From Amdahl's Law to Big Data: *A Story of* Mathematics and Technology

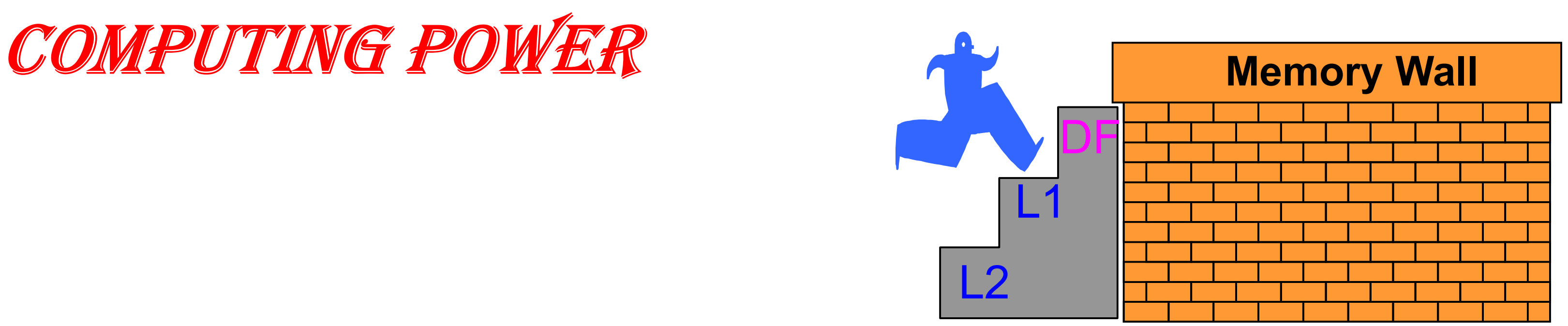
Xian-He Sun

Illinois Institute of Technology
sun@iit.edu

Asian Science Camp, July 30, 2019

Hot Issues

- AI and Deep Learning
- Big Data
- *High Performance and Could Computing*



Summit: the World Fastest Computer

- 148.6 petaflops (187.66 petaflop theoretical peak)
- 2,282,544 IBM Power 9 core
- 2,090,880 Nvidia Volta GV100 core
- Power efficiency 11.324gigaflop/watt

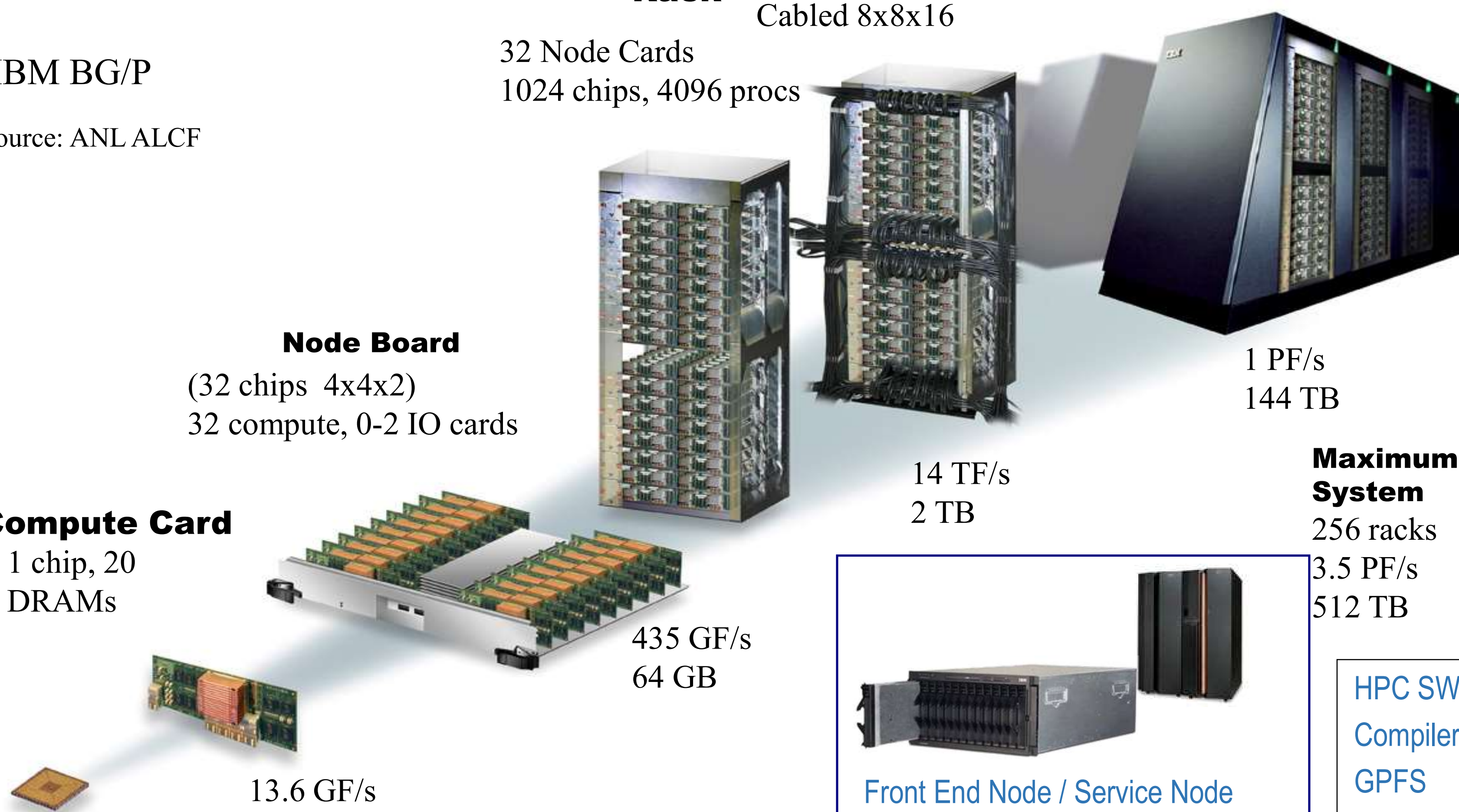
What is Parallel Processing

- Parallel Processing
  - Several working entities work together toward a common goal
- Parallel Computer
  - A computer designed for parallel processing
- Scalable Computing
  - A parallel computing which can be scaled up to larger size without losing efficiency
- Supercomputer (high performance computer, high end computer, advanced computer)
  - A general-purpose computer capable of solving individual problems at extremely high computation speed

Parallel Processing & Scalable Computing

IBM BG/P

Source: ANL ALCF



Chip
4 cores

850 MHz
8 MB EDRAM

Compute Card
1 chip, 20
DRAMs

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

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

435 GF/s
64 GB

Rack
32 Node Cards
1024 chips, 4096 procs

Cabled 8x8x16

14 TF/s
2 TB

Petaflops System
72 Racks

1 PF/s
144 TB

Maximum System
256 racks
3.5 PF/s
512 TB

Front End Node / Service Node
System p Servers
Linux SLES10

HPC SW:
Compilers
GPFS
ESSL
Loadleveler

Why Scalable Computing

– Discretization

–Scalable

More accurate solution

Sufficient parallelism

Maintain efficiency

–Efficient in parallel computing

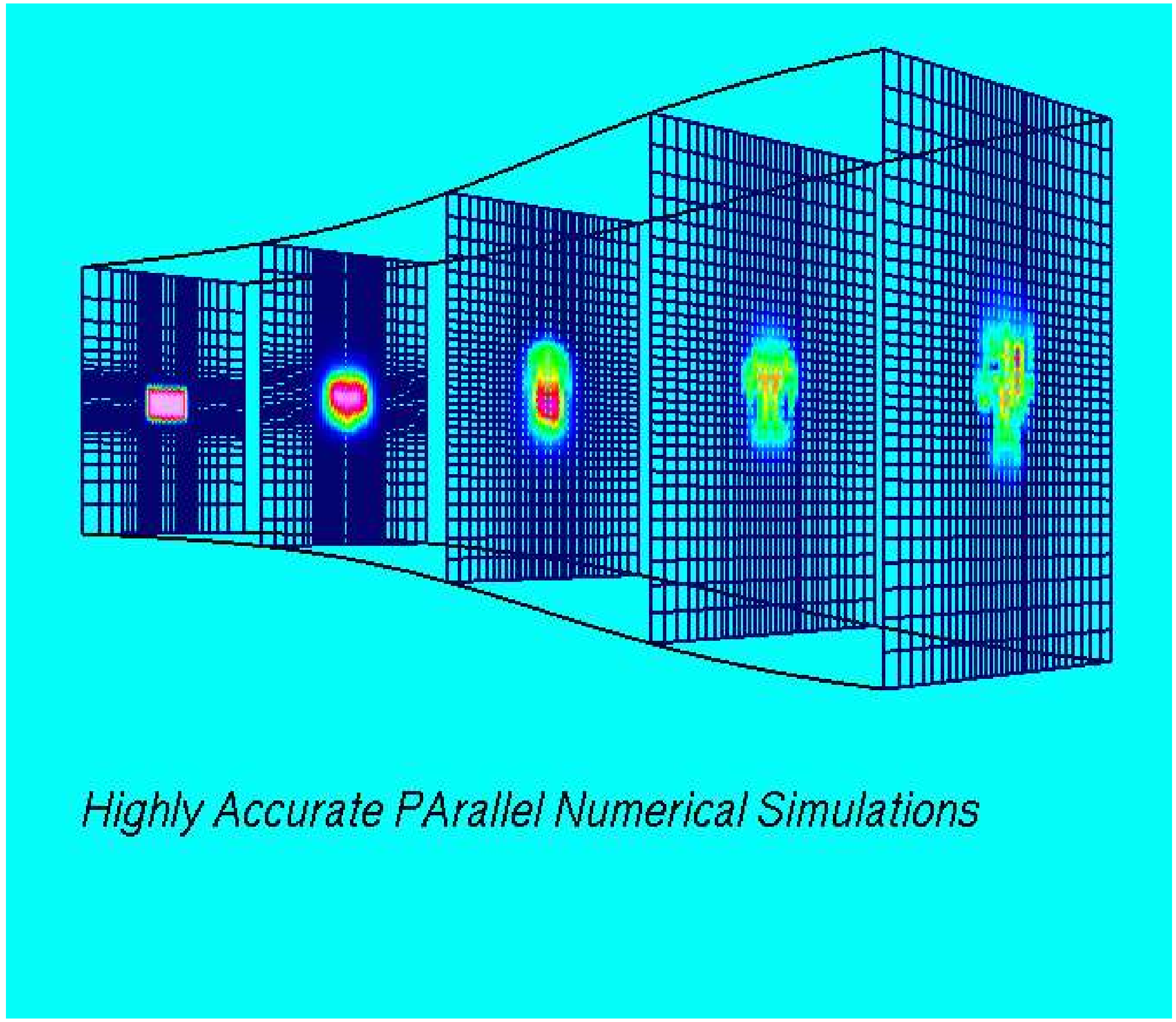
Load balance

Communication

– Mathematically effective

Adaptive

Accuracy



Cloud Computing & Big Data

From High Performance Computing to Cloud to Big Data

The Journey of Supercomputing

- The Background of Parallel Processing
  - Speedup
  - Sources of overhead
- The Laws of Scalable Computing
  - The Amdahl's law
  - The Gustafson's law
  - The Sun-Ni's law
- Impacts and Discussions

Performance of Parallel Processing

Models of Speedup

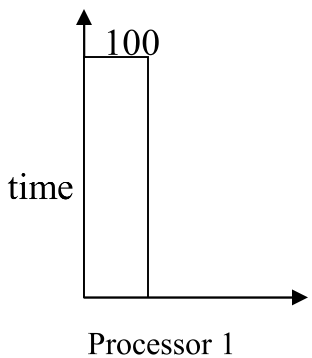
- Speedup
  - Ts = time for the best serial algorithm
  - Tp= time for parallel algorithm using p processors

$$S_p = \frac{T_s}{T_p}$$

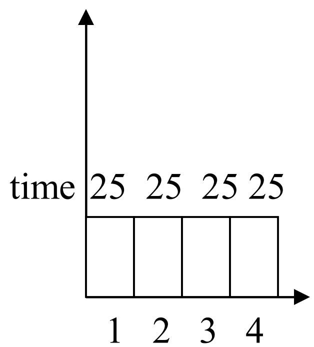
- Simple enough, but also unexpected complex

$$S_p = \frac{\text{Uniprocessor Execution Time}}{\text{Parallel Execution Time}}$$

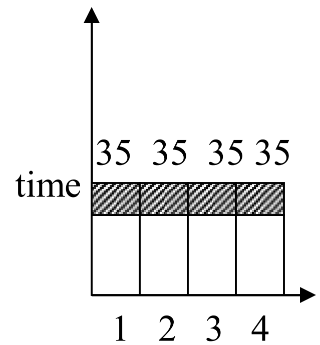
Example



(a)



(b)



(c)

$$S_p = \frac{100}{25} = 4.0,$$

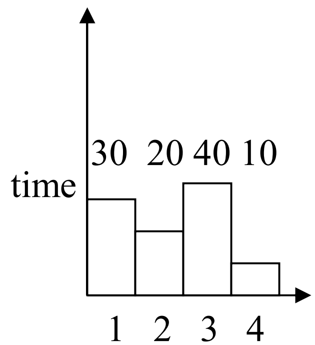
perfect parallelization

$$S_p = \frac{100}{35} = 2.85,$$

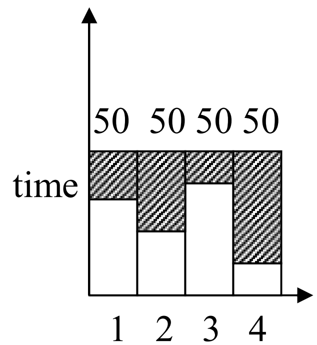
perfect load balancing

but synch cost is 10

Example (cont.)



(d)



(e)

$$S_p = \frac{100}{40} = 2.5,$$

no synch

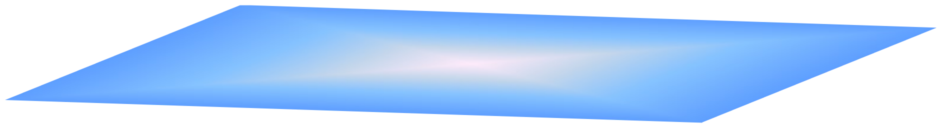
but load imbalance

$$S_p = \frac{100}{50} = 2.0,$$

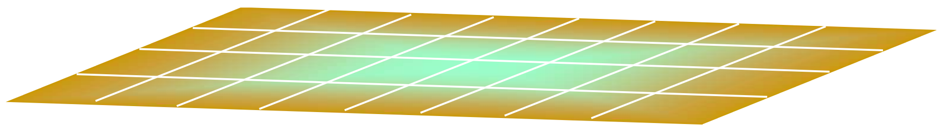
load imbalance

and synch cost

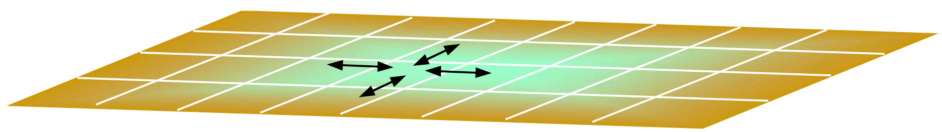
Degradations of Parallel Processing



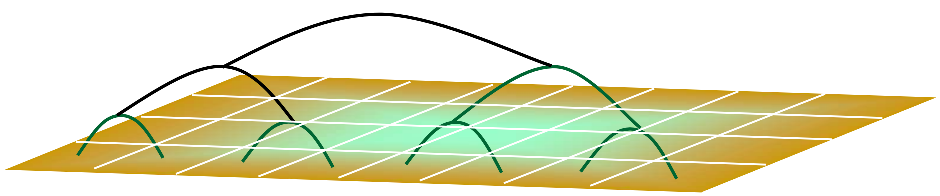
Unbalanced Workload



Communication Delay



Overhead Increases with the Ensemble Size

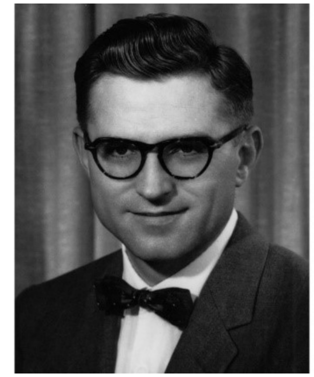


Overheads

- *communication*
- *Load imbalance*
- *Synchronization*
- *Extra computation*

Principals of Architecture Design

- Make common case fast (90/10 Rule)
- Amdahl's Law
  - Law of diminishing returns
- Speedup
  - Achieved performance improvement over original



Gene Amdahl

$$\text{Speedup Overall} = \frac{\text{speed new}}{\text{speed old}} = \frac{\text{execution time old}}{\text{execution time new}}$$

Here performance is measured in **Speed**

Amdahl's Law

Execution time of any code has two portions

Portion I: not affected by enhancement
Portion II: affected by enhancement

$$\text{execution time}_{old} = \underbrace{\text{execution time}_{p1}}_{} + \text{execution time}_{p2}$$

α is % of original code that cannot benefit from enhancement

As p -> infinity, execution time$_{new}$ -> α * execution time$_{old}$

$$\text{execution time}_{new} = \underbrace{(\alpha) * \text{execution time}_{old}}_{\text{Execution time}_{p1}} + \underbrace{(1-\alpha) * \frac{\text{execution time}_{old}}{p}}_{\text{Execution time}_{p2}}$$

p is speedup factor of old/new execution times for portion II

Amdahl's Law for Parallel Processing (1967)

- Let α= fraction of program (algorithm) that is <u>serial</u> and <u>cannot be parallelized</u>. For instance:
  - Loop initialization
  - Reading/writing to a single disk
  - Procedure call overhead
- Parallel run time is given by

$$\text{execution time}_{\text{new}} = (\alpha) * \text{execution time}_{\text{old}} + (1-\alpha) * \frac{\text{execution time}_{\text{old}}}{p}$$

$$T_p = (\alpha + \frac{1-\alpha}{p}) \bullet T_s$$

Gene M Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," AFIPS spring joint computer conference, 1967

Amdahl's Law

- Amdahl's law gives a limit on speedup in terms of α

$$S_p = \frac{T_s}{T_p} = \frac{T_s}{\alpha T_s + \frac{(1-\alpha)T_s}{p}} = \frac{1}{\alpha + \frac{1-\alpha}{p}}$$

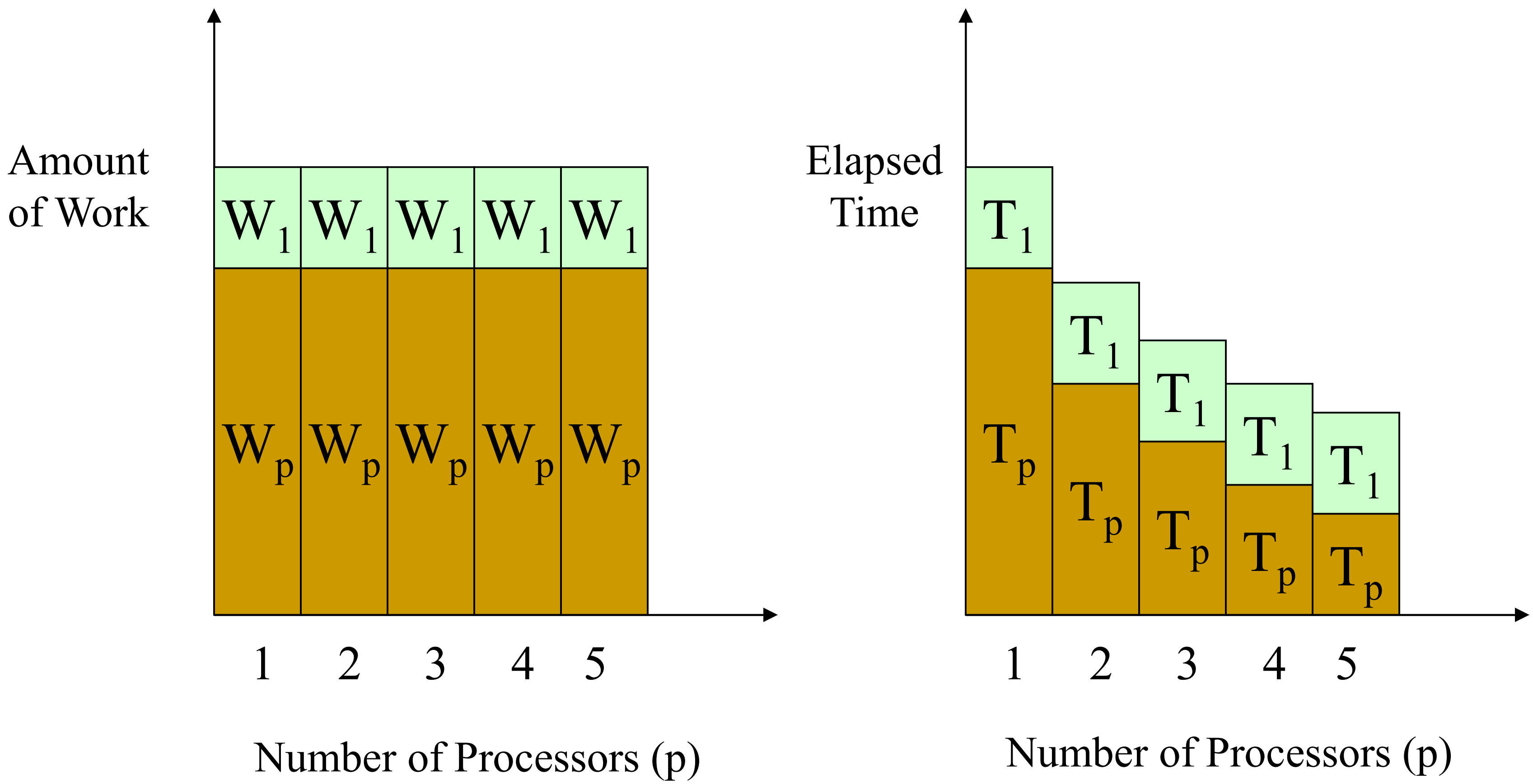
- If we assume that the serial fraction is fixed, then the speedup for infinite processors is limited by $1/\alpha$

$$\lim_{p \to \infty} S_p = \frac{1}{\alpha}$$

- For example, **if α=10%,** then the maximum speedup is **10,** even if we use an infinite number of processors

Amdahl Law

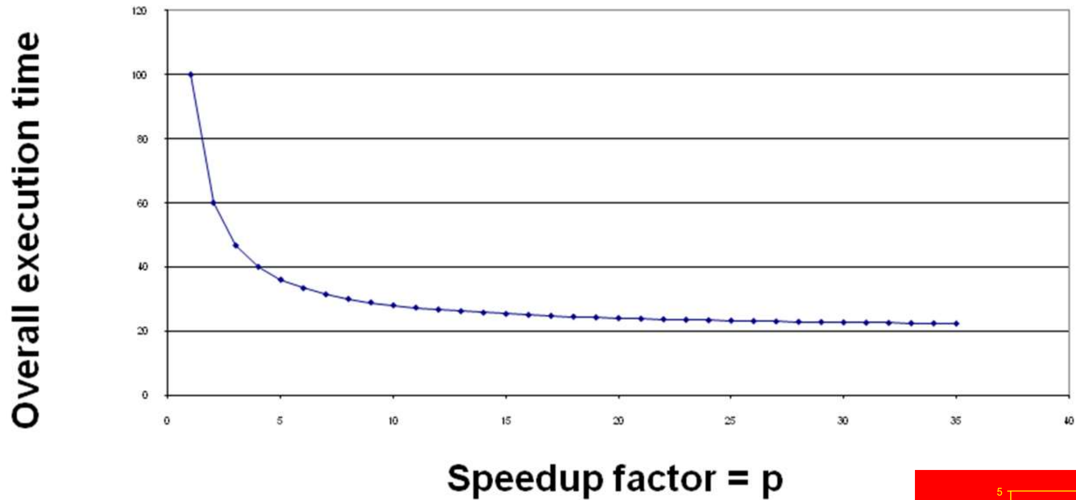
- The sequential part becomes the dominate factor quickly



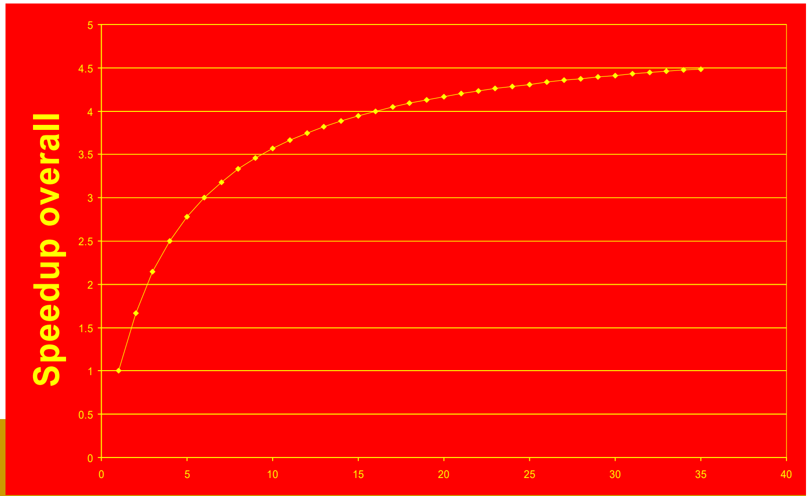
Amdahl's Law

$$\text{execution time}_{\text{new}} = (\alpha) * \text{execution time}_{\text{old}} + (1-\alpha) * \frac{\text{execution time}_{\text{old}}}{p}$$

Example: alpha = 20%



$$\text{Speedup}_{\text{overall}} = \frac{\text{execution time}_{\text{old}}}{\text{execution time}_{\text{new}}} = \frac{1}{(\alpha) + \frac{1-\alpha}{p}}$$



Amdahl's Law with Overhead

- To include overhead will be even worse
- The overhead includes parallelism and interaction overheads

$$Speedup_{FS} = \frac{T_1}{\alpha T_1 + \frac{(1-\alpha)T_1}{p} + T_{overhead}} \rightarrow \frac{1}{\alpha + \frac{T_{overhead}}{T_1}} \; as \;\; p \rightarrow \infty$$

Amdahl's law: argument against massively parallel systems

History back to 1988

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

$$\lim_{p \to \infty} Speedup_{Amdahl} = \frac{1}{\alpha}$$

IBM 7030 Stretch

IBM 7950 Harvest

Cray X-MP
Fastest computer 1983-1985

Cray Y-MP

Gene Amdahl

Bombshell: *Gustafson, etc. Got Speedup of more than 1,000 on Three Applications*

- On a 1024-processor nCUBE parallel computer
- For three applications: wave mechanics, fluid dynamics, and structural analysis.
- Introduced the concept of **Scalable Computing,** *problem size increases with the machine size*

John L. Gustafson, Gary R. Montry, and Robert E. Benner, "Development of Parallel Methods for a 1024-Processor Hypercube," SIAM Journal on Scientific and Statistical Computing, Vol. 9, No.4, 1988 (submitted 3/10/1988, accepted 3/25/1988, appeared April 1988)

John Gustafson, "Reevaluation of Amdahl's Law," Communications of the ACM, Vol. 31, No. 5, May 1988.

Reevaluate Amdahl's Law

- **Amdahl's Law** is designed for technology improvement, but has been widely used to against parallel processing in terms of reducing execution time

- **But**: large computers are not (only) designed for solving existing problem faster, they are designed for solving otherwise unsolvable large problems

- The introduction of **scalable computing,** where *problem size increases with the machine size*

- Fixed-Time Speedup (Gustafson, 88)
  - Emphasis on work finished in a fixed time
  - Problem size is **scaled** from W to W'
  - W': Work finished within the fixed time with parallel processing

John L. Gustafson

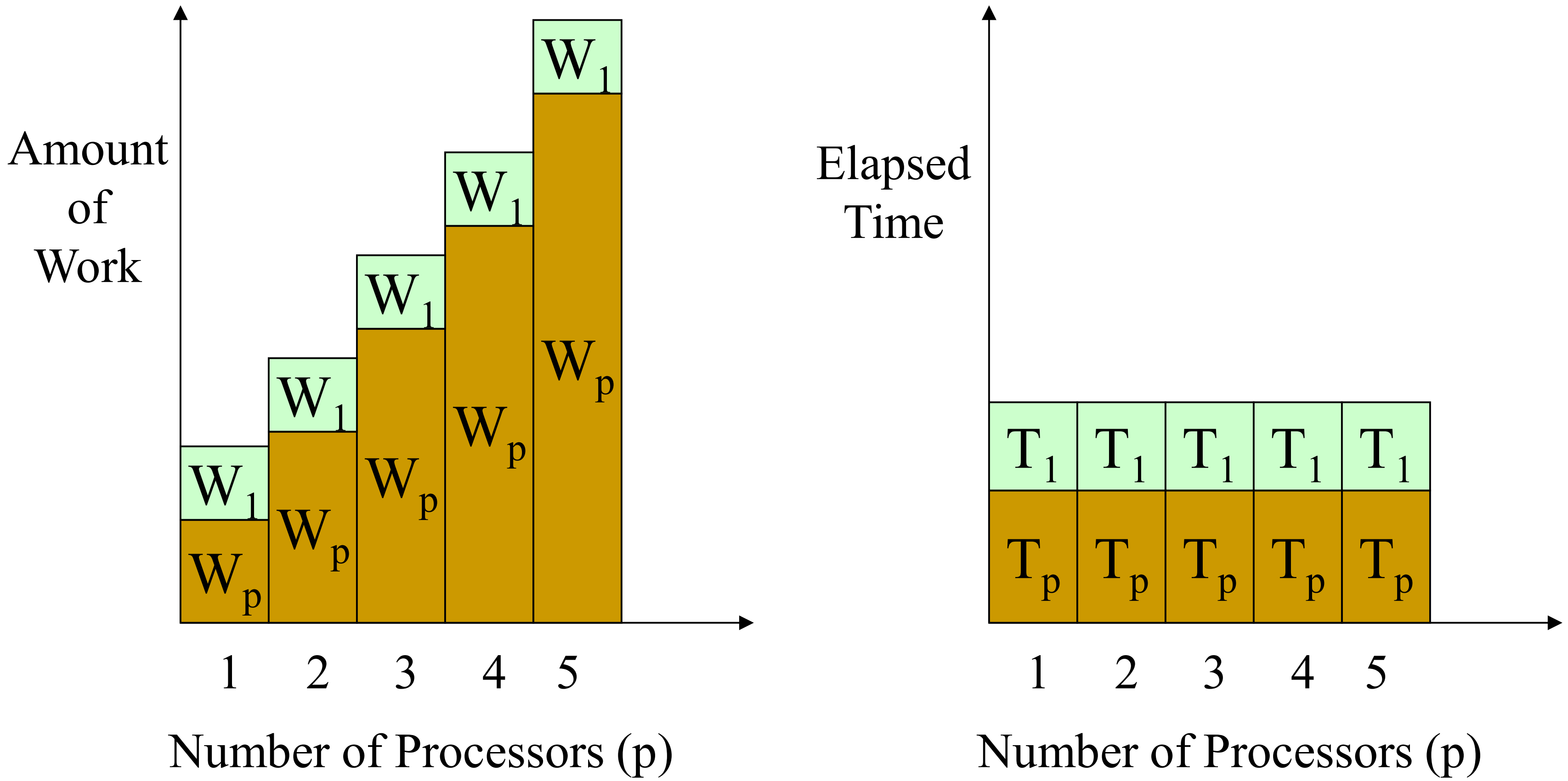
$$S'_p = \frac{\text{Uniprocessor Time of Solving } W'}{\text{Parallel Time of Solving } W'}$$

$$= \frac{\text{Uniprocessor Time of Solving } W'}{\text{Uniprocessor Time of Solving } W}$$

$$= \frac{W'}{W}$$

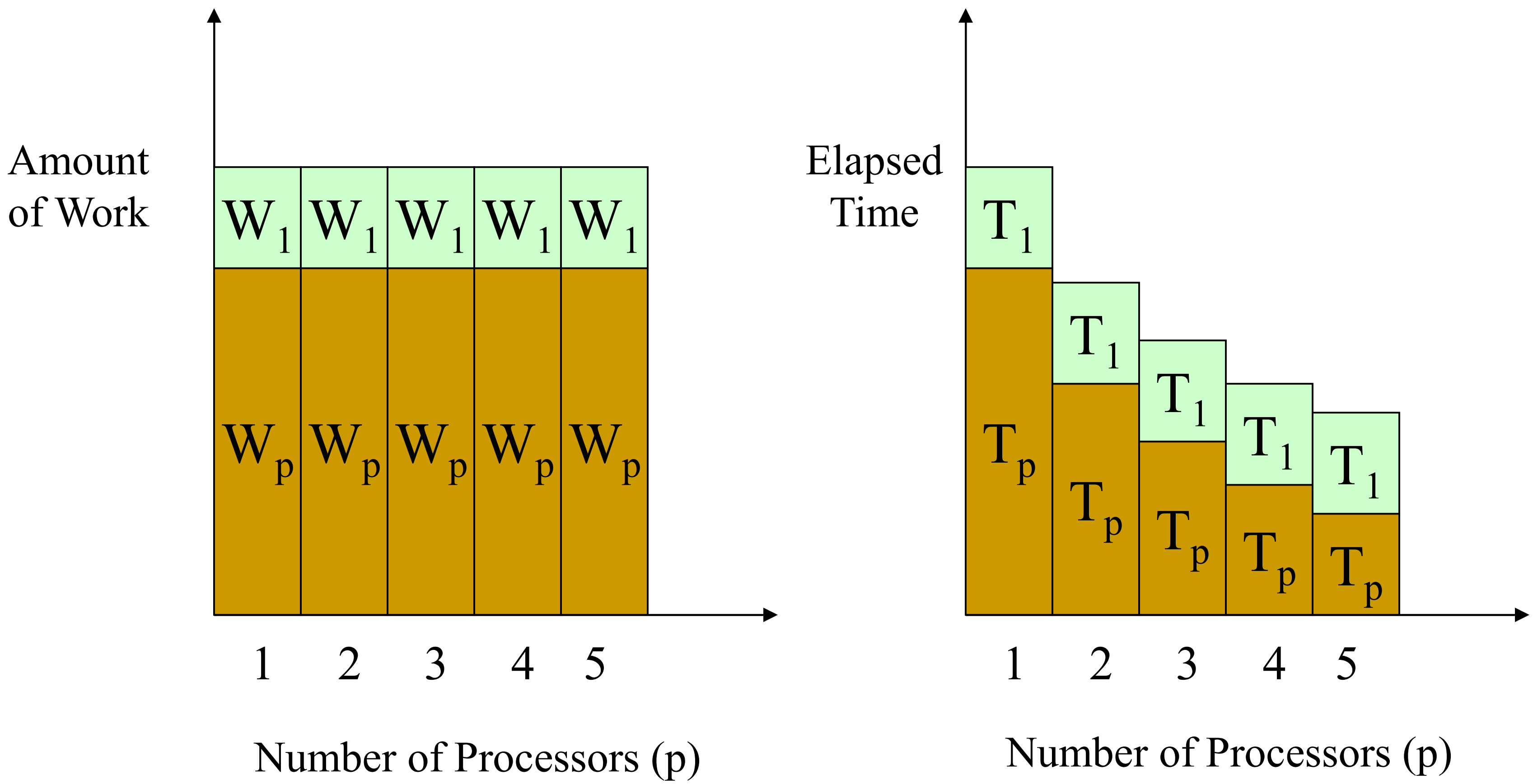
Fixed-Time Speedup (Gustafson)

- Solving a larger application within the time limit



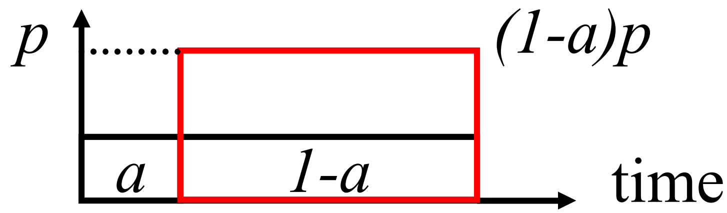
Reexam Amdahl Law (Fixed-Size Speedup)

- It is on time reduction for solving a fixed problem (size)



Gustafson's Law (Without Overhead)

- Under **Gustafson's Law** the parallel processing part is changing with the number of processors, *p,* and problem size
- Linear speedup



$$\alpha = \frac{t_s}{t_s + t_p}$$

$$Speedup_{FT} = \frac{Work(p)}{Work(1)} = \frac{\alpha W + (1-\alpha)pW}{W} = \alpha + (1-\alpha)p$$

If α=0.1

$$Speedup_{FT} = \alpha + (1-\alpha)p = 0.1 + 0.9p$$

But: *Gustafson's Applications are not Scalable*

- Most applications cannot get more than 1,000 speedup on a 1024-processor nCUBE parallel computer

  Parallel Processing overhead

- Even the three applications are not **Scalable** (increase *problem size further does not help)*

  Why?

Memory Constrained Scaling:

Sun and Ni's Law

- **Scaling is limited by memory space** (disk will increase overhead significantly), e.g. fixed memory capacity/usage per processor
  - (ex) N-body problem
- Problem size is scaled from W to W*, W* is the work executed under memory limitation
- The relation between memory & computing requirement is determined by the underlying algorithm/program
- **Memory-scaling function**

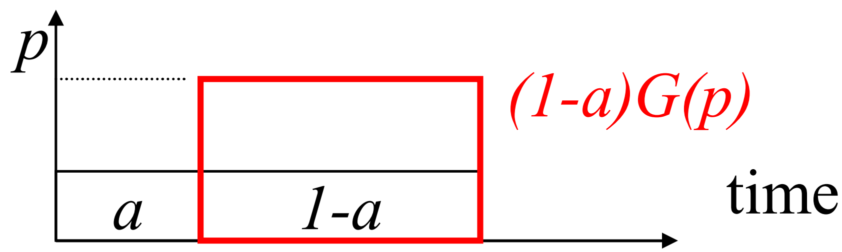
$$W^* = G(p * M)$$

X.H. Sun, and L. Ni , *"Scalable Problems and Memory-Bounded Speedup,"* *Journal of Parallel and Distributed Computing*, Vol. 19, pp.27-37, Sept. 1993 (**SC90**).

Sun & Ni's Law

存储受限理论

Xian-He Sun



Lionel M. Ni

$$Speedup_{MB} = \frac{Work(p)/Time(p)}{Work(1)/Time(1)} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)G(p)/p}$$

Assuming $\alpha = 0.1$, the problem needs $2n^3$ computation and $3n^2$ memory

Then $G(p) = G(\mathrm{p}) = p^{\frac{3}{2}}$, and

$$Speedup_{MB} = \left(0.1 + 0.9 \times p^{\frac{3}{2}}\right) / \left(0.1 + (0.9 \times p^{\frac{3}{2}})/p\right)$$

Memory-Bounded Speedup 存储受限理论

(Sun & Ni, 90)

- Emphasis on work finished under current physical limitation
  - Problem size is scaled from W to W^*
  - W^*: Work executed under memory limitation with parallel processing

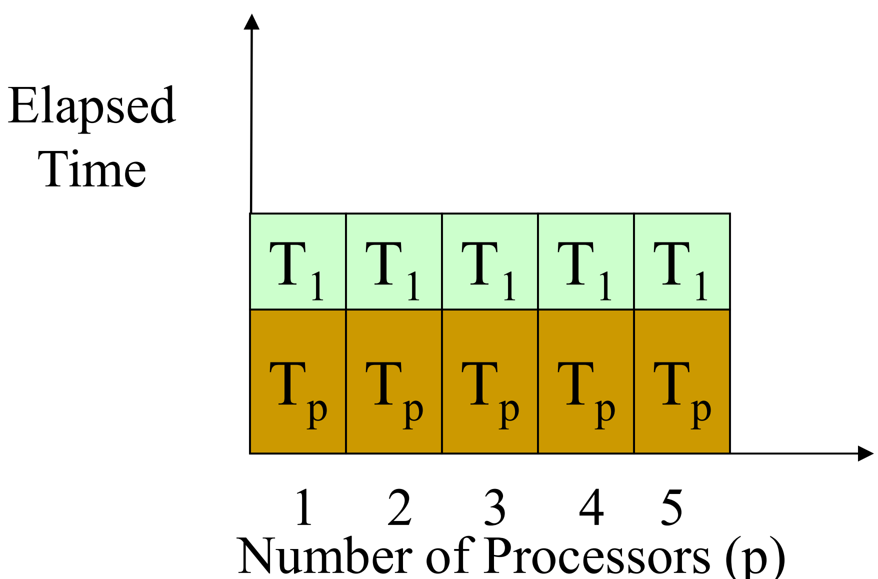
$$S_p^* = \frac{\text{Uniprocessor Time of Solving } W^*}{\text{Parallel Time of Solving } W^*}$$

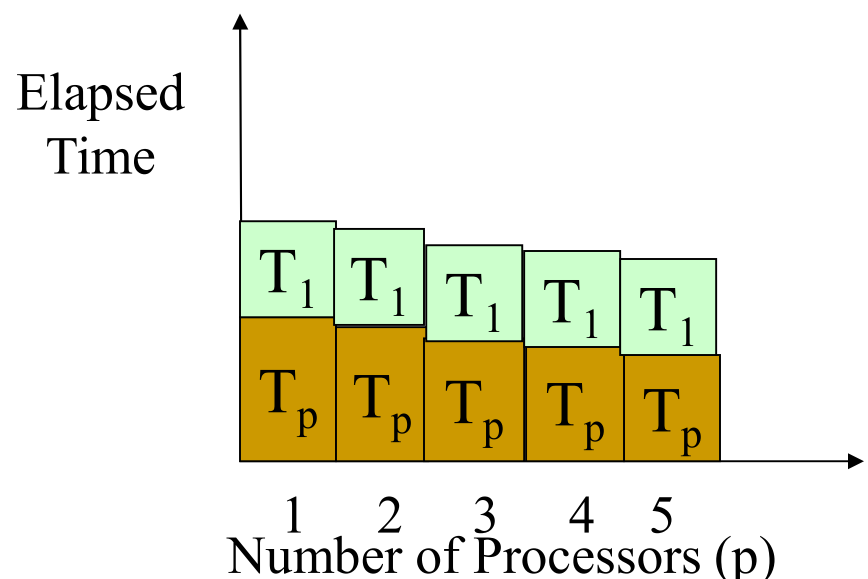
Memory-Bounded Speedup (Sun & Ni)

- In practice, memory-bounded performs better than fixed-time but both hard to achieve linear speedup

$$Speedup_{MB} = \frac{Work(p)/Time(p)}{Work(1)/Time(1)} = \frac{\alpha + (1-\alpha)G(p)}{\alpha + \frac{(1-\alpha)G(p)}{p} + overhead(p, G(p))}$$

$$Speedup_{FT} = \frac{Work(p)}{Work(1)} = \alpha + (1 - \alpha - \frac{T_{overhead}}{T_1})p$$





Rethinking of Speedup

- Speedup

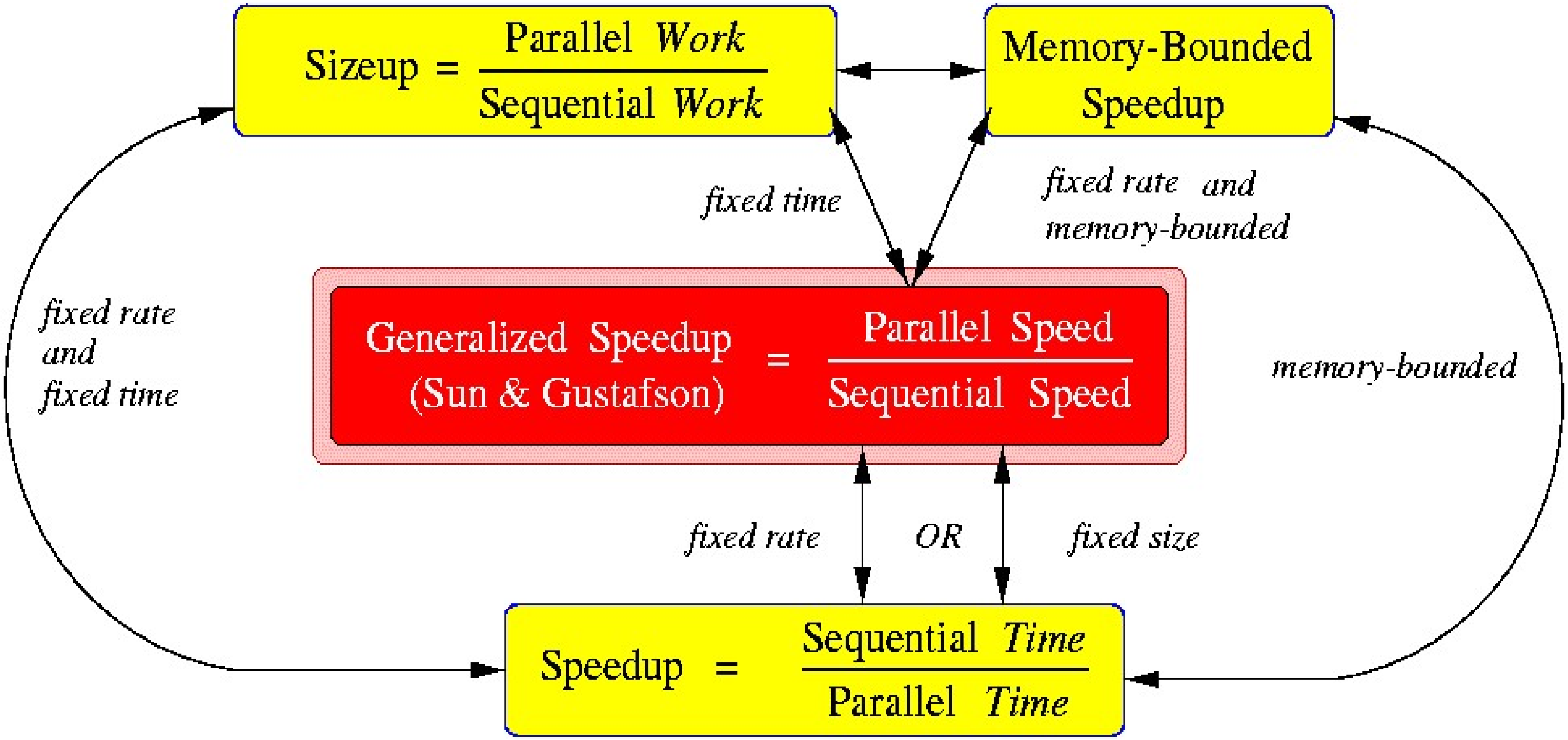
$$S_p = \frac{Uniprocessor\ Execution\,Time}{Parallel\ Execution\,Time}$$

- It is only the true speedup if problem size is fixed, but now we have scalable computing
- Generalized speedup

$$S_p = \frac{\text{Parallel Speed}}{\text{Sequential Speed}}$$

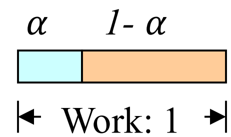
X.H. Sun, and J. Gustafson, *"Toward A Better Parallel Performance Metric," Parallel Computing*, Vol. 17, pp.1093-1109, Dec. 1991.

Models of Speedup

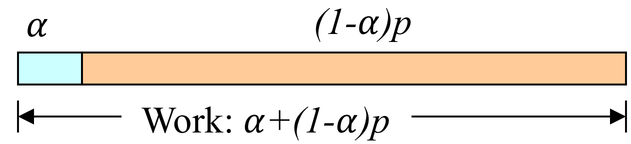


The Three Laws

- Tacit assumption of Amdahl's law
  - Problem size is fixed
  - Speedup emphasizes on time reduction

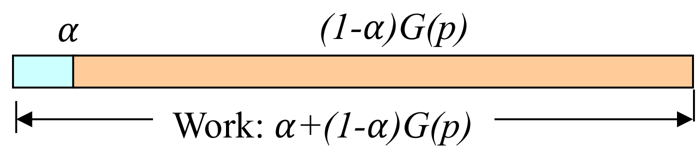


- Gustafson's Law, 1988
  - Fixed-time speedup model



$$Speedup_{fixed-time} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload}$$

$$= \alpha + (1-\alpha)p$$



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

$$Speedup_{memory-bound} = \frac{Sequential\ Time\ of\ Solving\ Scaled\ Workload}{Parallel\ Time\ of\ Solving\ Scaled\ Workload}$$

$$= \frac{\alpha + (1-\alpha)G(p)}{\alpha + (1-\alpha)\,G(p)/p}$$

X.-H. Sun, and L. Ni, "Another View of Parallel Speedup," Proc. of IEEE Supercomputing'90, NY, NY, Nov.12--Nov.16, 1990.

The Three Laws: and their impact

I can improve Amdahl's law

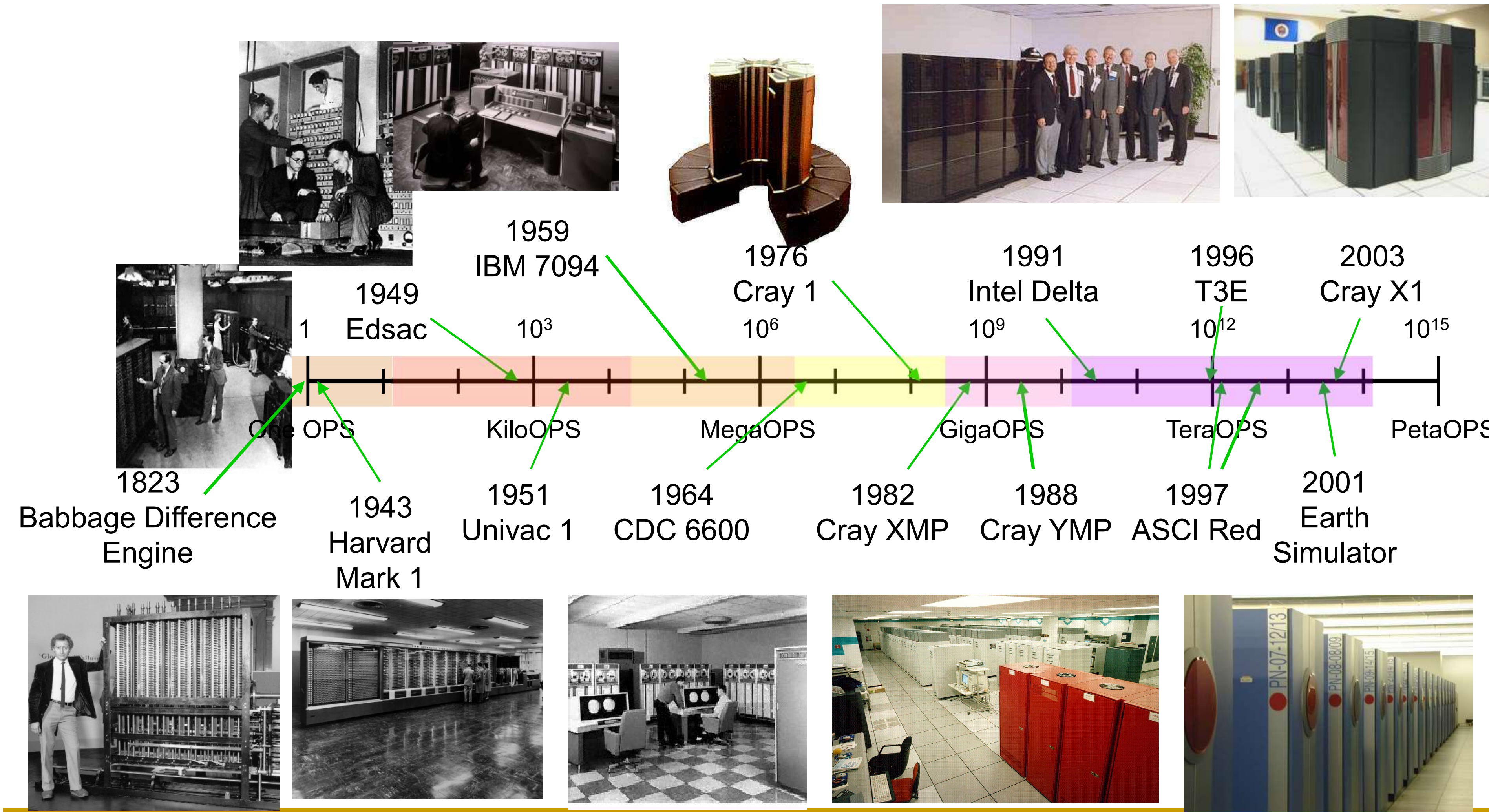
- **Amdahl's law** (1967) shows the inherent limitation of parallel processing
- **Gustafson's law** (scalable computing, 1988) shows there is no inherent limitation for scalable parallel computing, except engineering issues
- **Sun-Ni's law** (memory-bounded, 1990) shows memory (data) is the constraint of scalable computing (**the** engineering issue)

I have a huge memory

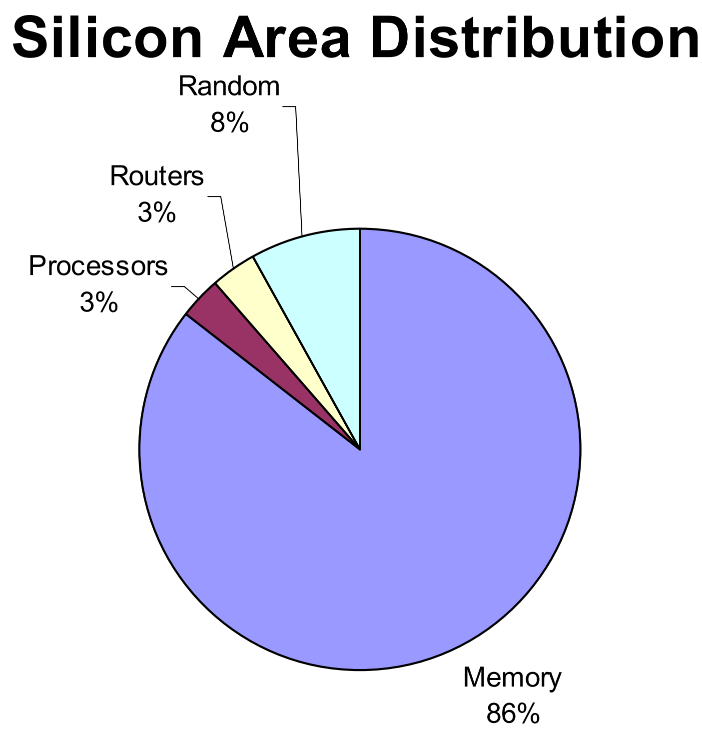
- The **Memory-Wall Problem** (1994) shows memory-bound is a general performance issue for computing, not just for parallel computing

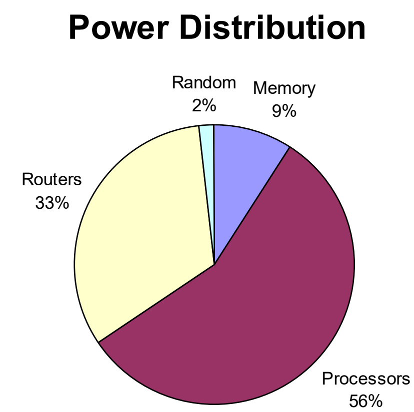
William Wulf, Sally Mckee, "Hitting the memory wall: implications of the obvious," ACM SIGARCH Computer Architecture News Homepage archive, Vol. 23 Issue 1, March 1995

Impact of Scalable Computing



Impact: Computing/Memory Trade-off





Modern microprocessors such as the Pentium Pro, Alpha 21164, Strong Arm SA110, and Longson-3A use 80% or more of their transistors for the on-chip cache

Courtesy of Peter Kogge, UND

Impact of Memory-Bounded Speedup

- **W = G(M)** shows the trade-off between computing & memory
  - W, the work in floating point operation
  - M, the memory requirement
  - G, the data reuse rate

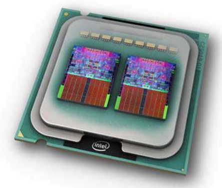
- **W = G(M)** unifies the models
  - G(p) = 1, Amdahl's law
  - G(p) = p, Gustafson's law

- Reveal memory is the performance bottleneck
  - Memory-bounded algorithms and analysis in

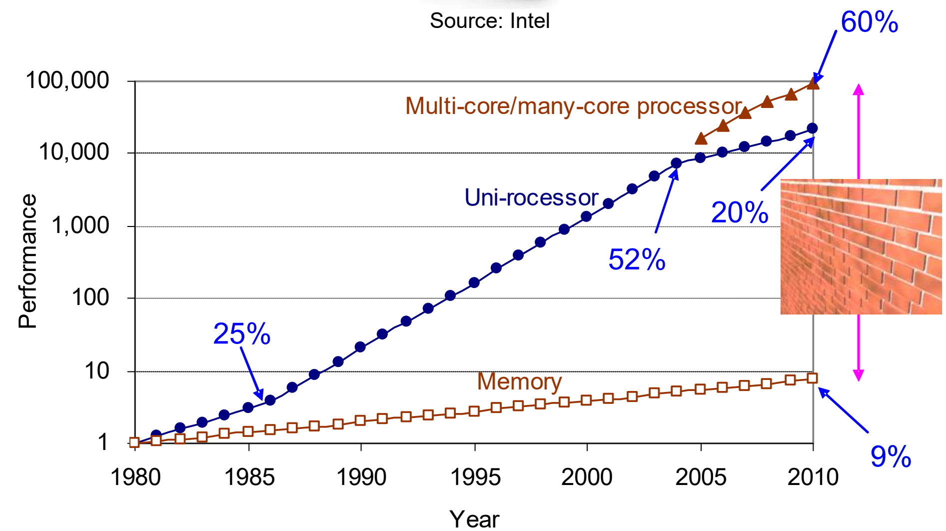
    Dynamic programming, distributed optimization, search, convolution, regression, etc.
  - The Memory-Wall problem (1994)

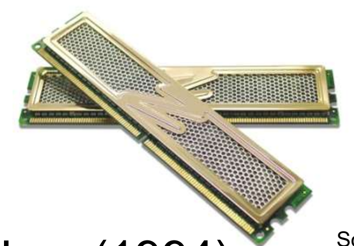
Impact: The Memory-wall Problem

- Processor performance increases rapidly
  - Uni-processor: ~52% until 2004
  - Aggregate multi-core/many-core processor performance even higher since 2004
- Memory: ~9% per year
  - Storage: ~6% per year
- Processor-memory speed gap keeps increasing



Source: Intel





Source: OCZ

Memory-bounded speedup (1990), Memory wall problem (1994)

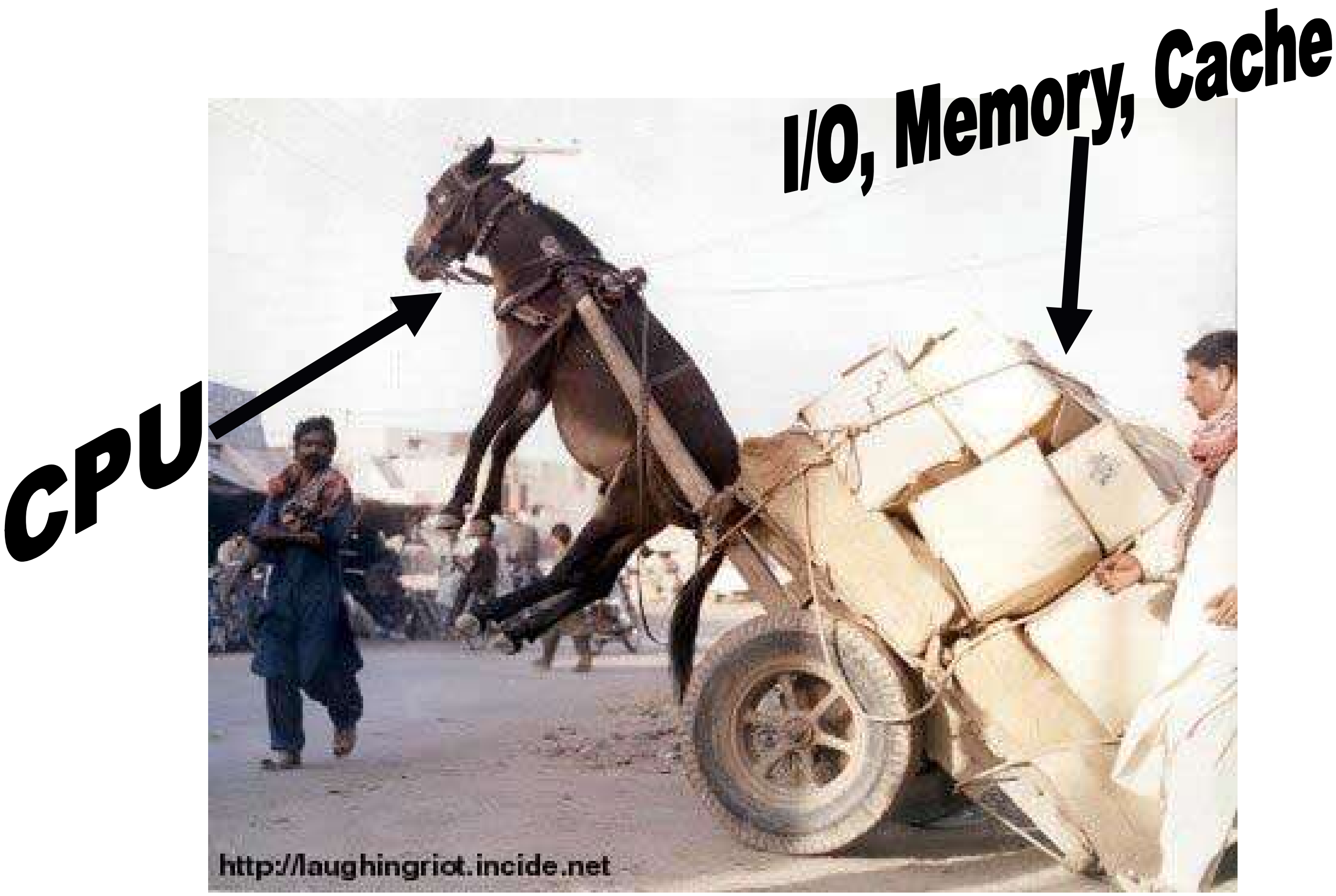
The Beauty of Mathematics

- The ability of abstract
- In depth understanding
  of the engineering issues
- Creative thinking

- Complex Specificity, Simple Genericity
- Abstract the complex specificity into simple genericity
- Engineering, mathematics, philosophy
- Everybody understand something, at a different level
- Your understanding determine your ability to apply it

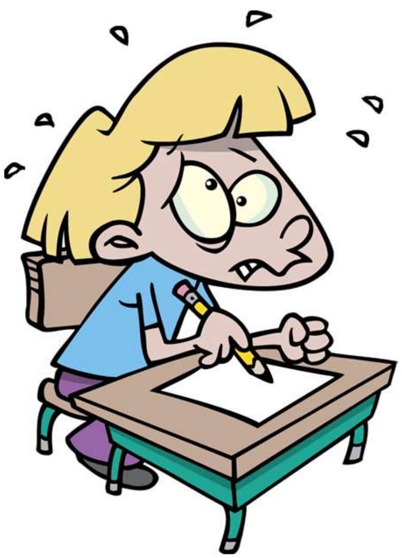
- 厚积薄发，可遇不可求

Big Data Makes Memory-Bound Even Worse

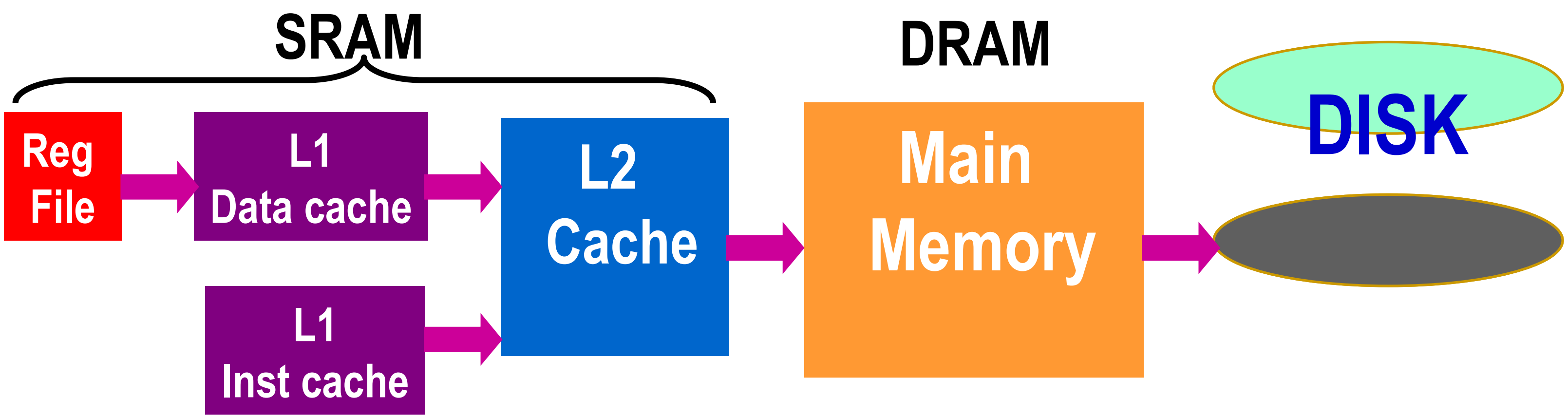


- Source: Bob Colwell keynote ISCA'29 2002 http://systems.cs.colorado.edu/ISCA2002/Colwell-ISCA-KEYNOTE-2002-final.ppt

How do we solve the memory-bound constraint or the memory-wall problem



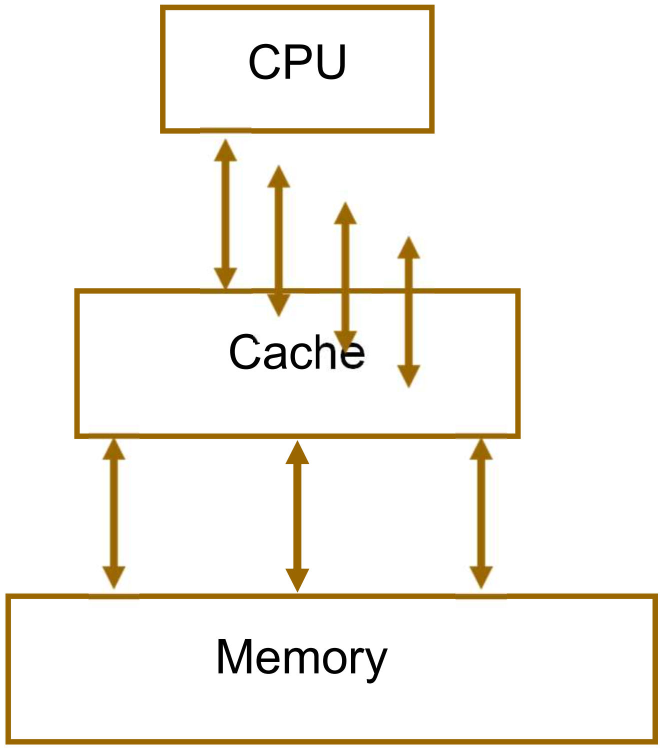
Solution: Memory Hierarchy



More on Memory Hierarchy & Concurrency

Multi-core
Multi-threading
Multi-issue

Out-of-order Execution
Speculative Execution
Runahead Execution



Multi-banked Cache
Multi-level Cache

Pipelined Cache
Non-blocking Cache
Data Prefetching
Write buffer

Multi-channel
Multi-rank
Multi-bank

Pipeline
Non-blocking
Prefetching
Write buffer

Input-Output (I/O)

Parallel File System

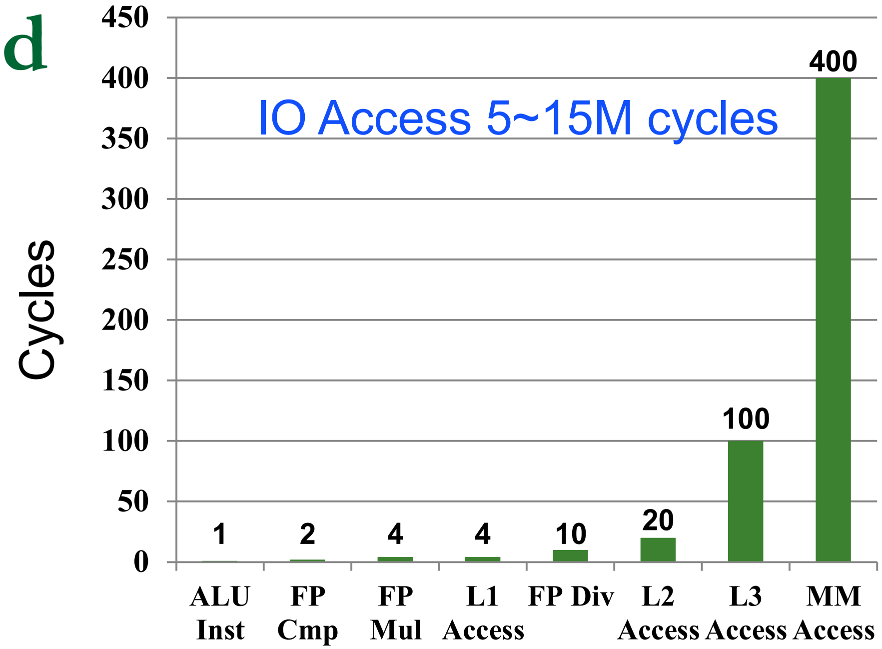
Disks

Assumption of Current Solutions

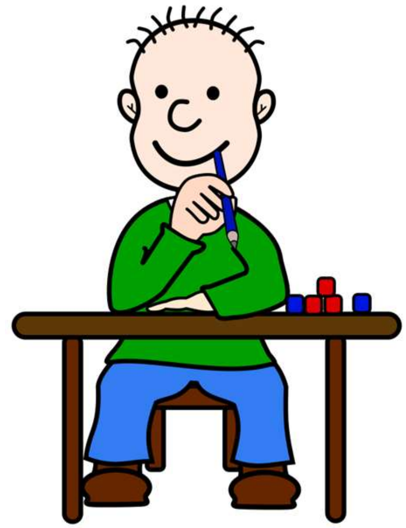
- ❑ Memory Hierarchy: Locality
- ❑ Concurrence: Data access pattern
  - Data stream

Extremely Unbalanced Operation Latency

Performances vary largely



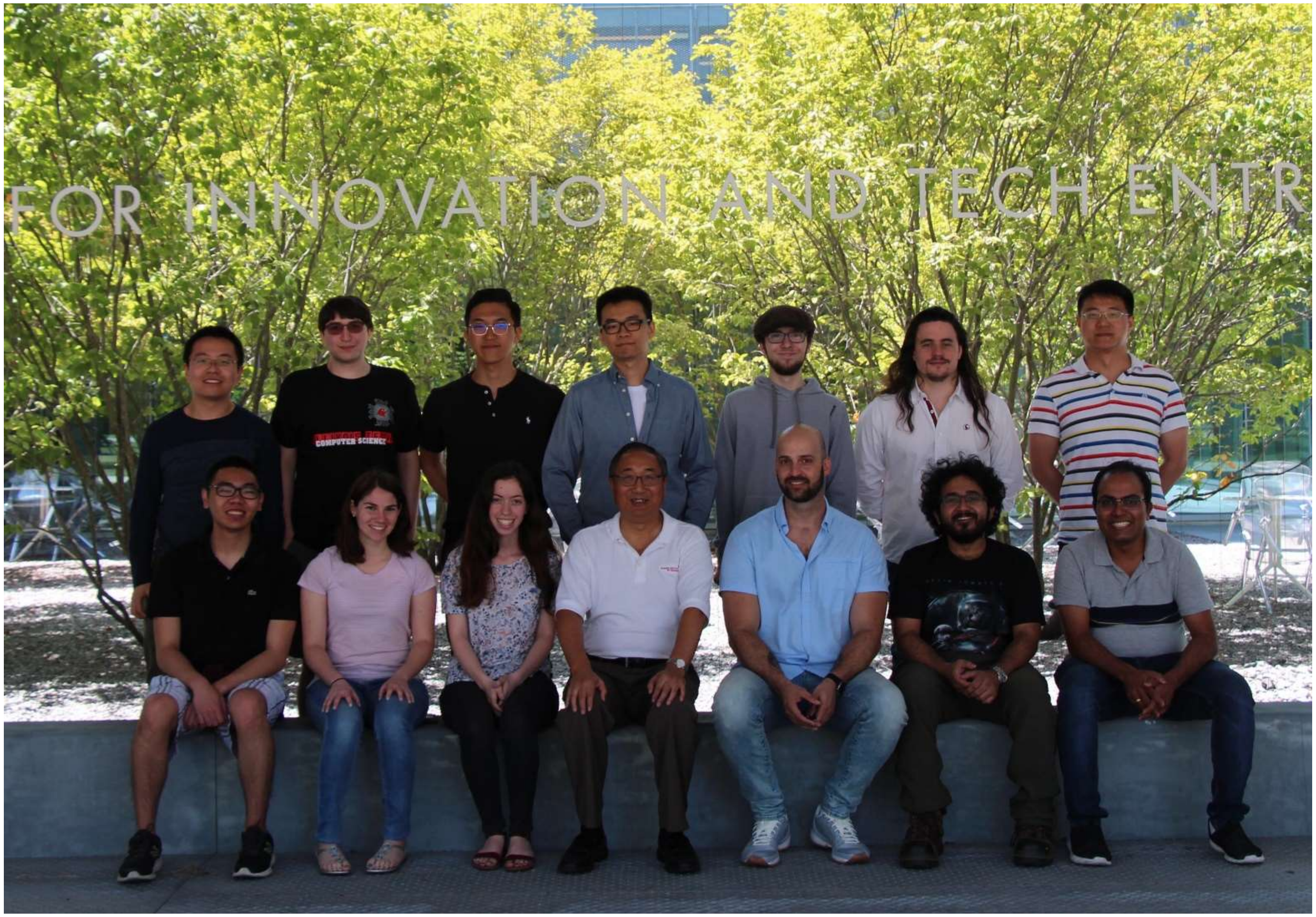
How do we further solve the memory-bound constraint or the memory-wall problem



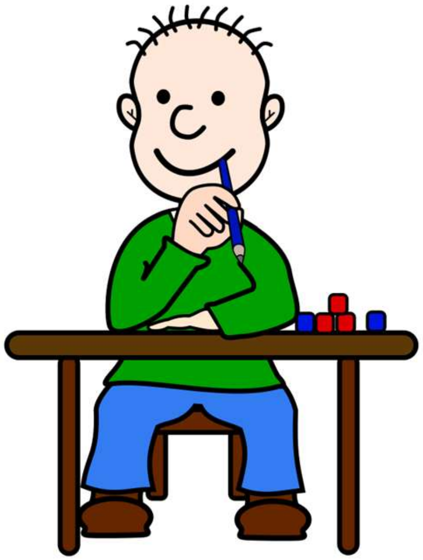
SEE YOU NEXT TIME

且听下回分解

Welcome to my Research Team



How can we produce classical research results?



SEE YOU NEXT TIME

且听下回分解