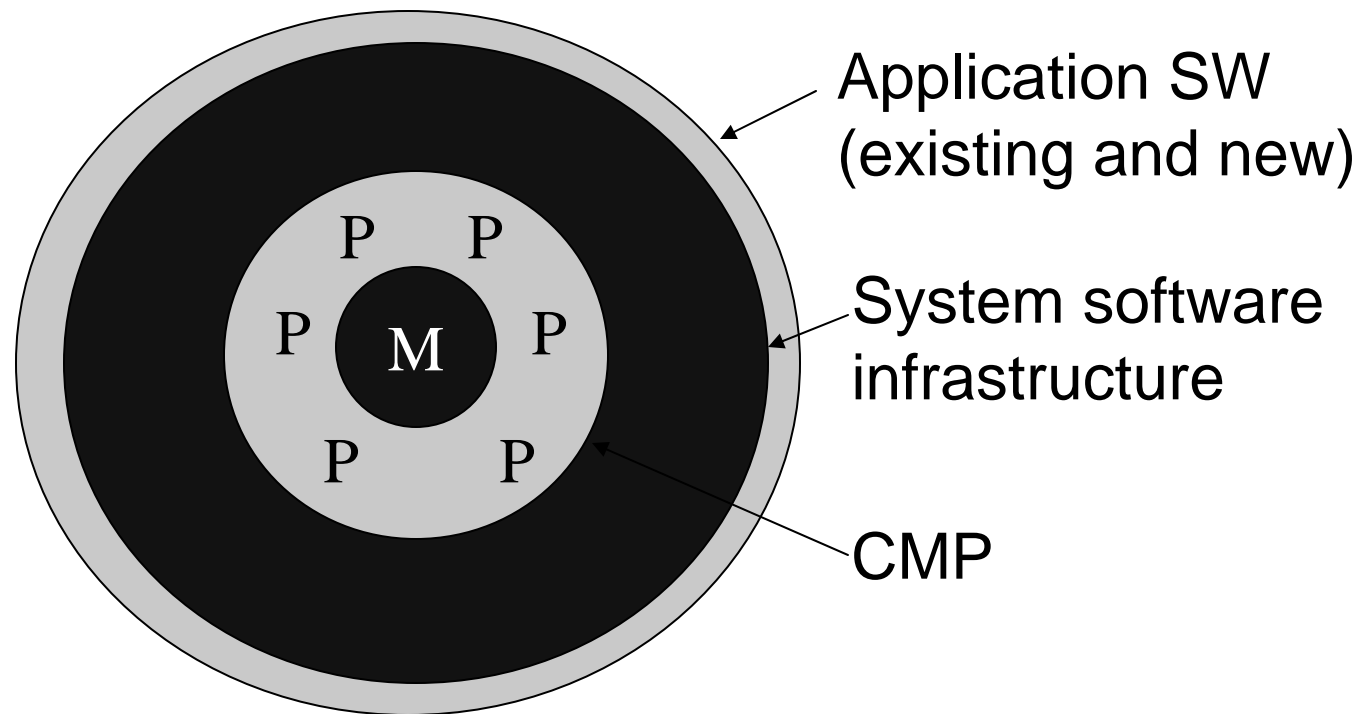
- Dual core as well as 8-way CMPs with 4 threads/core
- Today (2006) 4-way CMPs with 4 threads/core possible

Outlook..

Year	2006	2009	2012	2015
# Cores	4	8	16	32
# Threads	16	32	64	128

- With a doubling of cores every three years, a large-scale multiprocessor server will fit on a chip soon
- Incentive to exploit the many threads will increase dramatically
- Implication: Less reluctant view on parallel programming – question is *how much less?*

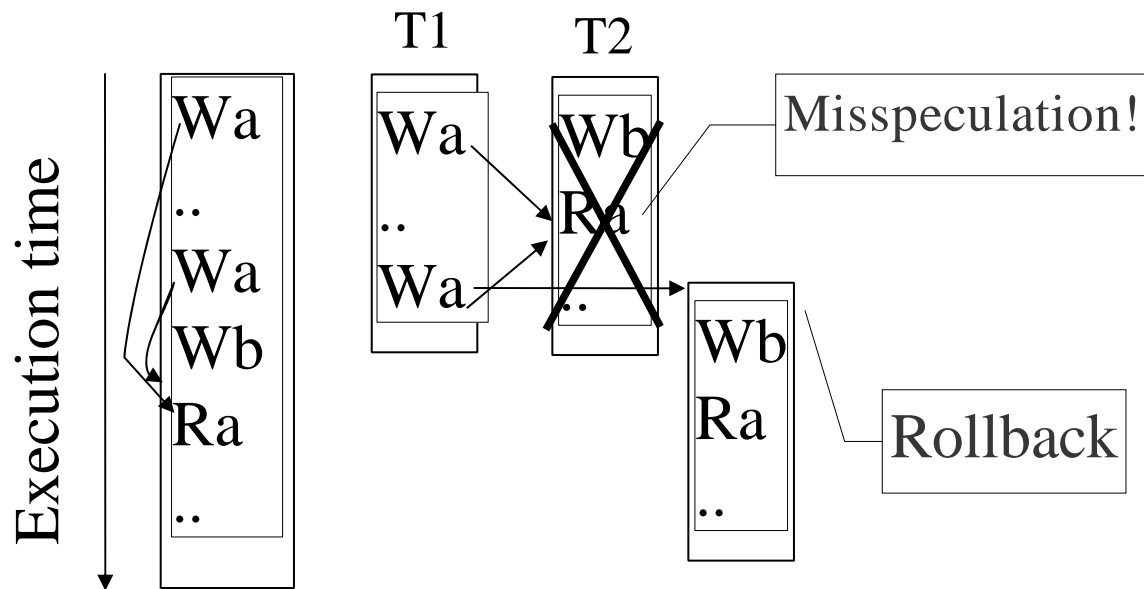
Challenge 1: How to Extract TLP - Implicitly?



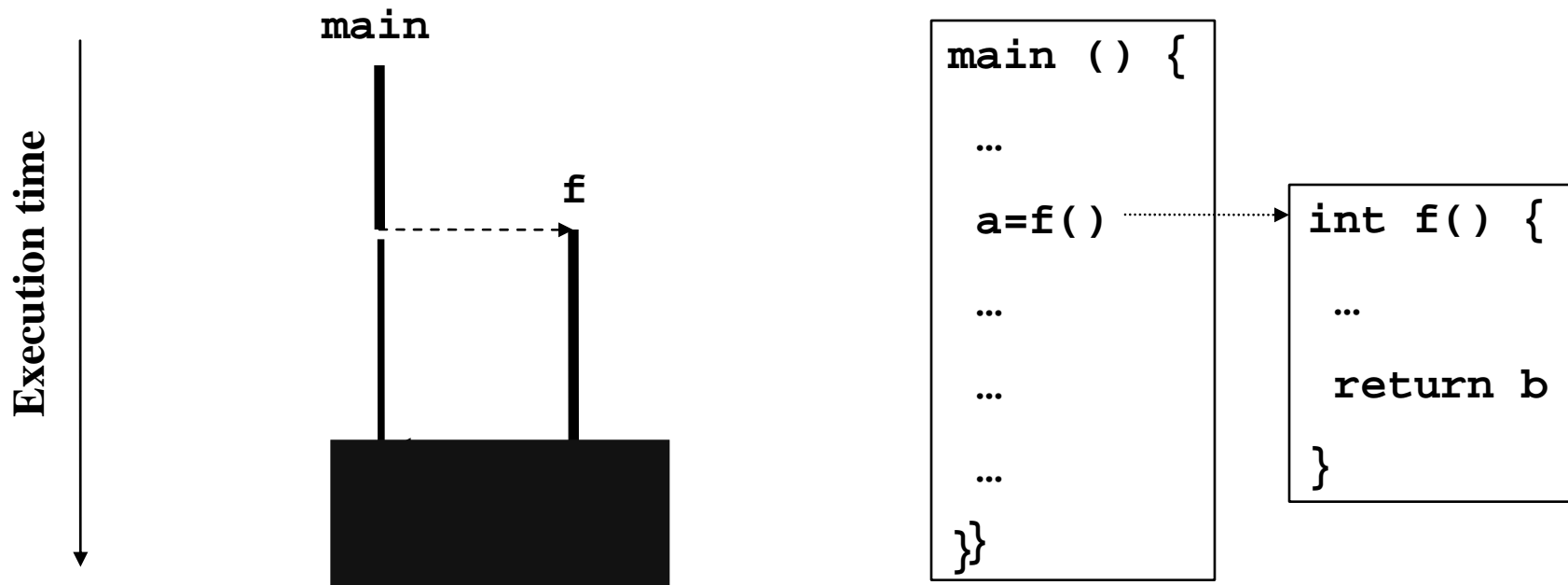
Requires a concerted action across layers:
programming model, compiler, architecture

Thread-level Speculation: What did we Learn?

Thread-level Speculation - a Crash Course

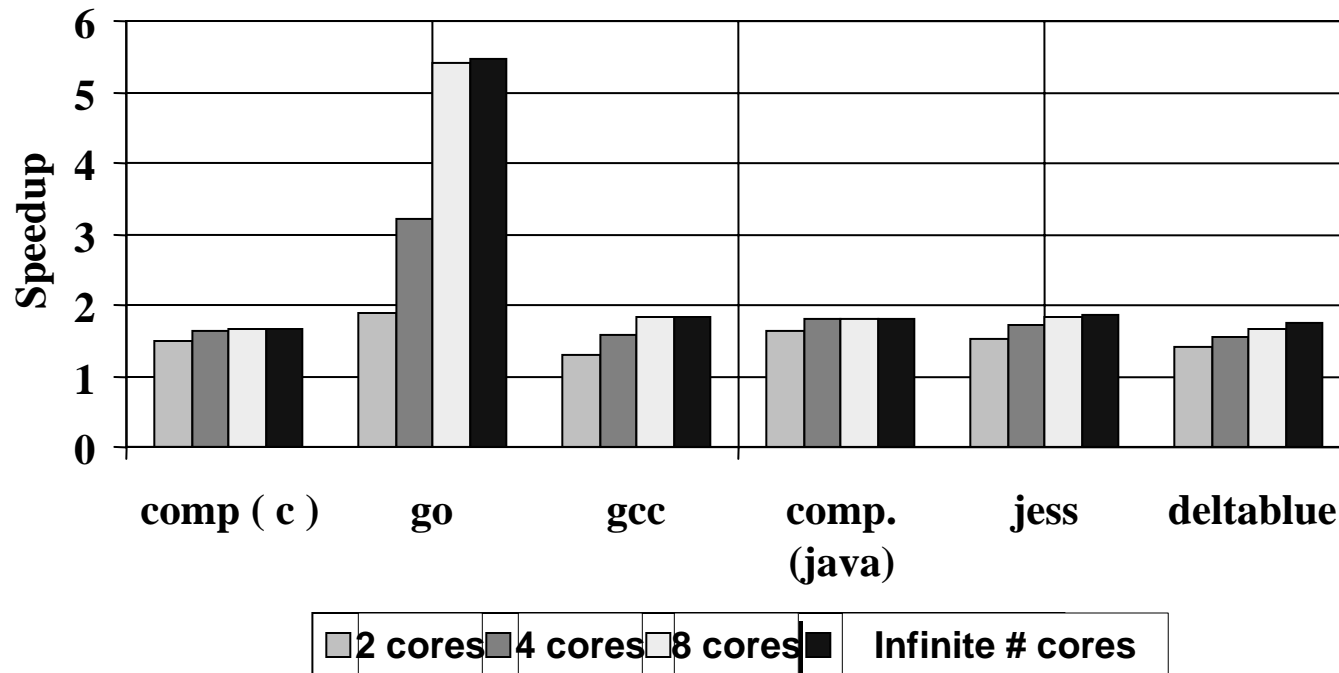


Module-Level Parallelism



New threads are spawned for **module continuations**. Return values are predicted

Module-level Parallelism: Potential



- Even at 2 cores, there is a fair amount of parallelism.
- But will not make use of more than a handful threads

Transactional Memory

- Built upon thread-level speculation substrates
- But programmer breaks up code into speculative threads – transactions
- Lessens synchronization & consistency obstacles
- Strikes a promising tradeoff between parallelization effort and performance

Unfortunately, there are other hardships too

The Four Hard Steps

1. Decomposition
2. Assignment
3. Orchestration
4. Mapping

Goal: expose concurrency but beware of thread mngmt overhead

Goal: balance load & reduce communication

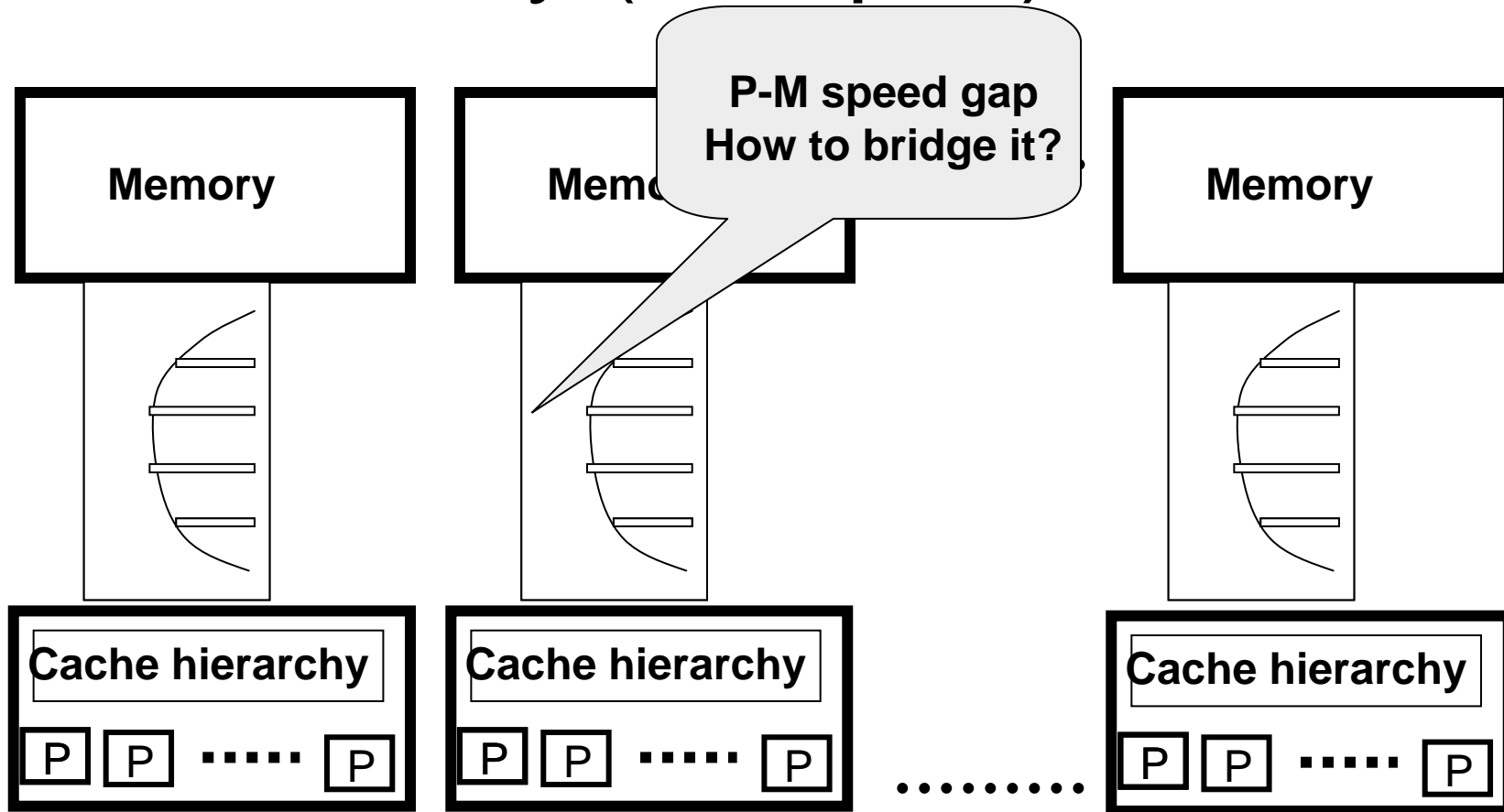
Goal: Orchestrate threads to reduce communication and synchronization costs

Current transactional memory proposals mainly address 3.

Opportunities

- Rethink all interfaces
 - HW/SW
 - Programming model
 - ...
- New tradeoffs at the chip level
- But, simplicity first, then performance

Challenge 2: How to Improve Efficiency (cost/perf) of Memory



The Art of Bridging a Speedgap..

Use more of everything...

- Deeper cache hierarchies
- Bigger flat main memory

Implications for multi-core systems:

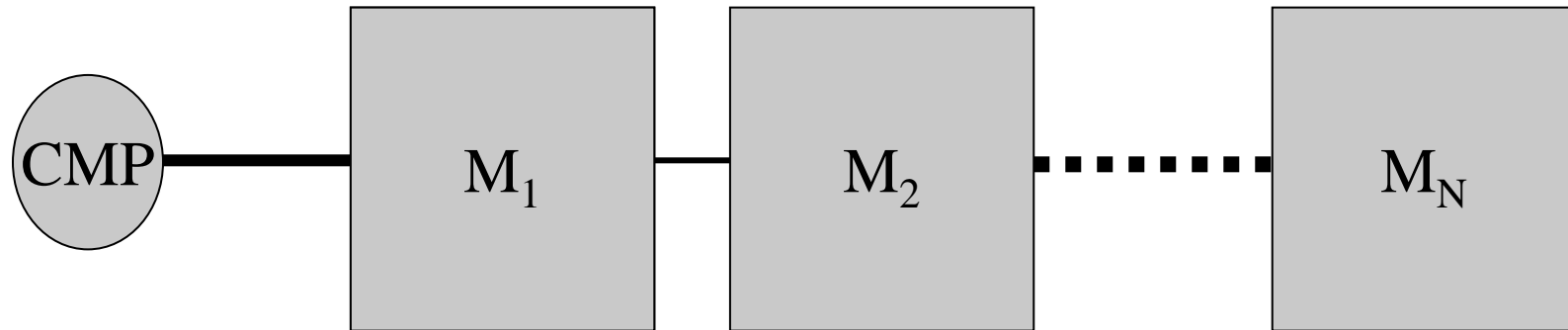
- Tradeoff in resources for cores/caches
- Amount of memory scales linearly with threads:
=> may change ratio #memory chip/ #processor chip

Memory Hierarchy Inefficiencies

Some observations:

- ~ 50% of all blocks in a cache are typically 'dead'
 - ~ 50% of data in a page is never accessed
 - ~ 50% of memory content is redundant
- There is room for improving resource efficiency

Multi-level Main Memory



Operation:

- No inclusion between levels
- Managed by software handlers, in page chunks

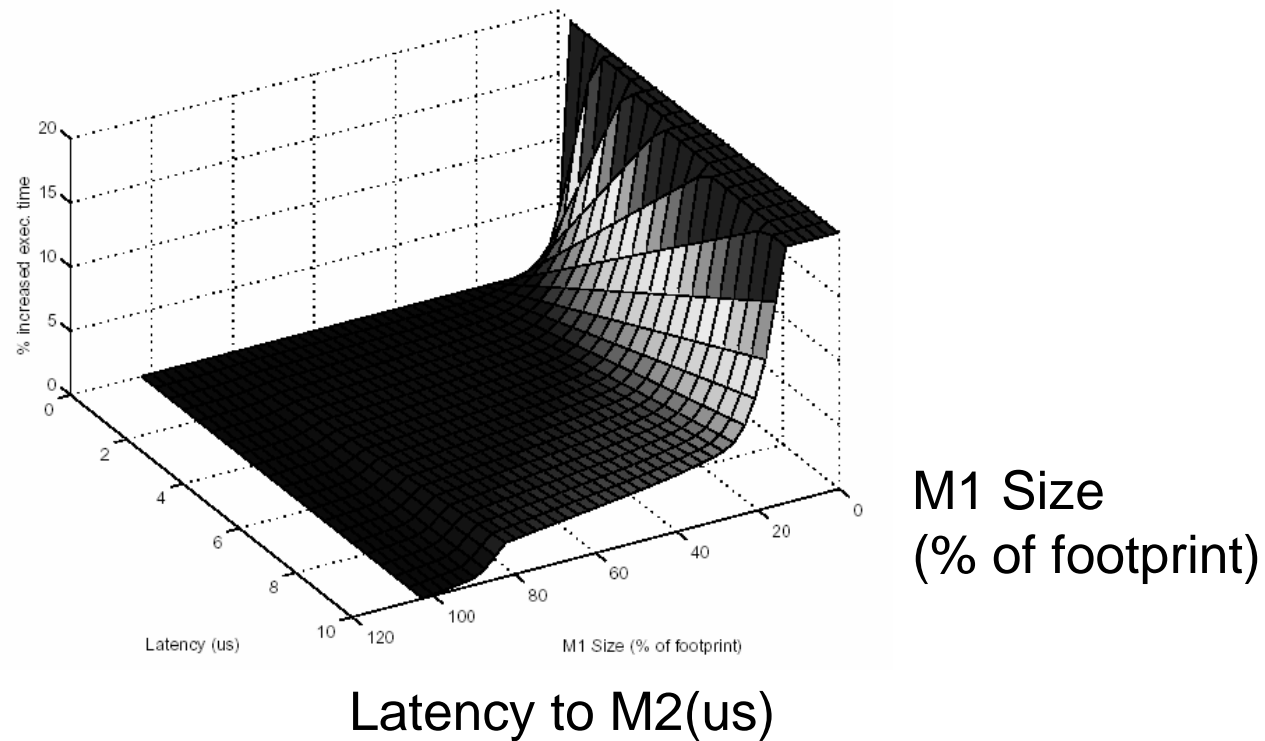
Rationale:

- Data in lower levels could be compressed or put in power down mode, hence slower
- Lower levels could use cheaper and slower memory technologies or be physically further apart

Is the locality principle powerful enough?

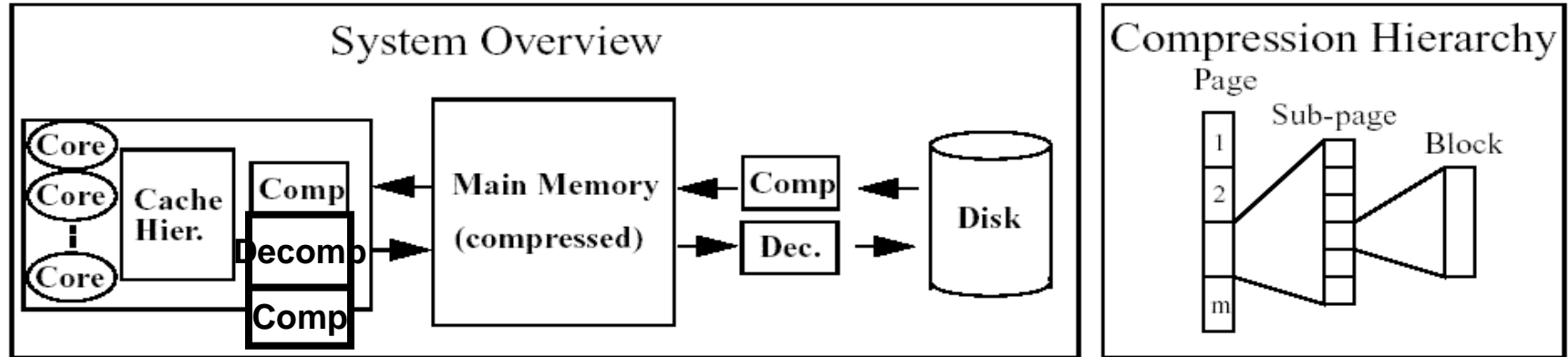
Performance of 2-Level Main Memory

Slowdown(%)



- We can afford to let 70% of footprint reside in ten times slower memory with negligible impact on performance
- Opens up for drastically new tradeoffs (Ekman & Stenstrom 2005 IPDPS)

Main Memory Compression



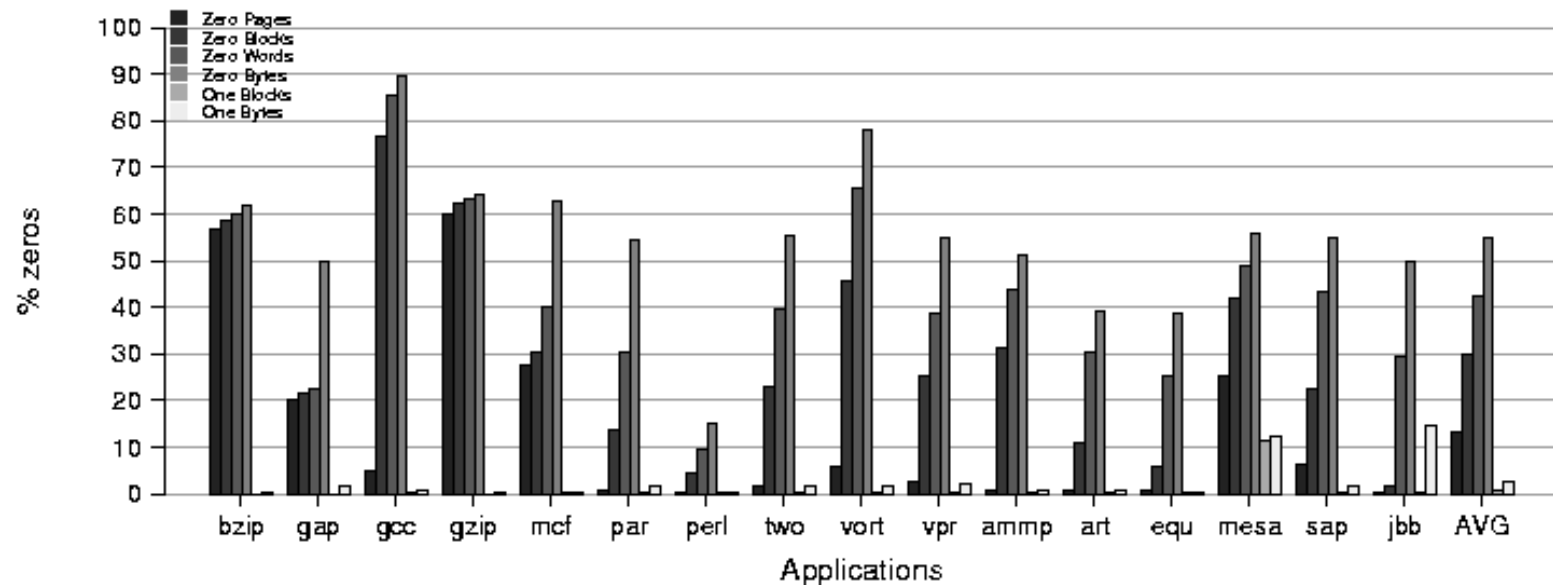
Performance issues:

- Decompression latency
- Address translation between linear and compressed space
- Fragmentation control

With simple (and fast) compression algorithms, 30% of memory resources can be freed up with no perf loss.

(Ekman and Stenstrom, ISCA'05)

Frequency of Zero-Valued Locations



Main observations (assuming 8-KB pages and 64-B blocks)

- 13%, 30%, 42%, and 55% of all pages, blocks, words, and bytes, resp., contain zero; strikes of ones (11...1) are however quite infrequent
- Zero-aware compression schemes have a great potential!

Outlook: Compression

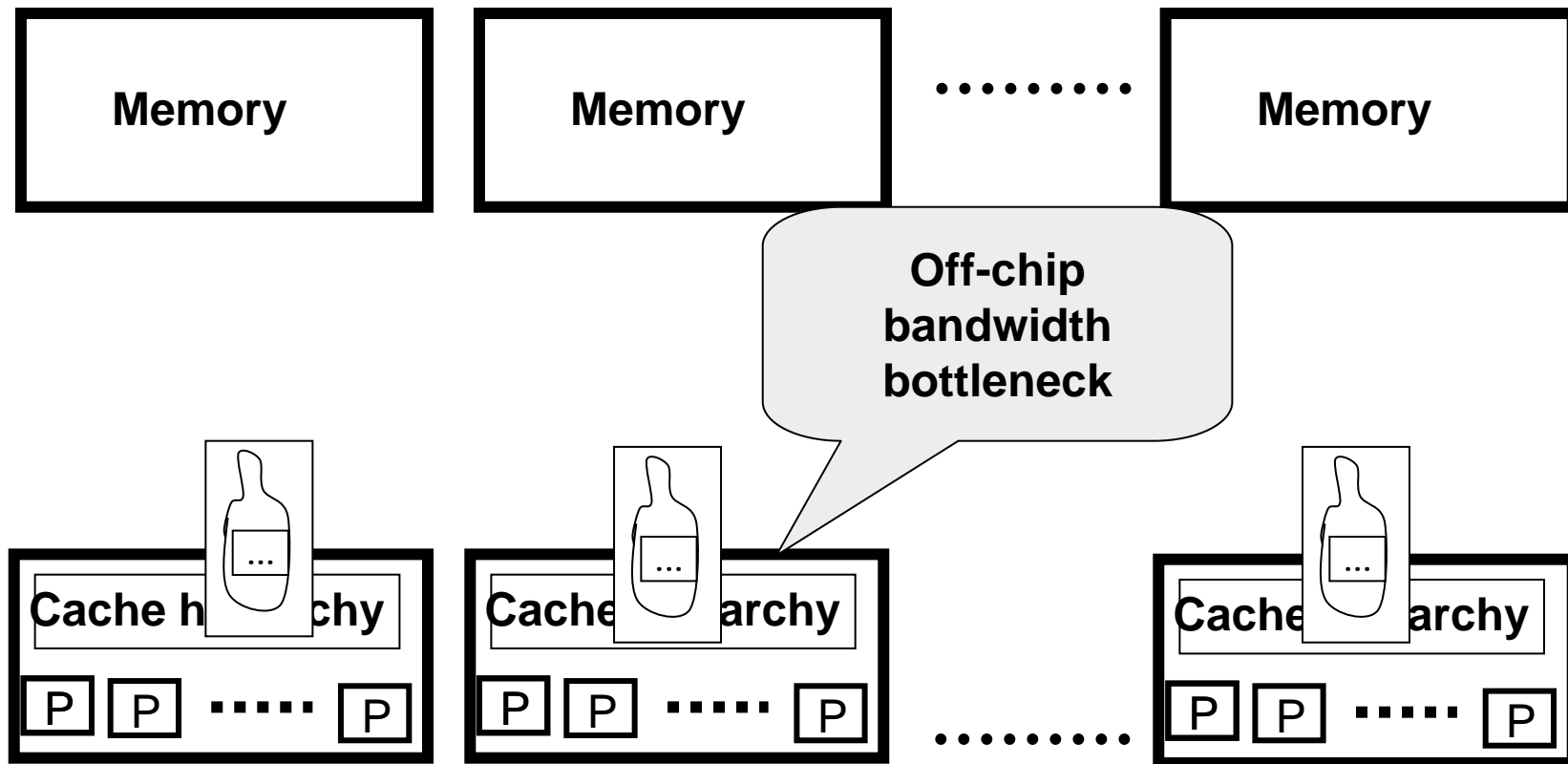
Compression has been applied at most levels:

- Memory
- Cache
- Microarchitecture (narrow operands common)

Compression can yield 2x resource efficiency improvement
eliminating dead blocks and spatial inefficiencies another 2x

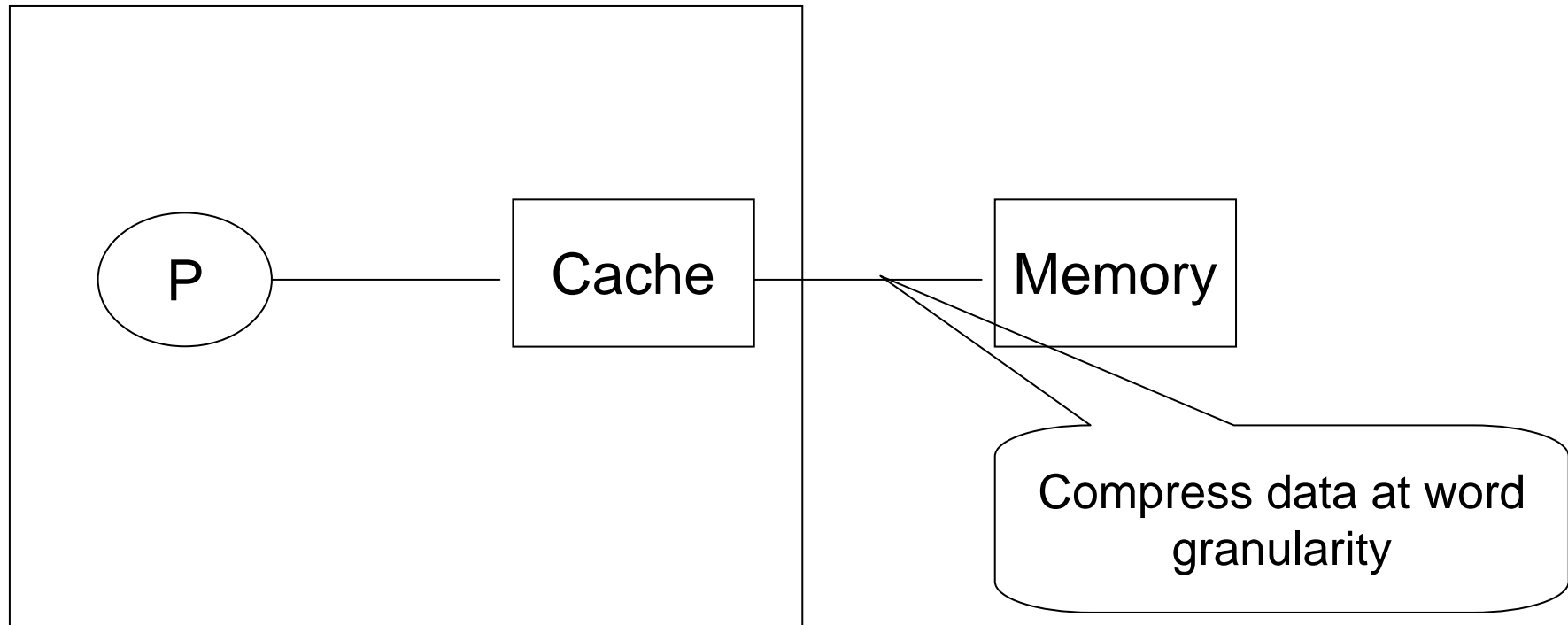
Question: How can we convert this into 4x resource efficiency improvement?

Challenge 3: Off Chip Bandwidth



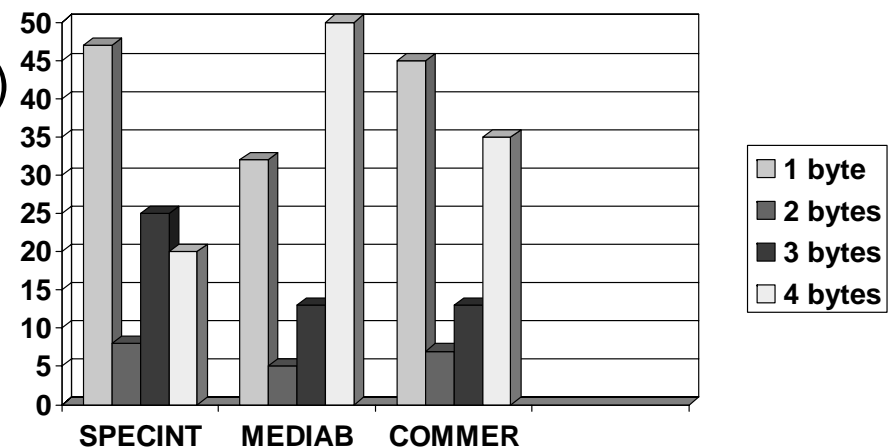
BW does not scale with Moore's law unless optics or other disruptive technologies change the rules of the game

Problem: How to Scale up Bandwidth for 100+ Threads?



Operand Size Distribution

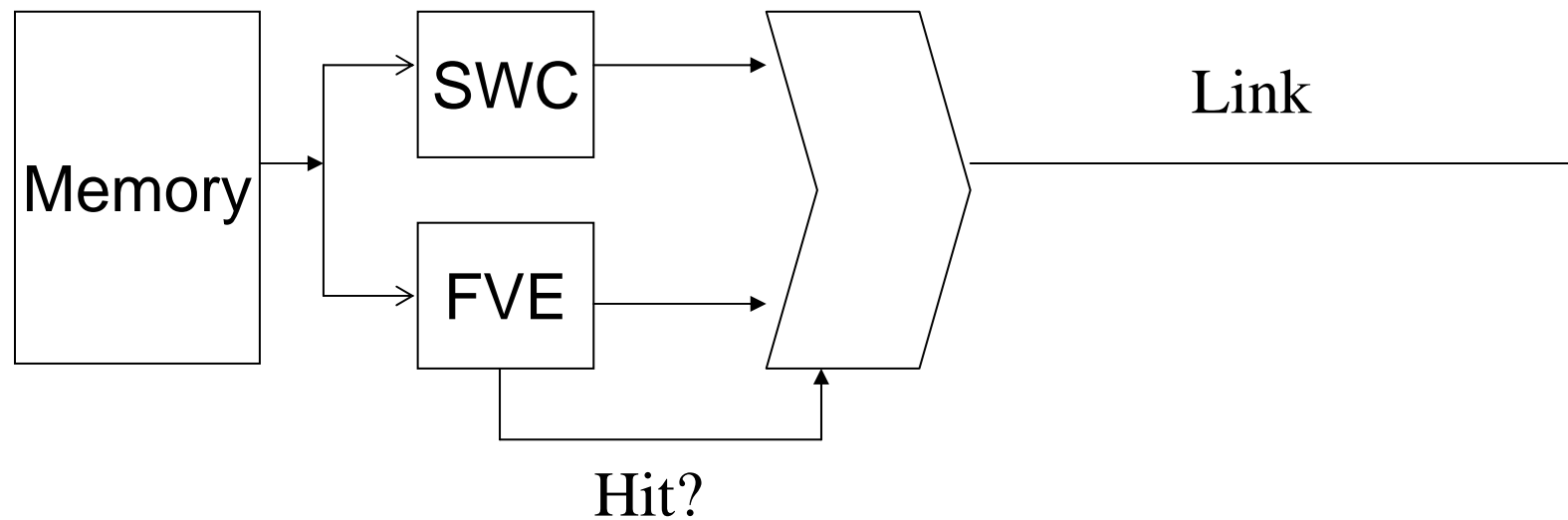
- 40% of all values are small 8-bit operands (in fact zero is most common)
- But values requiring more than 24 bits account for about 30%



Two implications:

- Interesting to consider significance width compression
- Interesting to consider partial value locality in most significant bits of 32-bit values

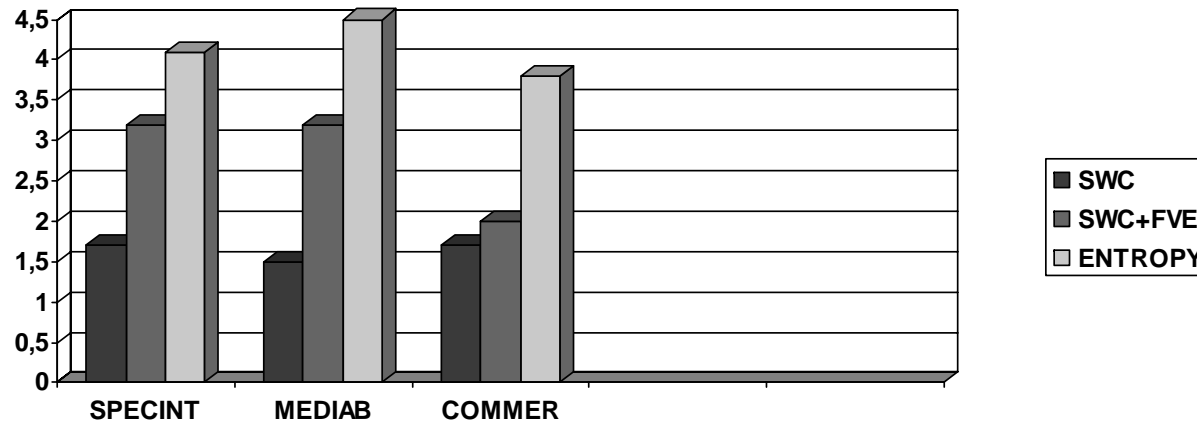
Compression Engine



- Value caches at both sides must be consistent with each other

Compression Results

- Use 16 registers to cache frequent values
- Using more registers provided small additional gains



Combination yields 2X-3X bandwidth improvement

Summary

Challenges at all levels

Chip-level parallel machines will boost adoption of parallel software but progress is needed in

- Parallel programming models to ease the task
- Parallelizing compiler technology
- Architectural approaches to exploit new forms

Memory system still a bottleneck

It is an exciting era that is in need of critically questioning existing frameworks