Lecture 3
Reasoning about Performance

- Performance
- Amdahl's Law
- Scaling
- Benchmarks

1

Why worry about Performance?

- Compare different systems
  - Select the most appropriate system for a given problem
- Make efficient use of available resources
- Scaling
  - Increase in resources should result in faster results
  - How does increase in resources effect overall runtime?
  - How does increase of problem size effect overall runtime?

2

Optimization Goals

- Execution time
  - Minimize the time between start and completion of a task
  - Typical goal in HPC
- Throughput
  - Maximize the number of tasks completed in a given time
  - Typical goal of large data centers (HTC)

3

Performance

4

Performance Definitions

$$Performance_x = \frac{1}{Execution\ time_x}$$

For two computers X and Y, if the performance of X is greater than the performance of Y, we have

$$Performance_x > Performance_y$$

$$\frac{1}{Execution\ time_x} > \frac{1}{Execution\ time_y}$$

$$Execution\ time_x > Execution\ time_y$$

5

Relative Performance

- Comparing different systems, we typically say: "Computer X is n times faster than Y":

$$\frac{Performance_x}{Performance_y} = \frac{Execution\ time_y}{Execution\ time_x} = n$$

- Computer A runs a program in 10 seconds, computer B in 15 seconds, hence A is 1.5 times faster than B

6

Measuring Performance

- Performance is measured in time units
- Different ways to measure time
  - Wall clock time or elapsed time
    - Time taken from start to end
    - Measures everything, including other tasks performed on multitasking systems
  - CPU time
    - Actual time the CPU spends computing for a specific task
    - Does not include time spent for other processes or I/O
    - CPU time < wall clock time
  - User CPU time
    - CPU time spent for user program
    - User CPU time < CPU time < wall clock time
  - System CPU time
    - CPU time spent on operating system tasks for user program
    - User/System CPU time difficult to measure

7

Factors of CPU performance

$$\text{CPU execution time} = \frac{\text{CPU clock cycles}}{\text{Clock rate}}$$

$$\text{CPU clock cycles} = \text{Instructions} \times \text{Average clock cycles per instruction (CPI)}$$

$$\text{CPU time} = \frac{\text{Instruction count} \times \text{CPI}}{\text{Clock rate}}$$

Components of performance	Units of measure
CPU execution time	Seconds for the program
Instruction count	Instructions executed for the program
Clock cycles per instruction (CPI)	Average number of clock cycles per instruction
Clock cycle time	Seconds per clock cycle

8

Other Performance Factors

- Memory subsystem
  - Cache misses
  - Amount and frequency of data to be moved
- I/O subsystem
  - Amount and frequency of data to be moved
- For parallel systems
  - Synchronization
  - Communication
  - Load balancing
- We will return to memory and I/O in lecture 6

9

Understanding Program Performance

Hardware or software component	Affects what?	How?
Algorithm	Instruction count, possibly CPI	The algorithm determines the number of source program instructions executed and hence the number of processor instructions executed. The algorithm may also affect the CPI, by favoring slower or faster instructions. For example, if the algorithm uses more floating-point operations, it will tend to have a higher CPI.
Programming language	Instruction count, CPI	The programming language certainly affects the instruction count, since statements in the language are translated to processor instructions, which determine instruction count. The language may also affect the CPI because of its features; for example a language with heavy support for data abstraction (e.g. Java) will require indirect calls, which will use higher CPI instructions.
Compiler	Instruction count, CPI	The efficiency of the compiler affects both the instruction count and average cycles per instruction, since the compiler determines the translation of the source language instructions into computer instructions. The compiler's role can be very complex and affect the CPI in complex ways.
Instruction set architecture	Instruction count, clock rate, CPI	The instruction set architecture affects all three aspects of CPU performance, since it affects the instructions needed for a function, the cost in cycles of each instruction, and the overall clock rate of the processor

10

Amdahl's Law

Amdahl's law

- Pitfall: Expecting the improvement of one aspect of a computer to increase overall performance by an amount proportional to the size of the improvement
- Gene Amdahl (1967):

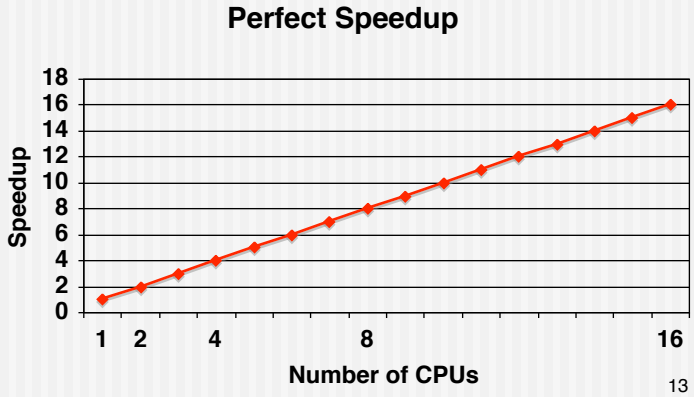
$$\text{Improved time} = \frac{\text{time effected by improvement}}{\text{Amount of improvement}} + \text{time unaffected}$$

- Example: Suppose a program runs for 100 seconds, with 80 seconds spent in multiply operations. Doubling the efficiency of multiply operations will result in new runtime of 60 seconds and thus a performance improvement of 1.67. How much do we need to improve multiply to achieve 5 times improvement?

Speedup

- Speedup (S) is defined as the improvement in execution time when increasing the amount of parallelism or sequential execution time (T_S) over parallel execution time (T_P)

$$S = \frac{T_S}{T_P}$$



13

Efficiency

- Speedup as percentage of number of processors

$$E = \frac{1}{P} \times \frac{T_S}{T_P} = \frac{1}{P} S$$

- A speedup of 90 with 100 processors yields 90% efficiency.

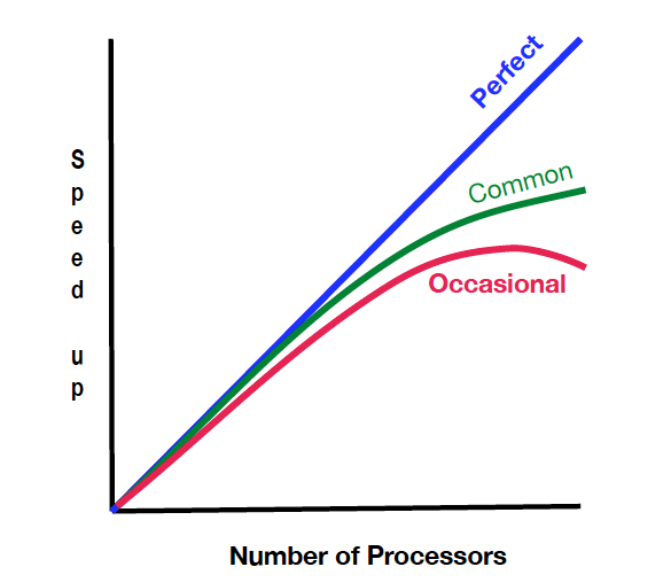
14

Superlinear Speedup

- Sometimes speedup is larger than number of processors
  - Very rare
- Main reasons:
  - Different parallel and sequential algorithms
  - Changes in memory behavior
    - Smaller problem size in parallel version fits main memory while sequential one doesn't
    - Changes in cache behavior

15

Typical Speedup Curves



16

Amdahl's Law and Parallel Processing

- According to Amdahl's law speedup is limited by the non-parallelizable fraction of a program
- Assume r_p is the parallelizable fraction of a program and r_s the sequential one. $r_p+r_s=1$
  We can compute the maximum theoretical speedup achievable on n processors with

$$S_{\max} = \frac{1}{r_s + \frac{r_p}{n}}$$

- If 20% of a program is sequential, the maximal achievable speedup is 5 for $n \Rightarrow \infty$

17

How to live with Amdahl's law

- Many real-world problems have significant parallel portions
- Yet, to use 100,000 cores with 90% efficiency, the sequential part needs to be limited to 0,00001%!
- Conclusion: minimize r_s and maximize r_p
- Increase amount of work done in the parallel (typically compute intensive) parts

18

Scaling Example

- Workload: sum of 10 scalars, and 10 × 10 matrix sum
  - Speed up from 10 to 100 processors
- Single processor: Time = (10 + 100) × t_{add}
- 10 processors
  - Time = 10 × t_{add} + 100/10 × t_{add} = 20 × t_{add}
  - Speedup = 110/20 = 5.5 (55% of potential)
- 100 processors
  - Time = 10 × t_{add} + 100/100 × t_{add} = 11 × t_{add}
  - Speedup = 110/11 = 10 (10% of potential)
- Assumes load can be balanced across processors

19

Scaling Example (cont)

- What if matrix size is 100 × 100?
- Single processor: Time = (10 + 10000) × t_{add}
- 10 processors
  - Time = 10 × t_{add} + 10000/10 × t_{add} = 1010 × t_{add}
  - Speedup = 10010/1010 = 9.9 (99% of potential)
- 100 processors
  - Time = 10 × t_{add} + 10000/100 × t_{add} = 110 × t_{add}
  - Speedup = 10010/110 = 91 (91% of potential)
- Assuming load balanced

20

Strong and Weak Scaling

- Strong scaling is the speedup achieved **without increasing** the size of the problem
- Weak scaling is the speedup achieved while **increasing** the size of the problem proportionally to the increase in number of processors

21

Weak Scaling Example

- 10 processors, 10 × 10 matrix
  - Time = $10 \times t_{add} + 100/10 \times t_{add} = 20 \times t_{add}$
- 100 processors, 32 × 32 matrix
  - Time = $10 \times t_{add} + 1000/100 \times t_{add} = 20 \times t_{add}$
- Constant performance in this example

22

Load Balancing

- Good speedup can only be achieved if the parallel workload is relatively equally spread over the available processors
- If workload is unevenly spread, overall performance is bound to the slowest processor (i.e. Processor with most workload)

23

Example Continued

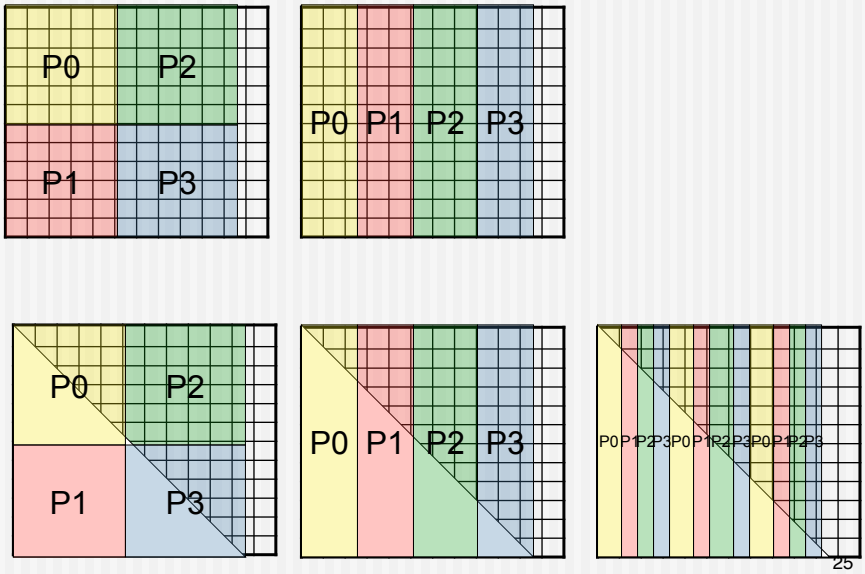
- 100 processors
  - Time = 10 × t_{add} + 10000/100 × t_{add} = 110 × t_{add}
  - Speedup = 10010/110 = 91 (91% of potential)
  - Assumes each processor gets 1% of workload
- Assume one processor get 2% (i.e. 200 matrix elements) and the rest of 9800 elements is equally distributed over the remaining 99 processors.

$$\text{Time} = \text{Max}\left(\frac{9800t}{99}, \frac{200t}{1}\right) + 10t = 210t$$

- Speedup = 10010/210 = 47.6

24

Load Balancing Examples



25

Synchronization and Communication

- Parallel programs need synchronization and communication to ensure correct program behavior
- Synchronization and communication adds significant overhead and thus reduces parallel efficiency

$$S = \frac{T_S}{T_P}$$ can be refined as

$$S = \frac{T_S}{T_{PC} + \text{synch wait time} + \text{communication time}}$$

with T_{PC} symbolizing the net parallel computation time

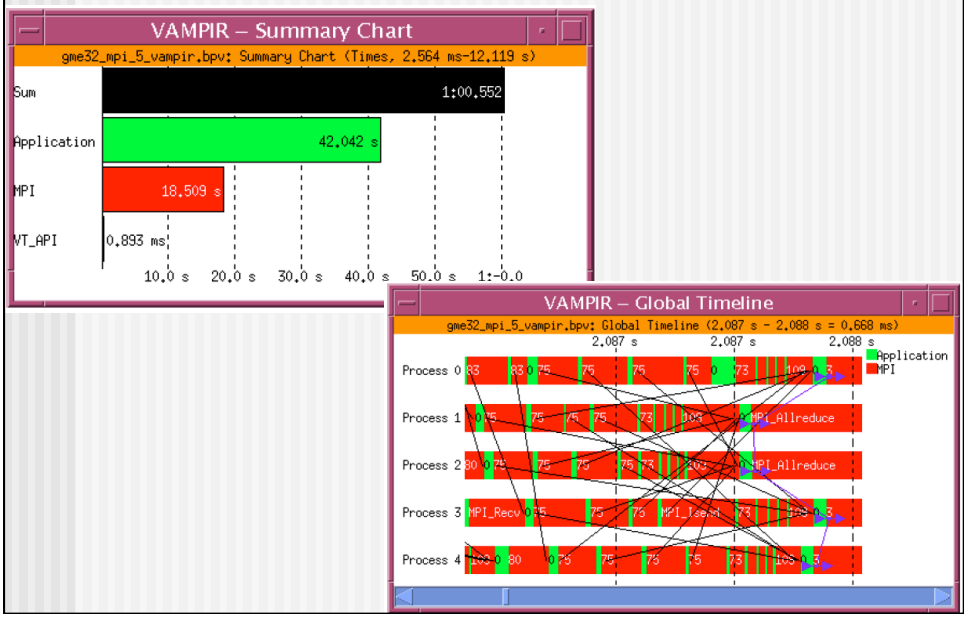
26

Synchronization and Communication Cont' d

- Goal is to avoid synchronization and communication
- Not always possible
- Overlap communication with computation and optimize communication
  - Communication overhead impacted by latency and bandwidth
    - Block communication
  - Use more efficient communication patterns
- Profiling tools can help identifying synchronization and communication overhead

27

Example: Vampir traces



The Impact of Data

- Apart from communication, data is effecting performance at many levels
- Memory hierarchy
- I/O
- Will be discussed in depth later in the course

29

Parallel Benchmarks

30

Compare Different Parallel Machines

- How to compare different parallel machines?
- Benchmarking
  - Run a specific program on different machines, collect performance information and compare results
- For specific problems the actual program could be run to find the best architecture
  - In many cases not practical (availability of machines, different usage pattern, etc.)
- A set of "standardized" benchmarks were developed over time for certain performance characteristics

31

SPEC CPU Benchmark

- To compare individual CPUs, the SPEC benchmarks are typically used
  - SPEC - Standard Performance Evaluation Corporation (http://www.spec.org)
  - Started with CPU benchmarks but now include also parallel performance, file system, etc.
  - 12 integer and (CINT2006) 17 floating-point (CFP2006) benchmarks; typically the geometric mean of normalized results are reported

32

SPEC Example

All SPEC CPU2006 Results Published by SPEC

These results have been submitted to SPEC; see the disclaimer before studying any results.

Search published CPU2006 results

Last update: *Thursday, 7 January 2010, 10:49*

| CINT2006 | CFP2006 | CINT2006 Rates | CFP2006 Rates |

CINT2006 (1481):

Test Sponsor	System Name	Auto Parallel	Processor				Results	
			Enabled Cores	Enabled Chips	Cores/ Chip	Threads/ Core	Base	Peak
ASUSTeK Computer Inc.	ASUS DSEB-DG Server Motherboard (Intel XEON X5482) HTML | CSV | Text | PDF | PS | Config	Yes	8	2	4	1	25.3	28.8
ASUSTeK Computer Inc.	ASUS DSEB-DG server motherboard (Intel Xeon X5492) HTML | CSV | Text | PDF | PS | Config	Yes	8	2	4	1	27.6	31.4
ASUSTeK Computer Inc.	ASUS P6T WS PRO workstation motherboard (Intel Core i7-965 Extreme Edition) HTML | CSV | Text | PDF | PS | Config	Yes	4	1	4	2	31.5	35.2
ASUSTeK Computer Inc.	ASUS P6T6 WS REVOLUTION workstation motherboard (Intel Core i7-965 Extreme Edition) HTML | CSV | Text | PDF | PS | Config	Yes	4	1	4	2	32.1	35.6
ASUSTeK Computer Inc.	ASUS Z8PE-D12X server motherboard (Intel Xeon X5570) HTML | CSV | Text | PDF | PS | Config	Yes	8	2	4	2	32.0	36.0
ASUSTeK Computer Inc.	ASUS RS700-E6 server system (Intel Xeon X5570) HTML | CSV | Text | PDF | PS | Config	Yes	8	2	4	2	32.1	36.1
ASUSTeK Computer Inc.	ASUS Z8NA-D6 server motherboard (Intel Xeon X5570) HTML | CSV | Text | PDF | PS | Config	Yes	8	2	4	2	32.0	36.0

3

Typical Parallel Benchmarks

Benchmark	Scaling?	Reprogram?	Description
Linpack	Weak	Yes	Dense matrix linear algebra
SPECrate	Weak	No	Independent job parallelism
Stanford Parallel Applications for Shared Memory SPLASH 2	Strong (although offers two problem sizes)	No	Complex 1D FFT Blocked LU Decomposition Blocked Sparse Cholesky Factorization Ocean Simulation Ray Tracer … Integer Radix Sort Adaptive Fast Multipole Barnes-Hut Hierarchical Radiosity Volume Renderer
NAS Parallel Benchmarks	Weak	Yes (C or Fortran only)	EP: embarrassingly parallel MG: simplified multigrid CG: unstructured grid for a conjugate gradient method FT: 3-D partial differential equation solution using FFTs IS: large integer sort
PARSEC Benchmark Suite	Weak	No	Blackscholes: option pricing with Black-Scholes PDE Bodytrack: Body tracking of a person Canneal: Simulated cache-aware annealing to optimize routing …
Berkeley Design Patterns	Strong or Weak	Yes	Finite-State Machine Graph Traversal Dense Matrix Spectral Methods (FFT) N-Body … Combinational Logic Structured Grid Sparse Matrix Dynamic Programming MapReduce

34

Problems of Standardized Benchmark

- Problem implementation is typically fixed
- Innovation limited to architecture and compiler
- But we need innovative data structures and algorithms to scale up to peta- and exaflop/s performance
- Berkeley Design Patterns try to overcome this problem by keeping definition on high level
  - E.g. Sparse matrix solver is welcome to use any data structure, algorithm, and programming language

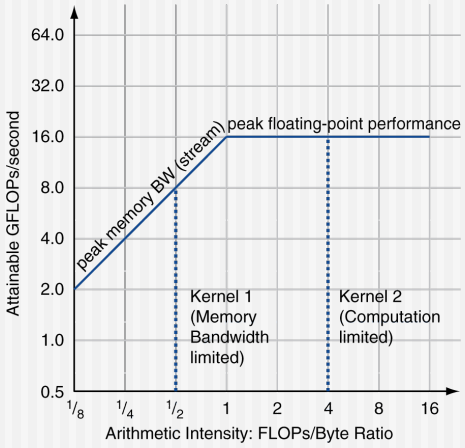
35

The Impact of Data

- We would like a model that combines compute (flop/s) and data (memory bandwidth/latency) characteristics
- Arithmetic intensity is the ration of floating-point operations to the number of data bytes access by a program from main memory
- For a given computer we need
  - Peak flop/s (from spec)
  - Peak memory bytes/sec (from stream benchmark)

36

Roofline Model

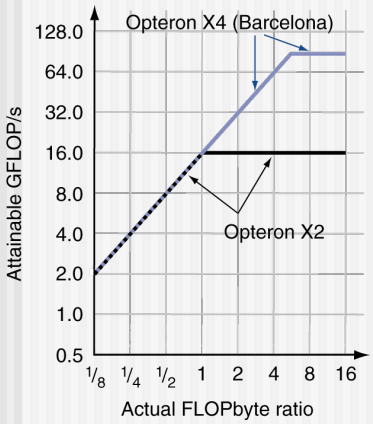


Attainable GFLOP/sec
= Max (Peak Memory BW × Arithmetic Intensity, Peak FP Performance)

37

Comparing Systems

- Example: Opteron X2 vs. Opteron X4
  - 2-core vs. 4-core, 2× FP performance/core, 2.2GHz vs. 2.3GHz
  - Same memory system

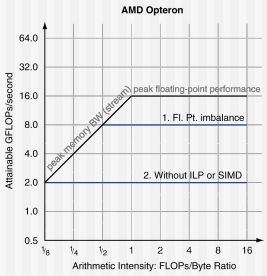


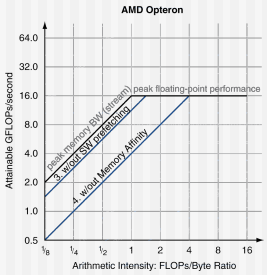
- To get higher performance on X4 than X2
  - Need high arithmetic intensity
  - Or working set must fit in X4's 2MB L-3 cache

38

Optimizing Performance

- Optimize FP performance
  - Balance adds & multiplies
  - Improve superscalar ILP and use of SIMD instructions
- Optimize memory usage
  - Software prefetch
    - Avoid load stalls
  - Memory affinity
    - Avoid non-local data accesses

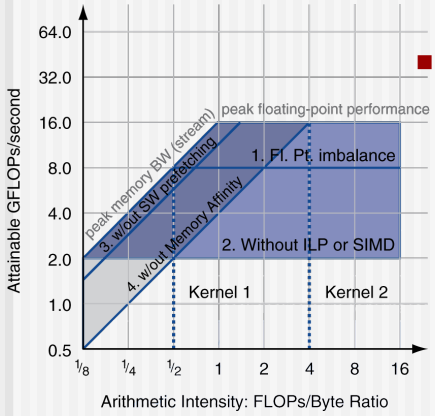




39

Optimizing Performance

- Choice of optimization depends on arithmetic intensity of code



- Arithmetic intensity is not always fixed
  - May scale with problem size
  - Caching reduces memory accesses
    - Increases arithmetic intensity

40

Summary and Outlook

- How to measure performance
- Performance impacts and characteristics
- Benchmarking and comparing systems

- We have now the basic terminology, system and performance characteristics to have a deeper look into ways to achieve parallelism in Lecture 4 and 5:
  - Instruction level parallelism
  - Task parallelism
  - Data parallelism

41

Further Readings

- **Computer Organization and Design - The Hardware/Software Interface,** 4th Edition, David A. Patterson and John L. Hennessy
  - Chapter 1.4 (Performance)
  - Chapter 7.1-2; 7.9-7.11

42

Exercises

9. Consider 3 different processors P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table:

Processor	Clock rate	CPI
P1	2 GHz	1.5
P2	1.5 GHz	1.0
P3	3 GHz	2.5

   1. Which processor has the highest performance?
   2. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions
   3. We are trying to reduce the time by 30% but this leads to an increase of 20% in CPI. What clock rate should we have to get this time reduction?

43

Exercises Cont'd

10. Consider a computer running programs with CPU times shown in the following table

    (total time broken down in floating-point, integer, load/store, and branch instructions)

	FP instr.	INT instr.	L/S instr.	Branch instr.	Total Time
a.	35s	85s	50s	30s	200s
b.	50s	80s	50s	30s	210s

   1. By how much is the total time reduced if the time for FP operations is reduced by 20%
   2. By how much is the time for INT operations reduced if the total time is reduced by 20%
   3. Can the total time be reduced by 20% by reducing only the time for branch instructions?

44

Exercises Cont'd

11. Consider a parallel program with the following computing and communication times on different numbers of processors:

# Proc.	Compute Time (ms)	Communication Time (ms)
2	176	11
4	96	13
8	49	17
16	30	22
32	14	23
64	6.5	26

    1. For each doubling of the number of processors, determine the ratio of new to old computing time and the ratio of new to old communication time.
    2. Using the geometric means of the ratios, extrapolate to find the computing time and routing time in a 128-processor system
    3. Find the computing time and routing time for a system with one processor

45

Exercises Cont'd

12. A shared memory system is used to process database transactions. Assume a two CPU/core system can process 5000 requests/sec.
    1. How many requests/sec can a 4,8,16 CPU/core system ideally process?
    2. Why is this ideal rate rarely achievable in practice?

46