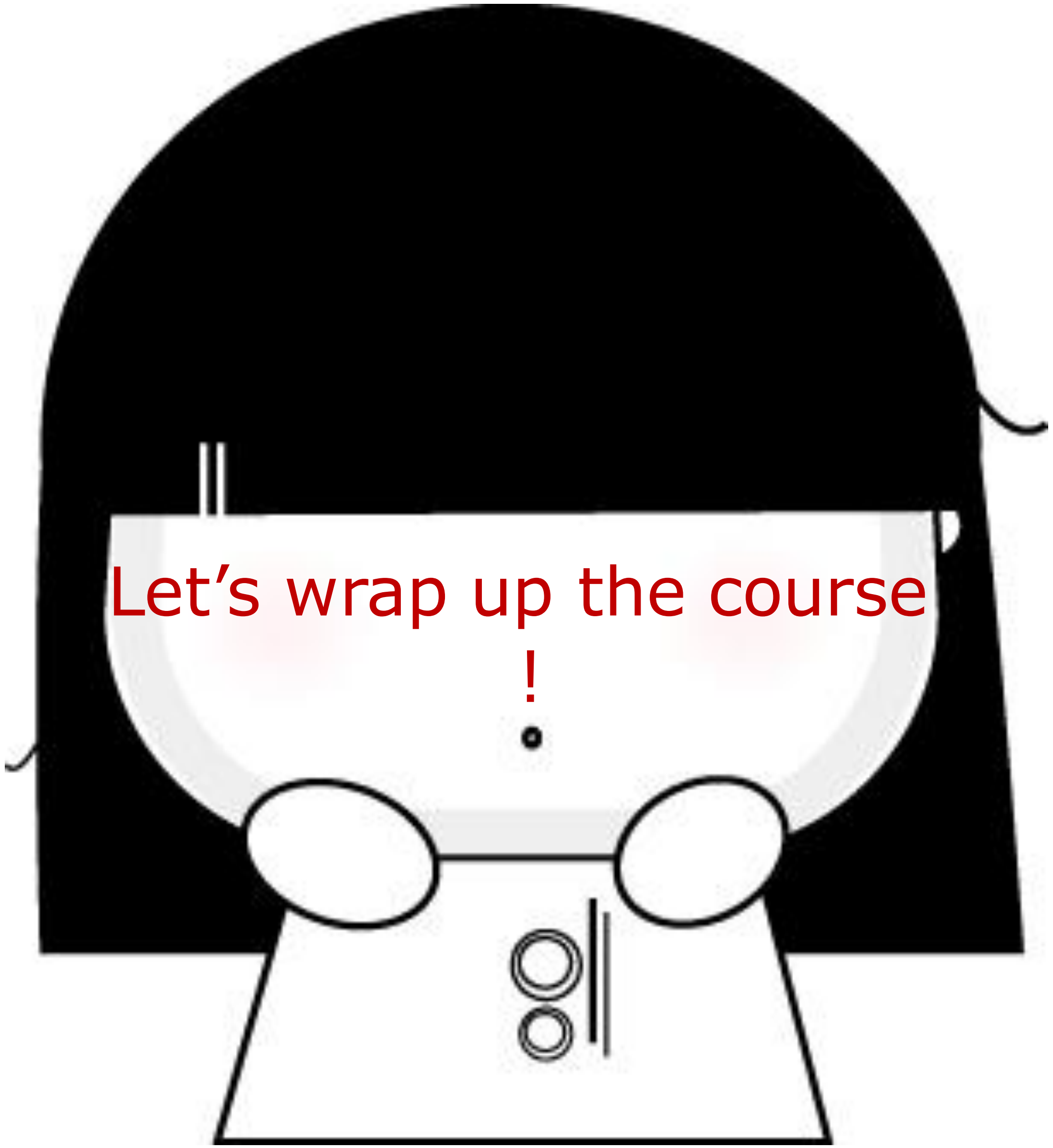
Psychoinformatics & Neuroinformatics

Week 15

Parallel & Distributed Computing of Big Data

by Tsung-Ren (Tren) Huang 黃從仁

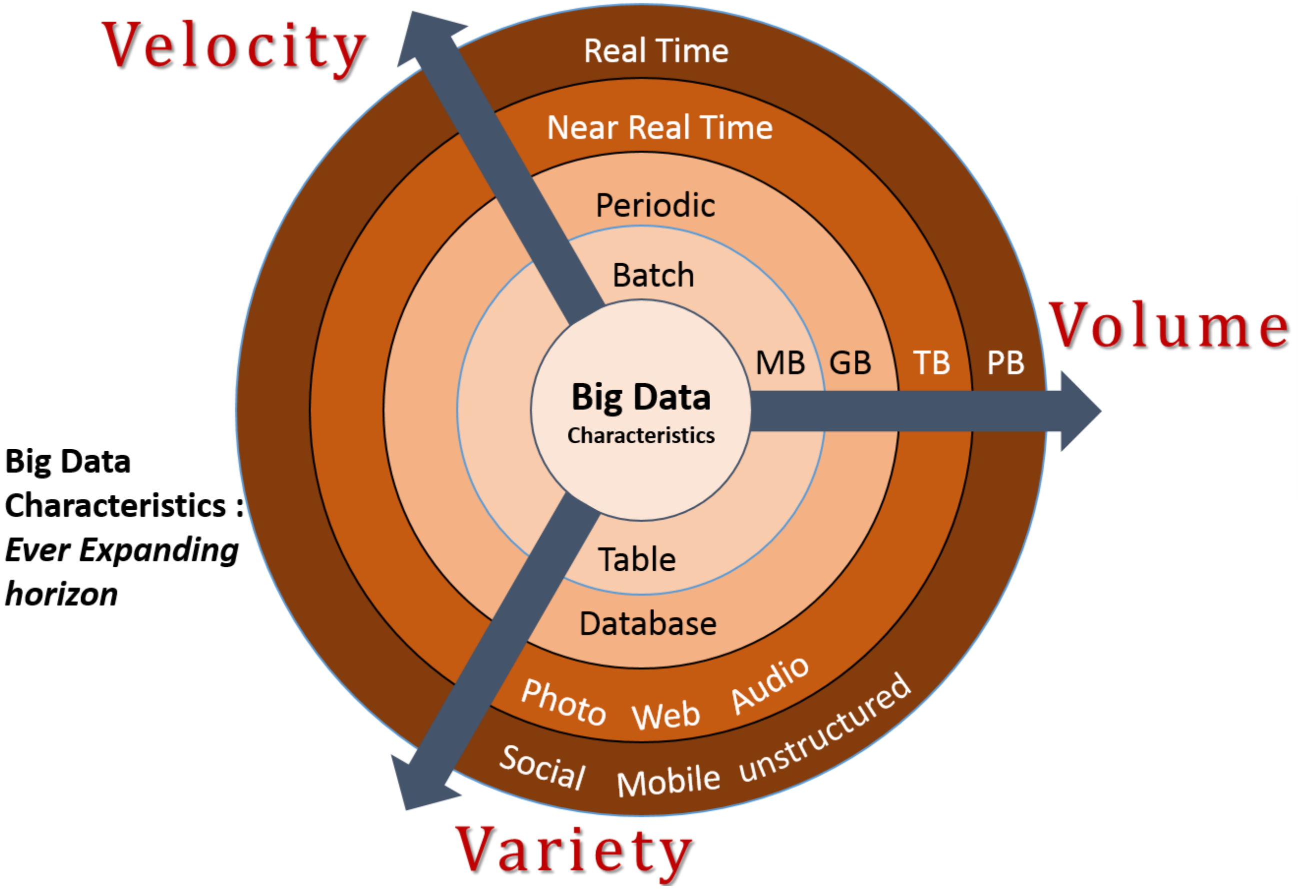


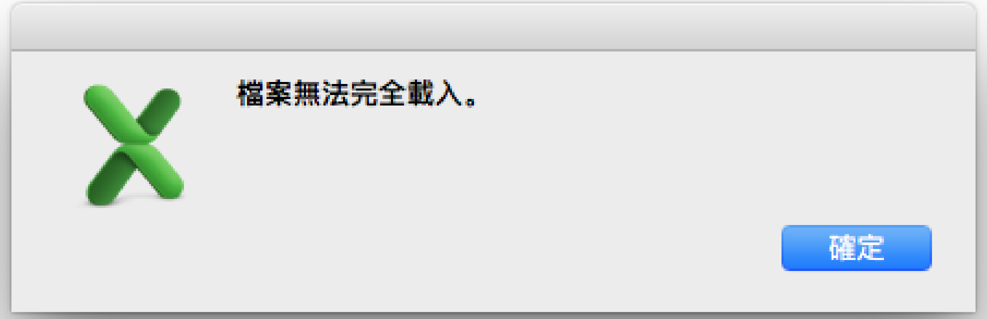
Let’s wrap up the course !

Analyzing Big Data

We're done w/ variety
& moving on to volume.

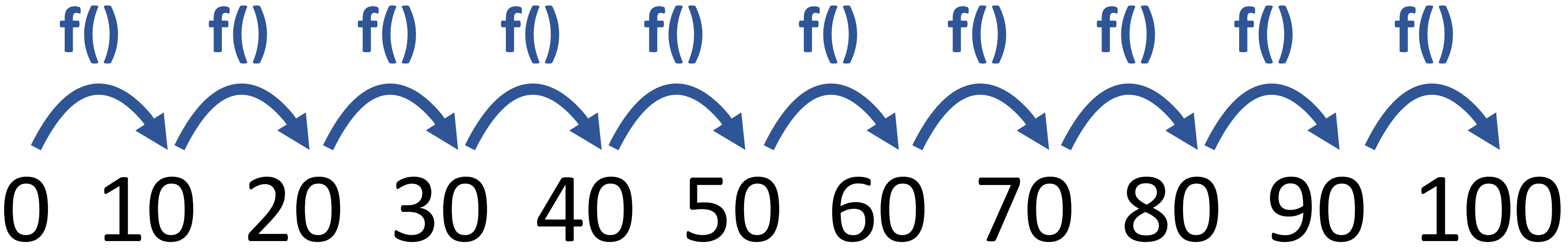
But how big is big?
Try loading info_15_network.txt,
which is only 32MB in size.




檔案無法完全載入。

Excel can only load up to
2^{20}=1048,576 rows.

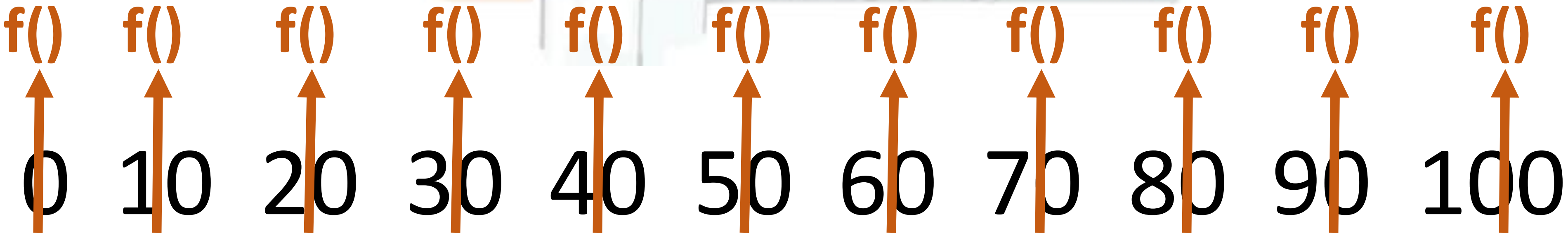
Sequential Computing vs. **Parallel Computing**



f() f() f() f() f() f() f() f() f() f()

0 10 20 30 40 50 60 70 80 90 100

(1 production line)



f() f() f() f() f() f() f() f() f() f() f()

0 10 20 30 40 50 60 70 80 90 100

(11 production lines)

Topics for today

Asynchronous Execution

on one thread

Parallel Computing

on one computer

Distributed Computing

across multiple computers

Topics for today

Asynchronous Execution

on one thread

Parallel Computing

on one computer

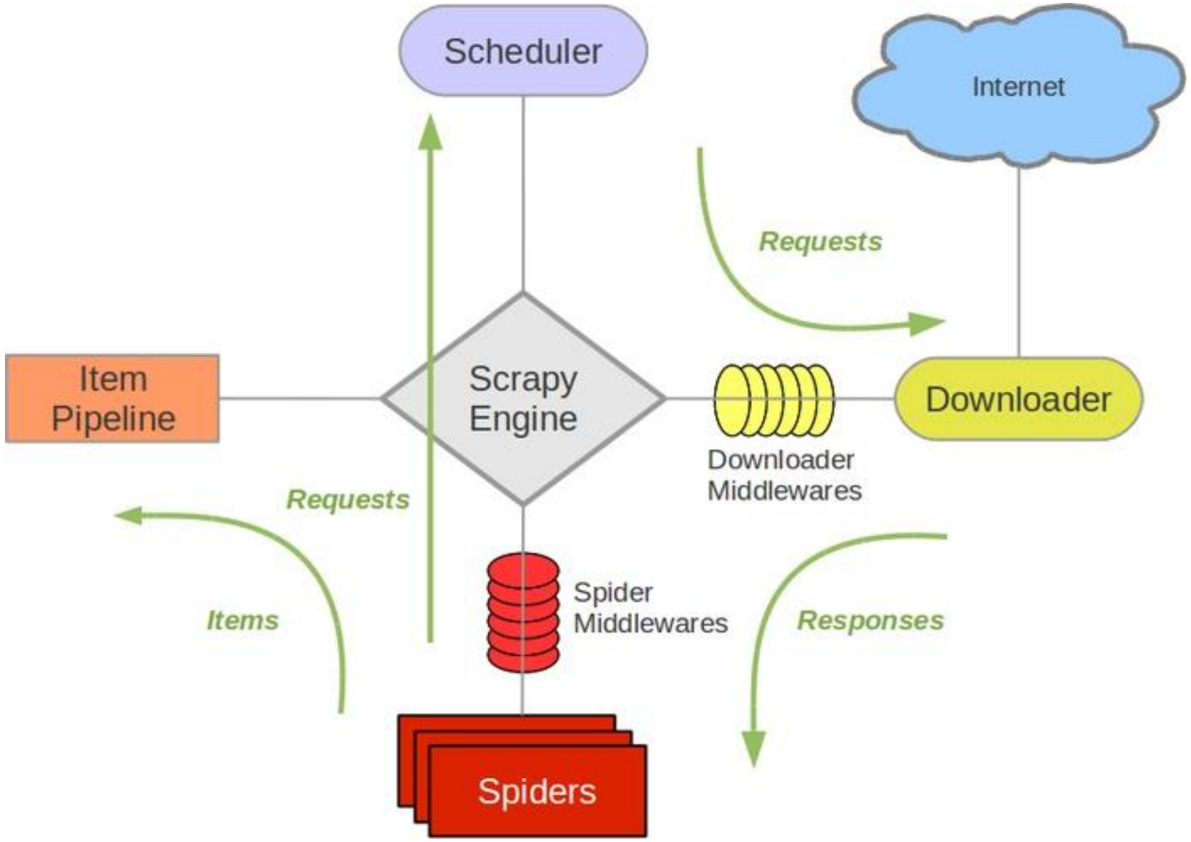
Distributed Computing

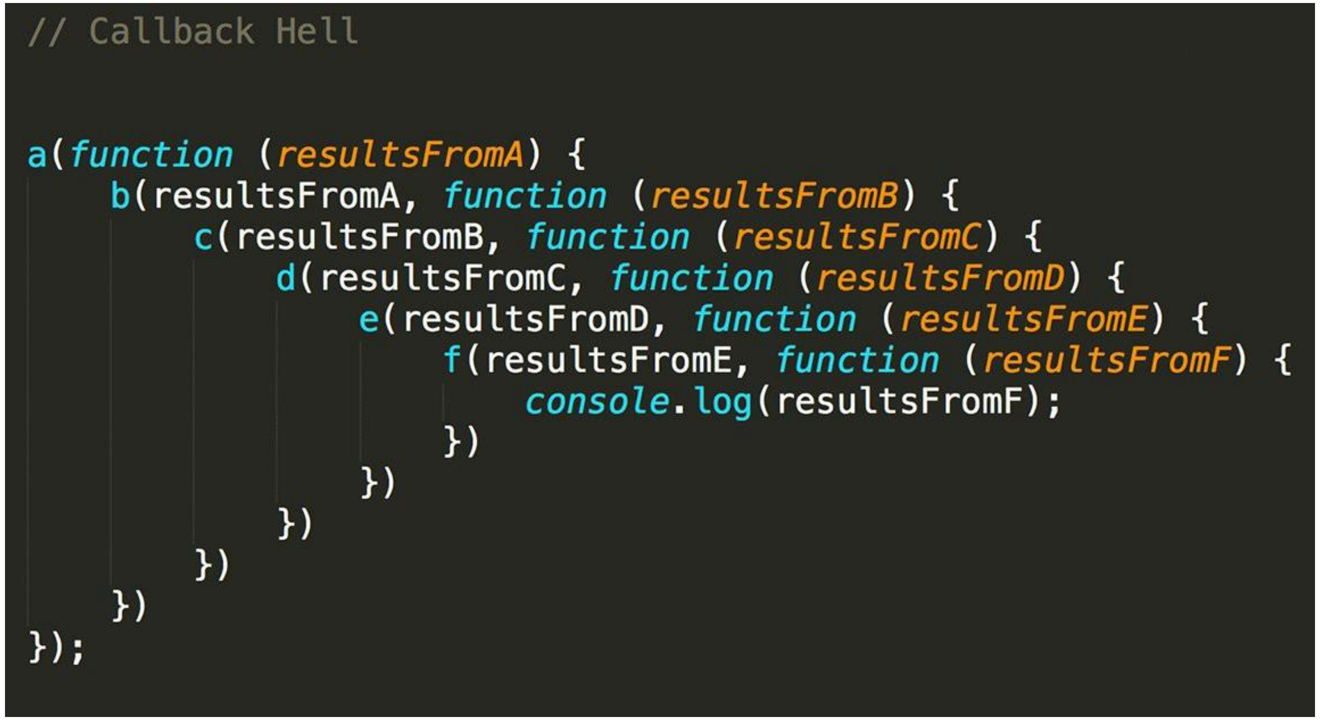
across multiple computers

Correlational vs. Experimental Methods

Powered by Twisted’s Async I/O

JS’s & Node’s Async I/O





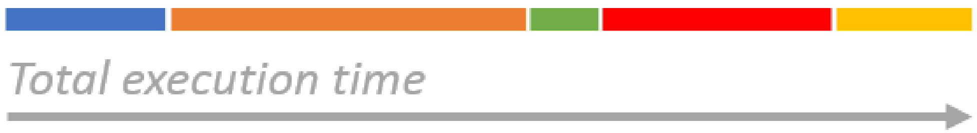
```
// Callback Hell


a(function (resultsFromA) {
    b(resultsFromA, function (resultsFromB) {
        c(resultsFromB, function (resultsFromC) {
            d(resultsFromC, function (resultsFromD) {
                e(resultsFromD, function (resultsFromE) {
                    f(resultsFromE, function (resultsFromF) {
                        console.log(resultsFromF);
                    })
                })
            })
        })
    })
});
```

Synchronous vs. Asynchronous Exec. (1/2)

Async exec. best for massive, slow, & non-independent I/O

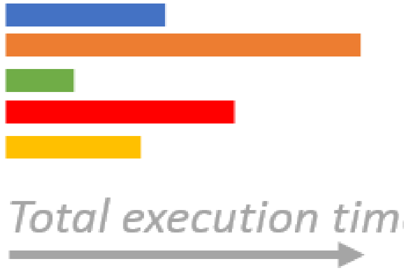
Synchronous

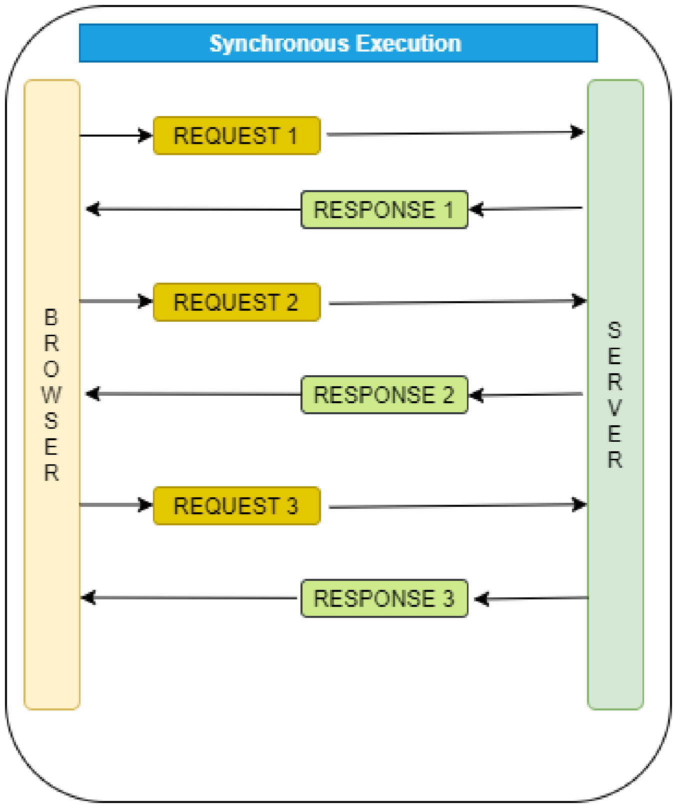
One request at a time

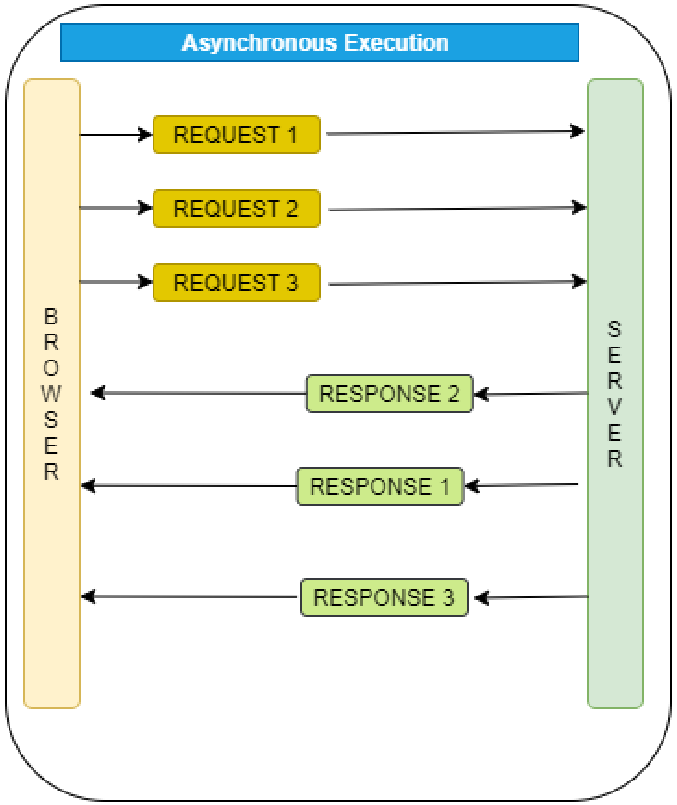


Asynchronous

Multiple requests at a time



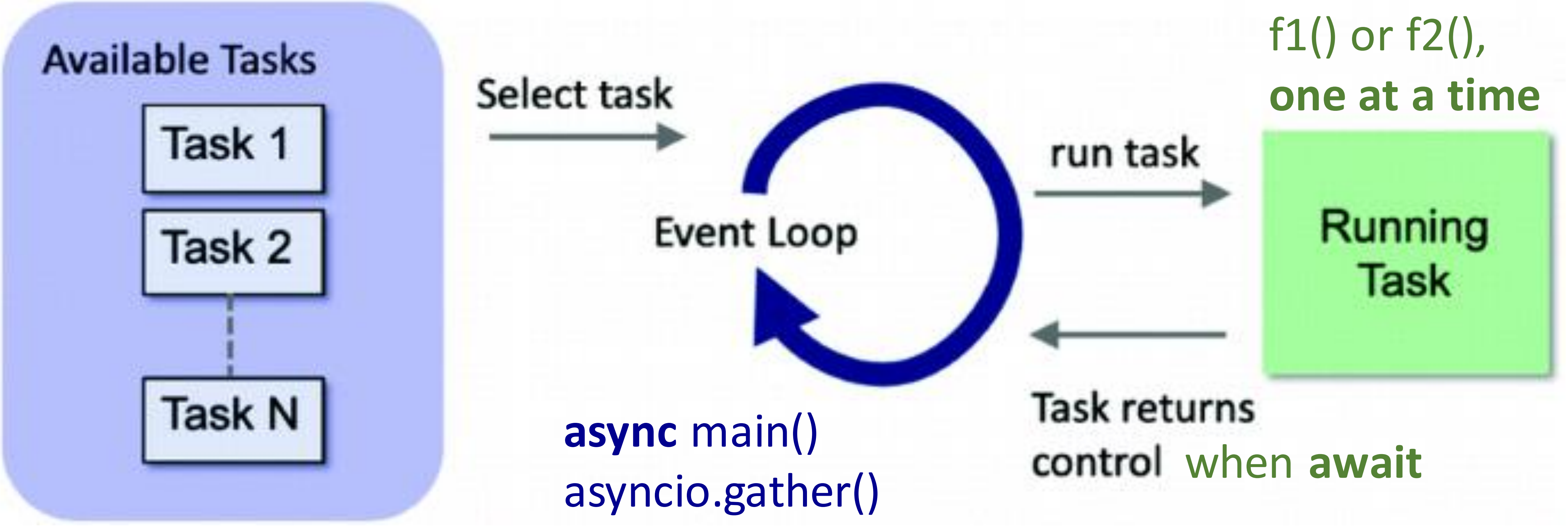




Async vs. Await in “asyncio”

Co-routines f1() & f(2) seem to run simultaneously

asyncio.run(**async** main()) or
asyncio.gather(**async** f1(), **async** f2())



Available Tasks

Task 1

Task 2

Task N

Select task

Event Loop

run task

Running Task

Task returns control when **await**

f1() or f2(),
one at a time

async main()
asyncio.gather()

Topics for today

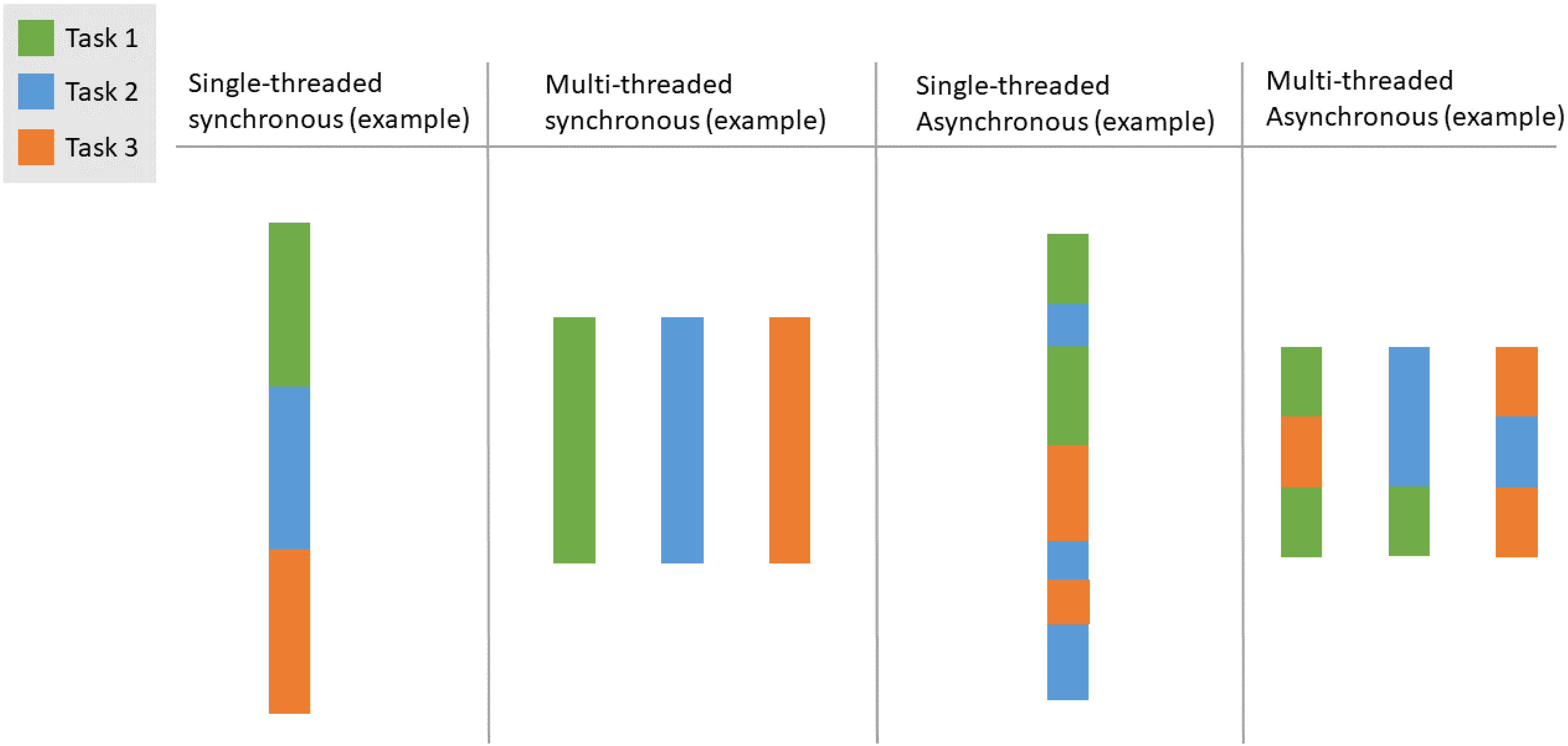
Asynchronous Execution
on one thread

Parallel Computing
on one computer

Distributed Computing
across multiple computers

Asynchronous vs Multithreading

Asynchronous execution uses only one thread *by default;*
Each thread can be executed by one CPU core!



🟩 Task 1

🟦 Task 2

🟧 Task 3

Single-threaded synchronous (example)

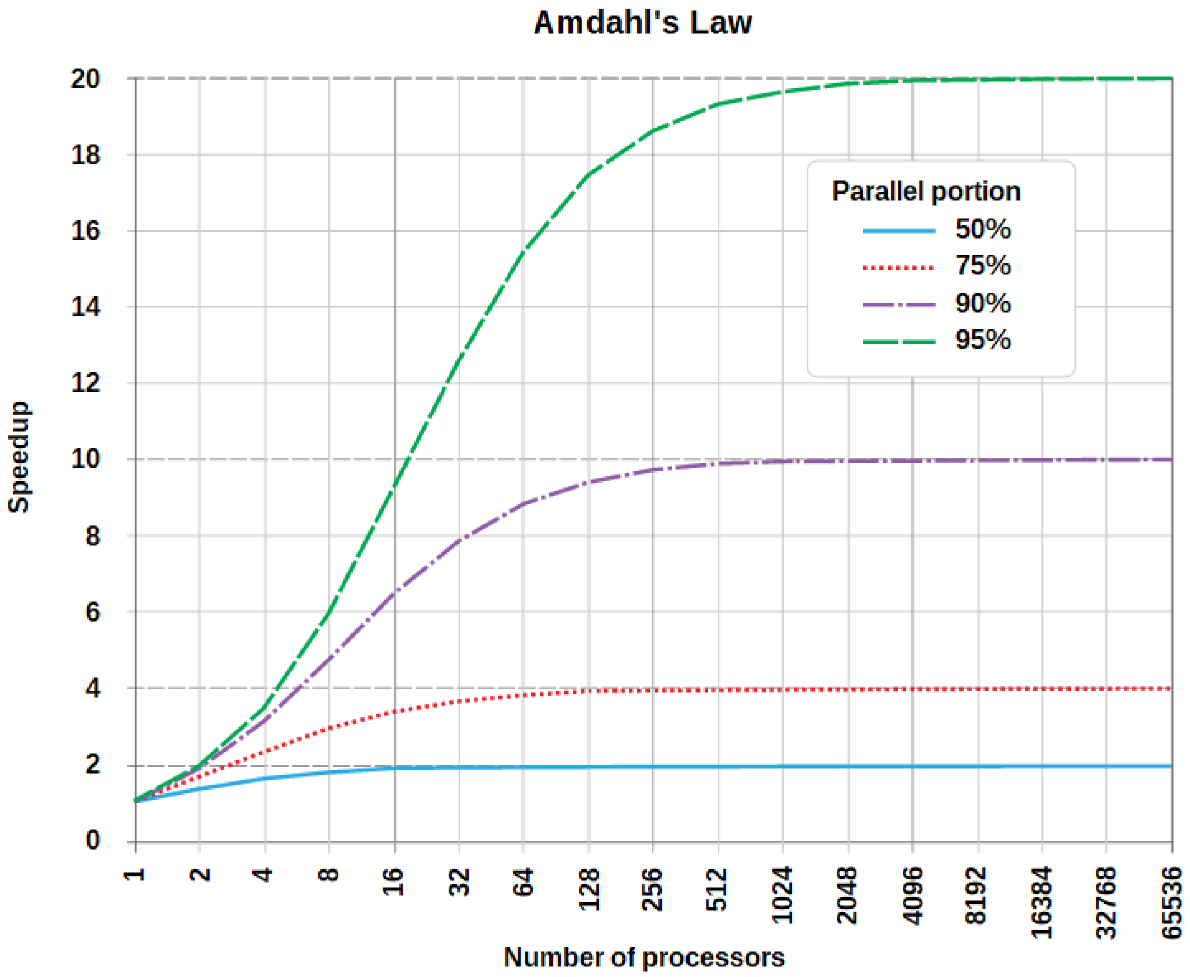
Multi-threaded synchronous (example)

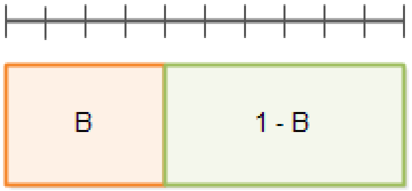
Single-threaded Asynchronous (example)

Multi-threaded Asynchronous (example)

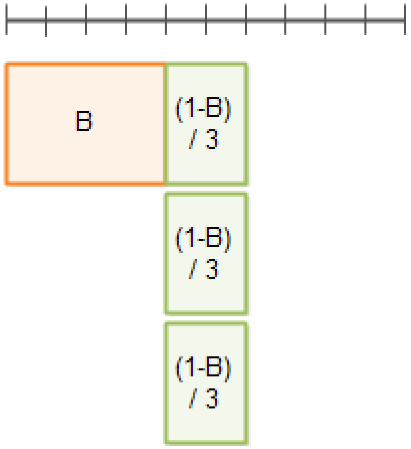
The upper bound of speedup

Amdahl’s Law: Sequential portion is the bottle-neck!





B = Non-parallelizable
1 - B = Parallelizable



Speedup(N)
$=1/[B+(1-B)/N]$
$\sim 1/B$ when $N=\infty$

Revisiting “map” from Week 1

```
import math
def adjust_score(old):
        new=math.sqrt(old)*10
        return new

print(list(map(adjust_score,range(0,101,10))))
```

Unlike the map() in multiprocessing or MapReduce, the map() here is actually a sequential operation.

Python: concurrent.futures

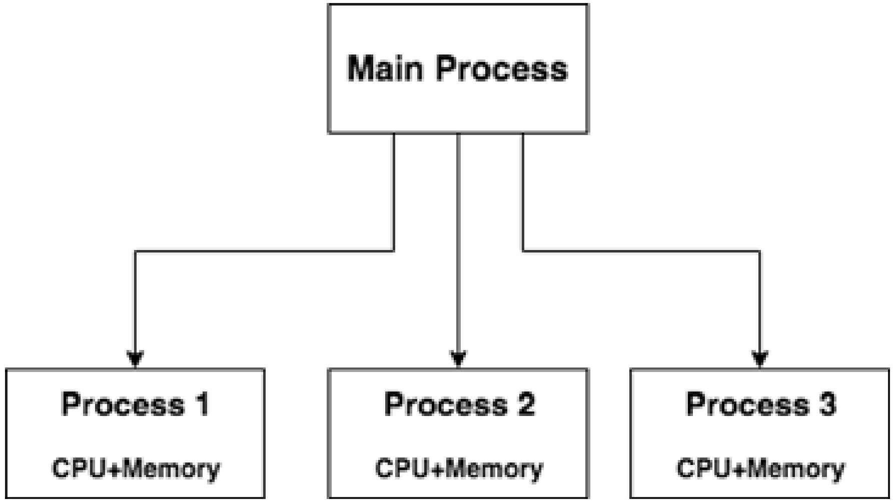
which allows for multithreading & multiprocessing

```
import math, concurrent.futures as cf
def adjust_score(old):
 new=math.sqrt(old)*10
 return new

with cf.ThreadPoolExecutor(max_workers=2) as pool:
#with cf.ProcessPoolExecutor(max_workers=2) as pool:
 new=pool.map(adjust_score, range(100))

list(new)
```

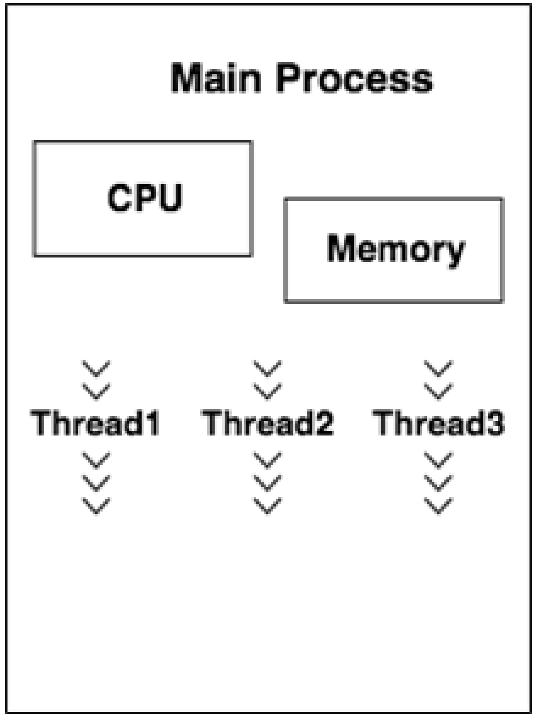
MultiProcessing vs. MultiThreading

A process = a program w/ its own CPU/RAM resources

Multiprocessing



Multithreading

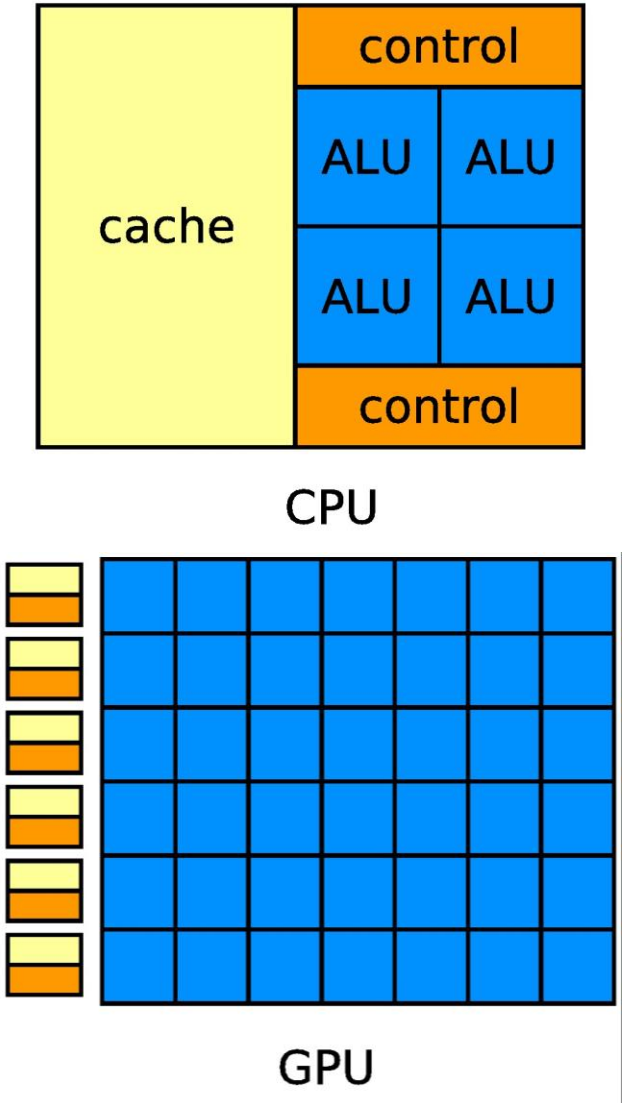


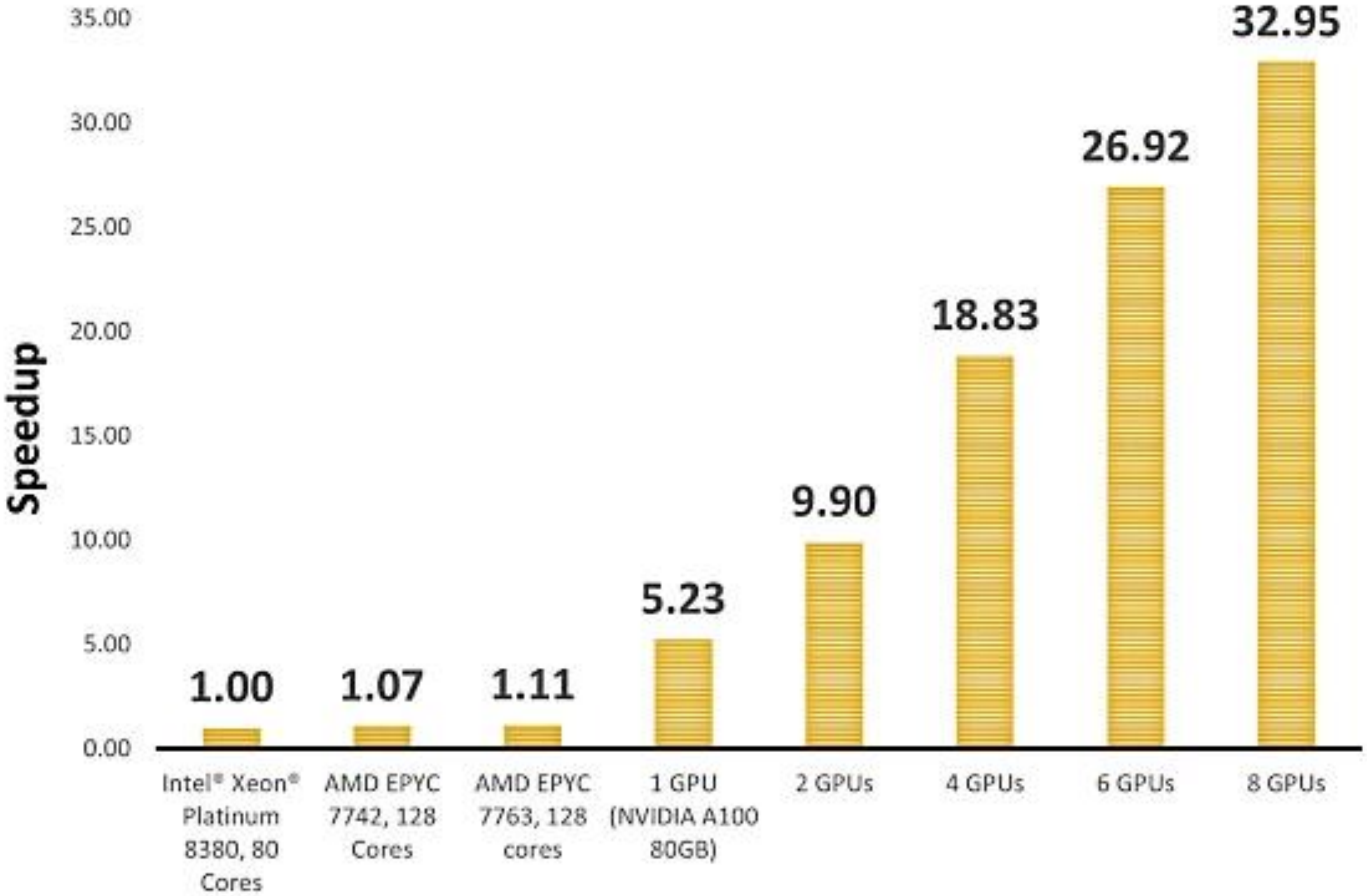
for CPU-heavy tasks

for I/O-heavy tasks

CPU vs. GPU

Compared to a CPU, a GPU has more simpler cores





Topics for today

Asynchronous Execution

on one thread

Parallel Computing

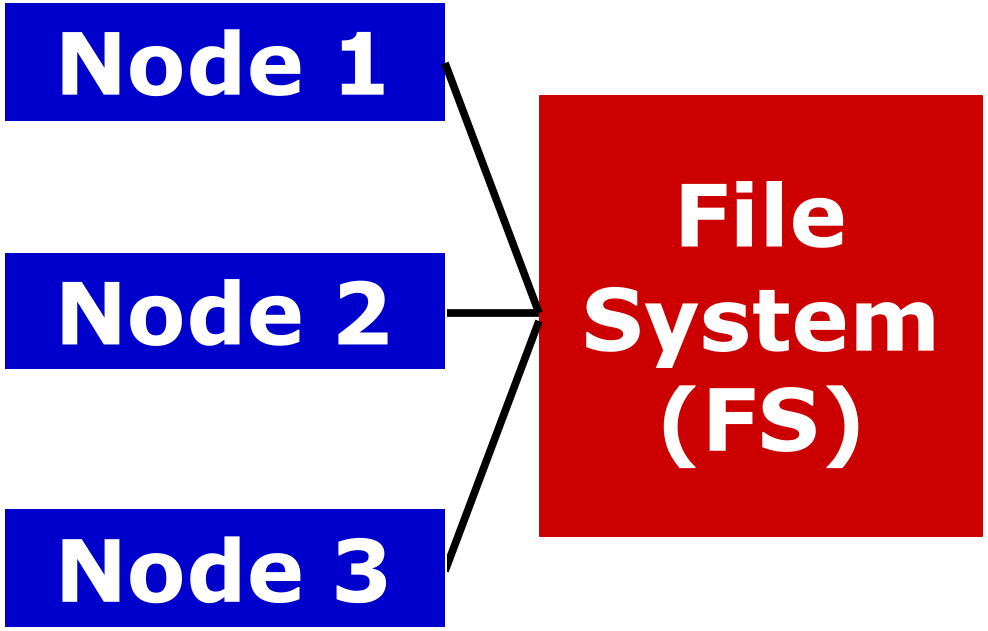
on one computer

Distributed Computing

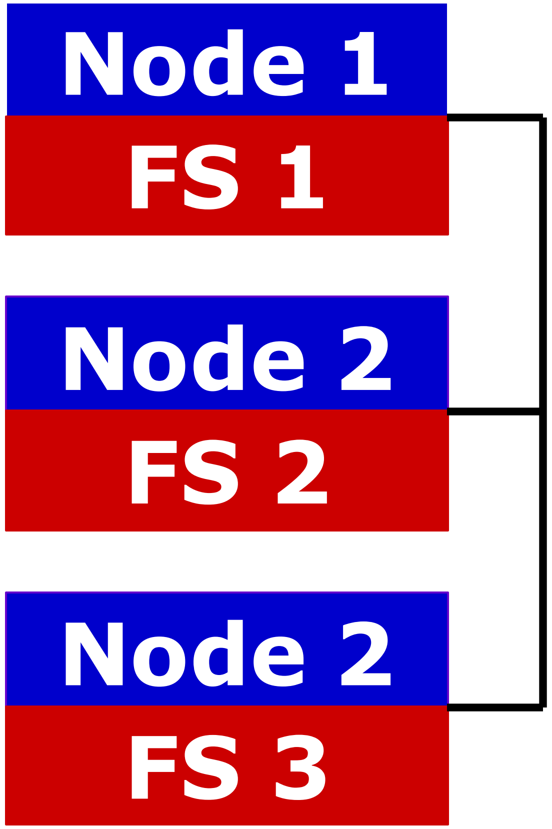
across multiple computers

Types of Computer Clusters

High-Performance Computing
(Centralized Storage)

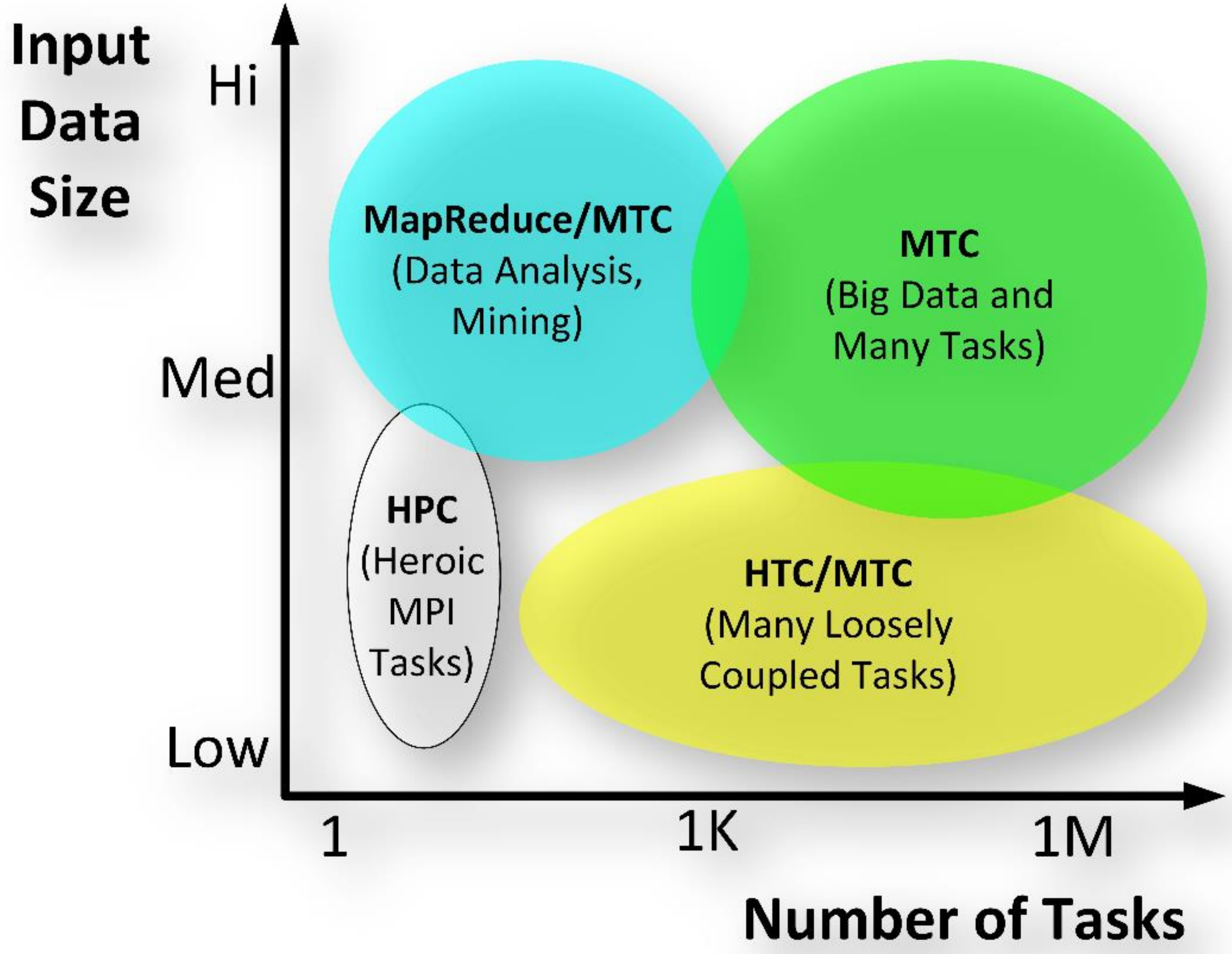


Big Data Analytics
(Distributed Storage)



Types of Computing Tasks

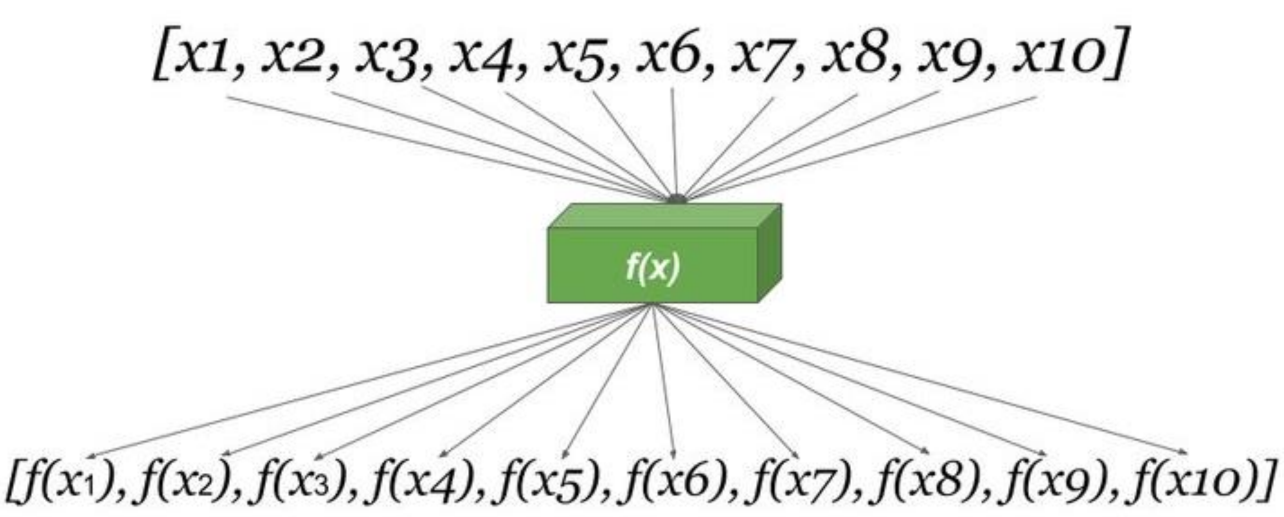
HPC=High Perf.; HTC=High Throughput; MTC=Many-Task



What is MapReduce?

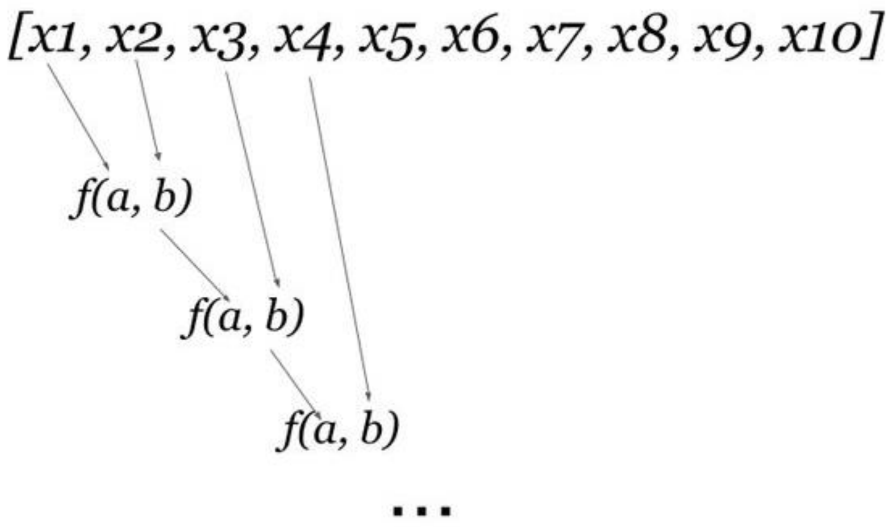
Same Instruction Multiple Data

Map

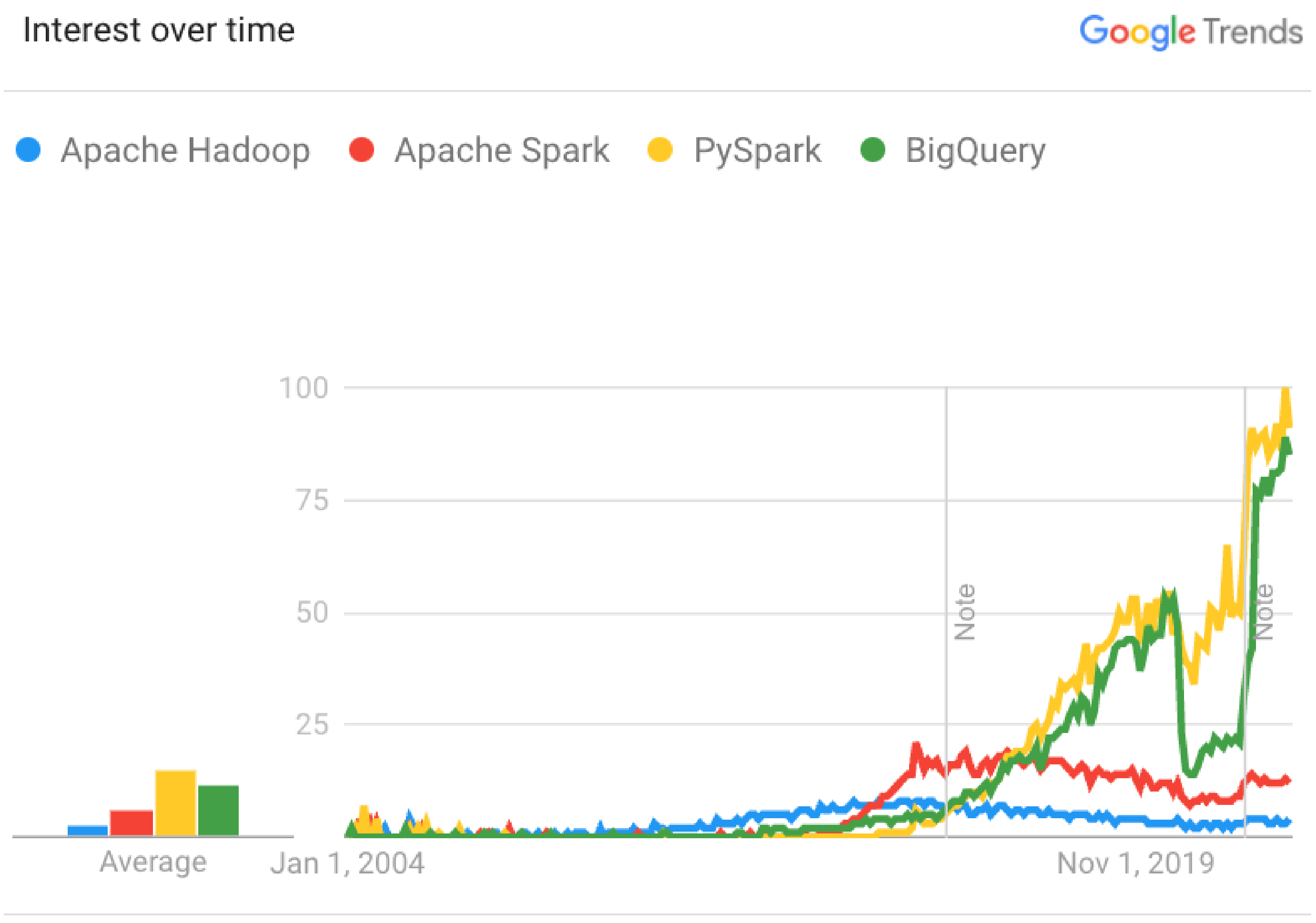

[x1, x2, x3, x4, x5, x6, x7, x8, x9, x10]
f(x)
[f(x1), f(x2), f(x3), f(x4), f(x5), f(x6), f(x7), f(x8), f(x9), f(x10)]

Merging MAP Results

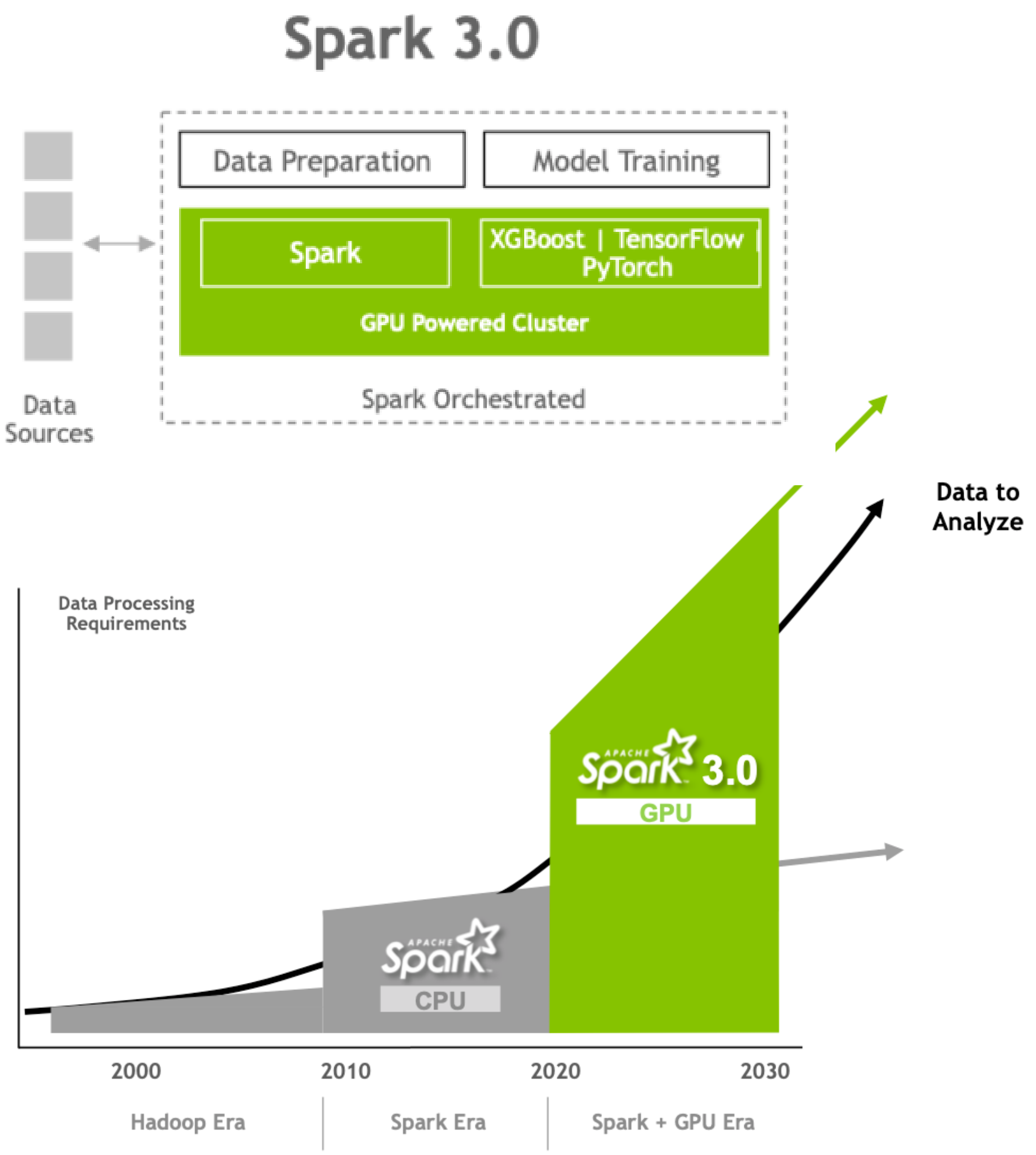
Reduce



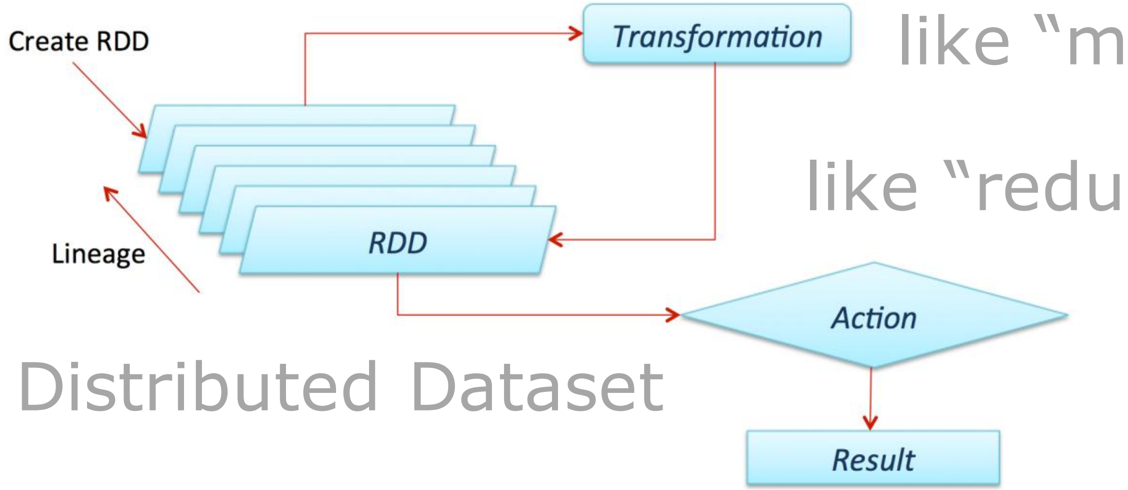
The Evolution of MapReduce Environments



Worldwide. 1/1/04 - 12/11/22. Web Search.



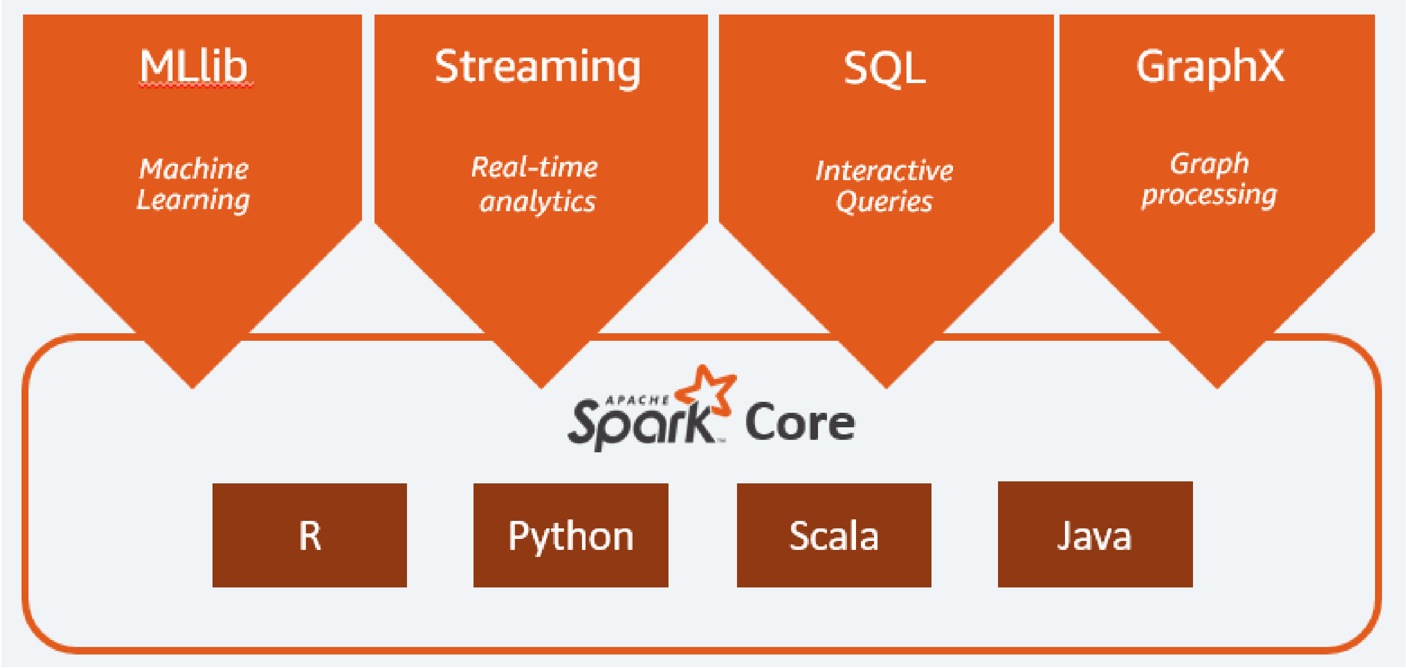
Apache Spark: The King of Big Data



like “map”

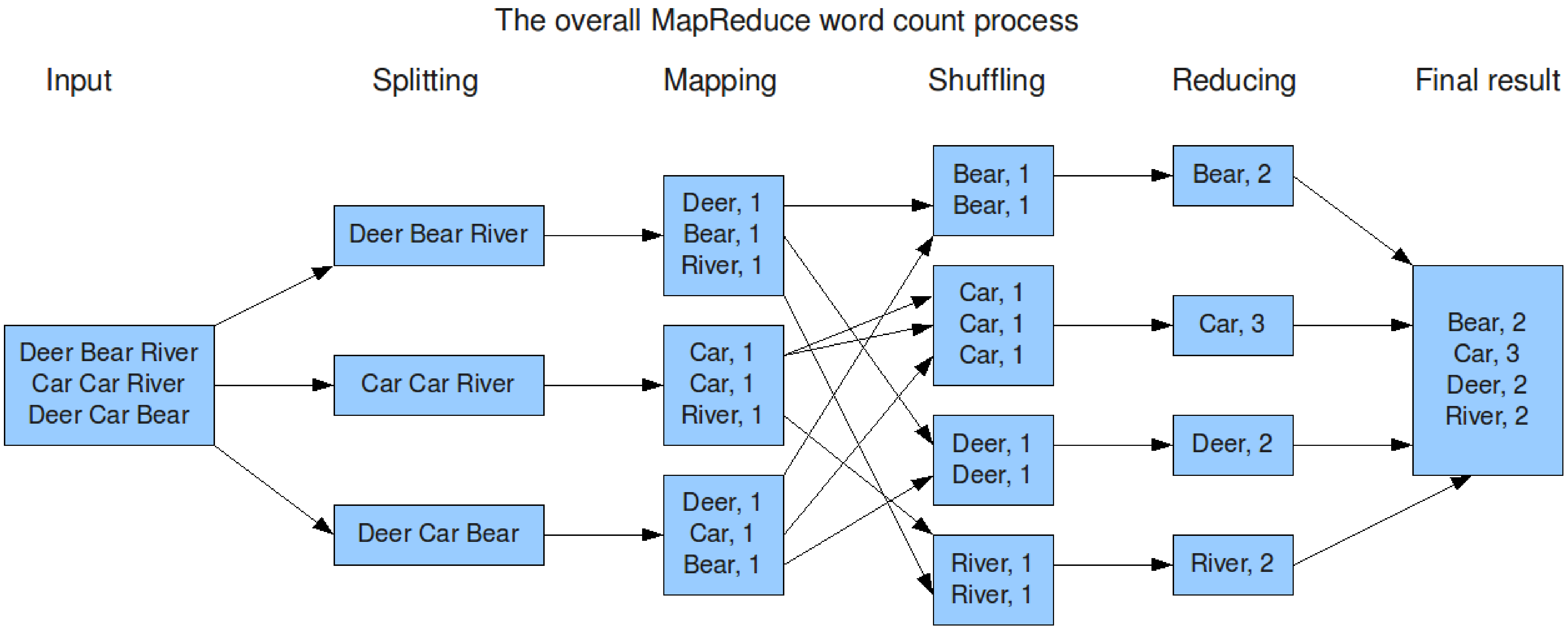
like “reduce”

Resilient Distributed Dataset



An Example of MapReduce in Spark

Even word counting is tedious when implemented by MapReduce



The overall MapReduce word count process

Input

Deer Bear River
Car Car River
Deer Car Bear

Splitting

Deer Bear River

Car Car River

Deer Car Bear

Mapping

Deer, 1
Bear, 1
River, 1

Car, 1
Car, 1
River, 1

Deer, 1
Car, 1
Bear, 1

Shuffling

Bear, 1
Bear, 1

Car, 1
Car, 1
Car, 1

Deer, 1
Deer, 1

River, 1
River, 1

Reducing

Bear, 2

Car, 3

Deer, 2

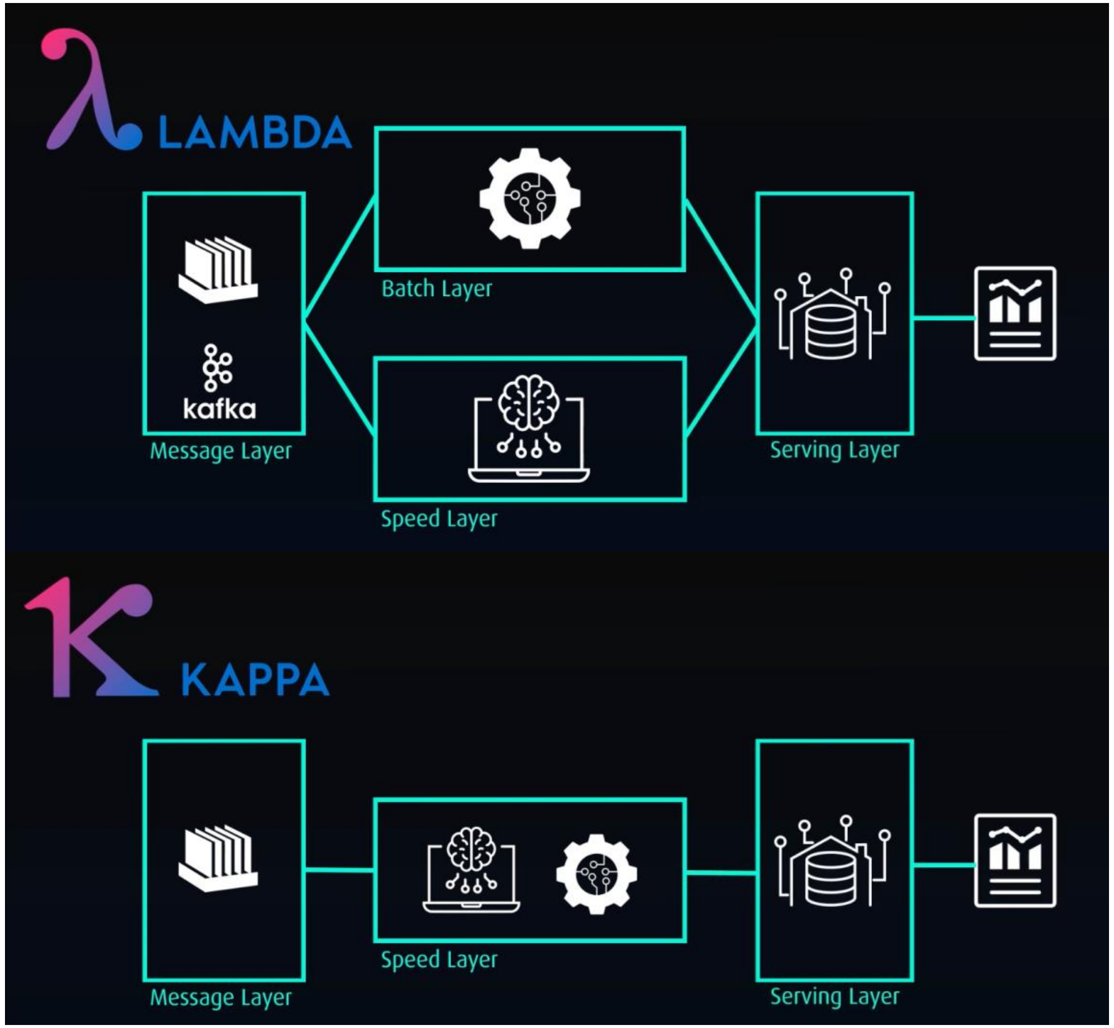
River, 2

Final result

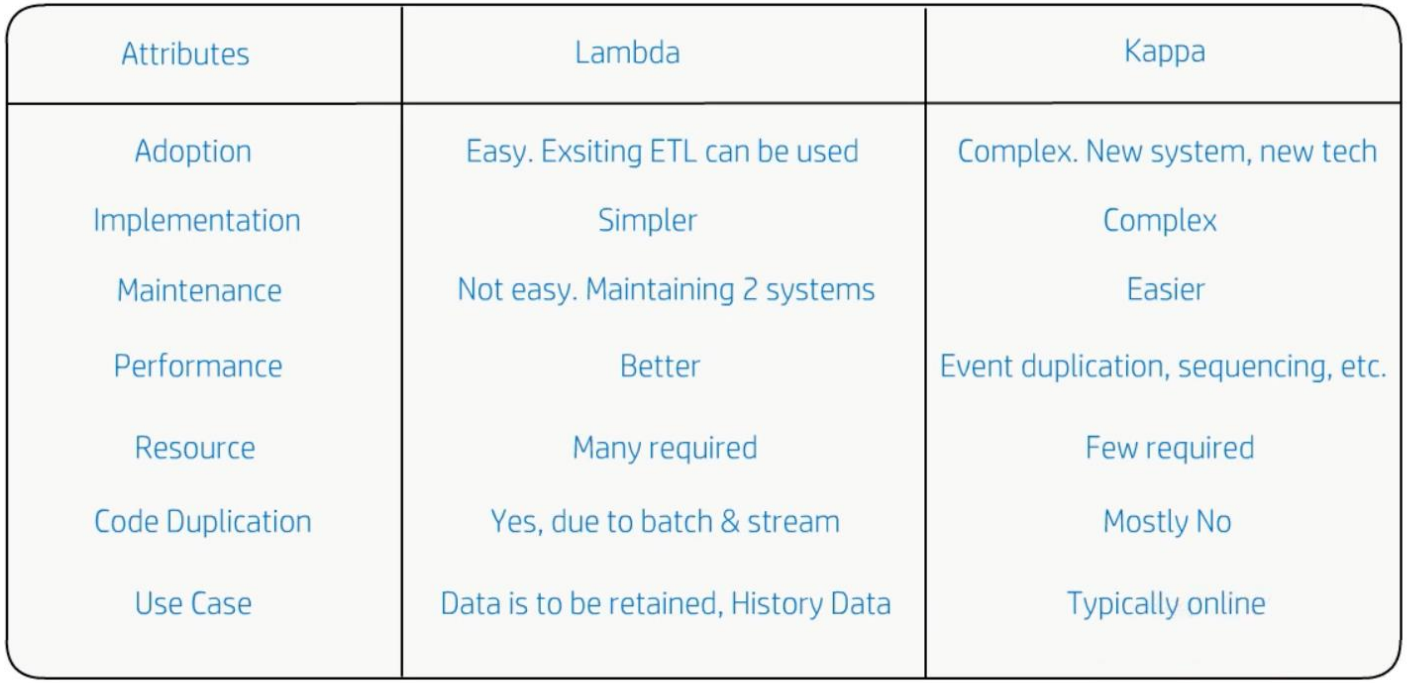
Bear, 2
Car, 3
Deer, 2
River, 2

RDD2 RDD3 RDD4

(Near) Real-Time Processing: λ vs. κ

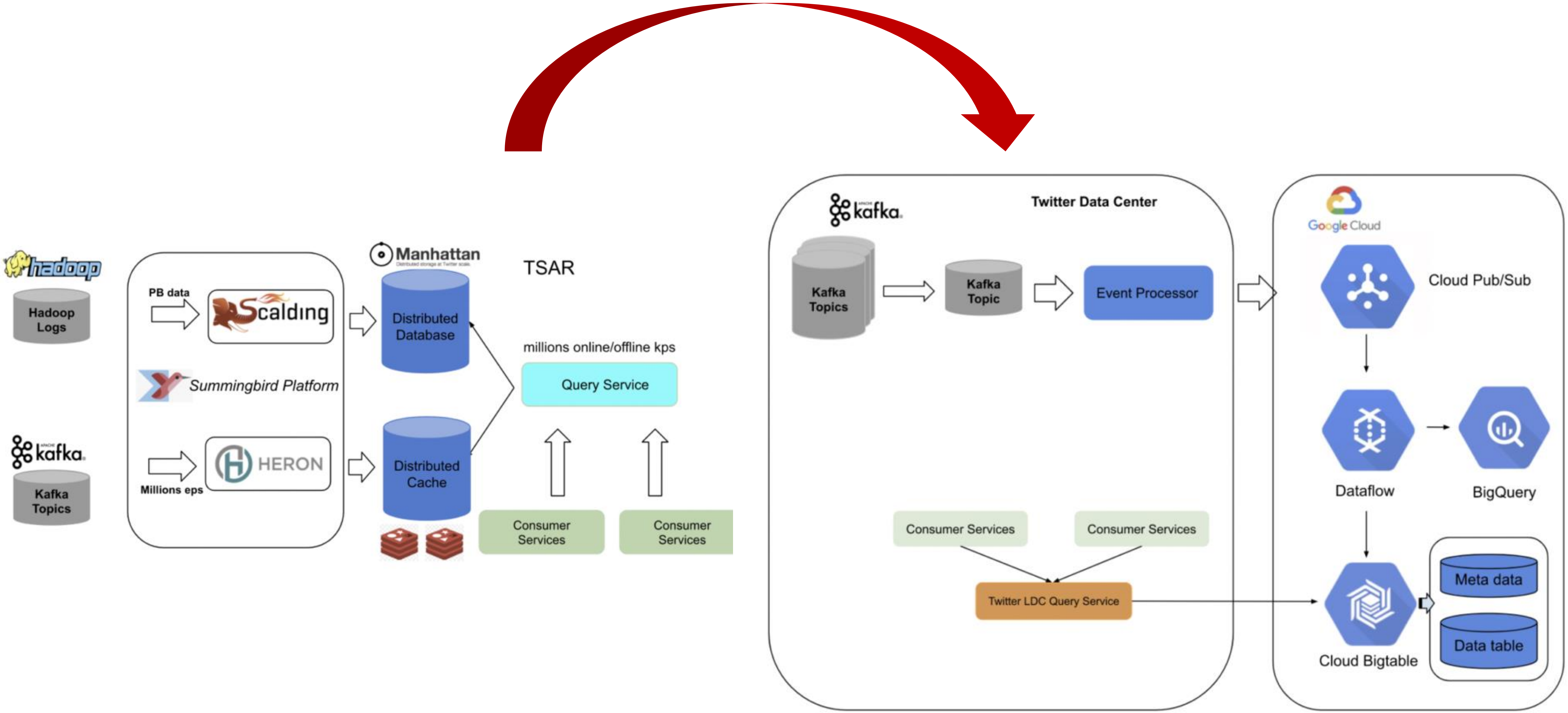


κ is gradually replacing λ

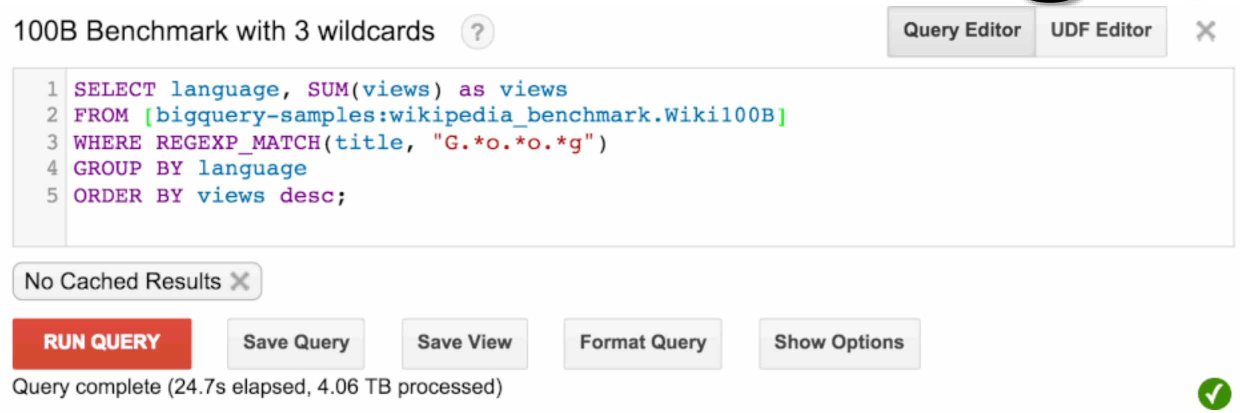


Attributes	Lambda	Kappa
Adoption	Easy. Exsiting ETL can be used	Complex. New system, new tech
Implementation	Simpler	Complex
Maintenance	Not easy. Maintaining 2 systems	Easier
Performance	Better	Event duplication, sequencing, etc.
Resource	Many required	Few required
Code Duplication	Yes, due to batch & stream	Mostly No
Use Case	Data is to be retained, History Data	Typically online

Twitter moving from λ to κ



Google Cloud Platform: BigQuery

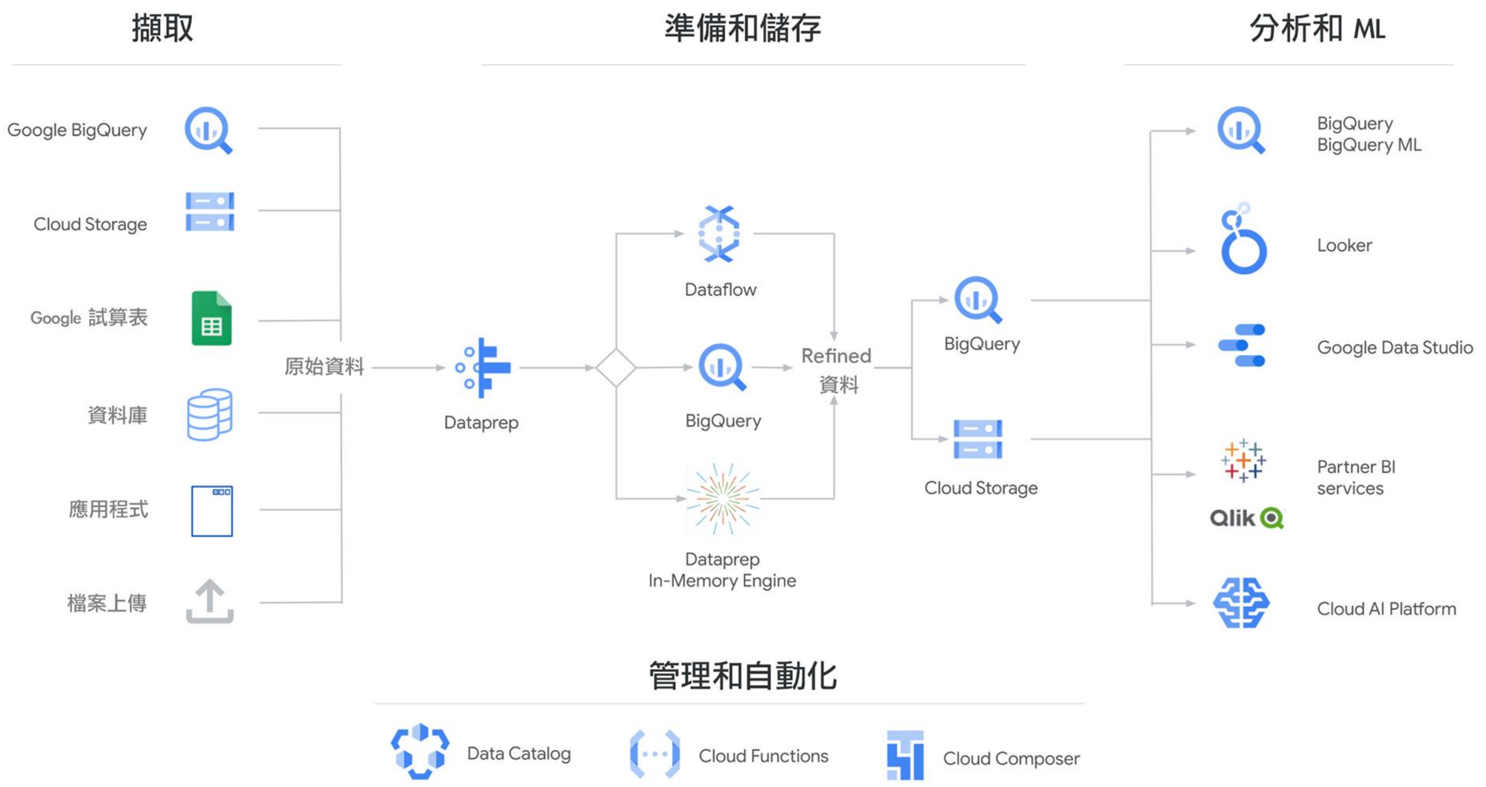


100B Benchmark with 3 wildcards

```
SELECT language, SUM(views) as views
FROM [bigquery-samples:wikipedia_benchmark.Wiki100B]
WHERE REGEXP_MATCH(title, "G.*o.*o.*g")
GROUP BY language
ORDER BY views desc;
```

No Cached Results ✕

Query complete (24.7s elapsed, 4.06 TB processed)



Topics for today

Asynchronous Execution

on one thread

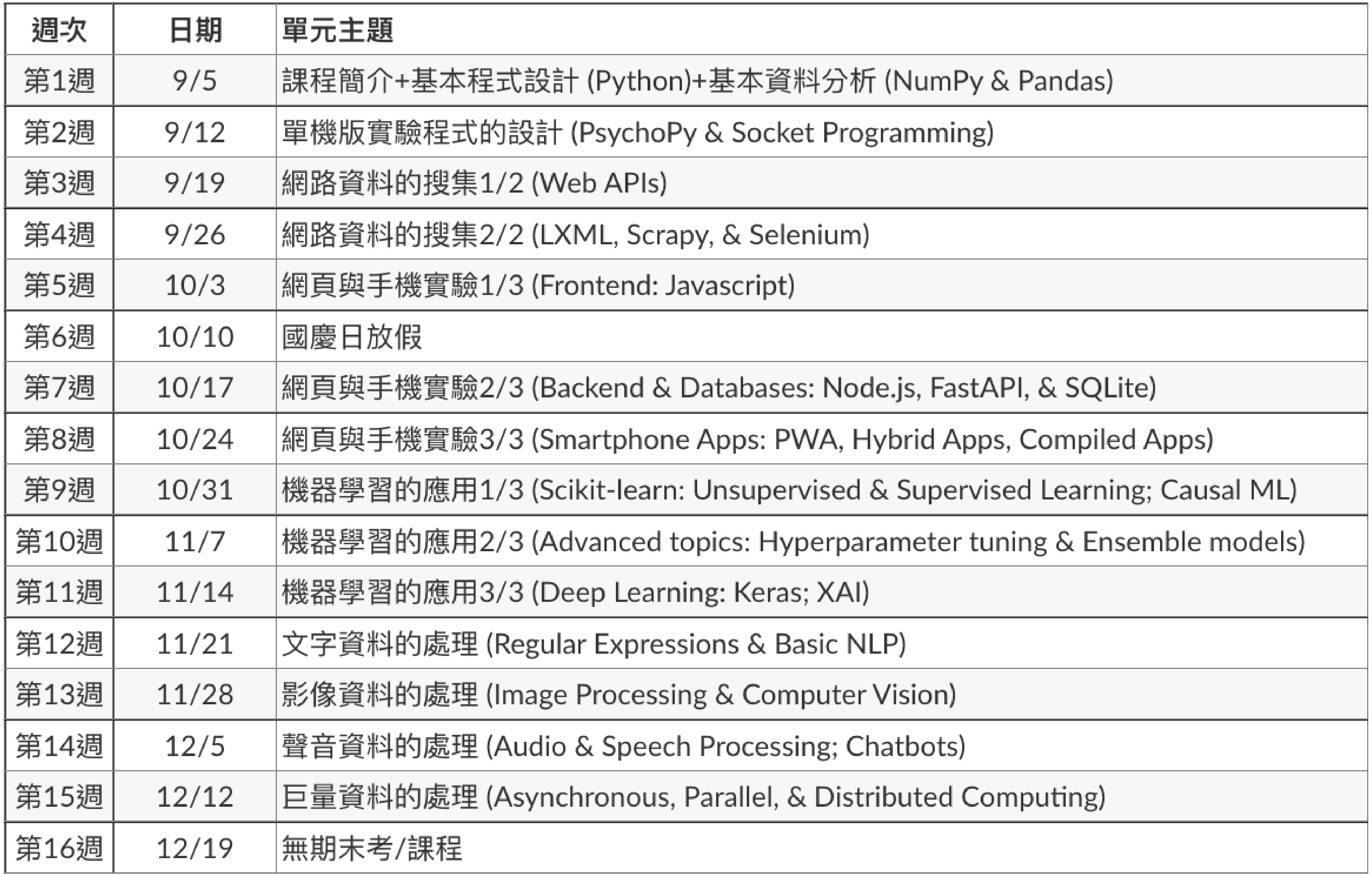
Parallel Computing

on one computer

Distributed Computing

across multiple computers

Reviewing the whole semester



週次	日期	單元主題
第1週	9/5	課程簡介+基本程式設計 (Python)+基本資料分析 (NumPy & Pandas)
第2週	9/12	單機版實驗程式的設計 (PsychoPy & Socket Programming)
第3週	9/19	網路資料的搜集1/2 (Web APIs)
第4週	9/26	網路資料的搜集2/2 (LXML, Scrapy, & Selenium)
第5週	10/3	網頁與手機實驗1/3 (Frontend: Javascript)
第6週	10/10	國慶日放假
第7週	10/17	網頁與手機實驗2/3 (Backend & Databases: Node.js, FastAPI, & SQLite)
第8週	10/24	網頁與手機實驗3/3 (Smartphone Apps: PWA, Hybrid Apps, Compiled Apps)
第9週	10/31	機器學習的應用1/3 (Scikit-learn: Unsupervised & Supervised Learning; Causal ML)
第10週	11/7	機器學習的應用2/3 (Advanced topics: Hyperparameter tuning & Ensemble models)
第11週	11/14	機器學習的應用3/3 (Deep Learning: Keras; XAI)
第12週	11/21	文字資料的處理 (Regular Expressions & Basic NLP)
第13週	11/28	影像資料的處理 (Image Processing & Computer Vision)
第14週	12/5	聲音資料的處理 (Audio & Speech Processing; Chatbots)
第15週	12/12	巨量資料的處理 (Asynchronous, Parallel, & Distributed Computing)
第16週	12/19	無期末考/課程