TEXAS TECH UNIVERSITY

CS4379: Parallel and Concurrent Programming
CS5379: Parallel Processing

Lecture 7

Dr. Yong Chen

Associate Professor

Computer Science Department

Texas Tech University

Course Info

- **Lecture Time**: TR, 12:30-1:50
- **Lecture Location**: ECE 217
- **Sessions**: CS4379-001, CS4379-002, CS5379-001, CS5379-D01
- **Instructor:** Yong Chen, Ph.D., Associate Professor
- **Email:** yong.chen@ttu.edu
- **Phone:** 806-834-0284
- **Office**: Engineering Center 315
- **Office Hours**: 2-4 p.m. on Wed., or by appointment
- **TA:** Mr. Ghazanfar Ali, Ghazanfar.Ali@ttu.edu
- **TA Office hours:** Tue. and Fri., 2-3 p.m., or by appointment
- **TA Office:** EC 201 A
- **More info:**
  - http://www.myweb.ttu.edu/yonchen
  - http://discl.cs.ttu.edu; http://cac.ttu.edu/; http://nsfcac.org

Announcements

- Two guest lectures by Mr. Misha Ahmadian next week on the subject of how to use parallel computers on campus
- If you have a laptop please consider bringing it with you and you can follow the lecture to try to use the systems

Outline

- Questions?
- Amdahl’s law revisited and scaled speedup
- Scalability of parallel systems
- Performance evaluation and analysis tools
- Quiz #2

Amdahl's Law

- Tacit assumption in Amdahl's law
  - The problem size or workload, W, is fixed
  - The speedup emphasizes time reduction

- Fixed-Size Speedup

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

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

Amdahl’s Law

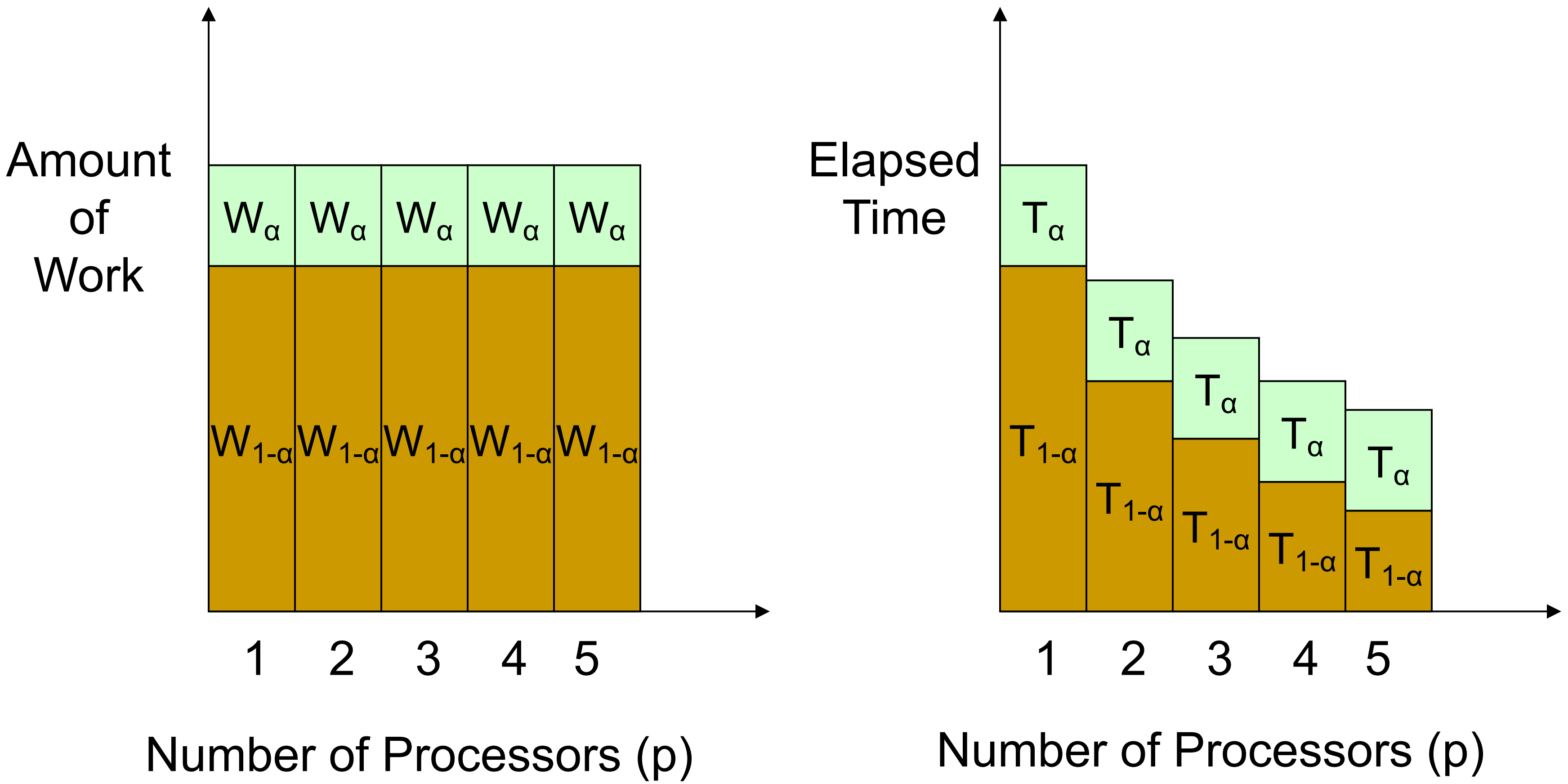
- Amdahl’s analysis argues a limit on speedup in terms of α

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

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

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

Fixed-Size Speedup



Impact of Amdahl's Law

IBM 7030 Stretch

IBM 7950 Harvest

All have up to 8/16/32 processors, citing Amdahl's law,

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

Cray X-MP
Fastest computer 1983-1985

Cray Y-MP

Scaled Speedup Model

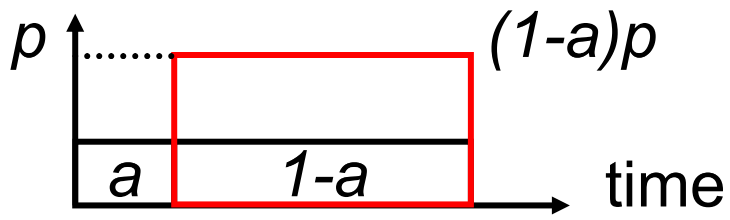
- Fixed-Time Speedup (Scaled Speedup) (Gustafson, 88)
  - Emphasis on work finished in a fixed time
  - Problem size is scaled from W to W'
  - W': Work finished within the fixed time with parallel processing

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

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

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

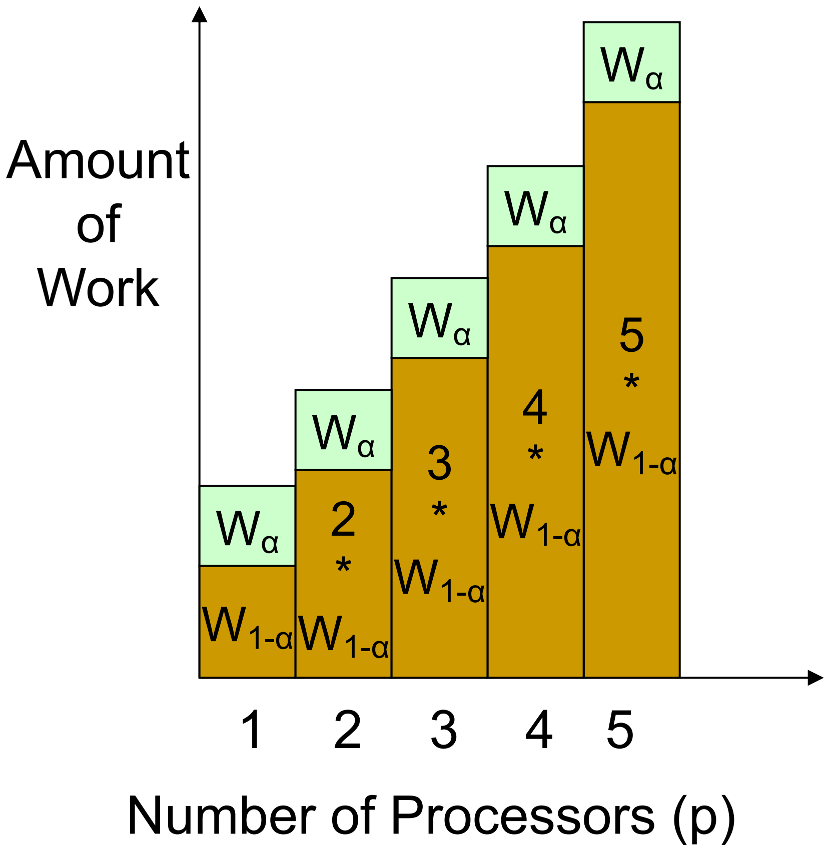
Gustafson's Law

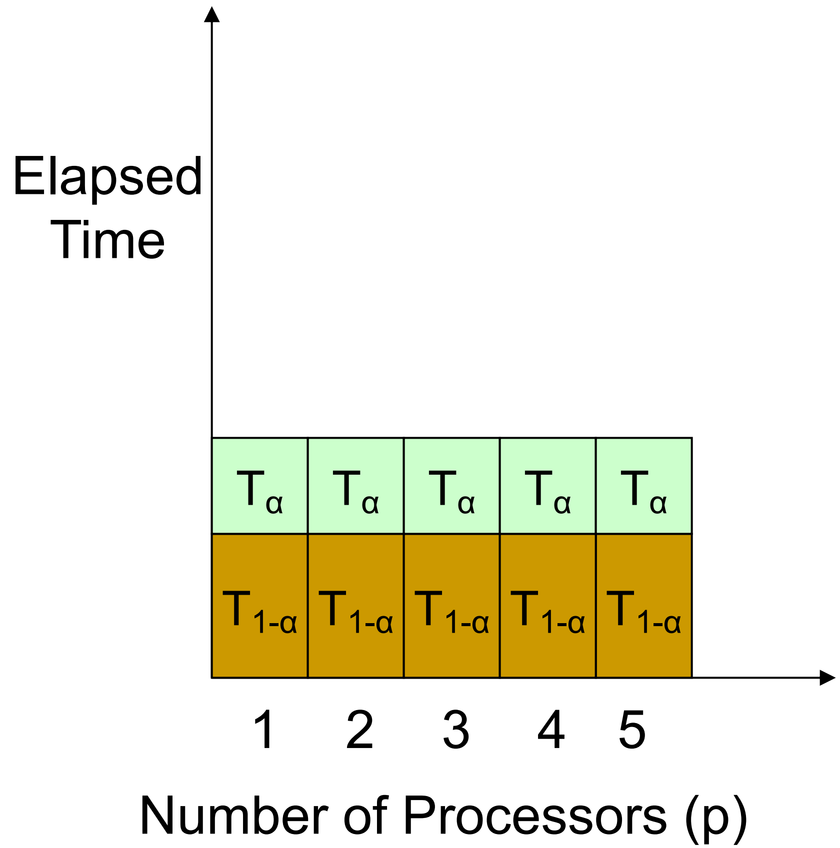


$$Speedup_{FT} = \frac{Work(p)}{Work(1)} = \frac{\alpha W + (1-\alpha)pW}{W} = \alpha + (1-\alpha)p$$

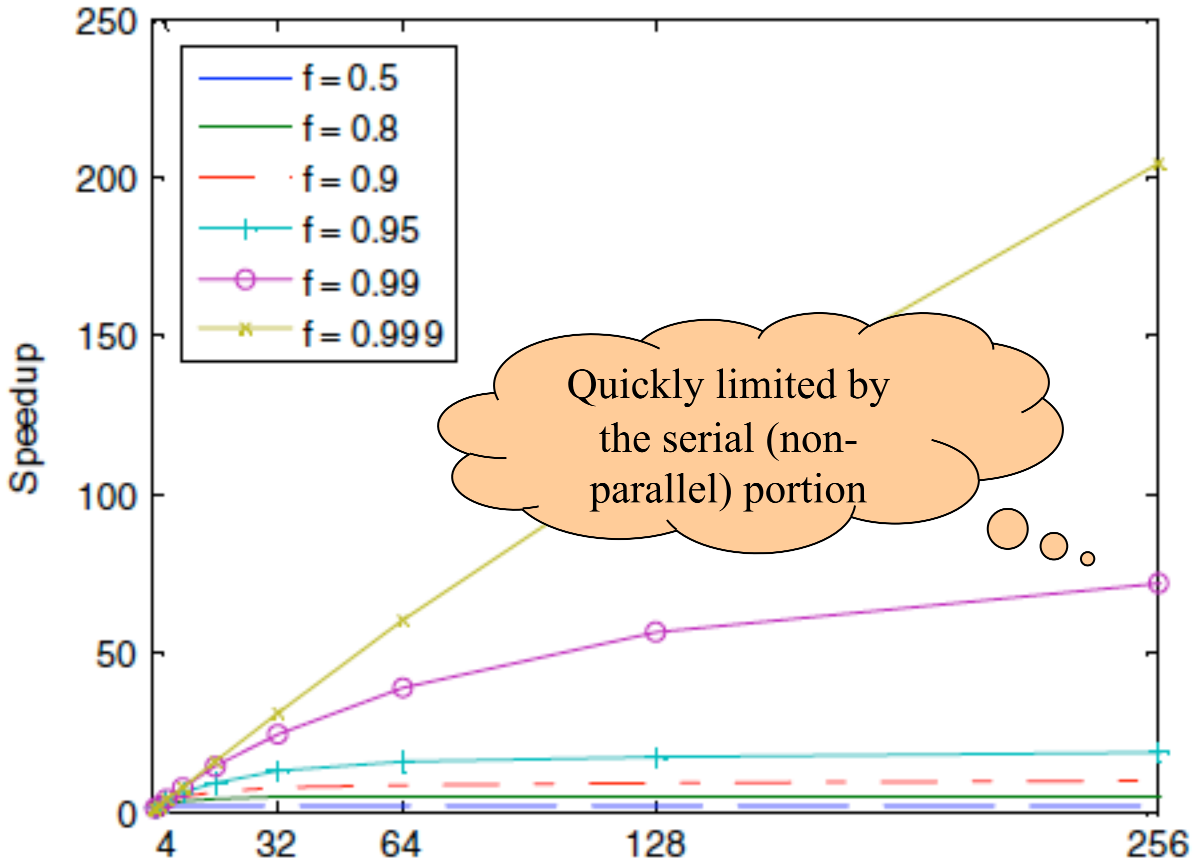
(FT stands for fixed-time)

Fixed-Time Speedup (Gustafson's Law)

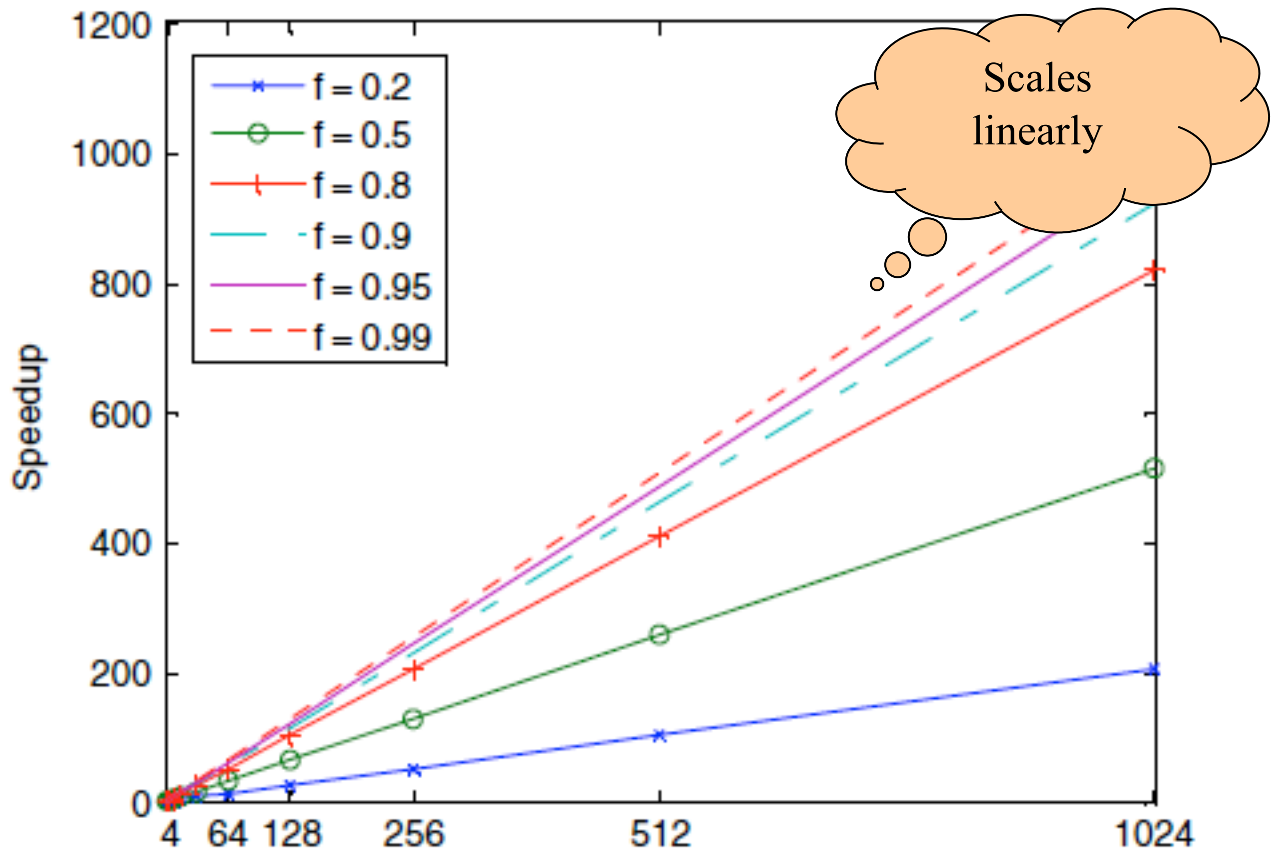




Fixed-size Speedup



Fixed-time Speedup



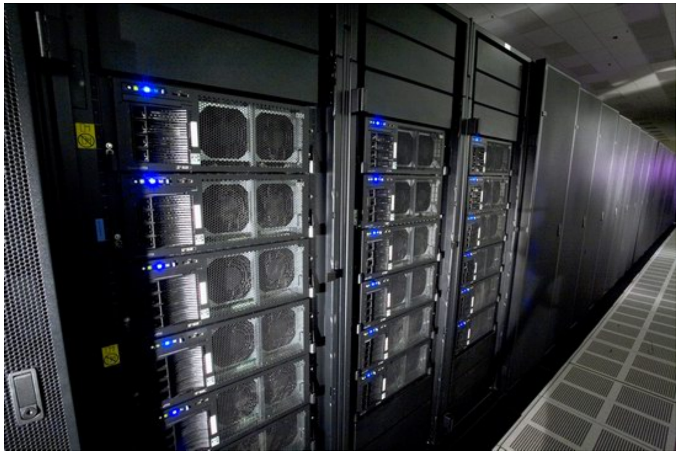
Terminologies

- Strong scaling: problem size fixed (i.e. what Amdahl's law focuses)
- Weak scaling: problem size scaled (i.e. what Gustafson's law focuses)
  - E.g. fixed-time speedup model
  - Other scaled speedup models also exist, e.g. memory-bounded scaled speedup model

Why Scaled Computing

- Solve larger problems
  - May not fit or can’t run for small machine
- Better solution, better accuracy
  - e.g. computing pi
- Maintain efficiency
- Provide real-time solution
  - May not be achievable with small machines

Scaled Computing Leads to Today's Systems



LANL Roadrunner:
25,200 processors, 2008
World's fastest supercomputer

TACC Ranger:
15,744 processors,
2008

ANL Intrepid:
20,480 processors, 2008

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

Note: data years ago

Outline

- Questions?

- Amdahl's law revisited and scaled speedup
- Scalability of parallel systems
- Performance evaluation and analysis tools

Scaling Characteristics of Parallel Programs

- The efficiency of a parallel program can be written as:

$$E = \frac{S}{p} = \frac{T_S}{pT_P}$$

or

$$E = \frac{1}{1 + \frac{T_o}{T_S}}.$$

$$T_o = p\,T_P - T_S$$

Scaling Characteristics of Parallel Programs

- For a given problem size (i.e., the value of T_S remains constant), as we increase p (*i.e.* the number of processing elements, or the system size), T_o increases
  - Problem size (workload) W: total number of operations required to solve a problem, e.g. $2N^3$ for matrix multiplication
  - i.e. the total overhead function T_o is an increasing function of p
- Thus the overall efficiency of the parallel program goes down

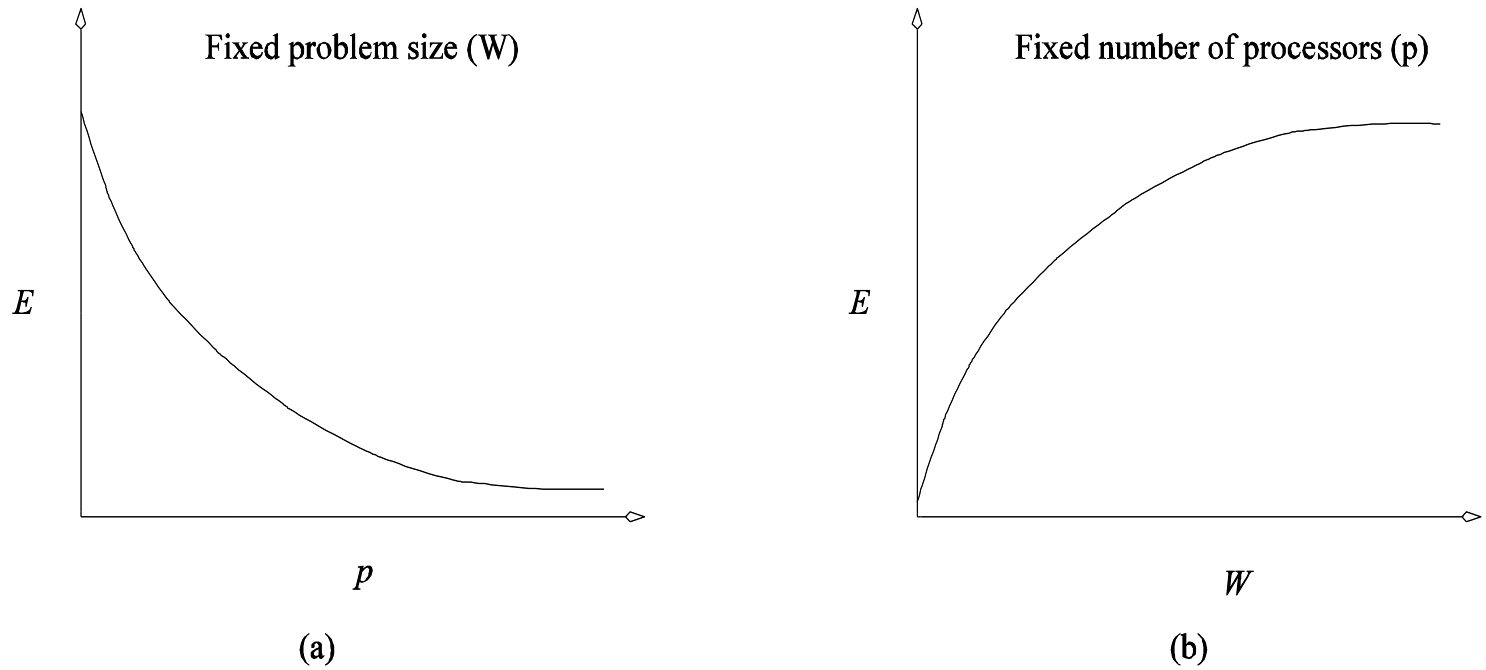
```
Matrix_multiplication(A, B, C)

for (i = 0; i<n; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < n; k++)
       C[i,j] = C[i, j] + A[i, k] * B[k, j]
```

Scaling Characteristics of Parallel Programs

- The total overhead function T_o is also a function of problem size W
- In many cases, T_o grows sub-linearly with respect to W
- In such cases, the efficiency increases if the problem size is increased, given the system size (the number of processing elements) remains constant

Variation of efficiency



Variation of efficiency: (a) as the number of processing elements is increased for a given problem size; and (b) as the problem size is increased for a given number of processing elements.

Scalable Parallel Systems

- We can simultaneously increase the problem size and the number of processors to keep efficiency constant.

- Scalable parallel systems: with the ability to maintain the efficiency at a fixed value by simultaneously increasing the number of processing elements (system size) and problem size

Isoefficiency Metric of Scalability

- What is the rate at which the problem size must increase with respect to the number of processing elements to keep the efficiency fixed?

- This rate determines the scalability of the system. The slower this rate, the better.

- Scalability of parallel systems: a measure of its capacity to increase speedup (in order to keep efficiency constant) in proportion to the number of processors given increasing the problem size
  - Isoefficiency scalability (see details from additional paper readings)

Scalability of Heterogeneous Computing

- What is a heterogeneous parallel system?

- Is the previously discussed scalability metric well applicable to heterogeneous computing?

- No, as system scaling cannot be well represented by *p* anymore
  - The efficiency definition is an issue
- System scaling: marked speed
- Contains homogeneous as a special case

Y. Chen, X.-H. Sun and M. Wu. Algorithm-System Scalability of Heterogeneous Computing. Journal of Parallel and Distributed Computing (JPDC), Vol. 68, No. 11, 1403 – 1412, 2008.

Outline

- Questions?
- Amdahl's law revisited and scaled speedup
- Scalability of parallel systems
- Performance evaluation and analysis tools
- Quiz #2

Performance Evaluation and Analysis

- Performance measurement
  - Define metrics (run time, speedup, scalability), adjust parameters / variables (controllable and non-controllable, e.g. noises, jitter, variation, etc.)
- Performance modeling, prediction
  - Build up a model, calculate/analyze the performance, like the π example we show earlier
  - A model can be used to predict the performance on a platform we don't have
- Performance diagnosis/optimization
  - Instrumentation/profiling
  - Post-execution, traces

Often combine these approaches too

Measurement

- Timing an entire program
  - UNIX *time* command outputs
    - User time
    - System time
    - Elapsed time
  - User time + system time = CPU time
  - Additional *time* output
    - Percent utilization
    - Average memory utilization
    - Blocked I/O operations
    - Page faults and swaps

Timing a Portion of a Program

- Record the time before you start doing x
- Do x
- Record the time at completion of x
- Subtract the start time from the completion time
- Often repeat x with *i* iterations (e.g. a million iterations), then divide the time by *i*

TEXAS TECH UNIVERSITY

Profiling

- Most compilers provide a facility to automatically insert timing calls into your program at the entry and exit of each routine at compile time
- A separate utility (e.g. *prof, gprof*) produces a report showing the percentage of time spent in each routine
- Many performance analysis tools also provide this capability

Types of Profiling

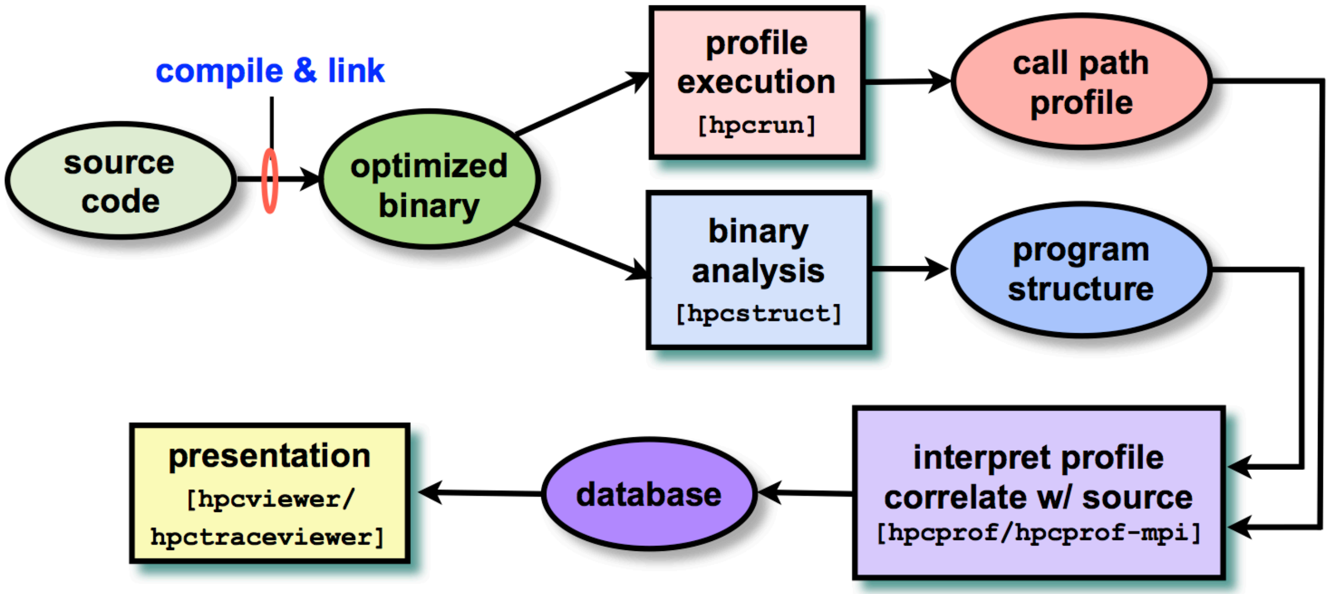
- Time-based
- Based on other metrics such as
  - Operation counts
  - Cache and memory event counts

PAPI

- http://icl.cs.utk.edu/papi/
- Provide support for accessing hardware performance counters available on most modern microprocessors

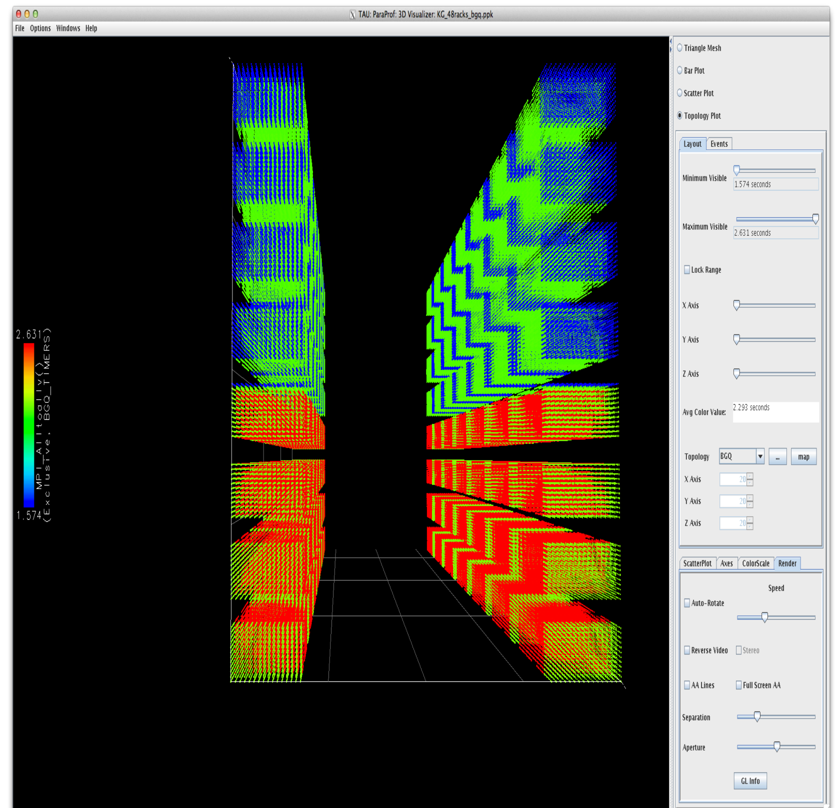
HPCToolkit

- http://hpctoolkit.org/
- An open-source suite of multi-platform tools for profile-based performance analysis of applications.



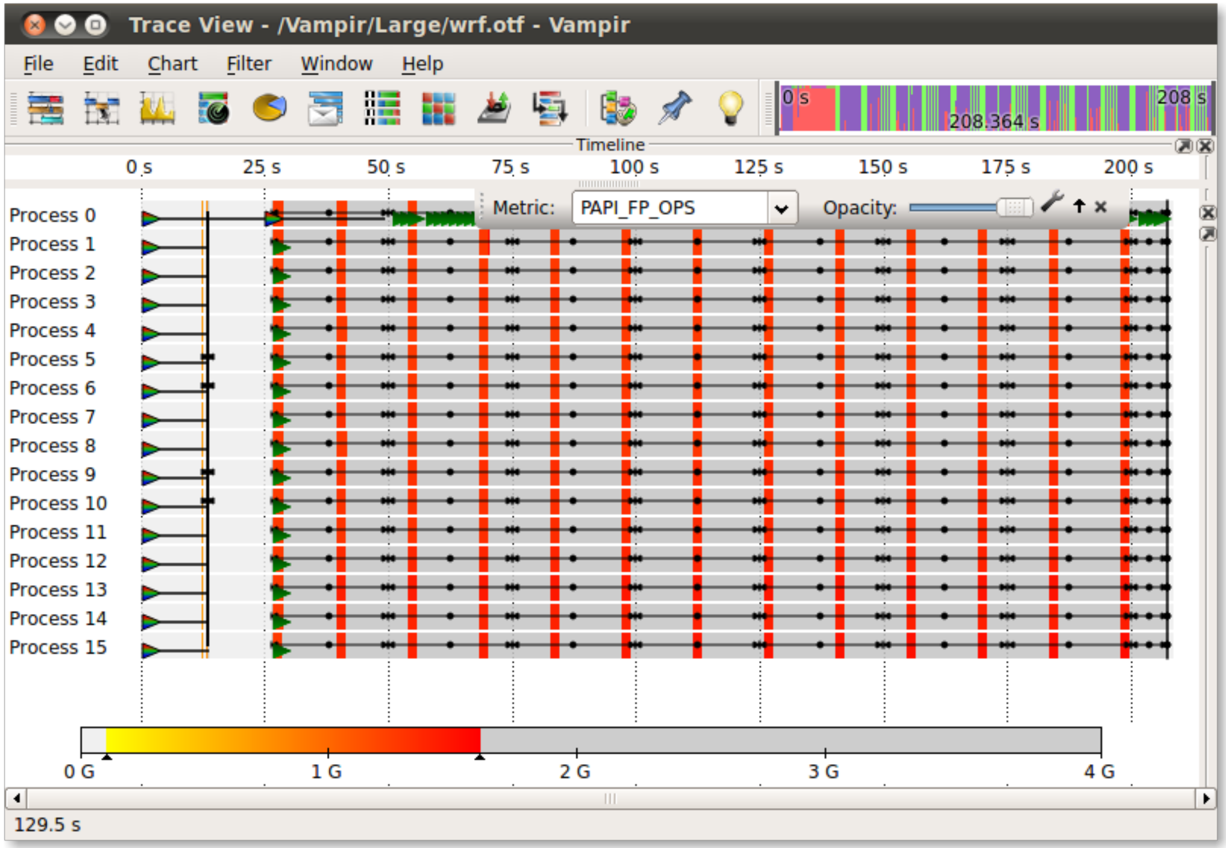
TAU

- http://www.cs.uoregon.edu/research/tau/home.php
- A portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, UPC, Java, Python



Vampir

- https://vampir.eu/
- A performance visualization and optimization tool



Performance Analysis Tools

- MPE logging.Jumpshot
- Pablo
- Paradyn
- Scalea
- VProf
- Prophesy
- DynaProf
- .....

Readings

- Reference book ITPC, Chapter 5
  - 5.1, 5.2, 5.3 (more examples), 5.4, 5.6, 5.7, 5.8
- Foster, DBPP, Chapter 3, 3.1-3.4
  - http://www.mcs.anl.gov/~itf/dbpp/text/book.html
- John L. Gustafson, "*Reevaluating Amdahl's Law*", 1988
- A. Y. Grama, A. Gupta, and V. Kumar. Isoefficiency : Measuring the Scalability of Parallel Algorithms and Architectures, 1993.
- Y. Chen, X.-H. Sun and M. Wu. Algorithm-System Scalability of Heterogeneous Computing. Journal of Parallel and Distributed Computing (JPDC), Vol. 68, No. 11, 2008. (Extended ICPP-2005)

Questions?

Questions/Suggestions/Comments are always welcome!

Write me: yong.chen@ttu.edu
Call me: 806-834-0284
See me: ENGCTR 315

If you write me an email for this class, please start the email subject with [CS4379] or [CS5379].