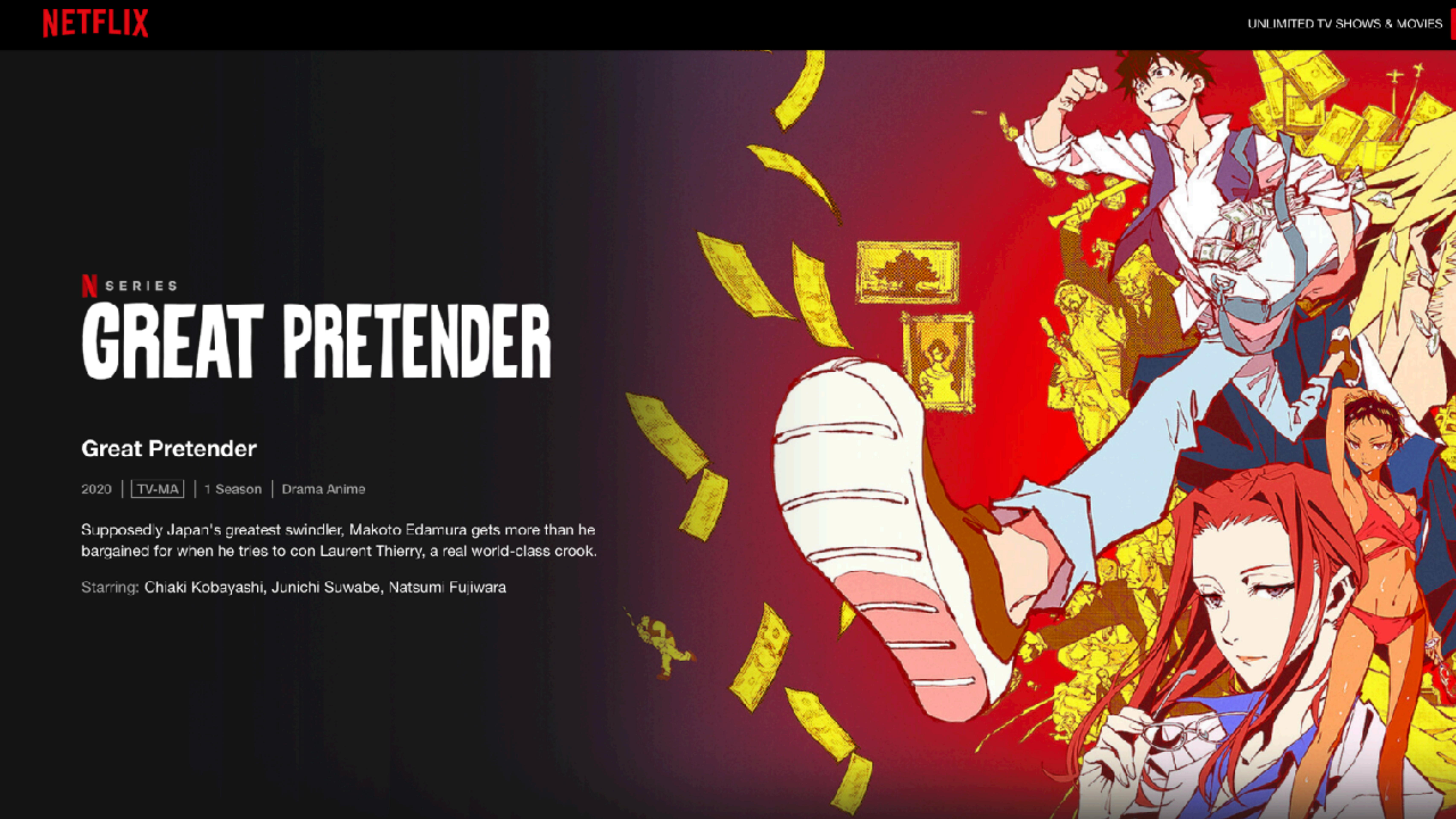
Performance (III): Amdahl’s Law (cont.) and Other Performance Metrics

Hung-Wei Tseng



NETFLIX

UNLIMITED TV SHOWS & MOVIES

N SERIES

GREAT PRETENDER

Great Pretender

2020 | TV-MA | 1 Season | Drama Anime

Supposedly Japan's greatest swindler, Makoto Edamura gets more than he bargained for when he tries to con Laurent Thierry, a real world-class crook.

Starring: Chiaki Kobayashi, Junichi Suwabe, Natsumi Fujiwara

Recap: Summary of CPU Performance Equation

$$Performance = \frac{1}{Execution\ Time}$$

$$Execution\ Time = \frac{Instructions}{Program} \times \frac{Cycles}{Instruction} \times \frac{Seconds}{Cycle}$$

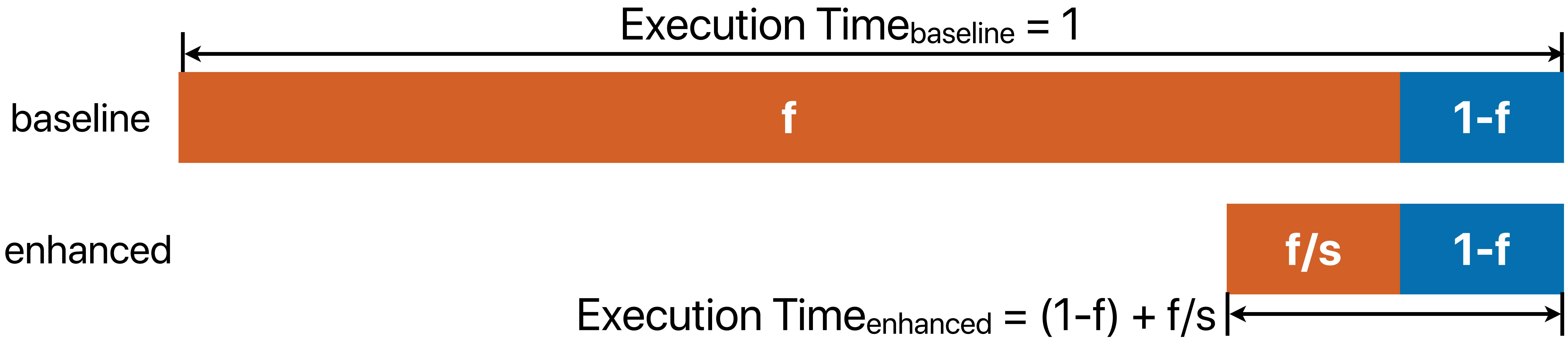
$$ET = IC \times CPI \times CT$$

$$Speedup = \frac{Execution\ Time_X}{Execution\ Time_Y}$$

- IC (Instruction Count)
  - ISA, Compiler, algorithm, programming language, **programmer**
- CPI (Cycles Per Instruction)
  - Machine Implementation, microarchitecture, compiler, application, algorithm, programming language, **programmer**
- Cycle Time (Seconds Per Cycle)
  - Process Technology, microarchitecture, **programmer**

Recap: Amdahl's Law

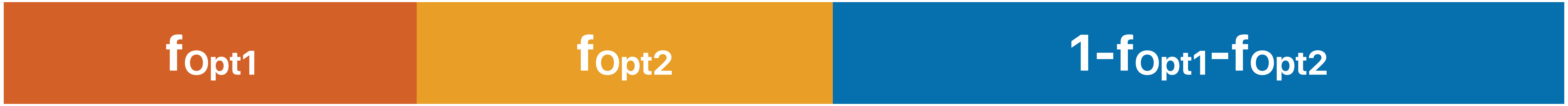
$$Speedup_{enhanced}(f, s) = \frac{1}{(1-f) + \frac{f}{s}}$$


Execution Time$_{baseline}$ = 1
baseline
f
1-f
enhanced
f/s
1-f
Execution Time$_{enhanced}$ = (1-f) + f/s

$$Speedup_{enhanced} = \frac{Execution\ Time_{baseline}}{Execution\ Time_{enhanced}} = \frac{1}{(1-f) + \frac{f}{s}}$$

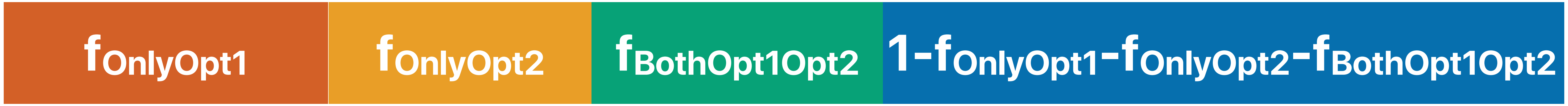
Recap: Amdahl's Law on Multiple Optimizations

- We can apply Amdahl's law for multiple optimizations
- These optimizations must be dis-joint!
  - If optimization #1 and optimization #2 are dis-joint:



$$Speedup_{enhanced}(f_{Opt1}, f_{Opt2}, s_{Opt1}, s_{Opt2}) = \frac{1}{(1-f_{Opt1}-f_{Opt2}) + \frac{f\_Opt1}{s\_Opt1} + \frac{f\_Opt2}{s\_Opt2}}$$

  - If optimization #1 and optimization #2 are not dis-joint:



$$Speedup_{enhanced}(f_{OnlyOpt1}, f_{OnlyOpt2}, f_{BothOpt1Opt2}, s_{OnlyOpt1}, s_{OnlyOpt2}, s_{BothOpt1Opt2})$$

$$= \frac{1}{(1-f_{OnlyOpt1}-f_{OnlyOpt2}-f_{BothOpt1Opt2}) + + \frac{f\_BothOpt1Opt2}{s\_BothOpt1Opt2} + \frac{f\_OnlyOpt1}{s\_OnlyOpt1} + \frac{f\_OnlyOpt2}{s\_OnlyOpt2}}$$

Outline

- Amdahl’s law and its implications
- Other performance metrics

https://www.pollev.com/hungweitseng close in 1:30

Practicing Amdahl's Law (2)

- Final Fantasy XV spends lots of time loading a map — within which period that 95% of the time on the accessing the H.D.D., the rest in the operating system, file system and the I/O protocol. If we replace the H.D.D. with a flash drive, which provides 100x faster access time and a better processor to accelerate the software overhead by 2x. By how much can we speed up the map loading process?
    A. ~7x
    B. ~10x
    C. ~17x
    D. ~29x
    E. ~100x

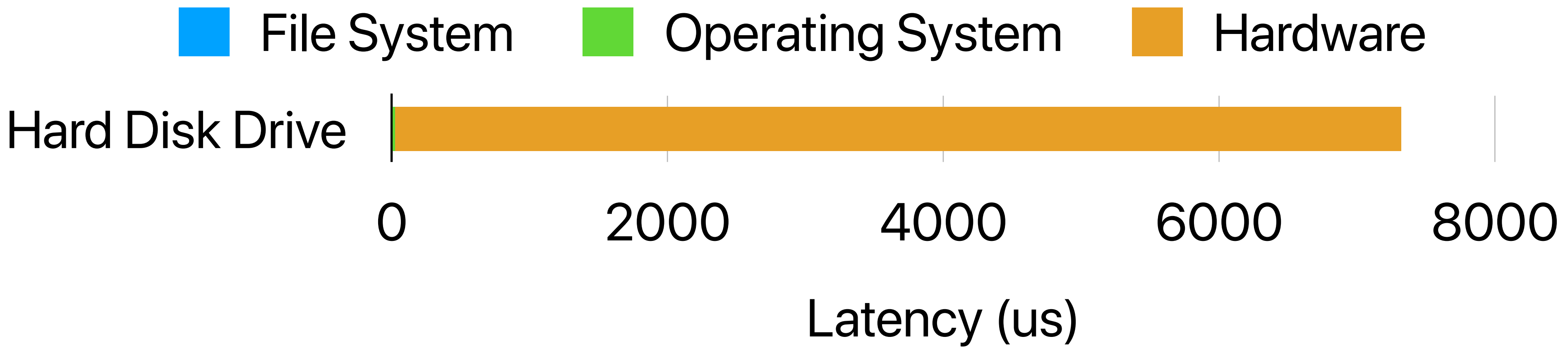
FINAL FANTASY XV WINDOWS EDITION

FINAL FANTASY XV WINDOWS EDITION > General Discussions > Topic Details

~Simulacrum Sakura 寂~ Mar 12, 2018 @ 7:10pm

Load times

I run this game from an 7200 RPM hardrive and load times are pretty long... do anyone run this game form an SSD? are load times good?



Practicing Amdahl’s Law (2)

- Final Fantasy XV spends lots of time loading a map — within which period that 95% of the time on the accessing the H.D.D., the rest in the operating system, file system and the I/O protocol. If we replace the H.D.D. with a flash drive, which provides 100x faster access time and a better processor to accelerate the software overhead by 2x. By how much can we speed up the map loading process?

FINAL FANTASY XV WINDOWS EDITION

FINAL FANTASY XV WINDOWS EDITION > General Discussions > Topic Details

~Simulacrum Sakura 寂~ Mar 12, 2018 @ 7:10pm

Load times

I run this game from an 7200 RPM hardrive and load times are pretty long... do anyone run this game form an SSD? are load times good?

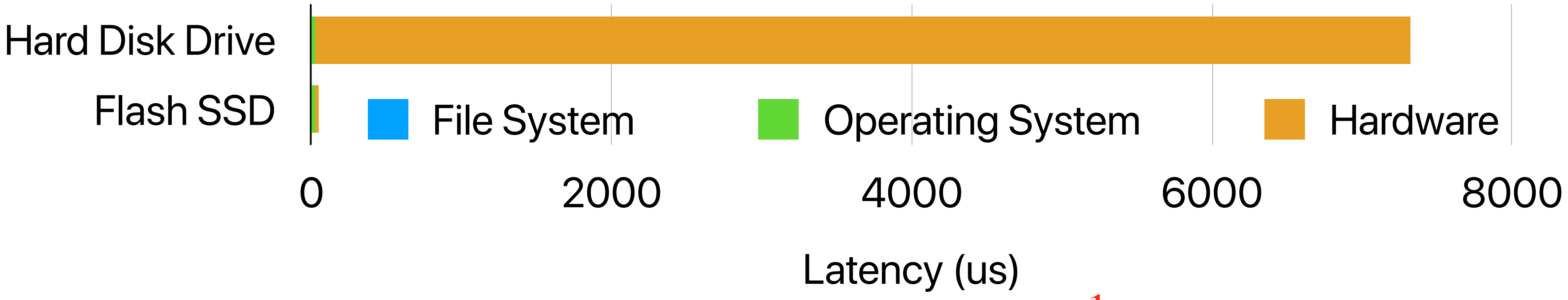
A. ~7x

B. ~10x

C. ~17x

[✓ D. ~29x]

E. ~100x



$$Speedup_{enhanced}(95\%,5\%,100,2) = \frac{1}{(1-95\%-5\%)+\frac{95\%}{100}+\frac{5\%}{2}} = 28.98\times$$

https://www.pollev.com/hungweitseng close in 1:30

Speedup further!

- With the latest flash memory technologies, the system spends 16% of time on accessing the flash, and the software overhead is now 84%. If we want to adopt a new memory technology to replace flash to achieve 2x speedup on loading maps, how much faster the new technology needs to be?
  A. ~5x
  B. ~10x
  C. ~20x
  D. ~100x
  E. None of the above

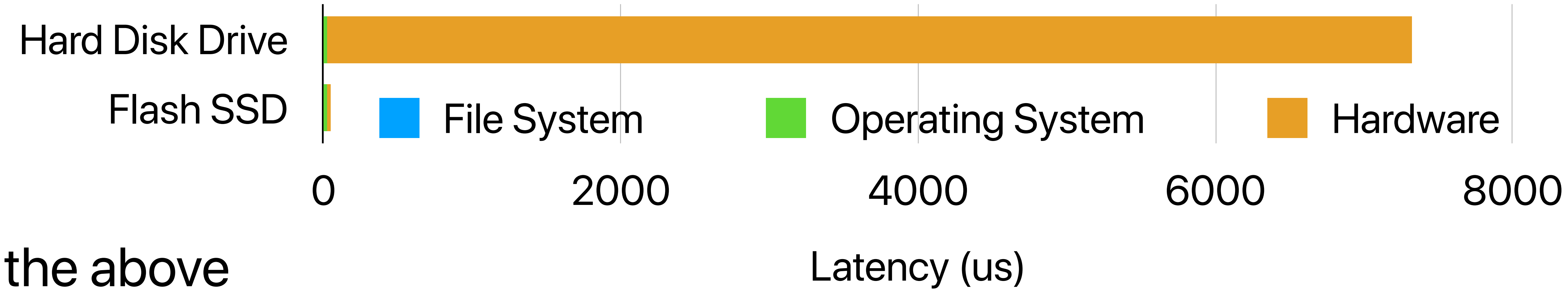
FINAL FANTASY XV WINDOWS EDITION

FINAL FANTASY XV WINDOWS EDITION > General Discussions > Topic Details

~Simulacrum Sakura 叔~ Mar 12, 2018 @ 7:10pm

Load times

I run this game from an 7200 RPM hardrive and load times are pretty long... do anyone run this game form an SSD? are load times good?



Speedup further!

- With the latest flash memory technologies, the system spends 16% of time on accessing the flash, and the software overhead is now 84%. If we want to adopt a new memory technology to replace flash to achieve 2x speedup on loading maps, how much faster the new technology needs to be?

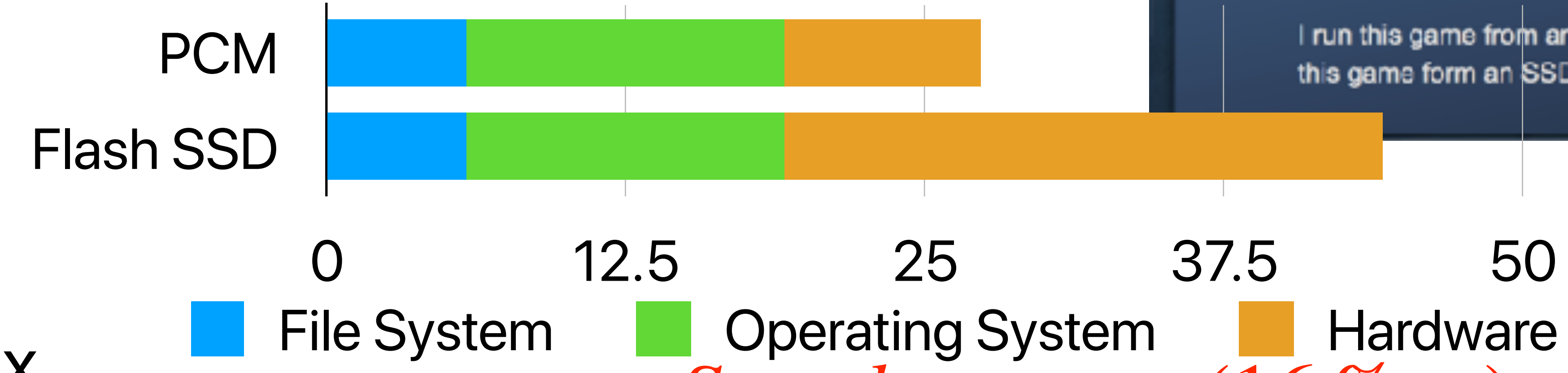
A. ~5x

B. ~10x

C. ~20x

D. ~100x

E. None of the above



FINAL FANTASY XV WINDOWS EDITION

FINAL FANTASY XV WINDOWS EDITION > General Discussions > Topic Details

~Simulacrum Sakura 寂~ Mar 12, 2018 @ 7:10pm

Load times

I run this game from an 7200 RPM hardrive and load times are pretty long... do anyone run this game form an SSD? are load times good?

$$Speedup_{enhanced}(16\%, x) = \frac{1}{(1-16\%) + \frac{16\%}{x}} = 2$$

$$x = 0.47$$

Does this make sense?

Amdahl's Law Corollary #1

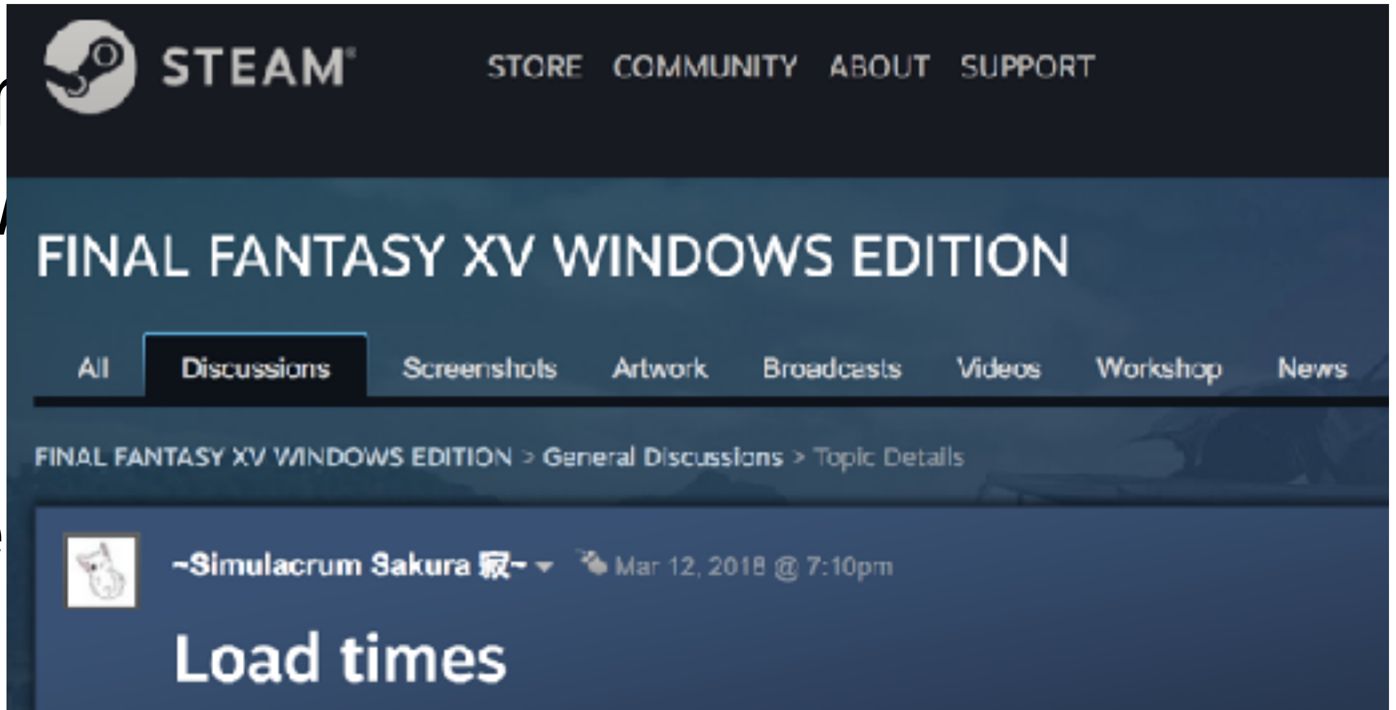
- The maximum speedup is bounded by

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f) + \frac{f}{\infty}}$$

$$Speedup_{max}(f, \infty) = \frac{1}{(1-f)}$$

Speedup further!

- With the latest flash memory technologies, the system spends 16% of time on accessi the flash, and the software overhead is now 84%. If we want to adopt a new memory technology to replace flash to achieve 2x speedup on loading maps, how much faste the new technology needs to be?



FINAL FANTASY XV WINDOWS EDITION

FINAL FANTASY XV WINDOWS EDITION > General Discussions > Topic Details

~Simulacrum Sakura 寂~ Mar 12, 2018 @ 7:10pm

Load times

I run this game from an 7200 RPM hardrive and load times are pretty long... do anyone run this game form an SSD? are load times good?

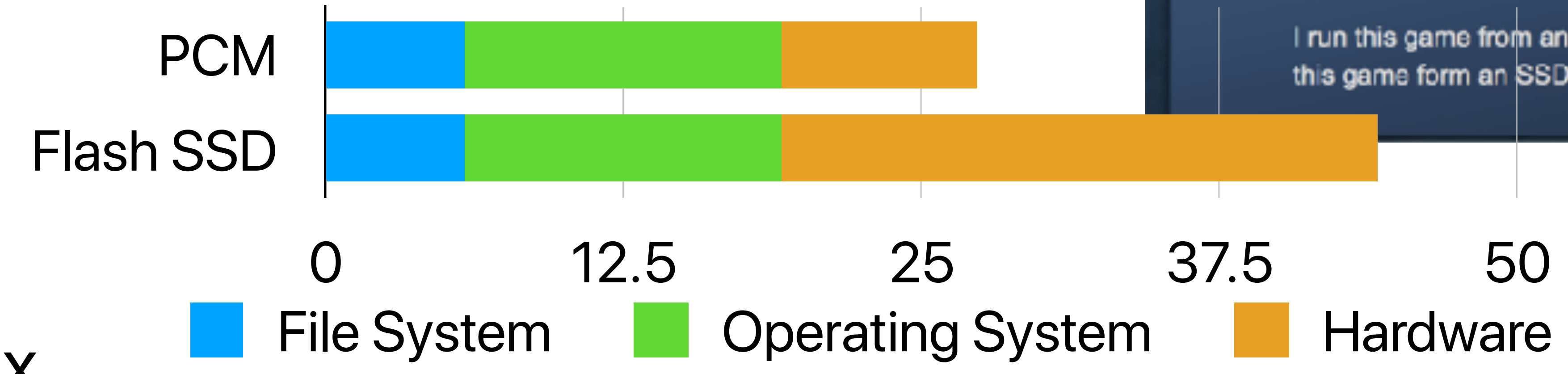
A. ~5x

B. ~10x

C. ~20x

D. ~100x

E. None of the above

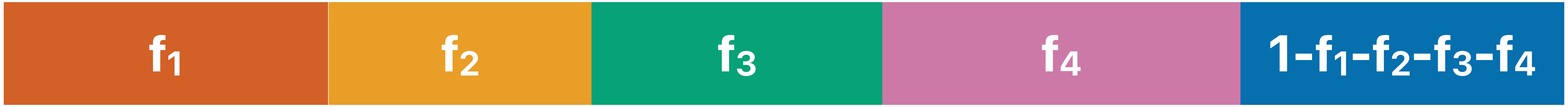


$$Speedup_{max}(16\,\%\,,\infty) = \frac{1}{(1-16\%)} = 1.19$$

2x is not possible

Corollary #1 on Multiple Optimizations

- If we can pick just one thing to work on/optimize



$$Speedup_{max}(f_1, \infty) = \frac{1}{(1-f_1)}$$

$$Speedup_{max}(f_2, \infty) = \frac{1}{(1-f_2)}$$

$$Speedup_{max}(f_3, \infty) = \frac{1}{(1-f_3)}$$

$$Speedup_{max}(f_4, \infty) = \frac{1}{(1-f_4)}$$

The biggest f_x would lead to the largest $Speedup_{max}$!

Corollary #2 — make the common case fast!

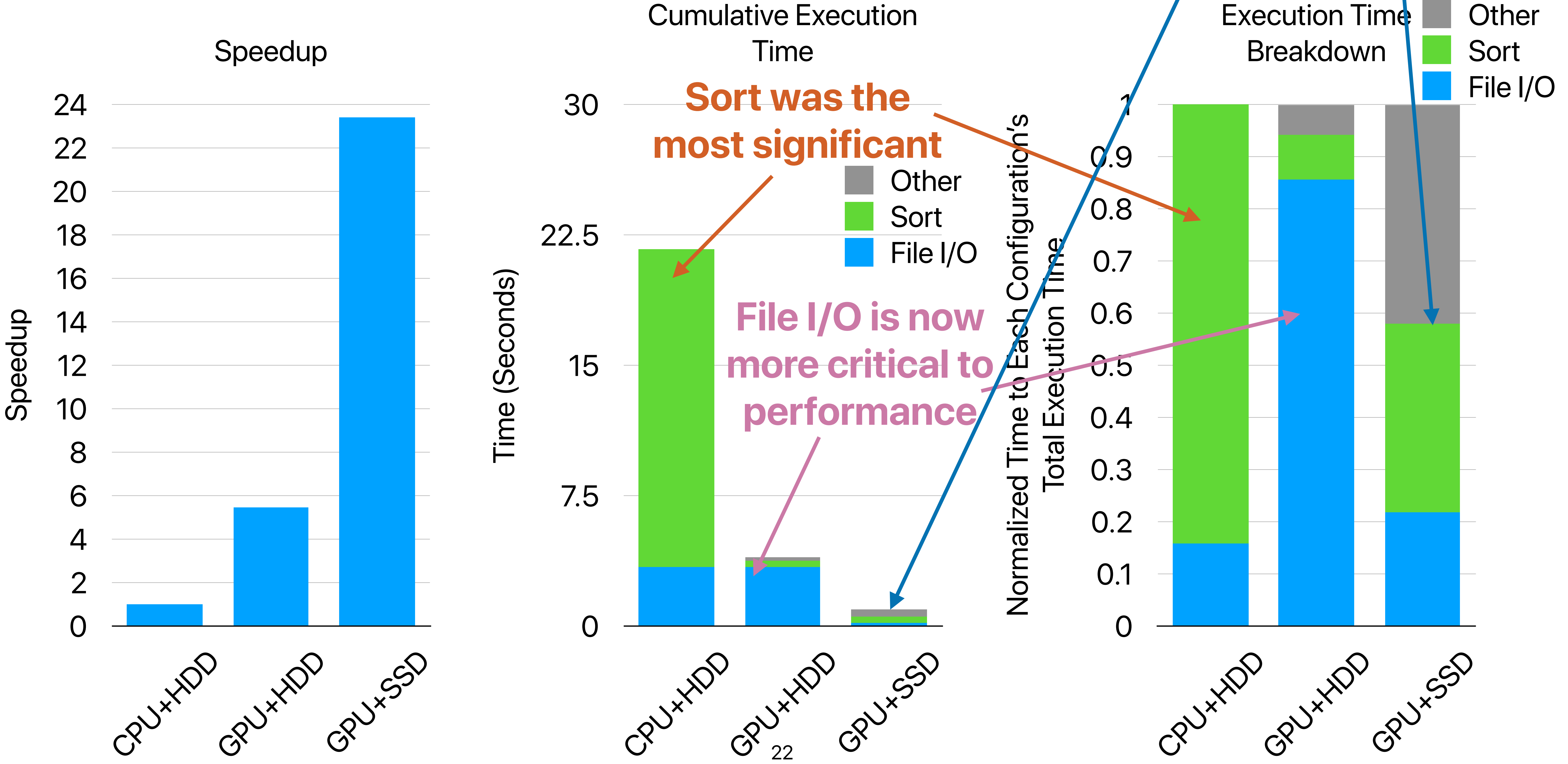
- When f is small, optimizations will have little effect.
- Common == **most time consuming** not necessarily the most frequent
- The uncommon case doesn’t make much difference
- The common case can change based on inputs, compiler options, optimizations you’ve applied, etc.

Identify the most time consuming part

- Compile your program with -pg flag
- Run the program
  - It will generate a gmon.out
  - gprof your_program gmon.out > your_program.prof
- It will give you the profiled result in your_program.prof

Demo — sort

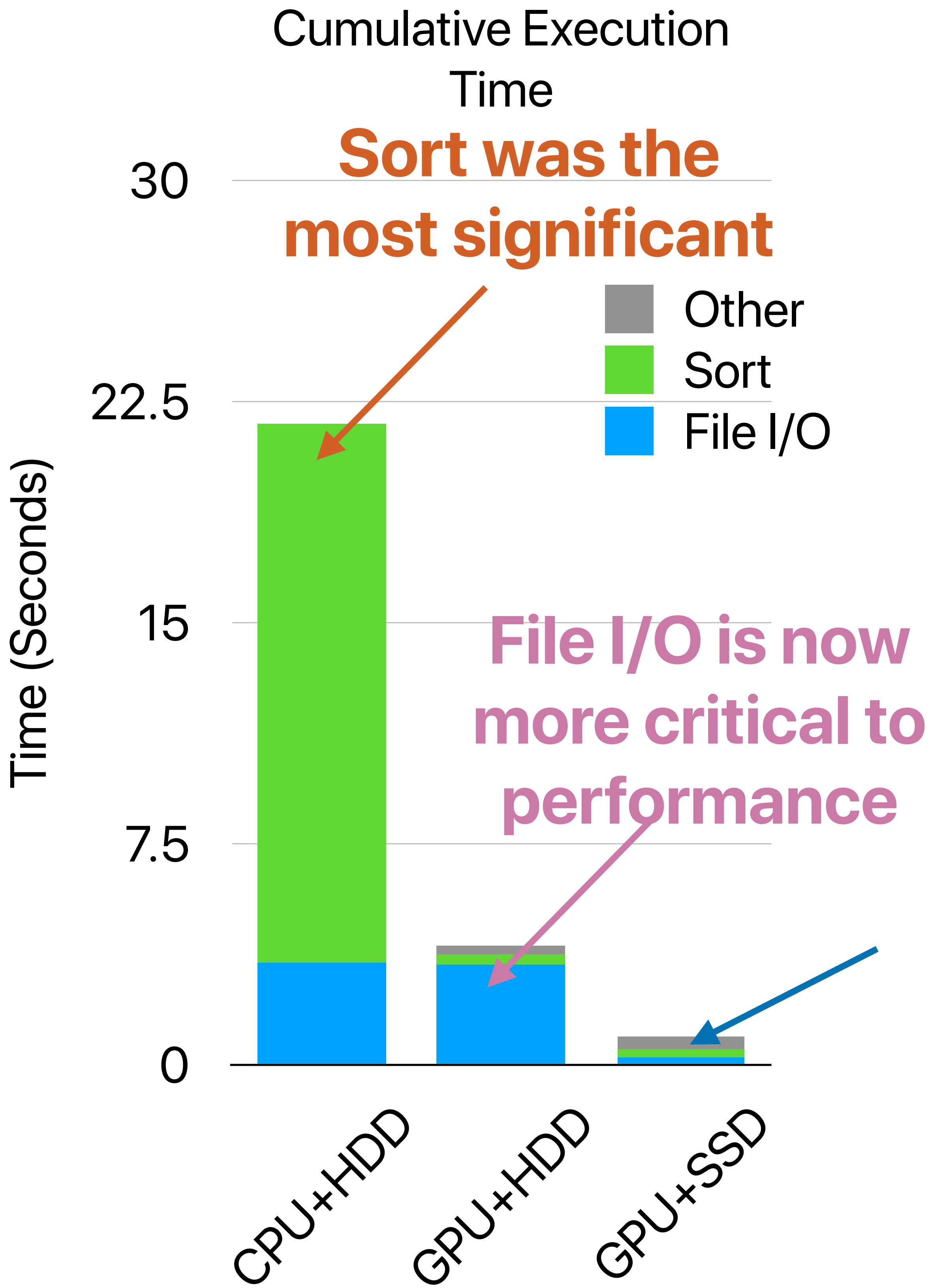
Something else (e.g., data movement) matters more



Sort was the most significant

File I/O is now more critical to performance

If we repeatedly optimizing our design based on Amdahl's law...



Sort was the most significant

File I/O is now more critical to performance

Something else (e.g., data movement) matters more now

- With optimization, the common becomes uncommon.
- An uncommon case will (hopefully) become the new common case.
- Now you have a new target for optimization — You have to revisit "Amdahl's Law" every time you applied some optimization

Don’t hurt non-common part too mach

- If the program spend 90% in A, 10% in B. Assume that an optimization can accelerate A by 9x, by hurts B by 10x…
- Assume the original execution time is T. The new execution time

$$ET_{new} = \frac{ET_{old} \times 90\,\%}{9} + ET_{old} \times 10\% \times 10$$

$$ET_{new} = 1.1 \times ET_{old}$$

$$Speedup = \frac{ET_{old}}{ET_{new}} = \frac{ET_{old}}{1.1 \times ET_{old}} = 0.91 \times$$ ……slowdown!

You may not use Amdahl’s Law for this case as Amdahl’s Law does NOT
(1) consider overhead
(2) bound to slowdown

Amdahl's Law on Multicore Architectures

- Symmetric multicore processor with n cores (if we assume the processor performance scales perfectly)

$$Speedup_{parallel}(f_{parallelizable}, n) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable}{n}}$$

https://www.pollev.com/hungweitseng close in 1:30

Amdahl’s Law on Multicore Architectures

- Regarding Amdahl’s Law on multicore architectures, how many of the following statements is/are correct?

  ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded

  ② With unlimited amount of parallel hardware units, single-core performance does not matter anymore

  ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts

  ④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Amdahl's Law on Multicore Architectures

- Regarding Amdahl's Law on multicore architectures, how many of the following statements is/are correct?

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable \times Speedup(<1)}{\infty}}$$

✓ ① If we have unlimited parallelism, the performance of each parallel piece does not matter as long as the performance slowdown in each piece is bounded

② With unlimited amount of parallel hardware units, single-core performance does not matter anymore

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

speedup is determined by 1-f

✓ ③ With unlimited amount of parallel hardware units, the maximum speedup will be bounded by the fraction of parallel parts

④ With unlimited amount of parallel hardware units, the effect of scheduling and data exchange overhead is minor

A. 0

B. 1

[✓ C. 2]

D. 3

E. 4

Demo — merge sort v.s. bitonic sort on GPUs

Merge Sort

$O(nlog_2n)$

Bitonic Sort

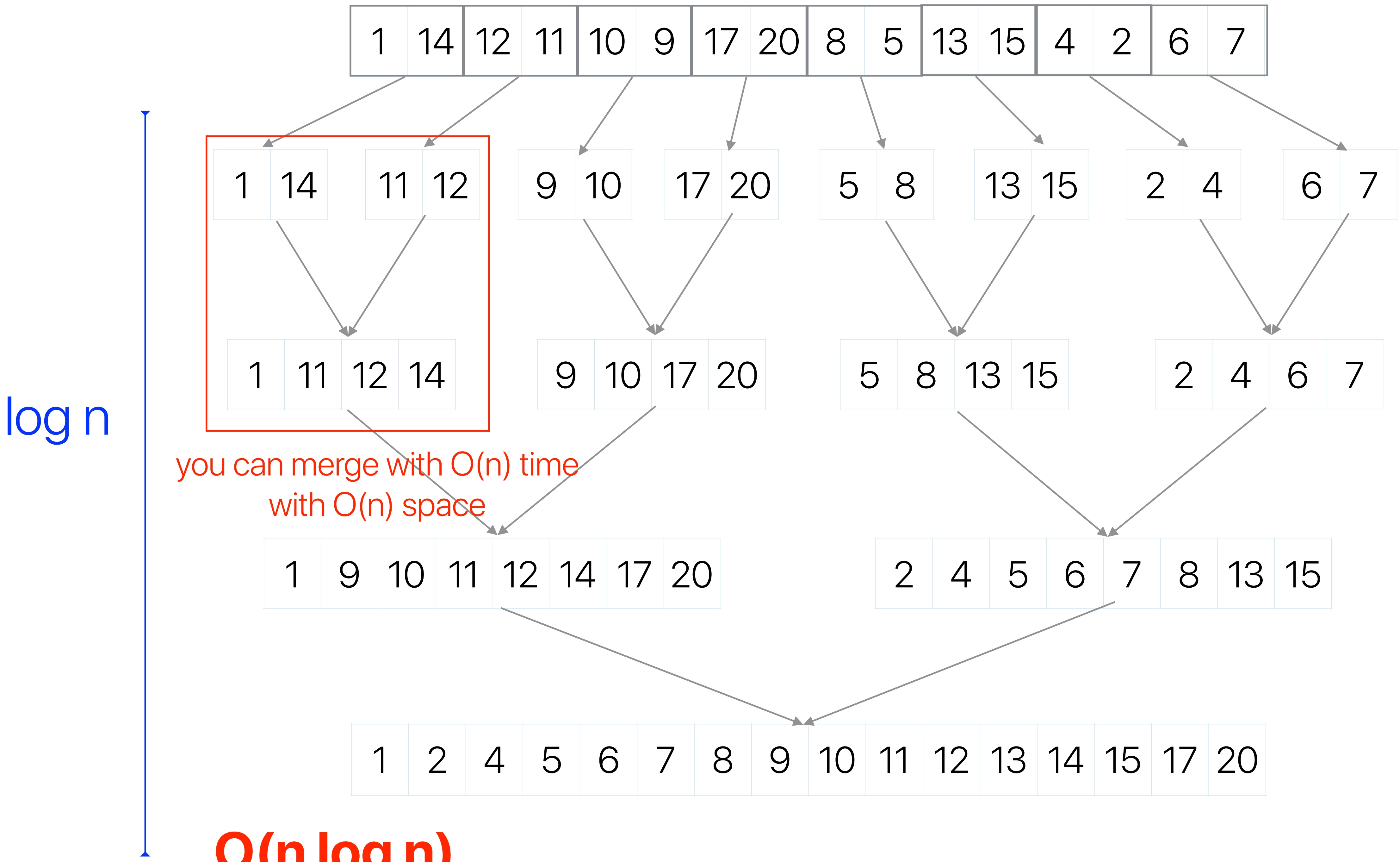
$O(nlog_2^2n)$

```
void BitonicSort() {

    int i,j,k;

    for (k=2; k<=N; k=2*k) {
        for (j=k>>1; j>0; j=j>>1) {
            for (i=0; i<N; i++) {
                int ij=i^j;
                if ((ij)>i) {
                    if ((i&k)==0 && a[i] > a[ij])
                        exchange(i,ij);
                    if ((i&k)!=0 && a[i] < a[ij])
                        exchange(i,ij);
                }
            }
        }
    }
}
```

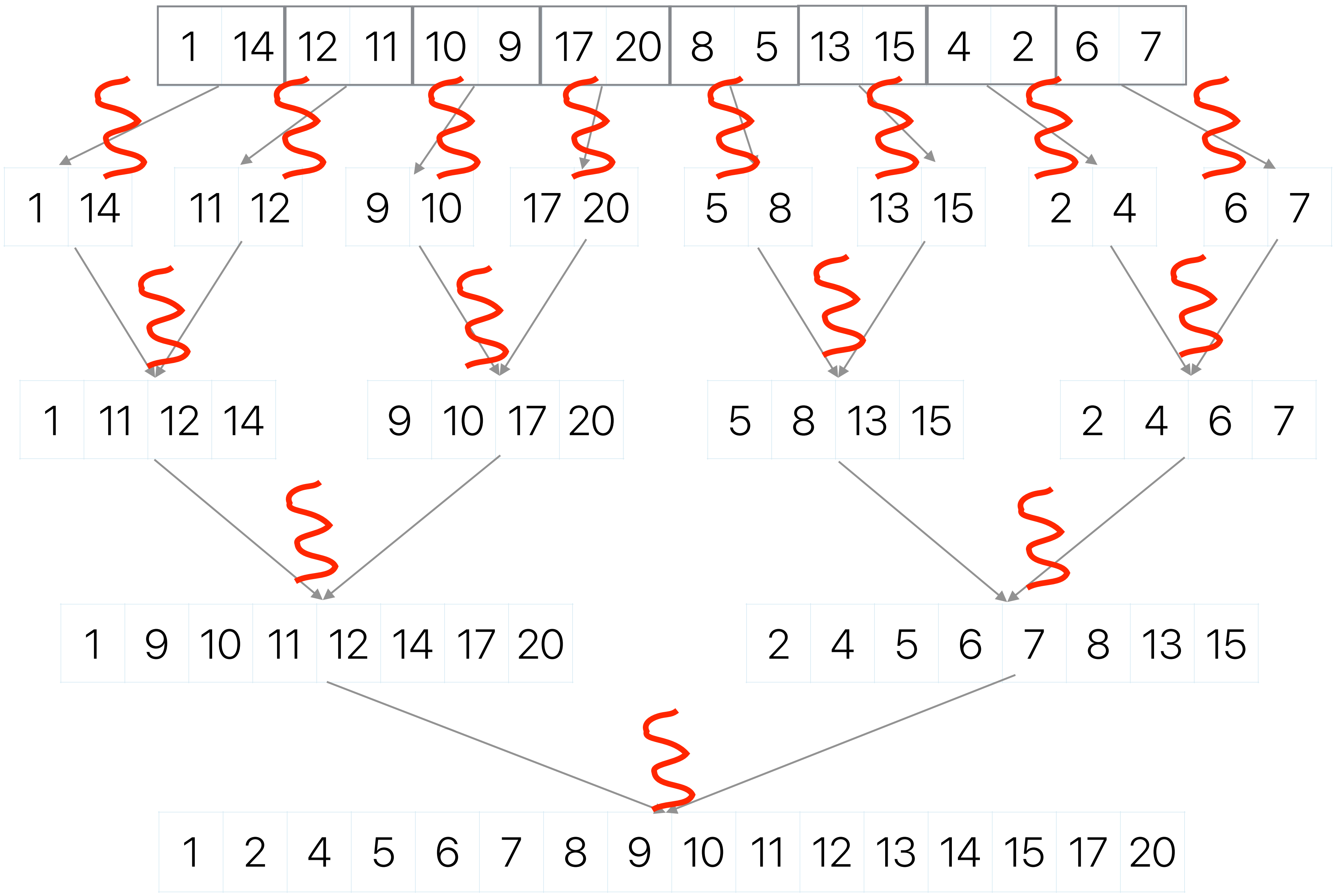
Merge sort



you can merge with O(n) time
with O(n) space

O(n log n)

Parallel merge sort



1 14 12 11 10 9 17 20 8 5 13 15 4 2 6 7

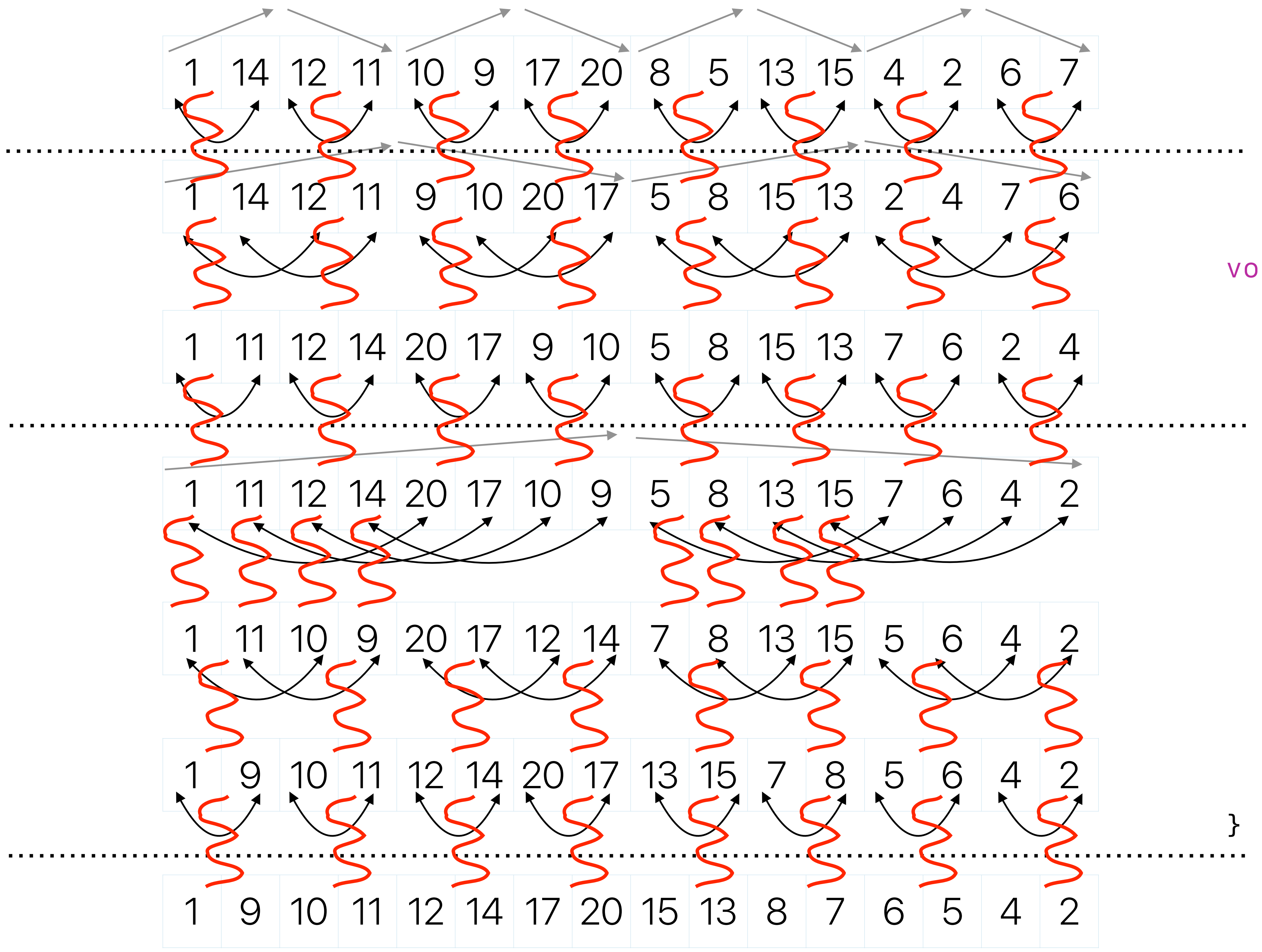
1 14 | 11 12 | 9 10 | 17 20 | 5 8 | 13 15 | 2 4 | 6 7

1 11 12 14 | 9 10 17 20 | 5 8 13 15 | 2 4 6 7

1 9 10 11 12 14 17 20 | 2 4 5 6 7 8 13 15

1 2 4 5 6 7 8 9 10 11 12 13 14 15 17 20

Bitonic sort



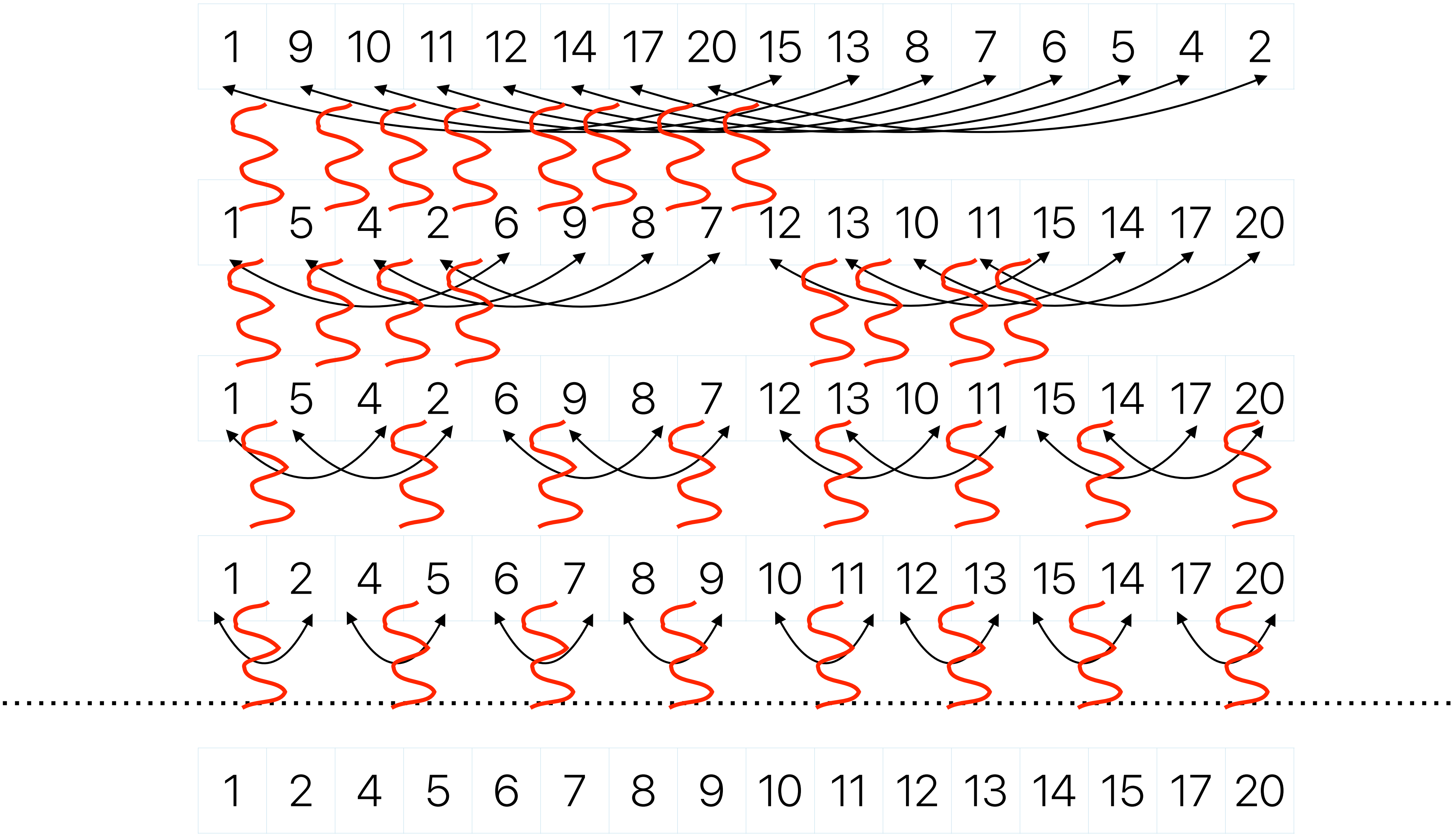
1	14	12	11	10	9	17	20	8	5	13	15	4	2	6	7
1	14	12	11	9	10	20	17	5	8	15	13	2	4	7	6
1	11	12	14	20	17	9	10	5	8	15	13	7	6	2	4
1	11	12	14	20	17	10	9	5	8	13	15	7	6	4	2
1	11	10	9	20	17	12	14	7	8	13	15	5	6	4	2
1	9	10	11	12	14	20	17	13	15	7	8	5	6	4	2
1	9	10	11	12	14	17	20	15	13	8	7	6	5	4	2

```
void BitonicSort() {

    int i,j,k;

    for (k=2; k<=N; k=2*k) {
        for (j=k>>1; j>0; j=j>>1) {
            for (i=0; i<N; i++) {
                int ij=i^j;
                if ((ij)>i) {
                    if ((i&k)==0 && a[i] > a[ij])
                        exchange(i,ij);
                    if ((i&k)!=0 && a[i] < a[ij])
                        exchange(i,ij);
                }
            }
        }
    }
}
```

Bitonic sort (cont.)



1 9 10 11 12 14 17 20 15 13 8 7 6 5 4 2

1 5 4 2 6 9 8 7 12 13 10 11 15 14 17 20

1 5 4 2 6 9 8 7 12 13 10 11 15 14 17 20

1 2 4 5 6 7 8 9 10 11 12 13 15 14 17 20

1 2 4 5 6 7 8 9 10 11 12 13 14 15 17 20

```
void BitonicSort() {

     int i,j,k;

     for (k=2; k<=N; k=2*k) {
          for (j=k>>1; j>0; j=j>>1) {
               for (i=0; i<N; i++) {
                    int ij=i^j;
                    if ((ij)>i) {
                         if ((i&k)==0 && a[i] > a[ij])
                              exchange(i,ij);
                         if ((i&k)!=0 && a[i] < a[ij])
                              exchange(i,ij);
                    }
               }
          }
     }
}
```

benefits — in-place merge (no additional space is necessary), very stable comparison patterns

O(n log² n) — hard to beat n(log n) if you can’t parallelize this a lot!

Corollary #4

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable}{\infty}}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

- If we can build a processor with unlimited parallelism
    - The complexity doesn’t matter as long as the algorithm can utilize all parallelism
    - That’s why bitonic sort or MapReduce works!
- **The future trend of software/application design is seeking for more parallelism rather than lower the computational complexity**

Corollary #3

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable}) + \frac{f\_parallelizable}{\infty}}$$

$$Speedup_{parallel}(f_{parallelizable}, \infty) = \frac{1}{(1 - f_{parallelizable})}$$

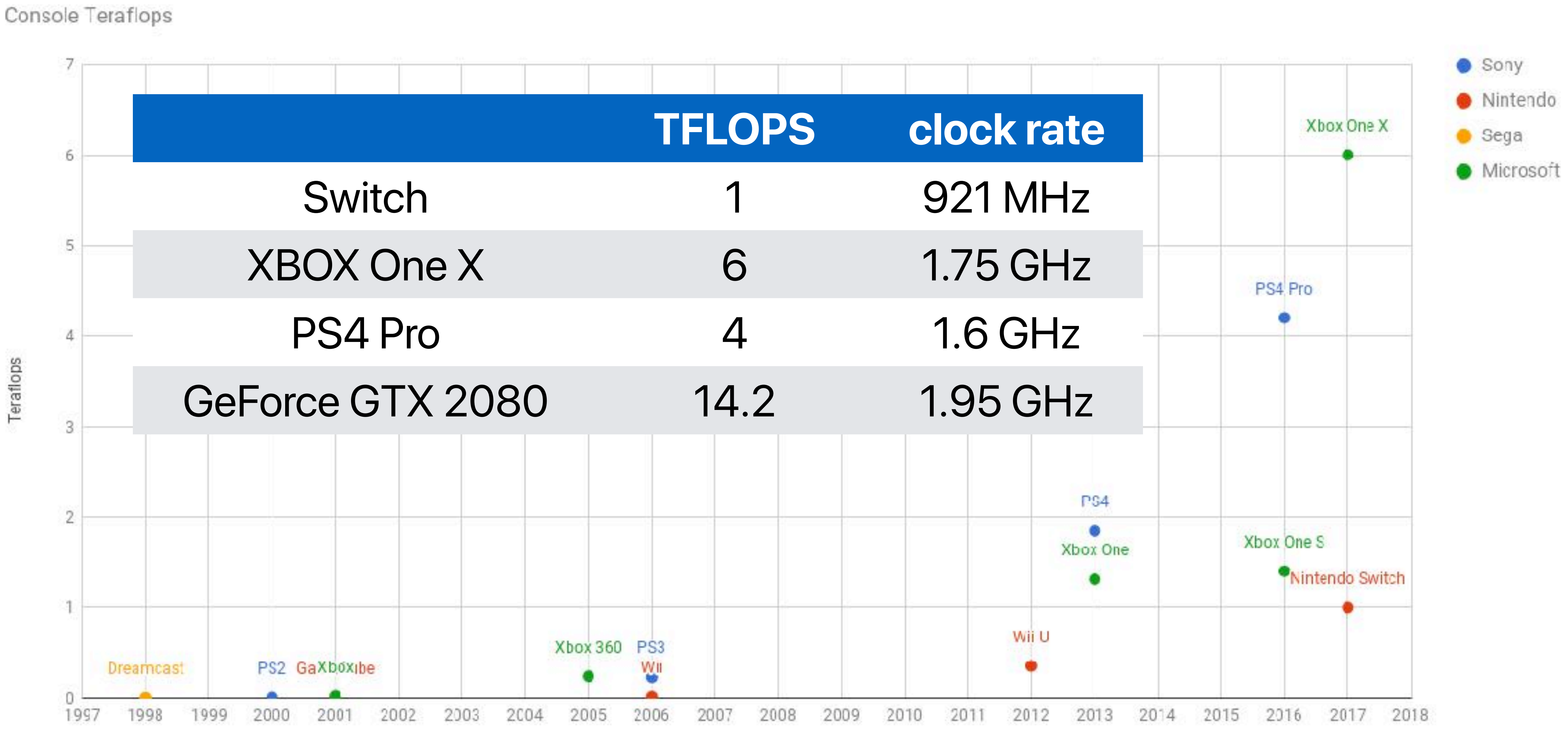
- Single-core performance still matters
  - It will eventually dominate the performance
  - If we cannot improve single-core performance further, finding more "parallelizable" parts is more important

“Fair” Comparisons

Andrew Davison. Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers. In Humour the Computer, MITP, 1995

V. Sze, Y. -H. Chen, T. -J. Yang and J. S. Emer. How to Evaluate Deep Neural Network Processors: TOPS/W (Alone) Considered Harmful. In IEEE Solid-State Circuits Magazine, vol. 12, no. 3, pp. 28-41, Summer 2020.

TFLOPS (Tera FLoating-point Operations Per Second)



	TFLOPS	clock rate
Switch	1	921 MHz
XBOX One X	6	1.75 GHz
PS4 Pro	4	1.6 GHz
GeForce GTX 2080	14.2	1.95 GHz

Is TFLOPS (Tera FLoating-point Operations Per Second) a good metric?

$$TFLOPS = \frac{\#\ of\ floating\ point\ instructions \times 10^{-12}}{Exection\ Time}$$

$$= \frac{IC \times \%\ of\ floating\ point\ instructions \times 10^{-12}}{IC \times CPI \times CT}$$

$$= \frac{\%\ of\ floating\ point\ instructions \times 10^{-12}}{CPI \times CT}$$

IC is gone!

- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- Does not make sense if the application is not floating point intensive

TFLOPS (Tera FLoating-point Operations Per Second)

- Cannot compare different ISA/compiler
  - What if the compiler can generate code with fewer instructions?
  - What if new architecture has more IC but also lower CPI?
- Does not make sense if the application is not floating point intensive

	TFLOPS	clock rate
Switch	1	921 MHz
XBOX One X	6	1.75 GHz
PS4 Pro	4	1.6 GHz
GeForce GTX 2080	14.2	1.95 GHz

Announcement

- Reading quiz due next Monday before the lecture
  - We will drop two of your least performing reading quizzes
  - You have two shots, both unlimited time
  - The commentary question in Quiz #2 needs manual grading — don’t be panic
- Assignment #1 will be up tonight
- Check our website for slides, eLearn for quizzes/assignments, piazza for discussions
- Youtube channel for lecture recordings: https://www.youtube.com/c/ProfUsagi/playlists

Computer
Science &
Engineering

203

つづく