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

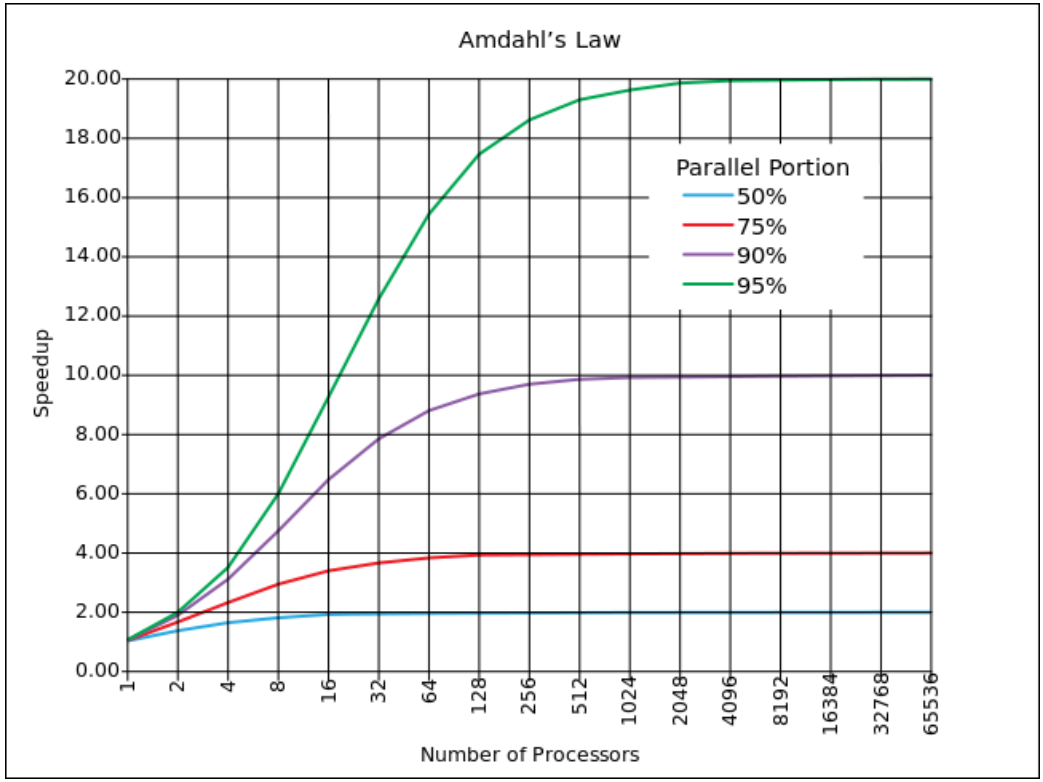
From paper “Reevaluating Amdahl's Law in the Multicore Era”

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)

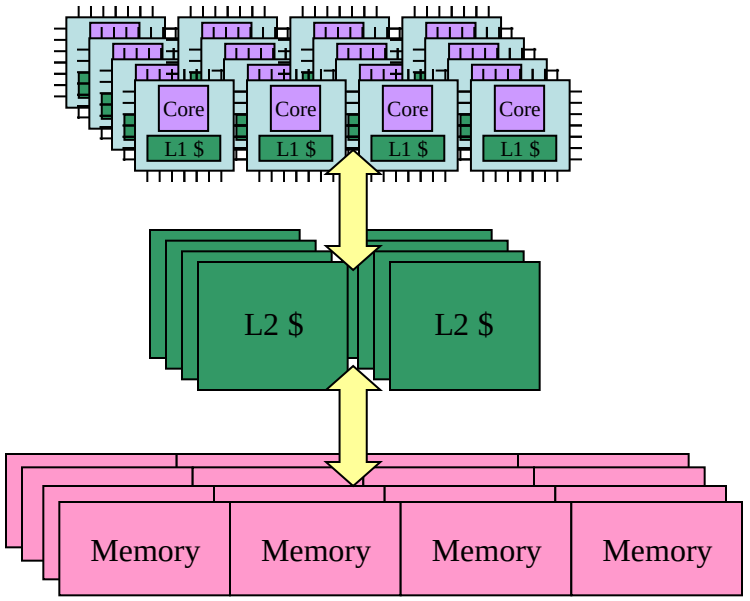
$$Speedup_{Amdahl} = \frac{1}{(1-f) + \frac{f}{n}}$$

$$\lim_{n \to \infty} Speedup_{Amdahl} = \frac{1}{1-f}$$



Recent Computer Architecture Evolvement

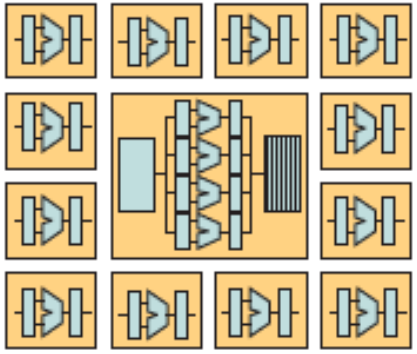
- Multicore architecture
  - Integrate multiple processing units into a single chip
  - Conquer the performance limitation of uni-processor
    - Pipeline depth (limited ILP)
    - Frequency
    - Power consumption
  - Cost-effective architecture
    - Pollack's Rule
  - Adds a new dimension of parallelism



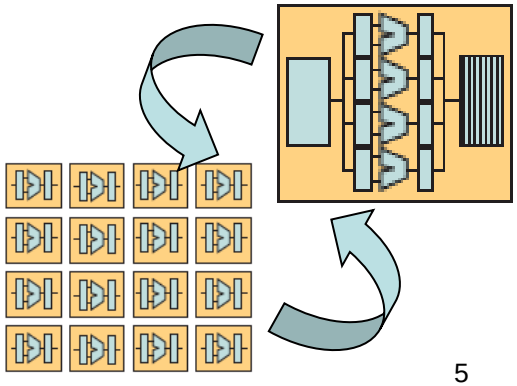
Amdahl's Law for Multicore

- Hill & Marty, "*Amdahl's Law in the Multicore Era*", IEEE Computer 2008
- 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

Symmetric



Asymmetric



Dynamic

Amdahl's Law for Multicore

- Speedup of symmetric architecture

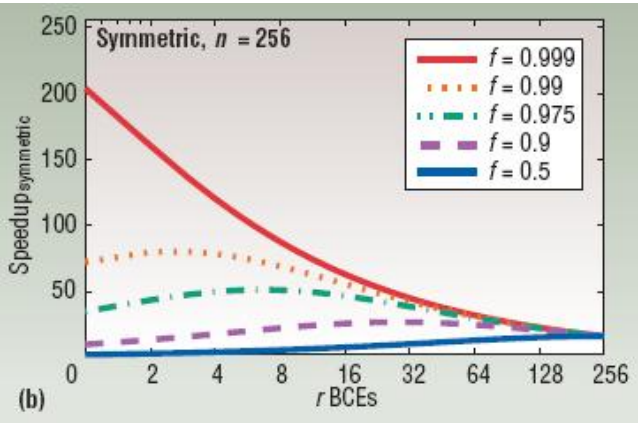
$$Speedup_{symmetric}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}$$

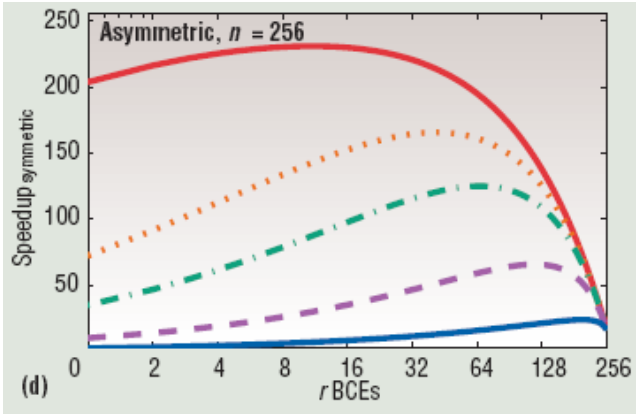
- Speedup of asymmetric architecture

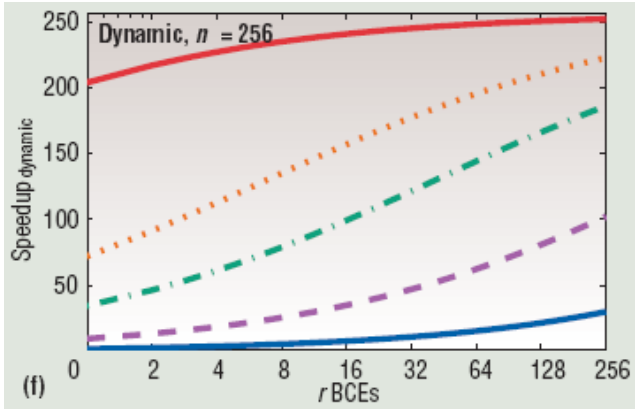
$$Speedup_{asymmetric}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{perf(r) + n - r}}$$

- Speedup of dynamic architecture

$$Speedup_{dynamic}(f,n,r) = \frac{1}{\frac{1-f}{perf(r)} + \frac{f}{n}}$$



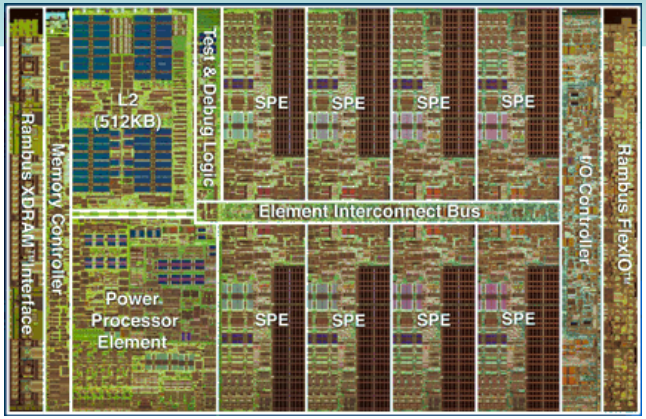




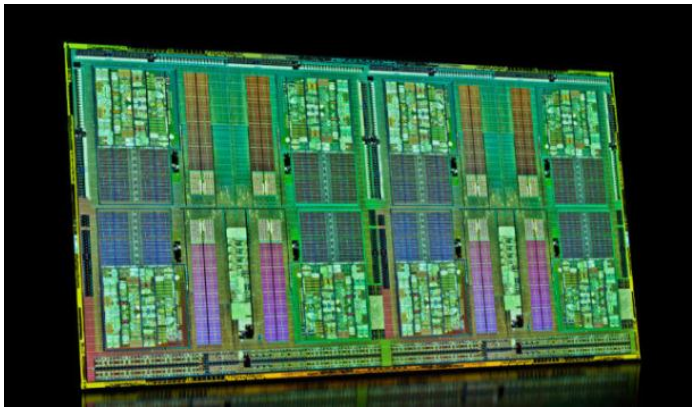
Amdahl's Law for Multicore

- Hill and Marty's study
  - Limited speedup at large-scale size
  - Dynamic architecture delivers a better speedup, but just an "ideal" situation, with $f \geq 0.975$
  - Suggest a large-scale multicore is less interesting
- Current major industries also cite Amdahl's law

Current Industries also Cite Amdahl's law

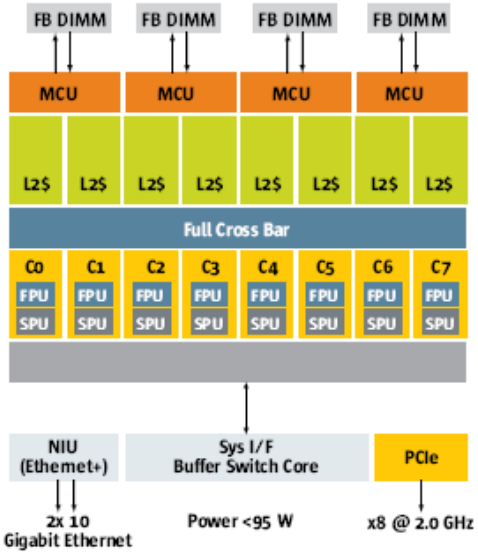


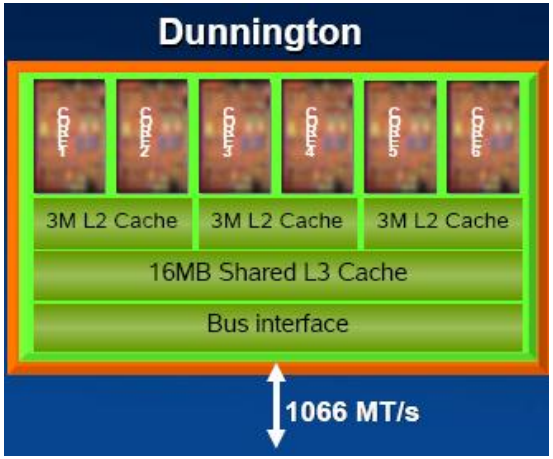
IBM Cell: 8 slave cores
+ 1 master core, 2005



AMD Bulldozer:
16 cores, 2011

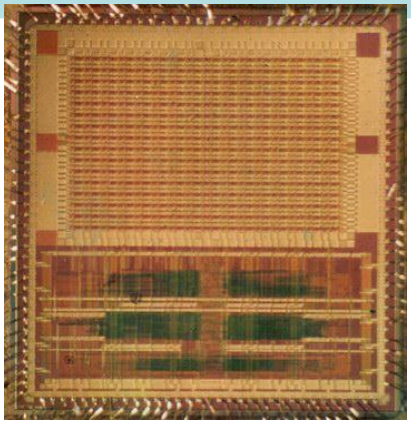
Sun T2: 8
cores, 2007



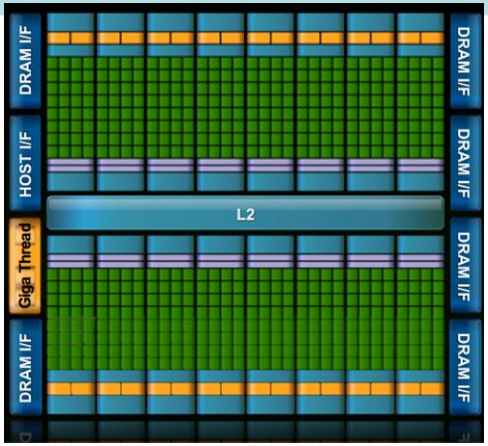


Intel Dunnington: 6 cores, 2008

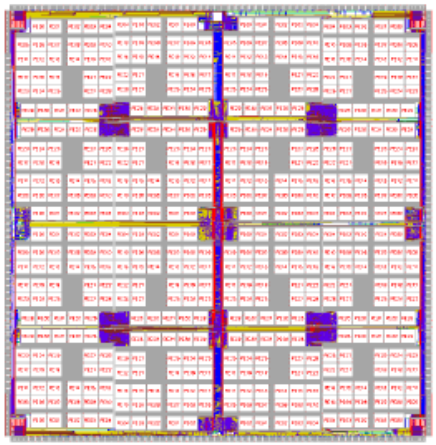
Whereas Technology is Available



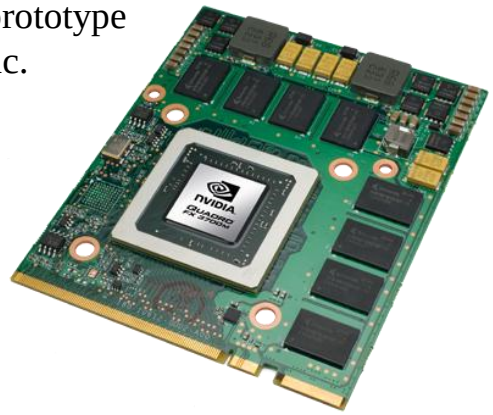
Kilocore: 256-core prototype
By Rapport Inc.



512-core GPUs
NVIDIA FERMI



GRAPE-DR chip:
512-core, By Japan



Quadro FX 3700M:
128-core, By nVIDIA

GRAPE-DR testboard

History Repeats Itself ?

IBM 7030 Stretch

IBM 7950 Harvest

All have up to 8 processors, citing Amdahl's law,

$$\lim_{n\to\infty} Speedup_{Amdahl} = \frac{1}{1-f}$$

Cray X-MP
Fastest computer 1983-1985

Cray Y-MP

Today is Not as Predicted by Amdahl's Law

TOP500 List - November 2011 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.

Power data in KW for entire system

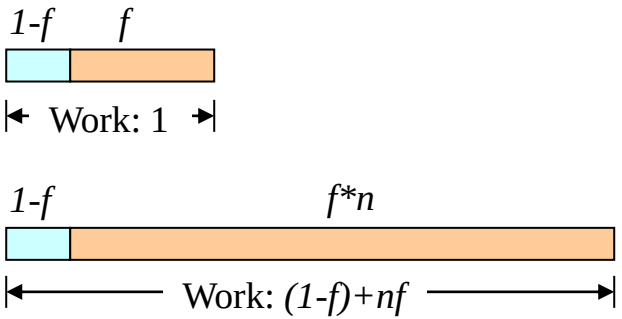
ne

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect / 2011 Fujitsu	705024	10510.00	11280.38	12659.9
2	National Supercomputing Center in Tianjin China	NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 / 2010 NUDT	186368	2566.00	4701.00	4040.0
3	DOE/SC/Oak Ridge National Laboratory United States	Cray XT5-HE Opteron 6-core 2.6 GHz / 2009 Cray Inc.	224162	1759.00	2331.00	6950.0
4	National Supercomputing Centre in Shenzhen (NSCS) China	Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 / 2010 Dawning	120640	1271.00	2984.30	2580.0
5	GSIC Center, Tokyo Institute of Technology Japan	HP ProLiant SL390s G7 Xeon 6C X5670, Nvidia GPU, Linux/Windows / 2010 NEC/HP	73278	1192.00	2287.63	1398.6
6	DOE/NNSA/LANL/SNL United States	Cray XE6, Opteron 6136 8C 2.40GHz, Custom / 2011 Cray Inc.	142272	1110.00	1365.81	3980.0
7	NASA/Ames Research Center/NAS United States	SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon 5570/5670 2.93 Ghz, Infiniband / 2011 SGI	111104	1088.00	1315.33	4102.0
8	DOE/SC/LBNL/NERSC United States	Cray XE6, Opteron 6172 12C 2.10GHz, Custom / 2010 Cray Inc.	153408	1054.00	1288.63	2910.0
9	Commissariat a l'Energie Atomique (CEA) France	Bull bullx super-node S6010/S6030 / 2010 Bull	138368	1050.00	1254.55	4590.0
10	DOE/NNSA/LANL United States	BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009	122400	1042.00	1375.78	2345.0

The scale size is far beyond implication of Amdahl's law

Scalable Computing

- Tacit assumption in Amdahl's law
  - The problem size is fixed
  - The speedup emphasizes time reduction
- Gustafson's Law, 1988
  - Fixed-time speedup model



$$Speedup_{fixed-time} = \frac{Sequential\ Time\ of\ \ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ \ Solving\ Scaled\ Workload}$$
$$= (1-f)+nf$$

- Sun and Ni's law, 1990
  - Memory-bounded speedup model

$$Speedup_{memory-bounded} = \frac{Sequential\ Time\ of\ \ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ \ Solving\ Scaled\ Workload}$$
$$= \frac{(1-f)+fG(n)}{(1-f)+fG(n)/n}$$

Scalable Computing in HPC

Petaflops System

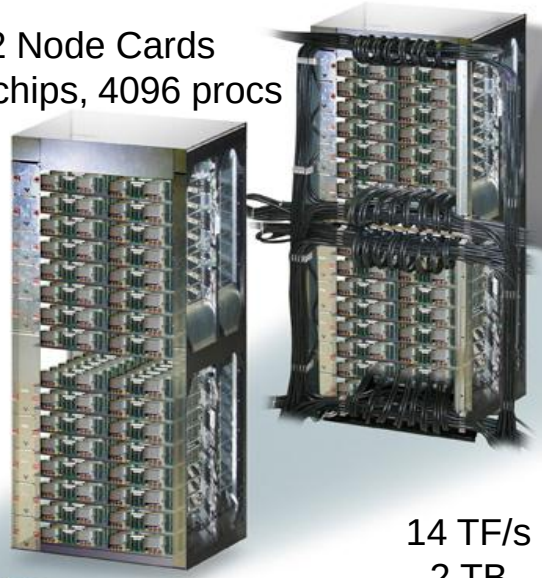
72 Racks

1 PF/s
144 TB

Rack

Cabled 8x8x16

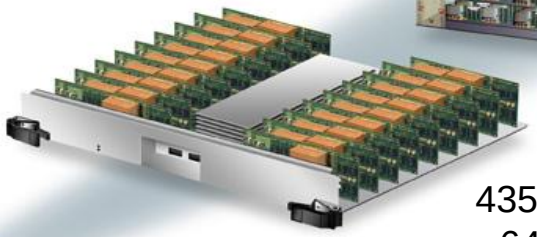
32 Node Cards
1024 chips, 4096 procs



14 TF/s
2 TB

Source: ANL ALCF

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



435 GF/s
64 GB

Compute Card
1 chip, 20 DRAMs

13.6 GF/s
2.0 GB DDR
Supports 4-way SMP

Chip
4 processors

850 MHz
8 MB EDRAM

Front End Node / Service Node
System p Servers
Linux SLES10

Maximum System
256 racks
3.5 PF/s
512 TB

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

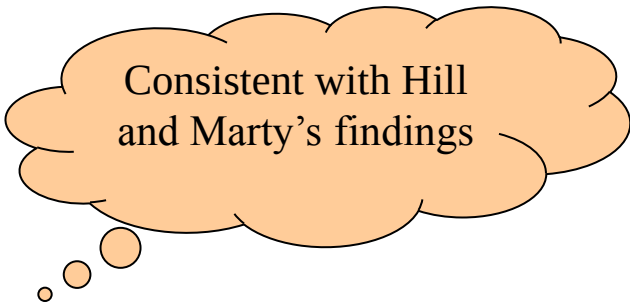
$$Speedup_{fixed-size} = \frac{T_{original}}{T_{enhanced}}$$

$$T_{original} = \frac{w}{perf(1)} = w$$

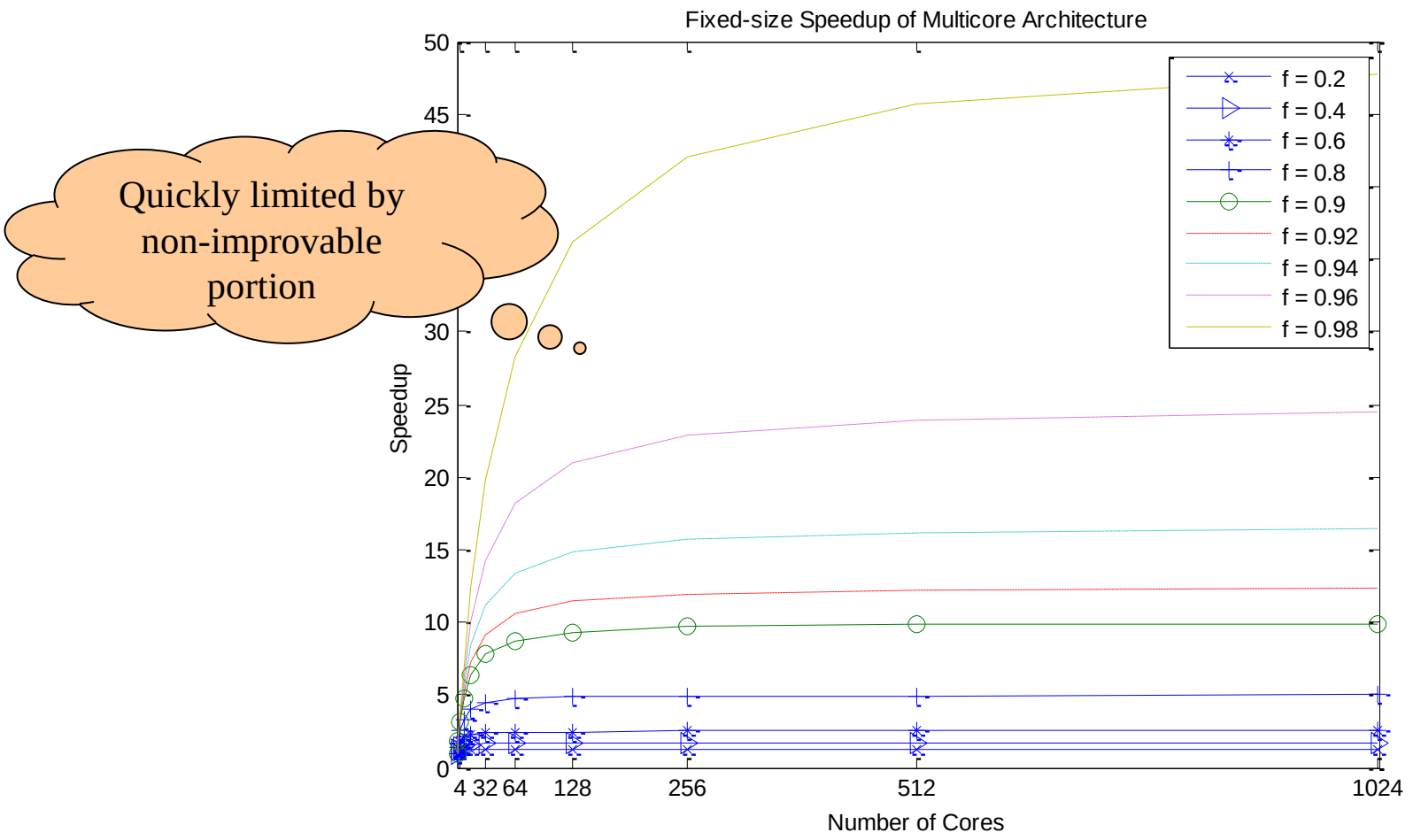
$$T_{enhanced} = \frac{(1-f)w}{perf(r)} + \frac{fw}{\frac{n}{r} \cdot perf(r)}$$

$$Speedup_{fixed-size} = \frac{\frac{w}{perf(1)}}{\frac{(1-f)w}{perf(r)} + \frac{fwr}{n \cdot perf(r)}}$$

$$= \frac{1}{\frac{1-f}{perf(r)} + \frac{f \cdot r}{perf(r) \cdot n}}$$



Fixed-size Speedup for Multicore



Fixed-time Model for Multicore

- 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*

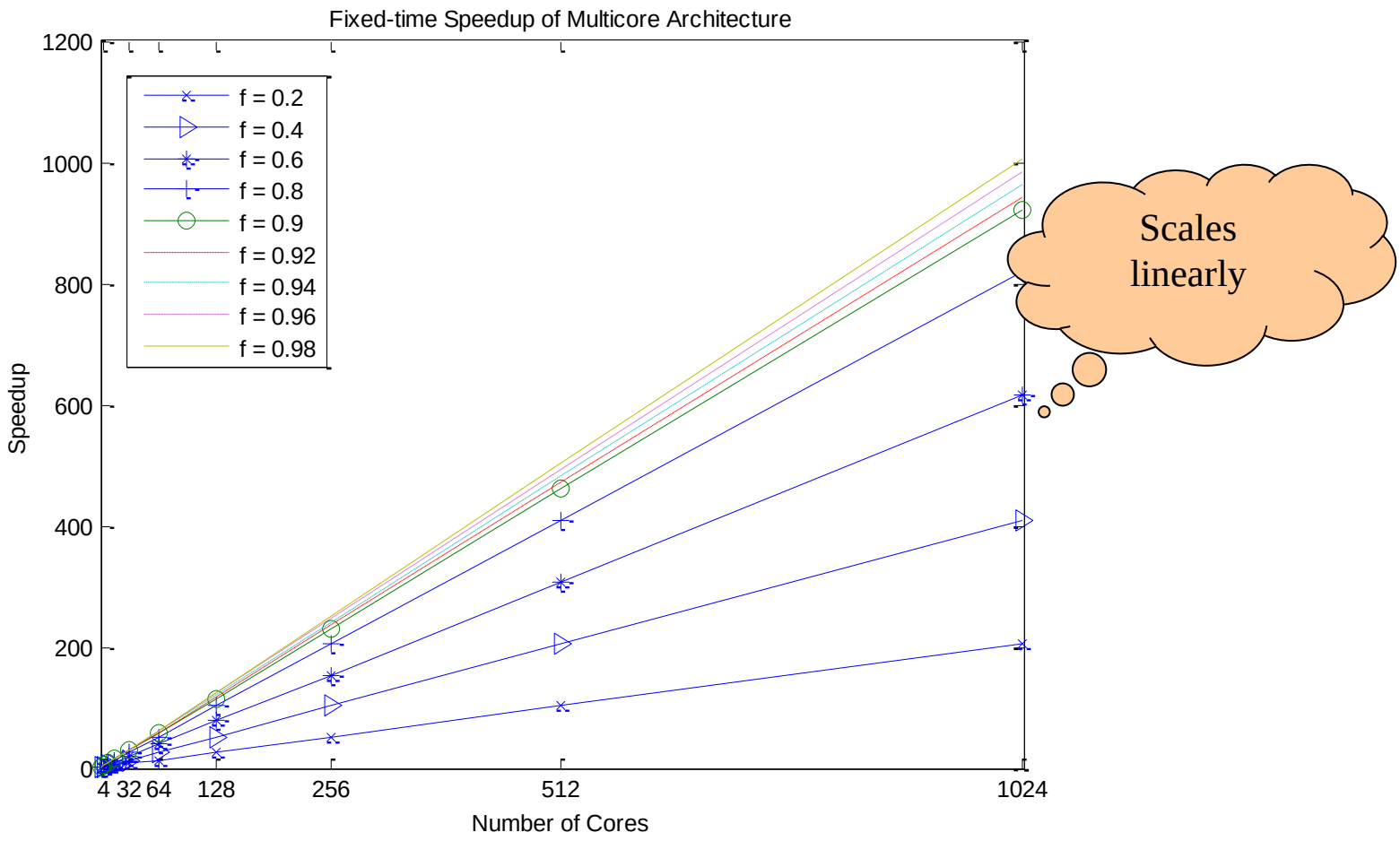
$$\frac{(1-f)w}{perf(r)} + \frac{fw}{perf(r)} = \frac{(1-f)w}{perf(r)} + \frac{fw'}{perf(r)m} \quad => \quad w' = mw$$

$$Speedup_{fixed-time} = \frac{Time\ of\ \ Solving\ w'\ in\ Original\ Mode}{Time\ of\ \ Solving\ w\ in\ Original\ Mode}$$

- The scaled fixed-time speedup

$$= \frac{\frac{(1-f)w}{perf(r)} + \frac{fw'}{perf(r)}}{\frac{w}{perf(r)}} = (1-f) + mf$$

Fixed-time Speedup for Multicore

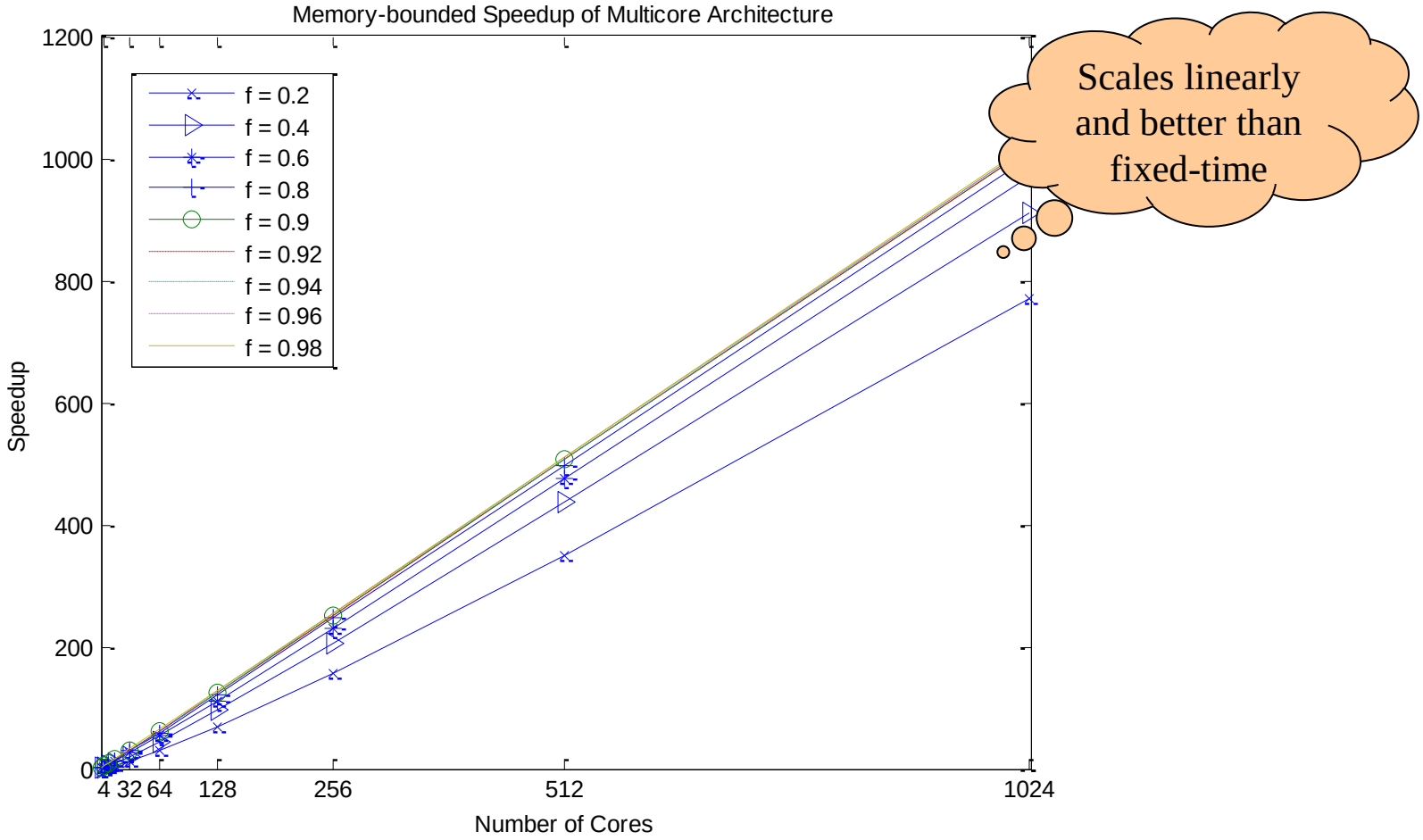


Memory-bounded Model for Multicore

- 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 ($2N^3$ v.s. $3N^2$)
- The scaled memory-bounded speedup

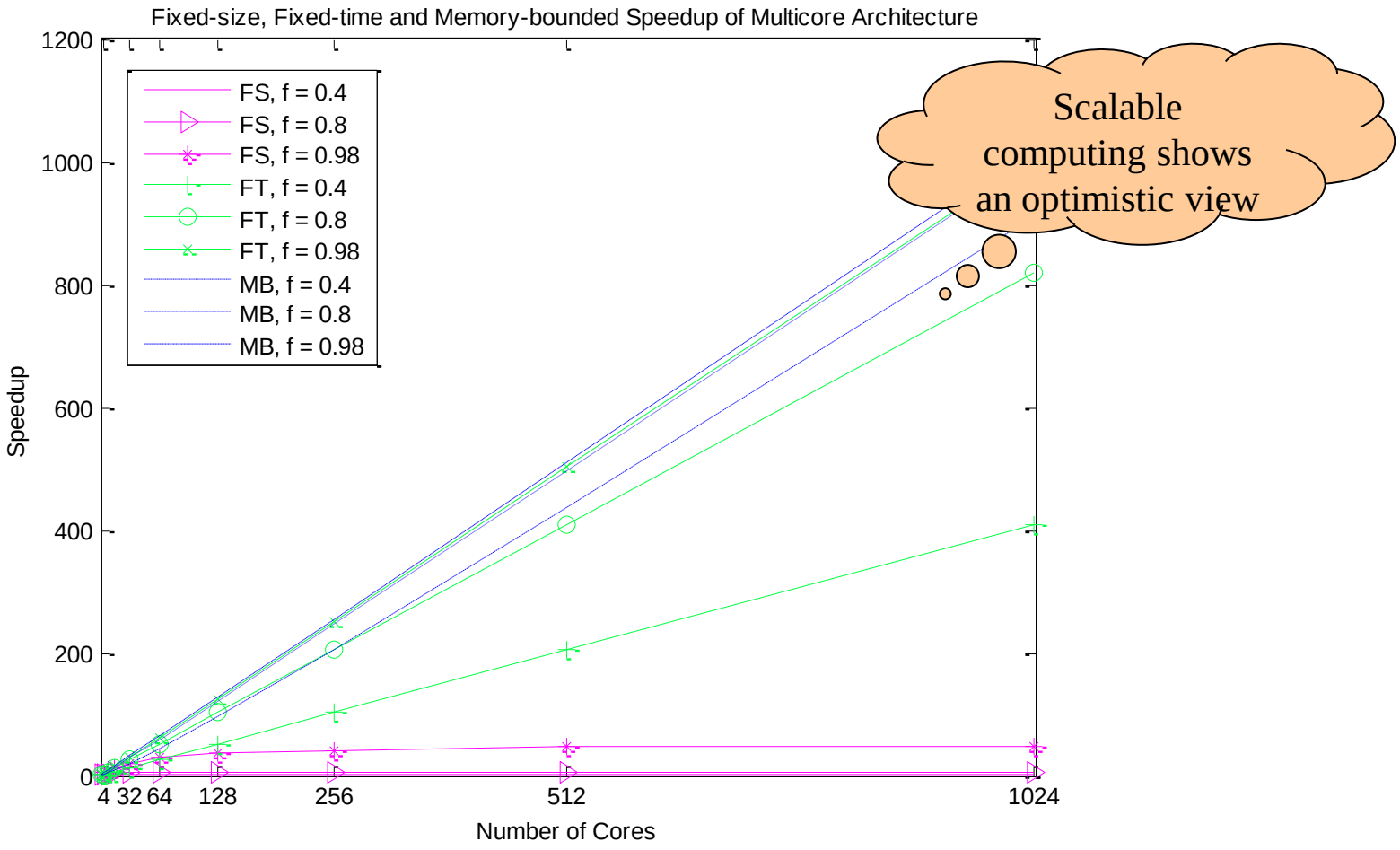
$$Speedup_{memory-bounded} = \frac{Time\ of\ \ Solving\ w^*\ in\ Original\ Mode}{Time\ of\ \ Solving\ w\ in\ Original\ Mode} = \frac{(1-f)+g(m)f}{(1-f)+\frac{g(m)f}{m}}$$

Memory-bounded Speedup for Multicore



$$g(m) = 0.38m^{3/2}$$

Putting It All Together



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

Memory-wall and Multicore Scalability

- 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

$$Speedup = \frac{1}{\frac{w_c}{perf(r)} + \frac{w_p \cdot r}{perf(r) \cdot n}}$$

Scaled Speedup under Memory-wall

- Fixed-time model constraint

$$\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)} = \frac{w_c}{perf(r)} + \frac{w_p'}{m \cdot perf(r)} \quad => \quad w_p' = m w_p$$

- Fixed-time scaled speedup

$$\frac{\frac{w_c}{perf(r)} + \frac{w_p'}{m \cdot perf(r)}}{\frac{w_c}{perf(r)} + \frac{w_p}{perf(r)}} = \frac{w_c + m \cdot w_p}{w_c + w_p} = (1 - f') + m f' \text{ , where } f' = \frac{w_p}{w_c + w_p}$$

- 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

Questions