CS 7172
Parallel and Distributed Computation

Introduction

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Outline

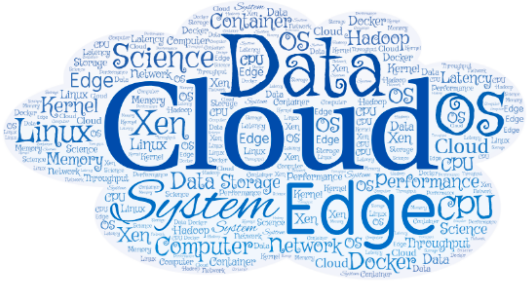
- Why study parallel and distributed computation?
- What to learn?
- Course structure
- Course policy
- An example of parallel and distributed computation

Self Introduction

- Kun Suo, Ph.D.
  - Homepage, https://kevinsuo.github.io/

- Research interests:
  - Cloud computing and virtualization;
  - Parallel and Distributed Computation, containers and kubernetes;
  - Software defined network (SDN) and network function virtualization (NFV)
  - Big data systems and machine learning systems

- Projects you may be interested in:
  - Several projects in Cloud & Data & Edge
  - https://kevinsuo.github.io/code-lab.html



Now it's your turn

- Name, program/year, where from
- Your interests in Computer Science
- Have you ever used or heard of distributed system? Can you name some of them?

https://www2.eecs.berkeley.edu/Research/Areas/CS/

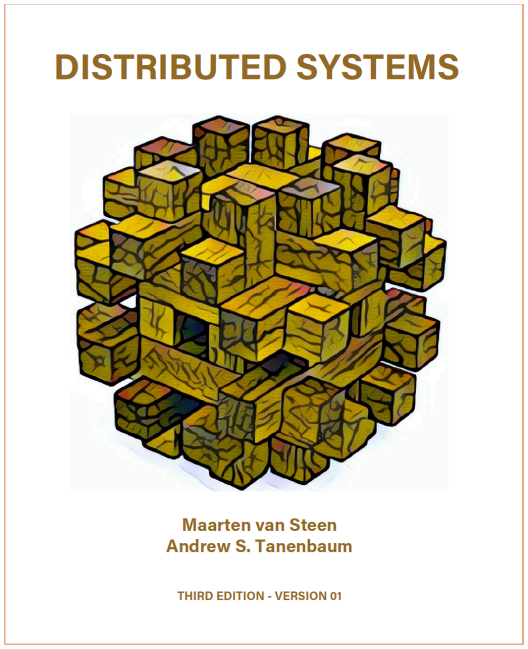
If you are in the online course, introduce yourself in D2L, Discussions → Self-Introduction

Course Information

- Instructor: Dr. Kun Suo
- Office: J-318
- Email: ksuo@kennesaw.edu
  - Only reply to e-mails that are sent from KSU student email accounts and title the course number [CS7172]
- Office Hours:
  - T/Th, 3pm-4pm
  - By appointment
- Course Materials
  - Homework assignments, lecture slides, and other materials will be posted in the webpage (https://kevinsuo.github.io/teaching/2020Spring/7172/class.html) and D2L.

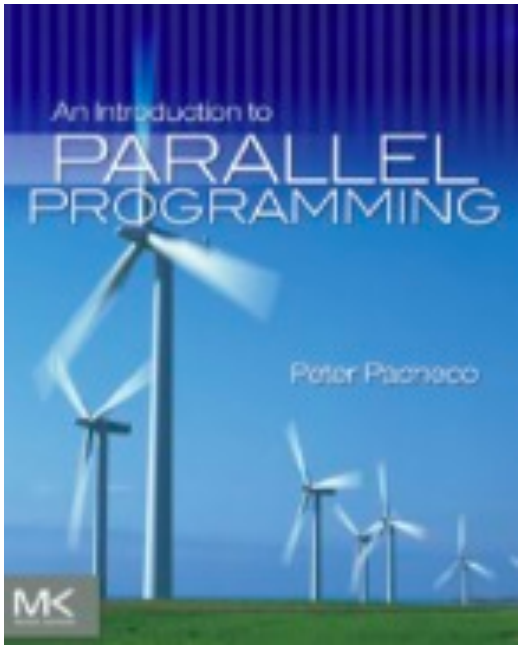
Reference Book

- "Distributed Systems 3rd edition (2017)" by M. van Steen and A.S. Tanenbaum:
  - ISBN-13: 978-1543057386
  - You can get a digital copy of this book for free: https://www.distributed-systems.net/index.php/books/distributed-systems-3rd-edition-2017/



Reference Book

- "An Introduction to Parallel Programming (2011)" by Peter S. Pacheco:



  - ISBN-13: 978-0-12-374260-5
  - You can get a digital copy of this book for free: http://www.e-tahtam.com/~turgaybilgin/2013-2014-guz/ParalelProgramlama/ParallelProg.pdf

Prerequisites

- Computer basics that are supposed to covered in *(CS 7172) Parallel and Distributed Computation* and *(CS 3503) Computer Organization and Architecture* course.

- C programming (code reading, kernel development and debugging). (Famous projects in C)

- Linux command line environment (compiling, Makefile, debugging, simple shell programming).

For C and Linux beginners

- C tutorial
  - https://www.tutorialspoint.com/cprogramming/
  - https://www.learn-c.org
  - https://www.cprogramming.com/tutorial/c-tutorial.html

- Linux tutorial
  - https://ryanstutorials.net/linuxtutorial/
  - http://www.ee.surrey.ac.uk/Teaching/Unix/
  - https://www.tutorialspoint.com/unix/

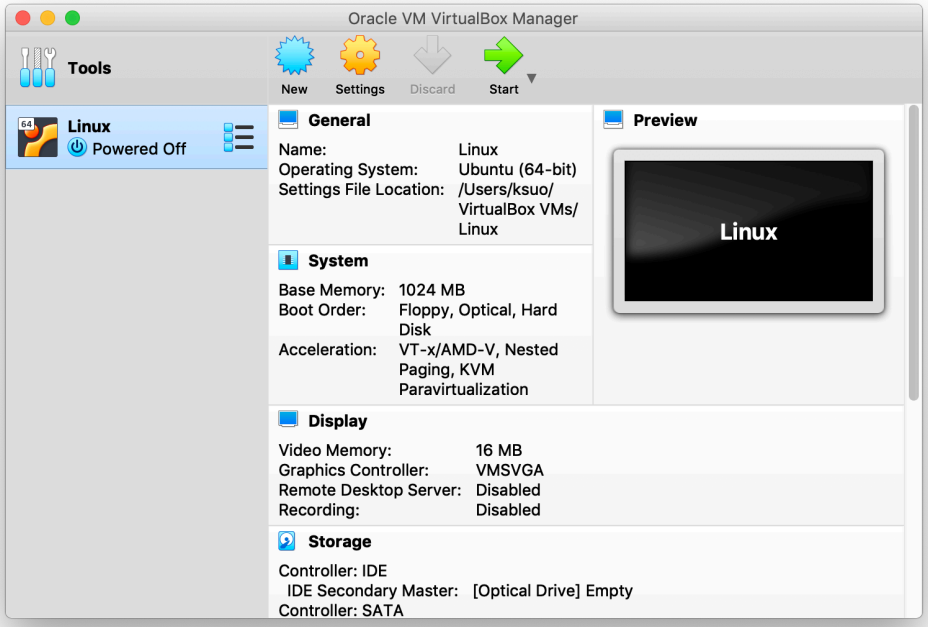
Project Environment

- Recommend project environment
  - VirtualBox + Ubuntu + Linux

Virtual machine — VM OS — VM OS Kernel

https://www.virtualbox.org/

https://ubuntu.com/download/desktop



Project Environment

- Recommend project environment
  - VirtualBox + Ubuntu + Linux

https://www.virtualbox.org/

https://ubuntu.com/download/desktop

- New to VirtualBox?
  - https://oracle-base.com/articles/vm/virtualbox-creating-a-new-vm
  - https://www.youtube.com/watch?v=sB_5fqiysi4

Why study parallel and distributed computation?

- Most computer systems today are a certain form of distributed systems
  - Internet, datacenters, super computers, mobile devices
- To learn useful techniques to build large systems
  - A system with 10,000 nodes is different from one with 100 nodes
- How to deal with imperfections
  - Machines can fail; network is slow; topology is not flat

What to learn

- Library (Pthread, OpenMP, MPI)
- Architectures
- Processes
- Communication
- Naming
- Synchronization
- Consistency and replication
- Fault tolerance and reliability
- Distributed file systems

Expected Outcomes

- Familiar with popular parallel programming libraries (Pthread, OpenMP, MPI)
- Familiar with fundamentals of distributed systems
- The ability to
  - Evaluate the performance of distributed systems
  - Write simple distributed programs
  - Understand the tradeoffs in distributed system design

Course Structure

- Lectures
  - M/W 8:00 PM – 9:15 PM
  - J-132
- One paper presentation
  - Second half term
- Projects
  - 3 programming assignments
- Exams
  - Midterm: in class, TBA.
  - Final: in class, TBA

Course Policy

- Grading scale

Percentage	Grade
90 - 100	A
80 - 89	B
70 - 79	C
60 - 69	D
Below 60	F

Grading Policy (cont.)

- Grading percentage
  - In-class discussion and attendance: 5%
  - Paper presentation: 15%
  - Projects (x3): 30%
  - Midterm: 20%
  - Final exam: 30%

Late submission policy: late submission will **not be accepted** and **no credits**

Academic Integrity

- Academic dishonesty
  - Cheating
  - Plagiarism
  - Collusion
  - The submission for credit of any work or materials that are attributable in whole or in part to another person
  - Taking an examination for another person
  - Any act designed to give unfair advantage to a student or the attempt to commit

Where to go for help ?

- Ask questions in class
- Ask questions outside class
  - Classmates and friends
- Attend office hours
  - Dr. Kun Suo: Tuesday/Thursday 3:00PM – 4:00PM, J-318
- Search on the web
  - Stand on the shoulder of giants

An example of parallel and distributed computation: Matrix multiply

$$\begin{array}{c} \\ \vec{a}_1 \rightarrow \\ \vec{a}_2 \rightarrow \\ \\ A \end{array} \begin{bmatrix} 1 & 7 \\ 2 & 4 \end{bmatrix} \cdot \begin{array}{c} \vec{b}_1 \quad \vec{b}_2 \\ \downarrow \quad \downarrow \\ \begin{bmatrix} 3 & 3 \\ 5 & 2 \end{bmatrix} \\ B \end{array} = \begin{array}{c} \\ \\ \\ C \end{array} \begin{bmatrix} \vec{a}_1 \cdot \vec{b}_1 & \vec{a}_1 \cdot \vec{b}_2 \\ \vec{a}_2 \cdot \vec{b}_1 & \vec{a}_2 \cdot \vec{b}_2 \end{bmatrix}$$

Matrix multiply

https://github.com/kevinsuo/CS7172/blob/master/matrix.c

```
int main()
{
        initMatrix();

        double time_spent = 0.0;
        clock_t begin = clock();

        matrixMultiply();

        clock_t end = clock();
        time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
        printf("Time elpased is %f seconds", time_spent);

        return 0;
}
```

Matrix multiply

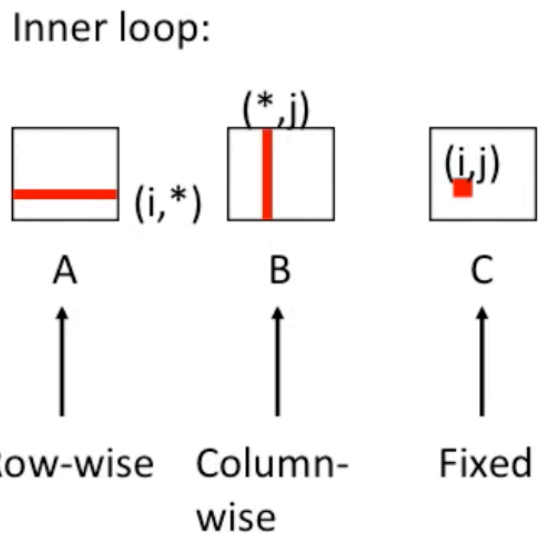
https://github.com/kevinsuo/CS7172/blob/master/matrix.c

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#define N 1000

double A[N][N], B[N][N], C[N][N];

void initMatrix()
{
        int i, j = 0;
        for (i = 0; i < N; i++) {
                for (j = 0; j < N; j++) {
                        A[i][j] = rand() % 100 + 1;  //generate a number between [1, 100]
                        B[i][j] = rand() % 100 + 1;  //generate a number between [1, 100]
                }
        }
}
```

Matrix multiply

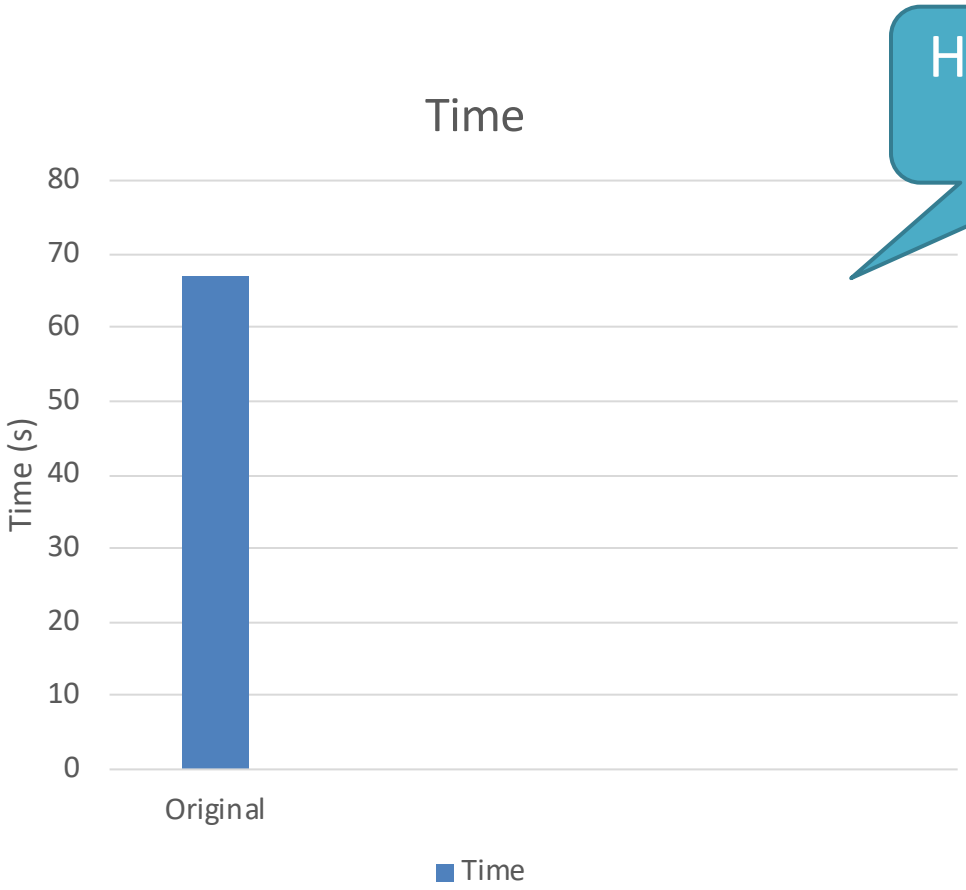
https://github.com/kevinsuo/CS7172/blob/master/matrix.c

```
void matrixMultiply() {
        int i, j, k = 0;
        for (i = 0; i < N; i++) {
                for (j = 0; j < N; j++) {
                        for (k = 0; k < N; k++) {
                                C[i][j] = A[i][k] * B[k][j];
                        }
                }
        }
}
```



Matrix multiply

https://github.com/kevinsuo/CS7172/blob/master/matrix.c



How to run it faster?

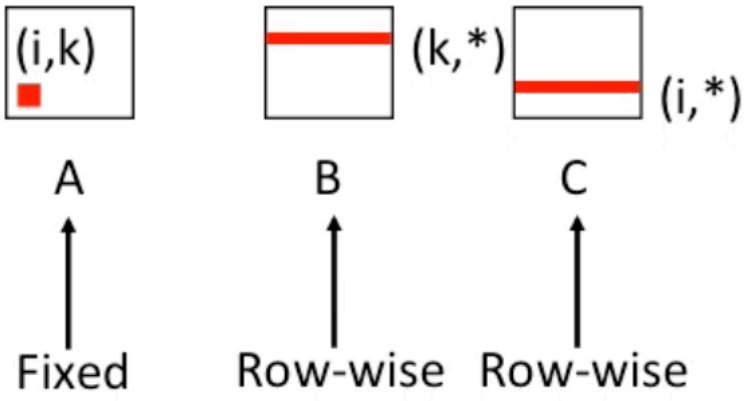
1. Accelerate serial execution

2. Accelerate in parallel

Option 1: Optimization using locality

```
void matrixMultiply() {
        int i, j, k = 0;
        for (k = 0; k < N; k++) {
                for (i = 0; i < N; i++) {
                        for (j = 0; j < N; j++) {
                                C[i][j] = A[i][k] * B[k][j];
                        }
                }
        }
}
```

Inner loop:



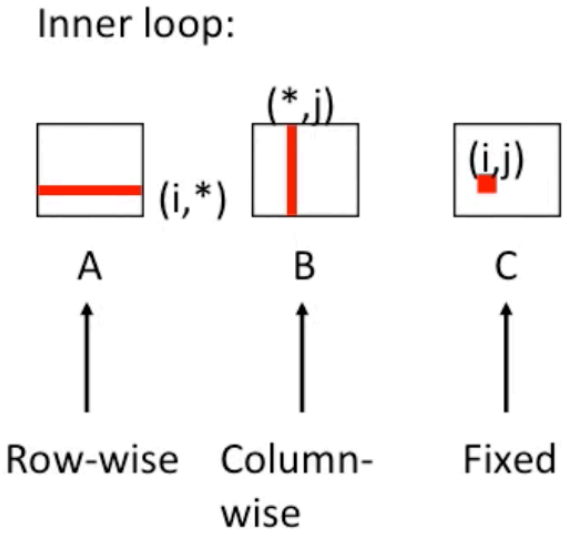
https://github.com/kevinsuo/CS7172/blob/master/matrix-opt.c

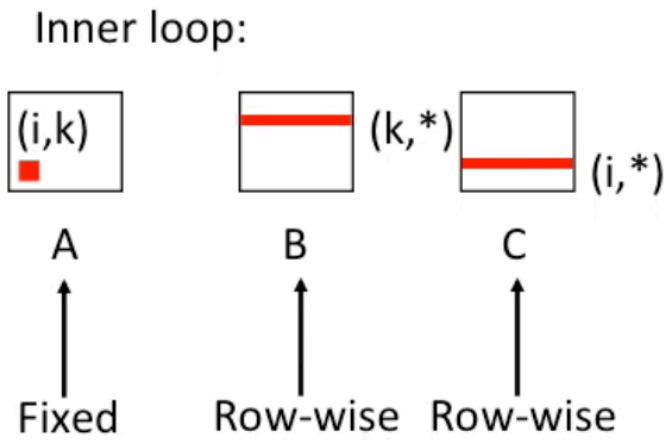
Option 1: Optimization using locality

```
ksuo@ksuo-VirtualBox ~/cs7172> ./a.o
Time elpased is 67.452589 seconds
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172> ./a2.o
Time elpased is 18.149353 seconds
```

N=2000

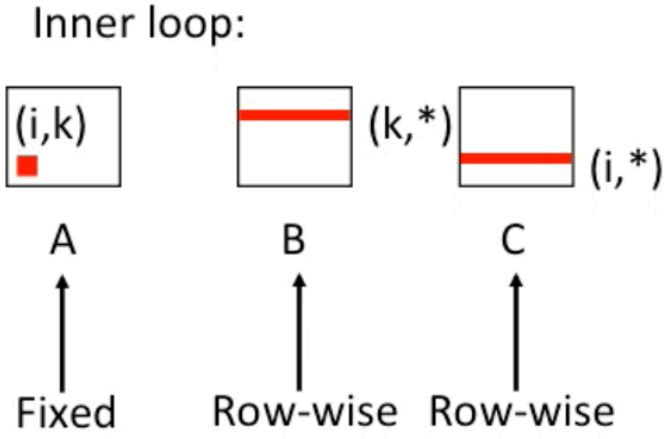
3.7x



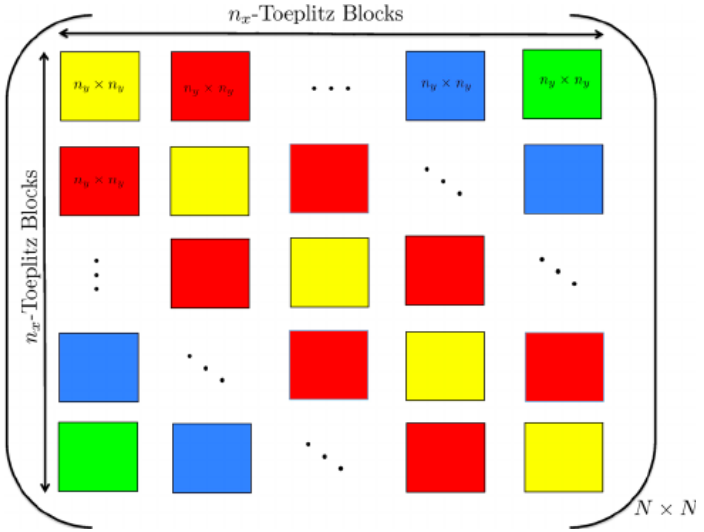


Option 1: Optimization using locality

- Temporal locality
  - Every inner loop reuse the value of A[i, k]



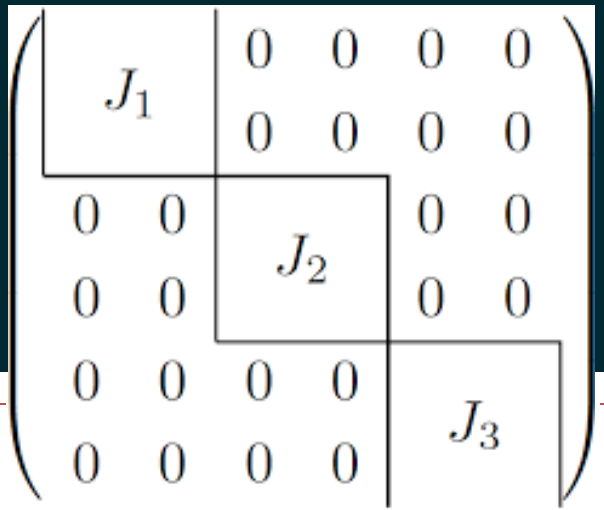
- Spatial locality
  - Divide the large matrix into smaller ones and put it inside the cache during calculation



Option 1: Optimization using locality

```
void matrixMultiply() {
        int i, j, k = 0;
        int i2, j2, k2 = 0;

        for (k2 = 0; k2 < N; k2+=BLOCK_SIZE) {
                for (i2 = 0; i2 < N; i2+=BLOCK_SIZE) {
                        for (j2 = 0; j2 < N; j2+=BLOCK_SIZE) {
                                //inside each block
                                for (k = k2; k < k2+BLOCK_SIZE; k++) {
                                        for (i = i2; i < i2+BLOCK_SIZE; i++) {
                                                for (j = j2; j < j2+BLOCK_SIZE; j++) {
                                                        C[i][j] = A[i][k] * B[k][j];
                                                }
                                        }
                                }

                        }
                }
        }
}
```

https://github.com/kevinsuo/CS7172/blob/master/matrix-opt2.c



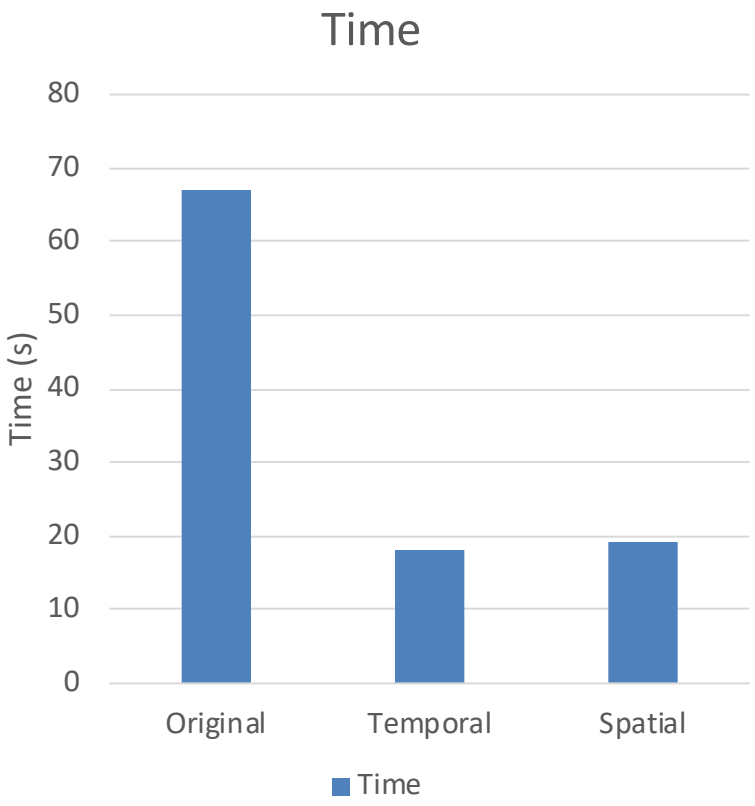
Option 1: Optimization using locality

$$A = \left(\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right) \Longrightarrow \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

$$A_{11} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, A_{12} = \begin{pmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{pmatrix}$$

$$A_{21} = \begin{pmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{pmatrix}, A_{22} = \begin{pmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{pmatrix}$$

Option 1: Optimization using locality




```
ksuo@ksuo-VirtualBox ~/cs7172> ./a.o
Time elpased is 67.845517 seconds
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172>
ksuo@ksuo-VirtualBox ~/cs7172> ./a3.o
Time elpased is 19.115410 seconds
```

Optimal 2: Optimization using parallel

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \cdot \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

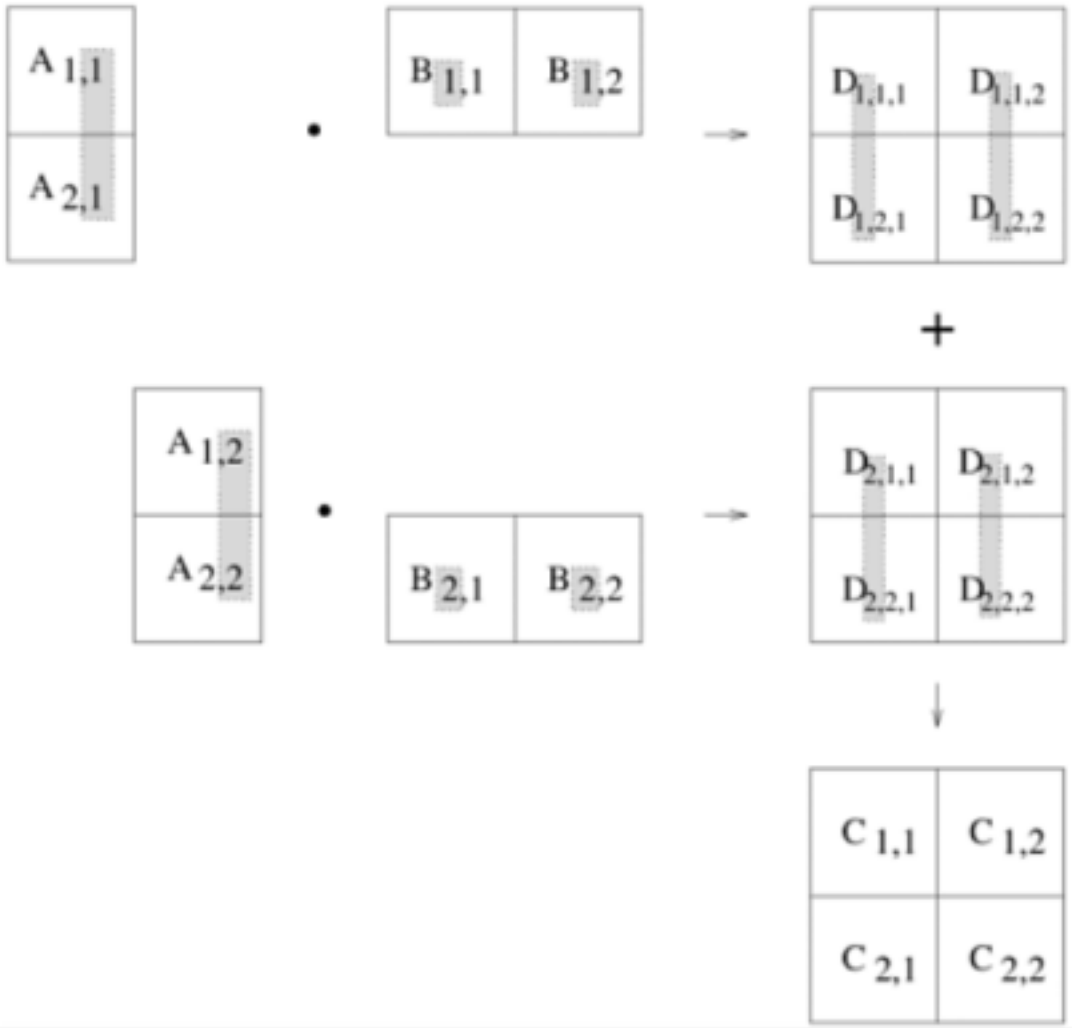
(a)

Task 1: $C_{1,1} = A_{1,1}B_{1,1} + A_{1,2}B_{2,1}$
Task 2: $C_{1,2} = A_{1,1}B_{1,2} + A_{1,2}B_{2,2}$
Task 3: $C_{2,1} = A_{2,1}B_{1,1} + A_{2,2}B_{2,1}$
Task 4: $C_{2,2} = A_{2,1}B_{1,2} + A_{2,2}B_{2,2}$

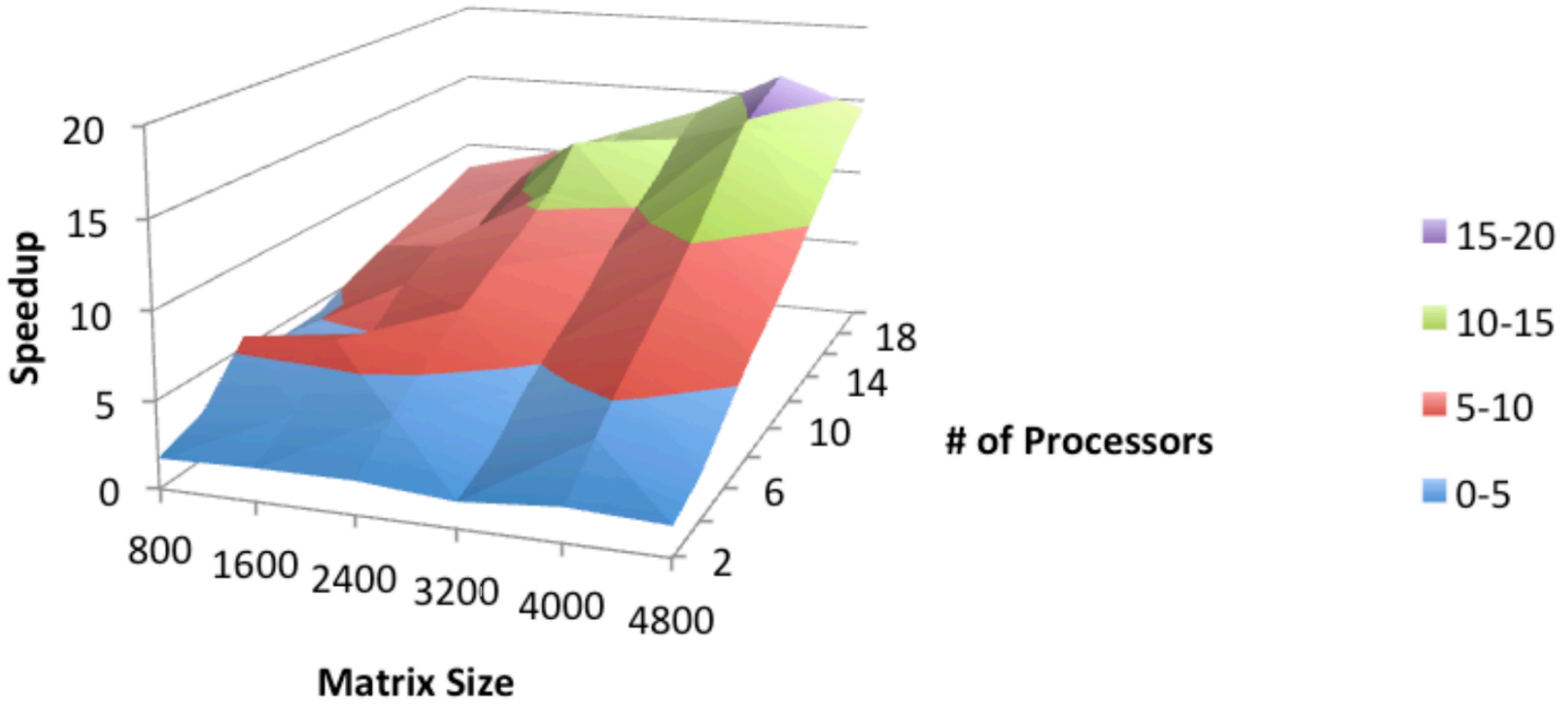
Optimal 2: Optimization using parallel

Thread 1:

Thread 2:



Optimal 2: Optimization using parallel



https://www.cse.unr.edu/~fredh/class/415/Nolan/matrix_multiplication/writeup.pdf

Example of distributed system: sorting

- Sorting on a single machine, e.g., Database

```
select field_a from table_b order by field_a limit 100, 10;
```

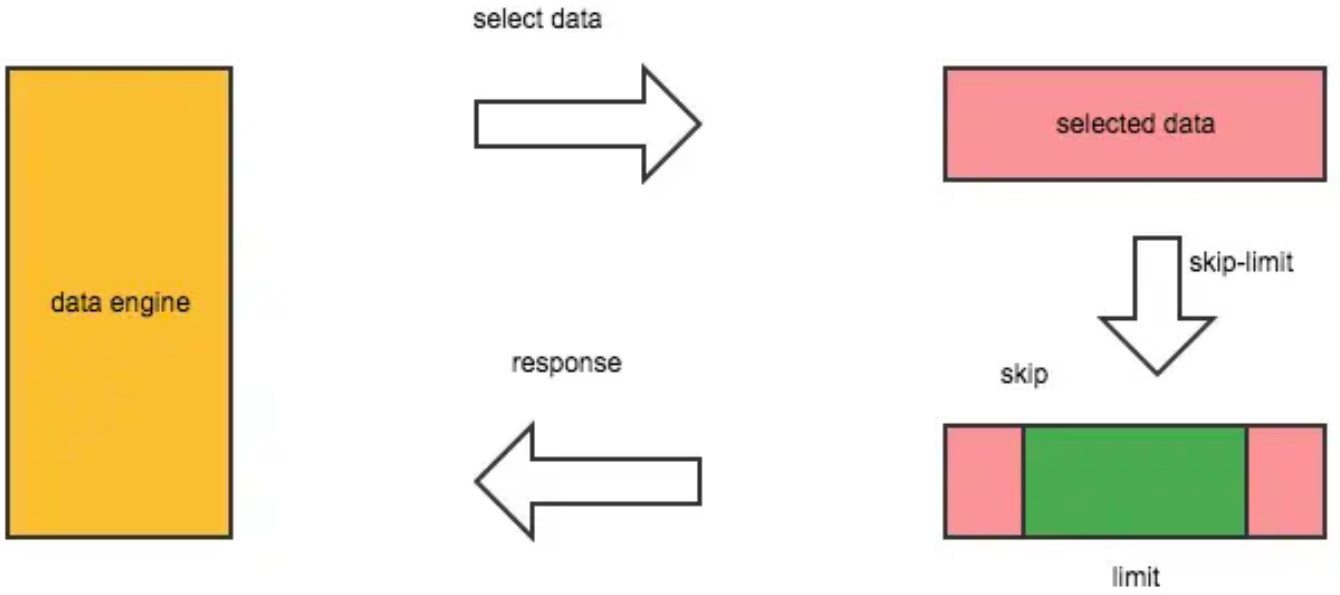
```
db.collection_b
.find()
.sort({"field_a":1})
.skip(100)
.limit(10);
```

	field_a	field_b	field_c
100			
…			
…			
110			

From line 100 to the next 10 lines of data

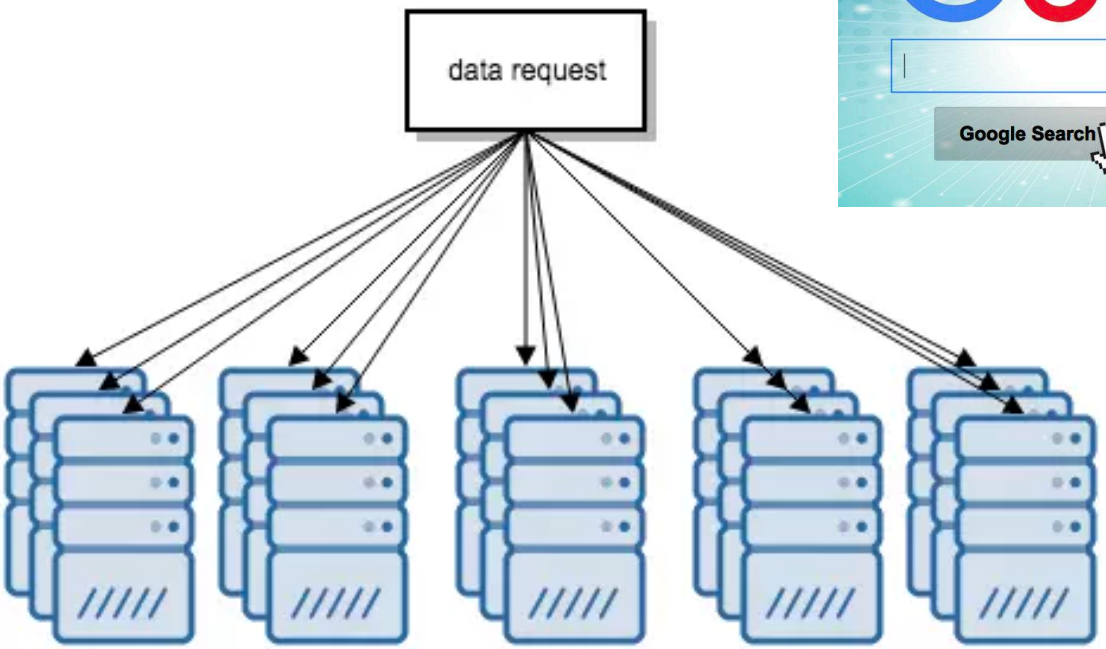
Example of distributed system: sorting

- Workflow



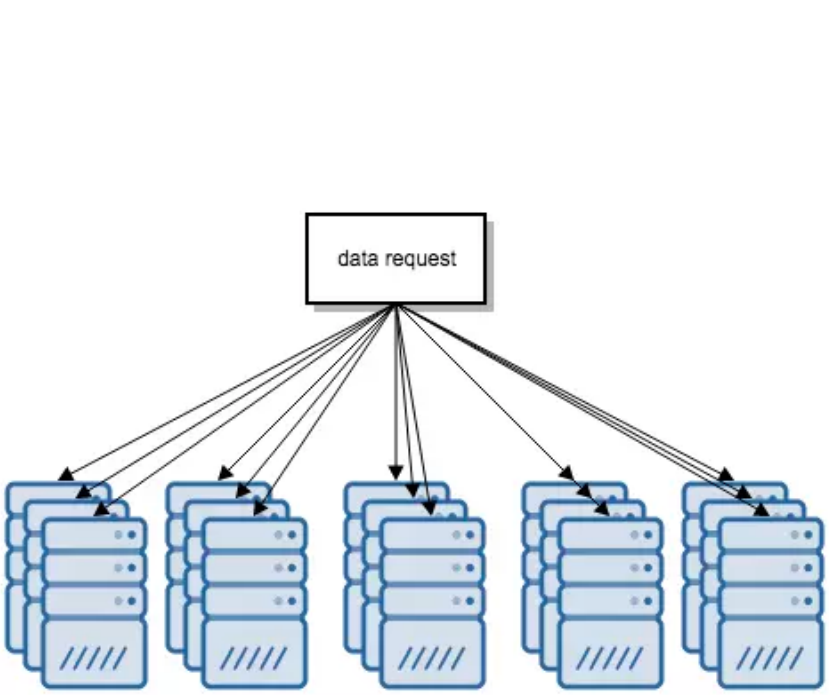
Example of distributed system: sorting

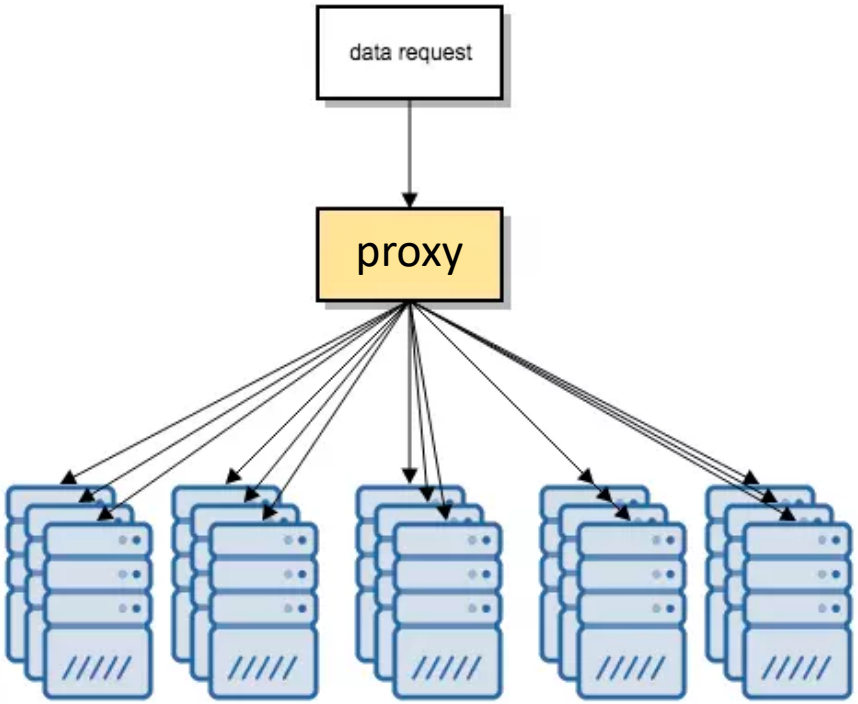
- If the data is too much and single node cannot hold



Example of distributed system: sorting

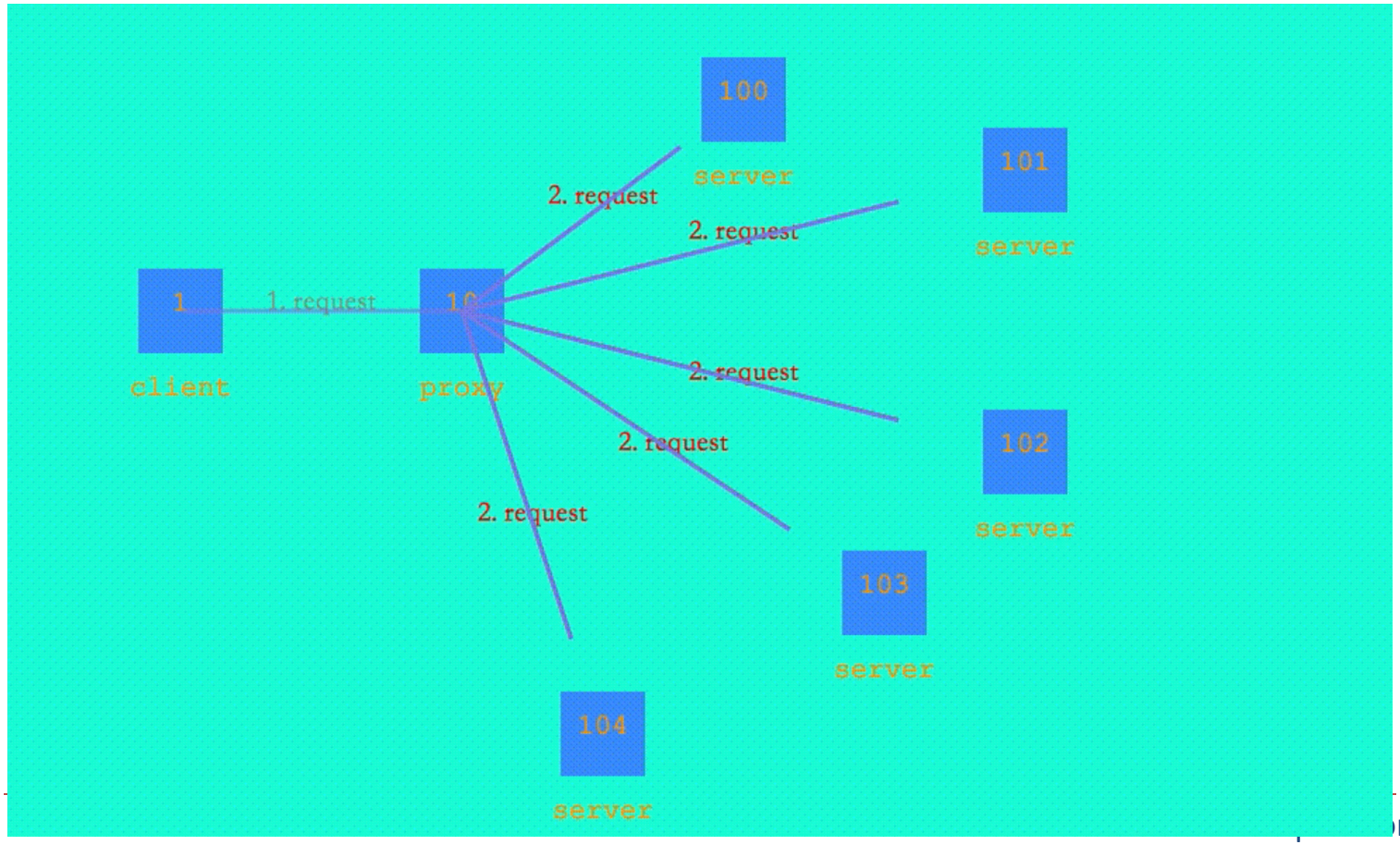
- Choose a node for merge processing





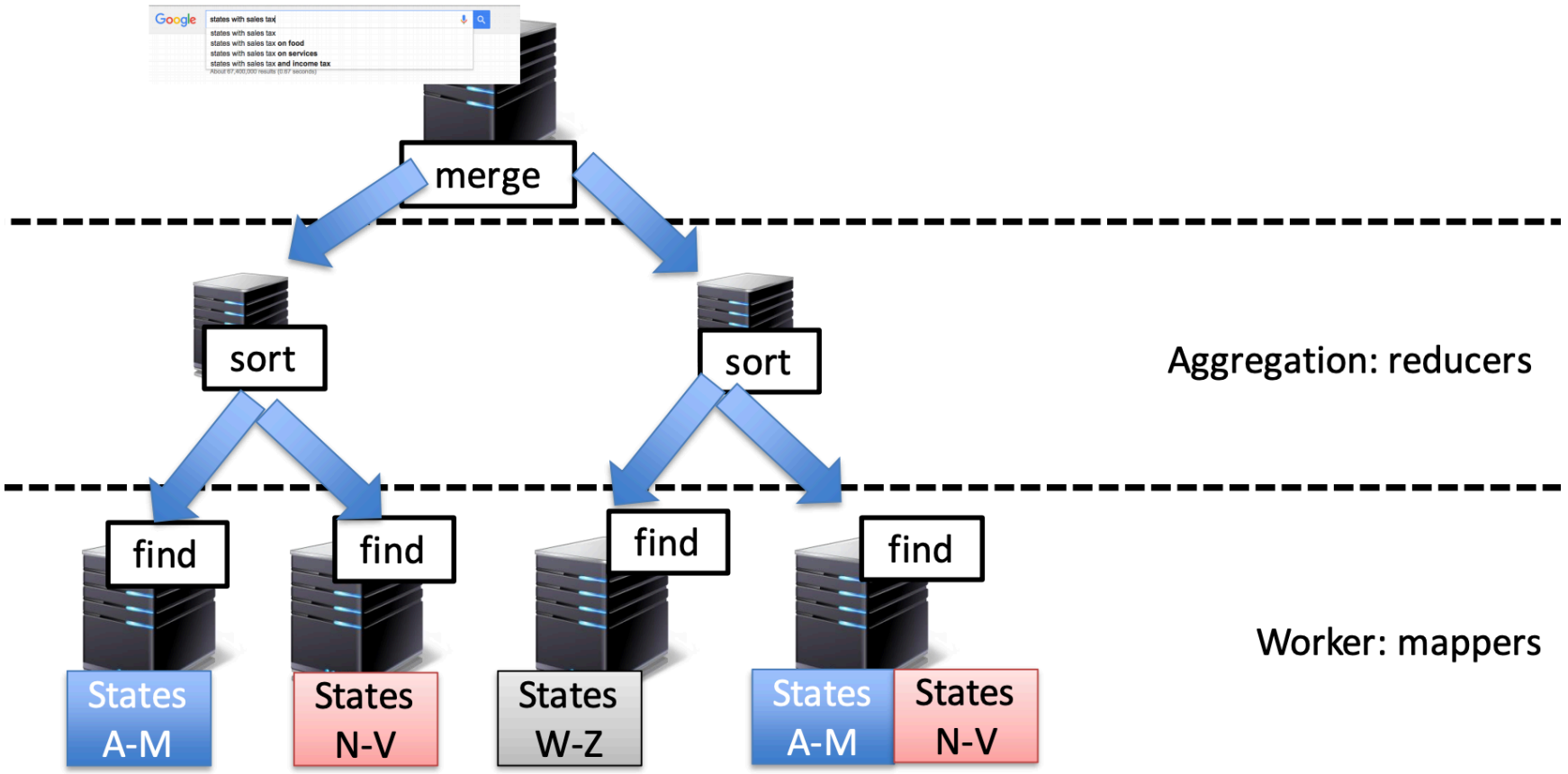
Example of distributed system: sorting

- Workflow



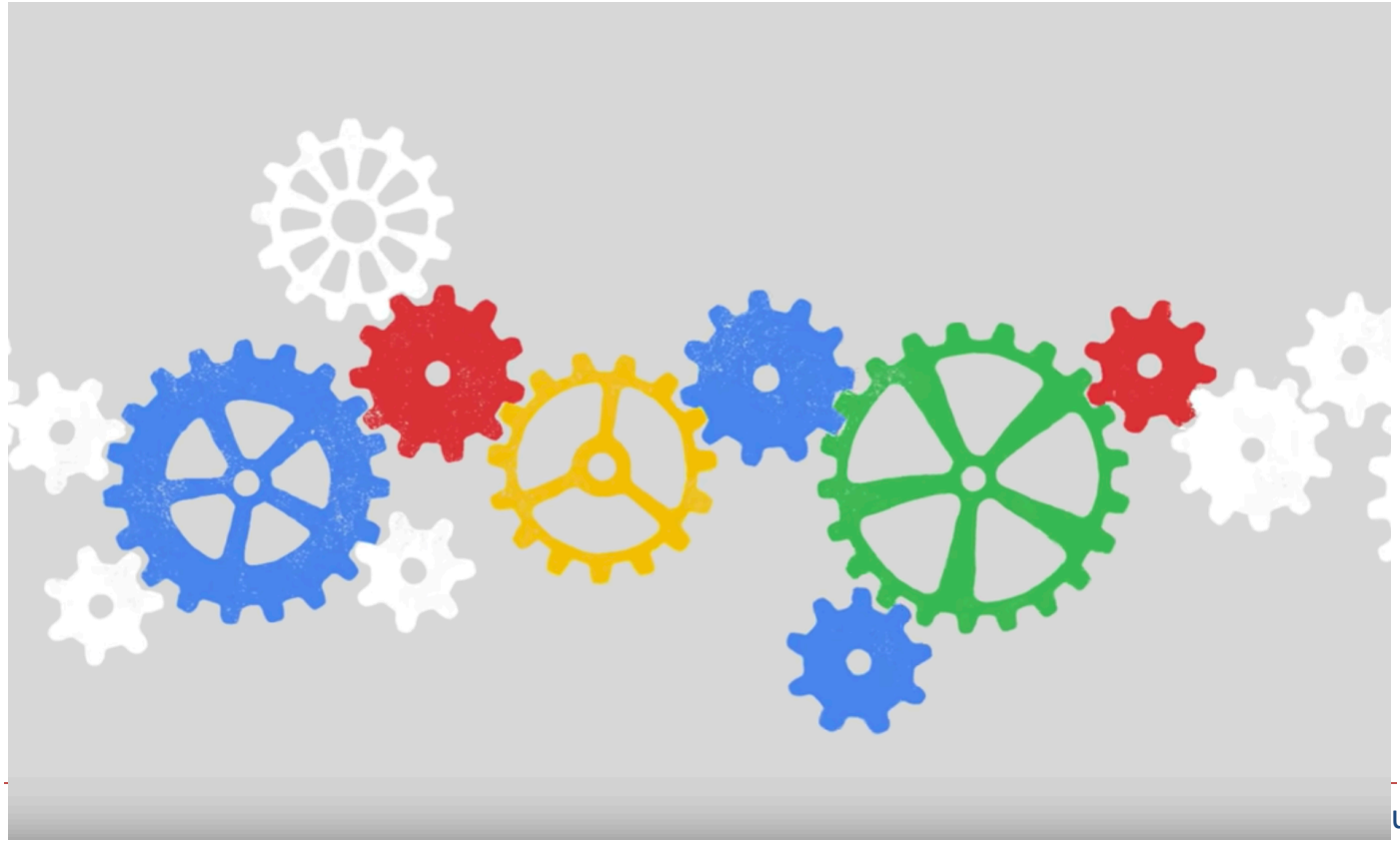
Example of distributed system: sorting

- How Do Request Get Processed in a Data Center



Example of distributed system: sorting

- How Google Search Works
- https://www.youtube.com/watch?v=0eKVizvYSUQ



Conclusion

- Why study parallel and distributed computation?
- What to learn?
- Course structure
- Course policy
- An example of parallel and distributed computation