Many-Core Computing

Ioan Raicu
Computer Science Department
Illinois Institute of Technology

CS 595: Data-Intensive Computing
November 16th, 2011
Slides compliment of
Yong Chen and Xian-He Sun
From paper “Reevaluating Amdahl's Law in the Multicore Era”
It All Starts with Amdahl’s Law

• Gene M. Amdahl, “Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities”, 1967

• Amdahl’s law (Amdahl’s speedup model)

\[
\text{Speedup}_{Amdahl} = \frac{1}{(1 - f) + \frac{f}{n}}
\]

\[
\lim_{n \to \infty} \text{Speedup}_{Amdahl} = \frac{1}{1 - f}
\]
Recent Computer Architecture Evolvement

- **Multicore architecture**
  - Integrate multiple processing units into a single chip
  - Conquer the performance limitation of uni-processor
    - Pipeline depth (limited ILP)
    - Frequency
    - Power consumption
  - Cost-effective architecture
    - Pollack’s Rule
  - Adds a new dimension of parallelism
Amdahl’s Law for Multicore

- Study the applicability of Amdahl’s law to multicore architecture
- Assumptions
  - $n$ BCEs (Base Core Equivalents)
  - A powerful $perf(r)$ core can be built with $r$ BCEs
- Analyze performance of three multicore architecture organizations
  - Symmetric, Asymmetric, Dynamic
• Speedup of symmetric architecture

\[ \text{Speedup}_{\text{symmetric}}(f, n, r) = \frac{1}{1 - f \frac{\text{perf}(r)}{\text{perf}(r) + f \cdot n}} \]

• Speedup of asymmetric architecture

\[ \text{Speedup}_{\text{asymmetric}}(f, n, r) = \frac{1}{1 - f \frac{\text{perf}(r)}{\text{perf}(r) + f + n - r}} \]

• Speedup of dynamic architecture

\[ \text{Speedup}_{\text{dynamic}}(f, n, r) = \frac{1}{1 - f \frac{\text{perf}(r)}{\text{perf}(r) + f \cdot n}} \]
Amdahl’s Law for Multicore

• Hill and Marty’s study
  – Limited speedup at large-scale size
  – Dynamic architecture delivers a better speedup, but just an “ideal” situation, with $t \geq 0.975$
  – Suggest a large-scale multicore is less interesting

• Current major industries also cite Amdahl’s law
IBM Cell: 8 slave cores + 1 master core, 2005

Sun T2: 8 cores, 2007

AMD Bulldozer: 16 cores, 2011

Intel Dunnington: 6 cores, 2008
Kilocore: 256-core prototype
By Rapport Inc.

Quadro FX 3700M:
128-core, By nVIDIA

512-core GPUs
NVIDIA FERMI

GRAPE-DR chip:
512-core, By Japan

GRAPE-DR testboard
Many-Core Computing

IBM 7030 Stretch

IBM 7950 Harvest

Cray X-MP
Fastest computer 1983-1985

Cray Y-MP

All have up to 8 processors, citing Amdahl’s law,
\[
\lim_{n \to \infty} \text{Speedup}^{\text{Amdahl}} = \frac{1}{1 - f}
\]
The scale size is far beyond implication of Amdahl’s law
• Tacit assumption in Amdahl’s law
  – The problem size is fixed
  – The speedup emphasizes time reduction
• Gustafson’s Law, 1988
  – Fixed-time speedup model

\[
\text{Speedup}_{\text{fixed-time}} = \frac{\text{Sequential Time of Solving Scaled Workload}}{\text{Parallel Time of Solving Scaled Workload}} = (1 - f) + nf
\]

• Sun and Ni’s law, 1990
  – Memory-bounded speedup model

\[
\text{Speedup}_{\text{memory-bounded}} = \frac{\text{Sequential Time of Solving Scaled Workload}}{\text{Parallel Time of Solving Scaled Workload}} = \frac{(1 - f) + fG(n)}{(1 - f) + fG(n)/n}
\]
Scalable Computing in HPC

Source: ANL ALCF

**Compute Card**
1 chip, 20 DRAMs

**Chip**
4 processors
850 MHz
8 MB EDRAM

**Node Card**
(32 chips 4x4x2)
32 compute, 0-2 IO cards

**Rack**
32 Node Cards
1024 chips, 4096 procs
Cabled 8x8x16

**Petaflops System**
72 Racks
1 PF/s
144 TB

**Maximum System**
256 racks
3.5 PF/s
512 TB

- 14 TF/s
- 2 TB

**Front End Node / Service Node**
System p Servers
Linux SLES10
HPC SW:
- Compilers
- GPFS
- ESSL
- Loadleveler
Scalable Computing for Multicore

- Multicore is a way of “parallel computing on chip”
  - Independent ALUs, FPUs
  - Independent register files
  - Independent pipelines
  - Independent memory (private cache)
  - Connected with ultra-highspeed on-chip interconnect
- **Scalable computing viewpoint** applies to multicore
- Applications demand quicker and more accurate results when possible
  - Video game (e.g. 3-D game)
  - High-quality multi-channel audio
  - Real-time applications (e.g. video-on-demand)
  - Scientific applications
  - Very unlikely to compute a fixed-size problem when have enormous computing
Definitions

- Definition 1. The *work* (or workload, or problem size) is defined as the number of instructions that are to be executed.
  - Denoted as $w$
  - Including improvable portion $fw$ and non-improvable $(1-f)w$

- Definition 2. The *execution time* is defined as the number of cycles spent for executing the instructions, either for computation or for data access.

- Definition 3. The *fixed-size speedup* is defined as the ratio of the execution time in the original architecture and the execution time in the enhanced architecture.
Fixed-size Model for Multicore

\[
\text{Speedup}_{\text{fixed-size}} = \frac{T_{\text{original}}}{T_{\text{enhanced}}}
\]

\[
T_{\text{original}} = \frac{w}{\text{perf}(1)} = w
\]

\[
T_{\text{enhanced}} = \frac{(1-f)w}{\text{perf}(r)} + \frac{fw}{n \cdot \text{perf}(r)}
\]

\[
\text{Speedup}_{\text{fixed-size}} = \frac{w}{\text{perf}(1)} \frac{1}{(1-f)w + \frac{fw}{\text{perf}(r)} + \frac{fw}{n \cdot \text{perf}(r)}}
\]

\[
= \frac{1}{1 - f} \frac{1}{\frac{fw}{\text{perf}(r)} + \frac{f}{\text{perf}(r) \cdot n}}
\]

Consistent with Hill and Marty’s findings
Fixed-size Speedup for Multicore

Quickly limited by non-improvable portion
Definition 4. The **fixed-time speedup** of multicore architecture is defined as the ratio of execution time of solving the scaled workload in the original mode to execution time of solving the scaled workload in enhanced mode, where the scaled workload is the amount of work that is finished in the enhanced mode within the same amount of time as in the original mode.

The fixed-time constraint, when the number of cores scales from \( r \) to \( mr \):

\[
\frac{(1-f)w}{\text{perf}(r)} + \frac{fw}{\text{perf}(r)} = \frac{(1-f)w}{\text{perf}(r)} + \frac{fw'}{\text{perf}(r)m} \Rightarrow w' = mw
\]

The scaled fixed-time speedup:

\[
\text{Speedup}_{\text{fixed-time}} = \frac{\text{Time of Solving } w' \text{ in Original Mode}}{\text{Time of Solving } w \text{ in Original Mode}} = \frac{(1-f)w + fw'}{\text{perf}(r) + \text{perf}(r)m} = \frac{w}{\text{perf}(r)} = (1-f) + mf
\]
Fixed-time Speedup for Multicore

Fixed-time Speedup of Multicore Architecture

Scales linearly
• Definition 5. The memory-bounded speedup of multicore architecture is defined as the ratio of execution time of solving the scaled workload in the original mode to execution time of solving the scaled workload in enhanced mode, where the scaled workload is the amount of work that is finished in the enhanced mode with a constraint on the memory capacity.

• Assume the scaled workload under the memory-bounded constraint is \( w^* = g(m)w \), where \( g(m) \) is the computing requirement in terms memory requirement, e.g. \( g(m) = 0.38m^{3/2} \), for matrix-multiplication \((2N^3 \text{ v.s. } 3N^2)\)

• The scaled memory-bounded speedup

\[
\text{Speedup}_{\text{memory\text{-}bounded}} = \frac{\text{Time of Solving } w^* \text{ in Original Mode}}{\text{Time of Solving } w \text{ in Original Mode}} = \frac{\frac{\text{Time of Solving } w^{*} \text{ in Original Mode}}{(1 - f) + g(m)f}}{(1 - f) + \frac{g(m)f}{m}}
\]
Memory-bounded Speedup for Multicore

Memory-bounded Speedup of Multicore Architecture

Scales linearly and better than fixed-time

\( g(m) = 0.38m^{3/2} \)
Scalable computing shows an optimistic view.
Results and Implications

- Result 1: Amdahl’s law presents a limited and pessimistic view on the multicore scalability.
- Implication 1: Micro-architects should jump out the pessimistic view to avoid the history to repeat itself.
- Result 2: The scalable computing concept and two scaled speedup models are applicable to multicore architecture.
- Implication 2: The manufactures should be actively move into building a large-scale multicore processor.
- Result 3: The memory-bounded model considers a realistic constraint and presents a practical and an even better view.
- Implication 3: The problem size scaling should prohibit extensive accesses across memory hierarchies.
• Sequential processing capability is enormous
• Long data access delay, a.k.a. memory-wall problem, is the identified performance bottleneck
• Assume a task has two parts, $w = w_p + w_c$
  – Data processing work, $w_p$
  – Data communication (access) work, $w_c$
• **Fixed-size speedup** with data-access processing consideration

\[
\text{Speedup} = \frac{1}{\frac{w_c}{\text{perf}(r)} + \frac{w_p \cdot r}{\text{perf}(r) \cdot n}}
\]
**Scaled Speedup under Memory-wall**

- **Fixed-time model constraint**
  \[
  \frac{w_c}{\text{perf}(r)} + \frac{w_p}{\text{perf}(r)} = \frac{w_c}{\text{perf}(r)} + \frac{w_p'}{m \cdot \text{perf}(r)} \Rightarrow w_p' = mw_p
  \]

- **Fixed-time scaled speedup**
  \[
  \frac{w_c}{\text{perf}(r)} + \frac{w_p'}{m \cdot \text{perf}(r)} = \frac{w_c + m \cdot w_p}{w_c + w_p} = (1 - f') + mf' , \text{ where } f' = \frac{w_p}{w_c + w_p}
  \]

- **Memory-bounded scaled speedup**
  - Computing requirement is generally greater than memory requirement
  - Likely to be greater than the fixed-time speedup
Results and Implications

• Result 4: Data-access delay remains as a dominant factor that decides the sustained performance of multicore architecture
  – When the processor-memory performance gap grows larger, data access delay has more impact on the overall system performance

• Implication 4: Architects should not only focus on-chip layout design to deliver a high peak performance, but also should focus on memory and storage component design to achieve a high sustained performance

• Result 5: With data-access delay consideration, scalable computing concept are still applicable to multicore architecture design

• Implication 5: Scalable computing concept characterizes the application requirements and reflect the inherent scalability constraints of multicore architecture well
Mitigating Memory-wall Effect

• Memory hierarchy
  – Principle of locality

• Data prefetching
  – Software prefetching technique
    □ Adaptive, compete for computing power, and costly
    – Hardware prefetching technique
      □ Fixed, simple, and less powerful

• Solutions
  – Data Access History Cache
  – Server-based Push Prefetching
  – Hybrid Adaptive Prefetching Architecture
Conclusion

- With scalable computing viewpoint, multicore architecture can scale up well and linearly.
- Scalable computing concept provides a theoretical foundation for building a large-scale multicore processor.
- Memory-bounded speedup model considers memory constraint on performance and indicates the tradeoff between memory capacity and computing power.
- Scalable computing view is applicable to task-level:
  - Multicore architecture is not built for single task.
  - Explore an overall performance speedup improvement.
  - Exploit a high-throughput computing.