Asymmetric Chip Multi-Core Applications, processors, and OS – initial thoughts

Uri Weiser
EE Technion

CMP day February 3rd
Sailing Basics

wind

Buoy
Sailing - wind shift

\[ \alpha \]

wind

Buoy

\[ \alpha \]

\[ \alpha \]
25th American Cap – September 23rd 1983
Liberty I against Australia II
7 rounds race, Score status: 3:0 to Liberty I, 4th round started
Sailing competition

- What is the Strategy of Boat 1?
- What is the Strategy of Boat 2?

Do not follow ➔ Invent
**Agenda**

- Today's environment
  - On die: Cores; Platform: NUMA
- The application environment
- Cores implications
- NUMA implications
- Scheduling
- Conclusions
Cores implications
CMP is ubiquitous since Paul Otellini announced the P-Due-2 in June 2006

Reasons: Power wall ➔
- Single Core Performance/power trend
- Process technology

Open questions
- How many Cores? 10s 1000s? Implications!
- Cache architecture?
- Are the cores Symmetric (aka Homogenous) or Asymmetric?
  - If Asymmetric
    - Same ISA
    - Different ISA extensions
    - Application specific accelerators
- Task/threads scheduling
implications
second boat chose NUMA in 2007

Reasons: Integration and Performance

- Implications
  - Synchronize Memory Controllers (MC)
  - Future Multiple MC on die

- Open issues/opportunities
  - All memory traffic runs through the CPU
  - Task/Threads assignments and scheduling
The Computer runs multiple programs

- CPUs & Caches
- Computation and “scratch pad”
- Computation engines e.g. Graphics engine, Media, Communication engines,
- Memory
- Mass storage
- Fast I/O e.g. camera, Accelerators
- Slow I/O e.g. human interface
- Display
- Communication

The “outside” world
Flying machines
Are they all the same?
Flying machines
Are they all the same?
Flying machines
Are they all the same?
Requirements and program behavior

- **QoS**
  - Application’s response ➔ Latency + Performance
  - Differential services in multiple applications environment
    - Priority of Programs
    - Priority of Data
  - Lossless or loss consent (e.g. Music, video, communication...)
  - Power constrains
  - Reliability
Requirements and program behavior

- Parallelism
  - Threads level parallelism
    - Many/Huge number of threads \( \Rightarrow \) 10s or 1000s
  - Data level parallelism \( \Rightarrow \) vector|SIMD
  - Instruction level parallelism \( \Rightarrow \) ILP
  - Pipelining (one dimension/few dimensions \( \Rightarrow \) graphics/systolic algorithms)

Remember Amdahl law
Requirements and program behavior (cont’)

- Memory BW
- Commutation vs. memory access ➔ e.g. G Byte/Flop, M 0.05 Byte/Flop
- Streaming ➔ e.g. Media, communication, stock exchange data
- Computation behavior ➔ FP, integer, branches, loops
- Locality of program/data ➔ impact on caches, TLBs, memory
- Sharing ➔
- Foot print Program/data
Gain

Frequency

GBWP = Gain Bandwidth Product = constant @ a given technology

e.g. $\text{Gain}_1 \times \text{BW}_1 = \text{Gain}_2 \times \text{BW}_2$
Analog Circuit Paradigm (cont.)

Gain

Frequency

Gain

BW₁

BWₙ
Application domain

Same ISA/ISA extensions/Application Specific Accelerators

Performance

Application specific – Accelerators (not our generic ISA)

General purpose

Apps range

Application specific engines achieve higher performance/power than general purpose engines
Applicati**on domain**

Same ISA/ISA extensions/Application Specific Accelerators

Performance

General purpose engines with ISA extensions
The memory Wall

- Many execution engines $\Rightarrow$ more performance $\Rightarrow$ more memory accesses
- How to reduce the memory latency performance impact?
  - Caches
  - Threads
The memory Wall

the Cache solution

The "MultiCore" machines solution ➔ use cache as a memory access "filter"

● Implications:
  – Number of cores: Cache area limits the number of cores
  – BW: Reduces memory access Bandwidth
  – Coherency: If keeping flat memory structure need coherent Cache
  – Cache Access time (see example: Nahalal)
Example: **Nahalal; Shared $ vs. Private $**

Overview of Nahalal cache organization

Aerial view of Nahalal cooperative village
27.41% improvement in average cache access time
- 41.1% in apache
NAHLAL Cache structure

Asymmetric NUMA Architecture

CPU
Private Cache
CPU
Private Cache
CPU
Private Cache
CPU
Private Cache

Private Memory
CPU
CPU
CPU
CPU
CPU
CPU
CPU
CPU

CPUs clusters

Bridge

"Shared Memory"
NUMA implication
a platform view

CPUs

Bridge

IOs

Accelerators
Communication
Storage
External devices

NUMA
Platform NUMA

- Can we improve platform data movements
  - Reduce unnecessary traffic
  - Efficient on-loading

Memory (aka M₃)

CPUs

Bridge

IOs
  - Accelerators
  - Communication
  - Storage
  - External devices
Task Scheduling

● Current platform tasks’ scheduling is being performed by OS based on:
  – Available resources
  – Ready tasks

● Are there other dimensions?
  – Scheduling based on thread’s essence
  – Can we impact scheduling based on “The data content”?*
ACCMP - Asymmetric Cluster CMP

- Big cores and small cores same ISA
- Serial phases execute on large core
- Parallel phases execute on all cores
ACCMP - Analysis

ACCMP Performance Vs. Power

Perf_{ACCMP} = \frac{\eta \sqrt{a}}{\lambda \left( \frac{P}{\gamma a} - \beta + \sqrt{\beta} \right)^{1/\gamma} + (1 - \lambda) \beta^{1/\gamma} + \lambda \sqrt{\eta (k_1 + nk_2)} a \sqrt{\frac{P}{\gamma a}}}

ACCMP delivers more performance per unit power!

$\alpha = 1, \beta = 4$

$\alpha = 0.33, \beta = 6$

Symmetric Upper Bound

Relative Performance

Relative Power

- Symmetric Upper Bound
- Asymmetric ($\alpha = 1, \beta = 4$)
- Asymmetric ($\alpha = 0.33, \beta = 6$)
Multiple Applications Scheduling

- **Observation**: Serial threads are the bottleneck of applications.
- **Current**: OS grants serial threads the same priority as parallel threads, resulting in lower system throughput and unfairness.
- **Solution**: Boost priority of serial threads.
- **ACCMP**: Serial threads will be granted more powerful cores.
Benchmark Throughput

Benchmark “B” Serial to Parallel ratio

Benchmark “A” Serial to Parallel ratio
Application’s Data’s requirements

- Differential services
  - Some data is latency non-tolerant (e.g. banking transactions)
  - Some data is BW non-tolerant (e.g. Video)
  - Some data is more “valuable”...

- Why not to handle data according to its “value” (in addition its processing requirement)?
Data Content Aware (DCA) Architecture*

Task assignment based on Data Content

- **System without DCA**
  - Symmetrical services

- **System with DCA**
  - DCA Content-aware
  - Differential services (1:99)
**Latency**

*Test does not include UDP zero copy*
Latency vs. load

- 10% priority traffic
- 5% priority traffic
- 1% priority traffic
Conclusions

- The environment is changing
  - Many open issues but interesting research
    - Core homogeneity?
    - Computation assignment and scheduling?
      - Thread essence base?
      - Data Content base?
    - Cache or not Cache?
    - Cache structure?
    - ...
    - ...
    - ...

Thank you