# ICME Fundamentals of Data Science

# Introduction to Parallel Computing

| | | | |
|---|---|---|---|
| **1) Algorithms** | Lecture | 9:00 | 10:10 |
| | Problem solving and Q&A | 10:10 | 10:20 |
| | **Break** | **10:20** | **10:30** |
| **2) Shared Memory** | Lecture | 10:30 | 11:45 |
| | Problem solving and Q&A | 11:45 | 12:00 |
| | **Lunch** | **12:00 pm** | **1:15** |
| | **Q&A** | **1:15** | **1:30** |
| **3) Distributed Memory** | Lecture | 1:30 | 2:40 |
| | Problem solving and Q&A | 2:40 | 2:50 |
| | **Break** | **2:50** | **3:00** |
| **4) Spark** | Lecture | 3:00 | 4:15 |
| | Problem solving and Q&A | 4:15 | 4:25 |
| | **Final Remarks** | **4:25** | **4:45** |

ICME Stanford Institute for Computational & Mathematical Engineering
SUMMER WORKSHOPS
FUNDAMENTALS OF DATA SCIENCE

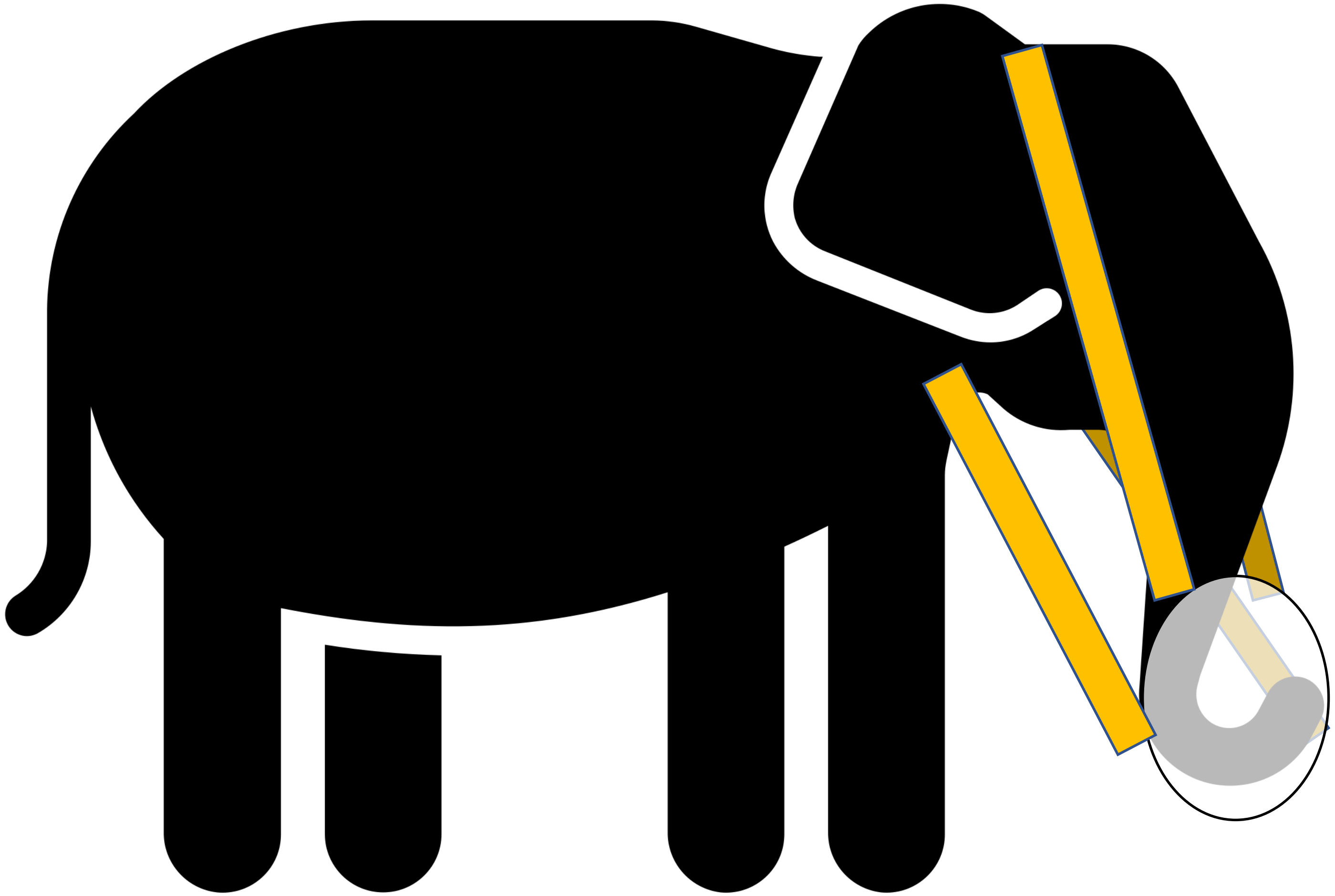
ICME

# Introduction to
# High Performance Computing

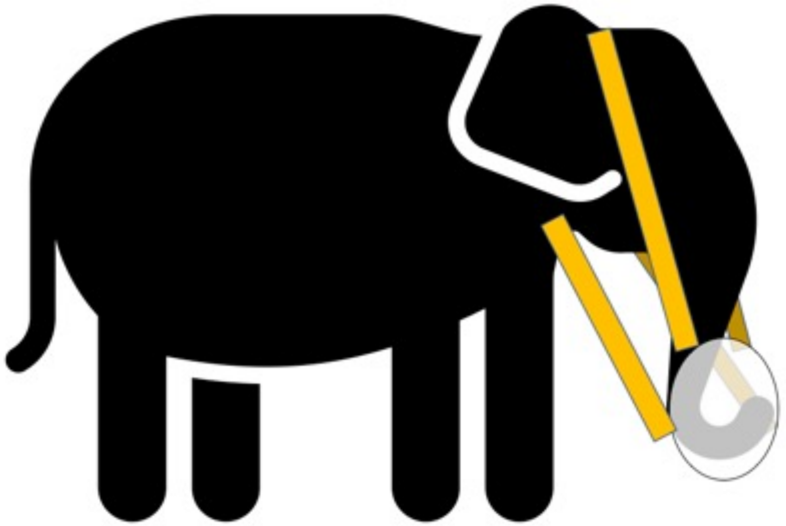
ICME Summer Workshop : Fundamentals of Data Science

Cindy C. Orozco Bohorquez

# First, the elephant in the room…



# First, the elephant in the room…



COVID-19 emergency

Synchronous vs. Asynchronous

Time Zones

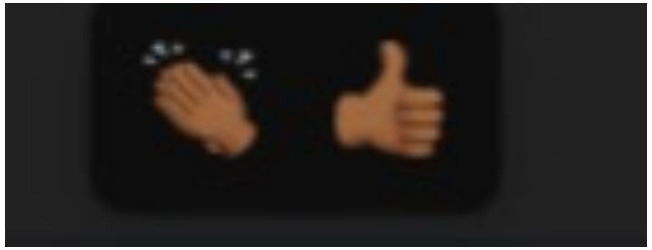
Responsibilities and conditions at home

Thank you!

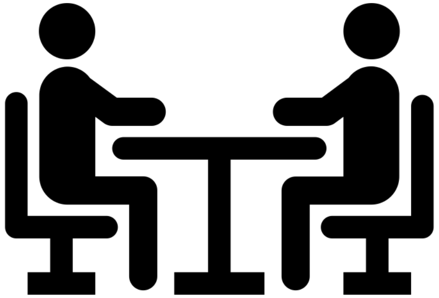
New experiment together

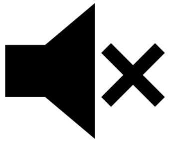
# Zoom etiquette

Zoom Reactions



Chat

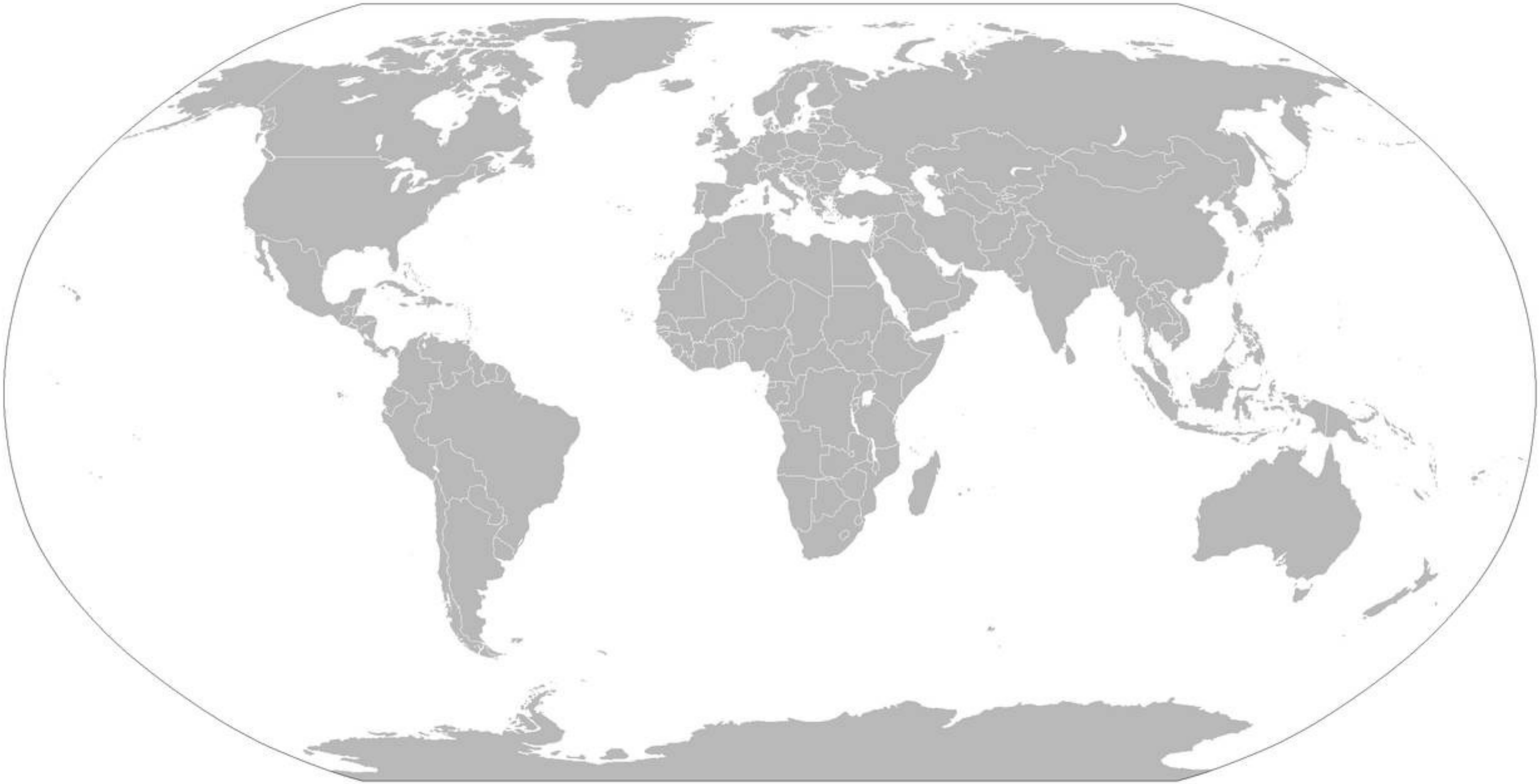


Sound 

Video 

# Getting to know each other

# From where are you taking this workshop?



# What is exciting about parallel computing?

# A bit about me …

- Cindy Orozco (orozcocc@stanford.edu)
- 6th year ICME PhD student

Numerical Analysis
(Engineering Simulations)

Data Science
(Learning from Data)


# Our TA today… Rahul Sarkar

ICME Ph.D. Student

Current Research Interests:

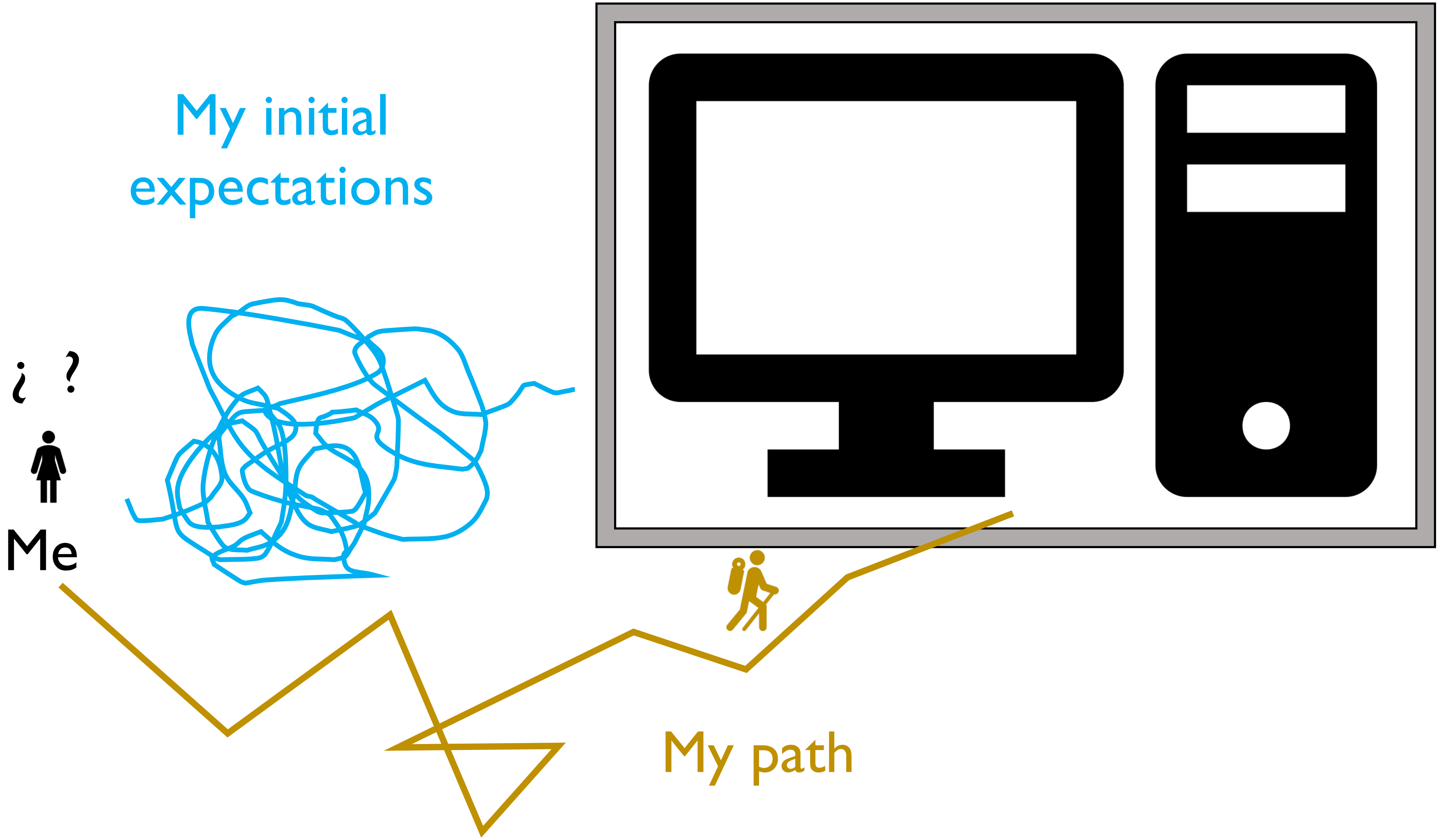
- Inverse Problems
- Mathematical Analysis
- Quantum Computing

**Contact**: rsarkar at stanford dot edu

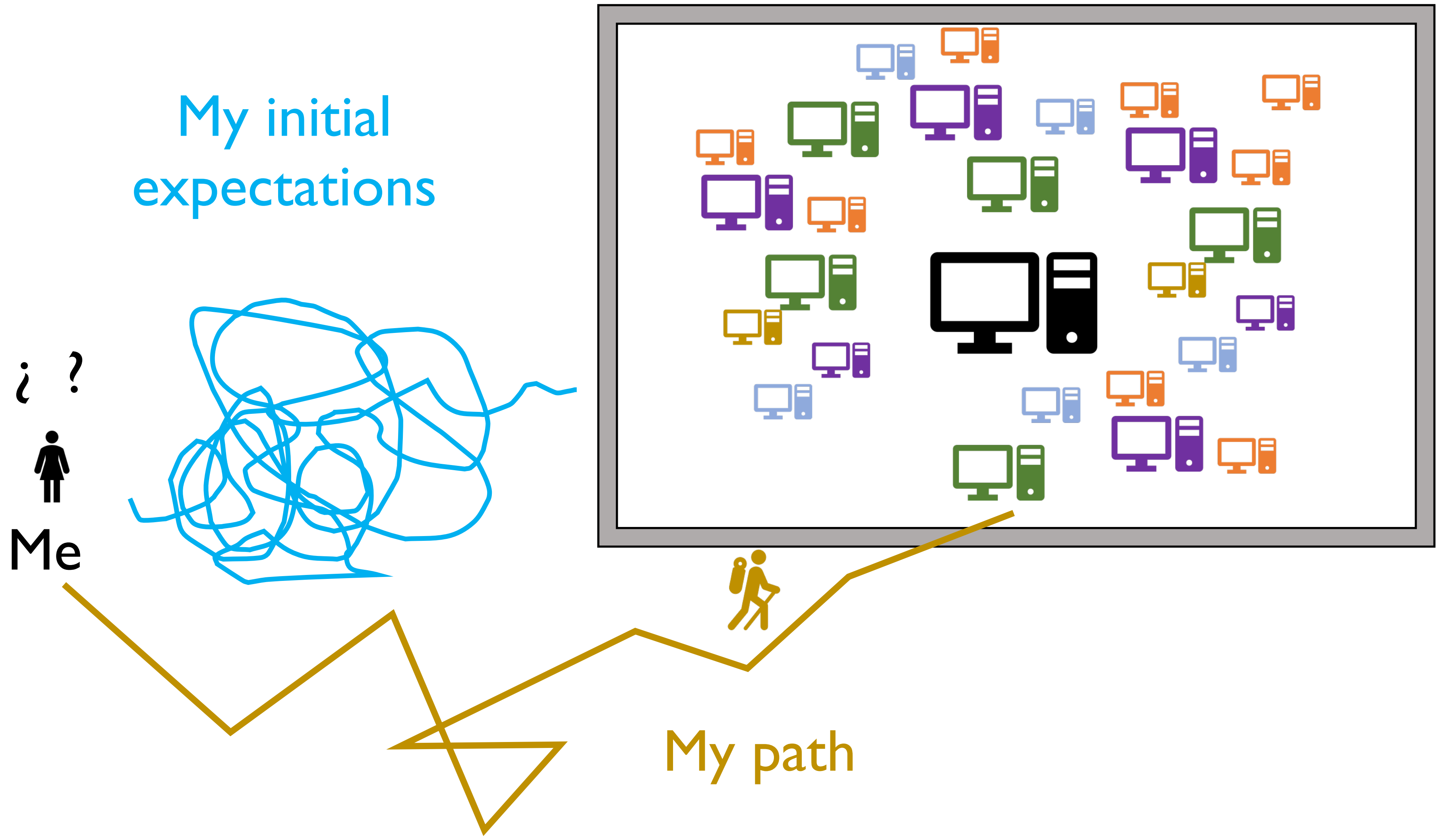
**Website**:

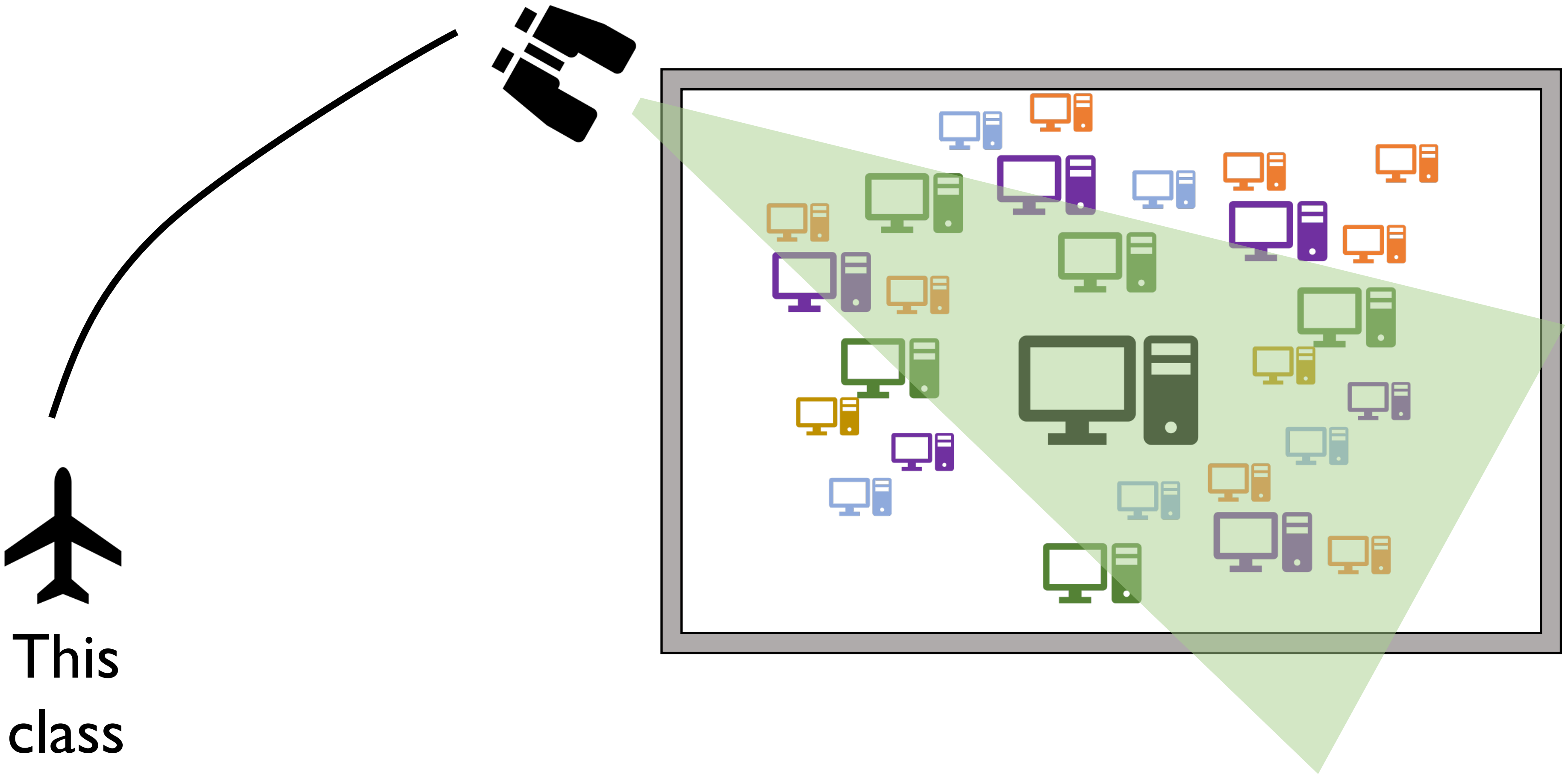
http://web.stanford.edu/~rsarkar/

# HPC = The largest super computer



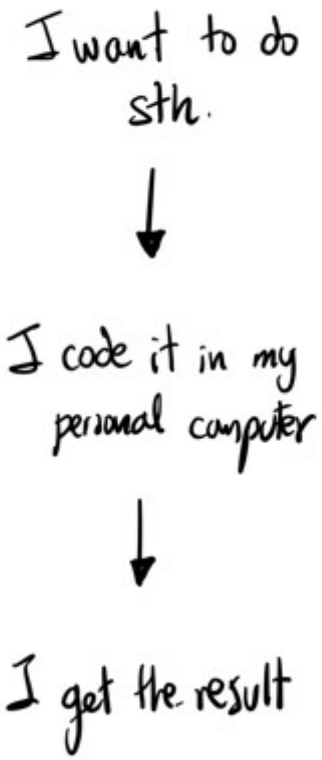
# HPC = The largest super computer… not quite



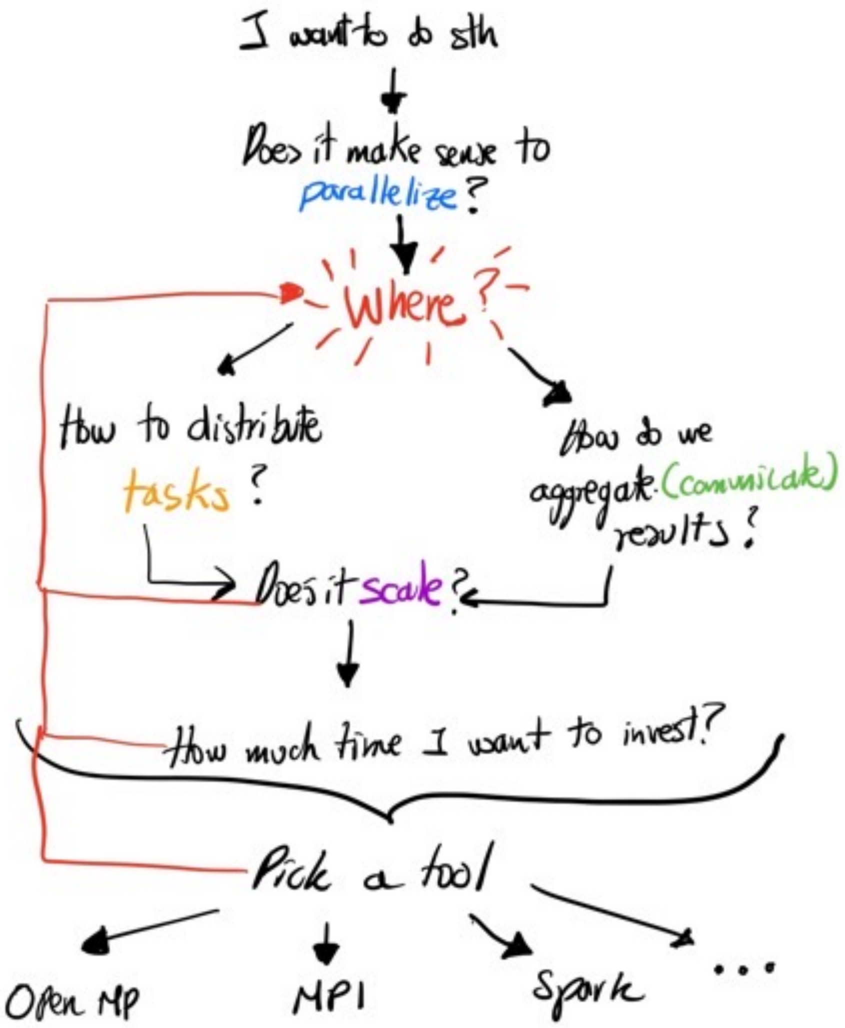

This
class

# What to expect at the end of today?

## Sequential




## Parallel



# What to expect at the end of today?

Parallel programming is an art. It requires a masterful combination of hardware knowledge, parallelization types and existing tools.

1) How do we think to parallelize something?

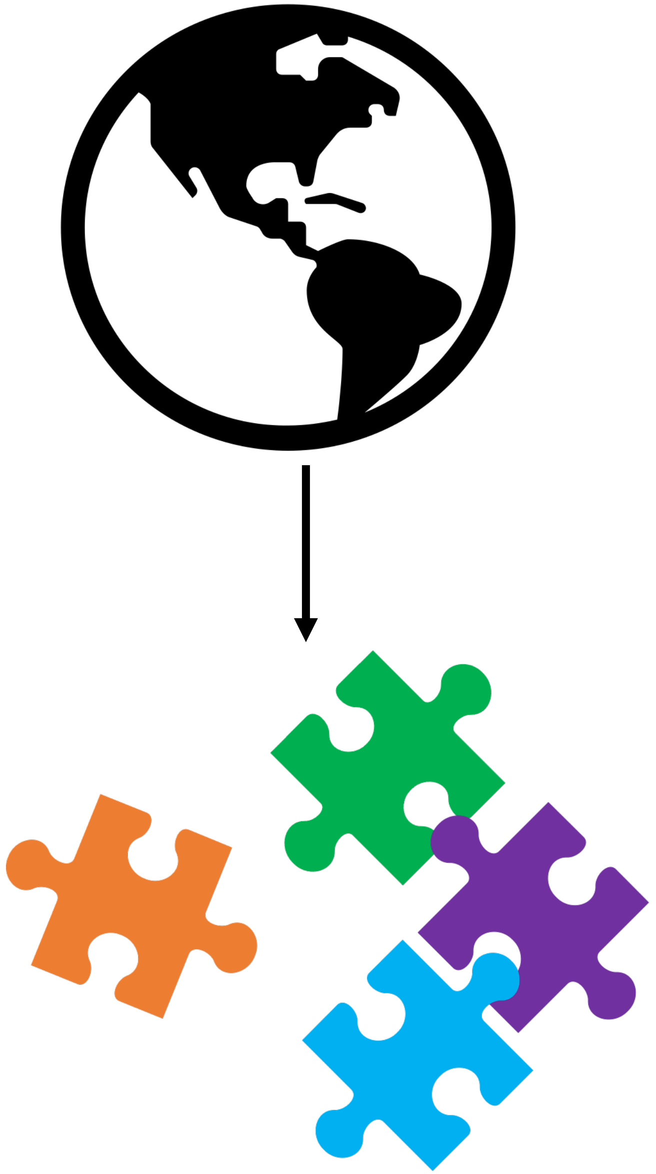
Algorithms

2) What do we do to parallelize something?

Computation vs Communication

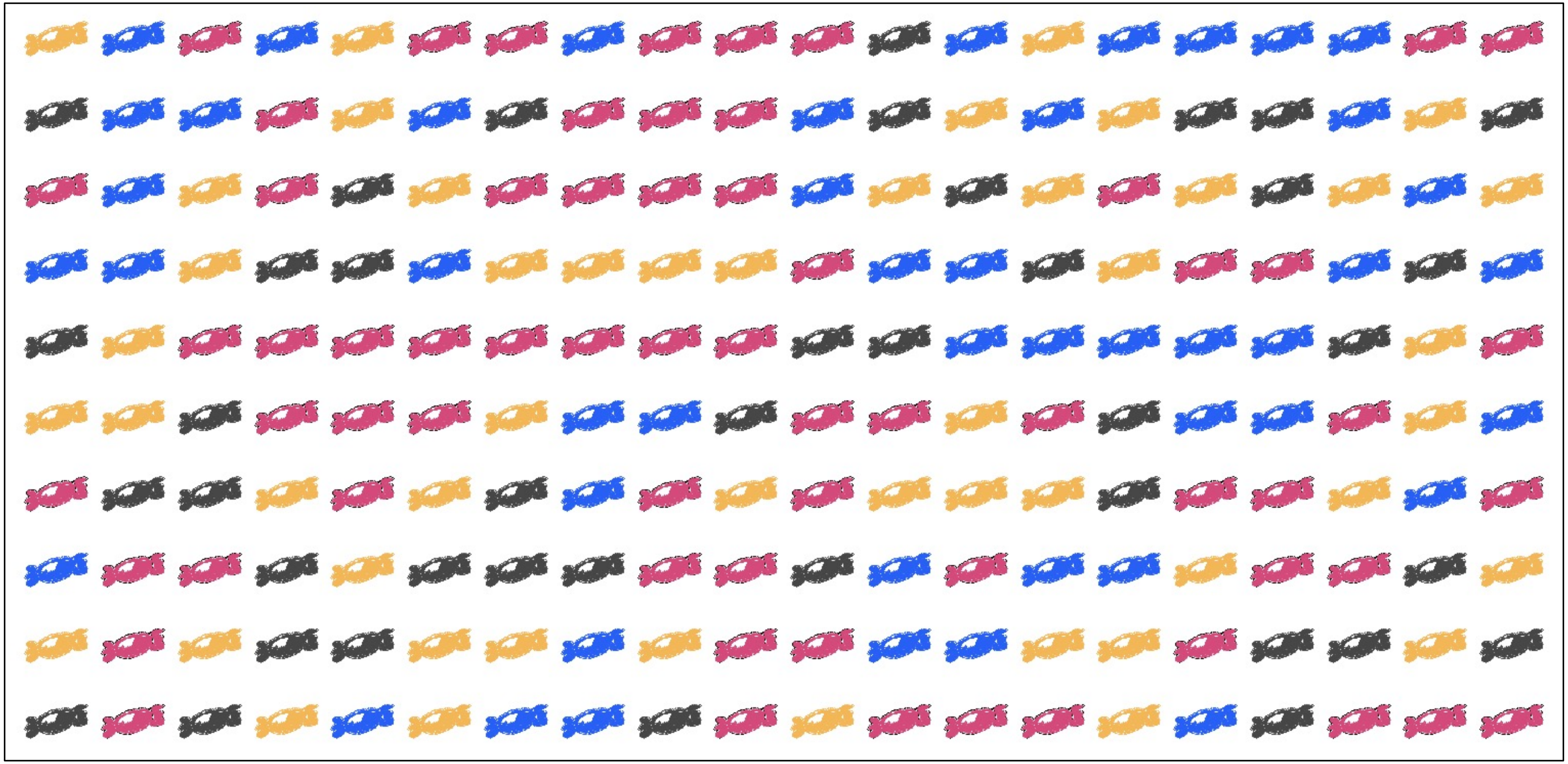
# 1)Algorithms

*Looking for Parallelization*



# What do we need to parallelize?

# What is the proportion of:



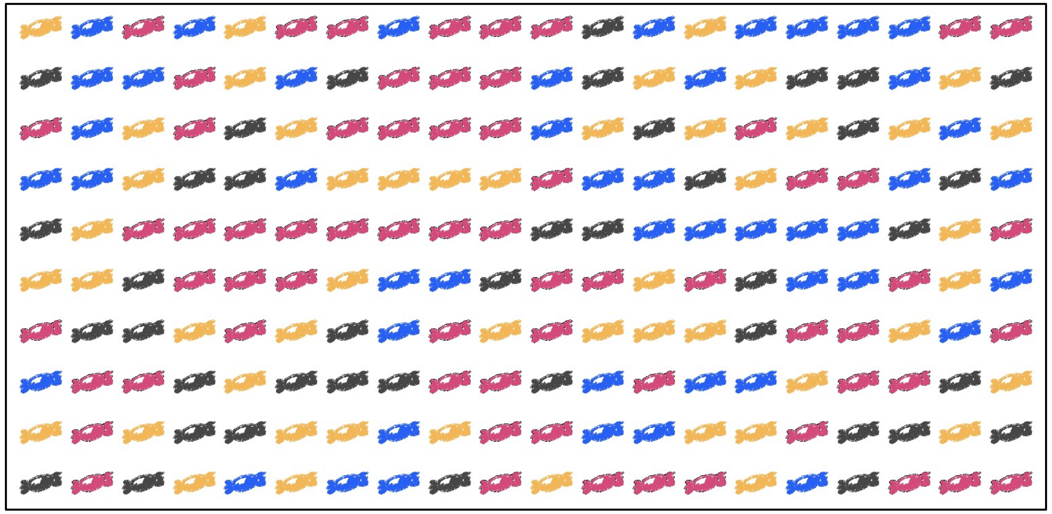
# What is the proportion of:

 =

 =

 =

 =



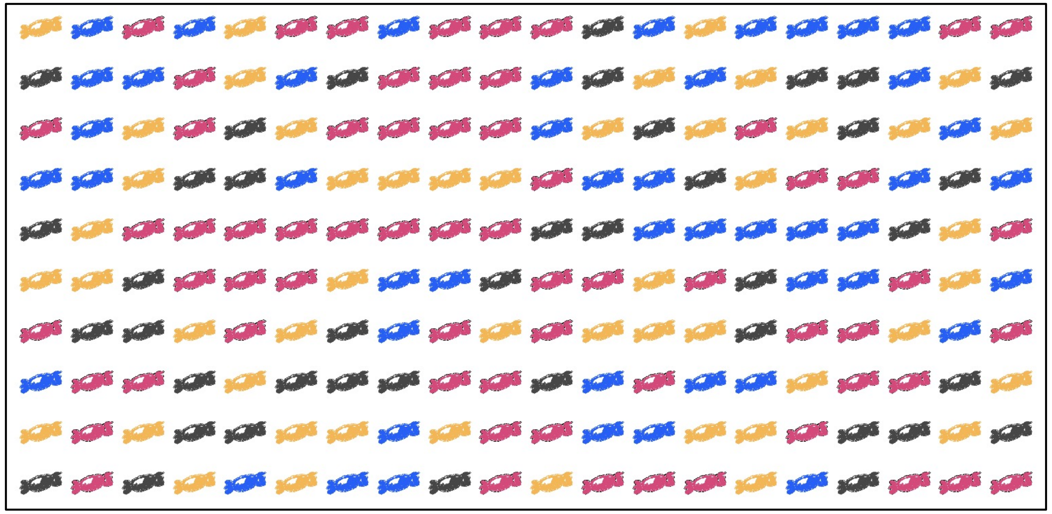
# What is the proportion of:

 =

 =

 =

 =



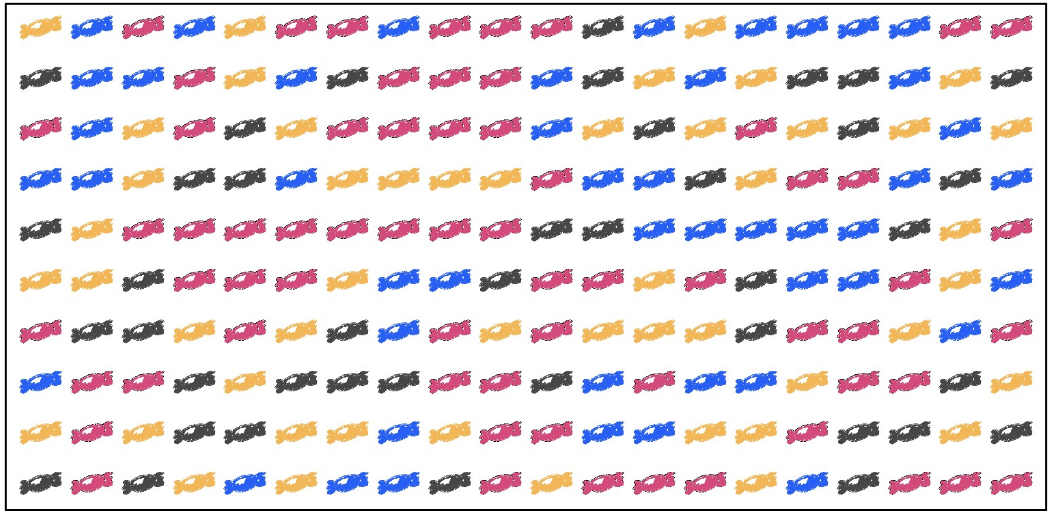
# What is the proportion of:

 =

 =

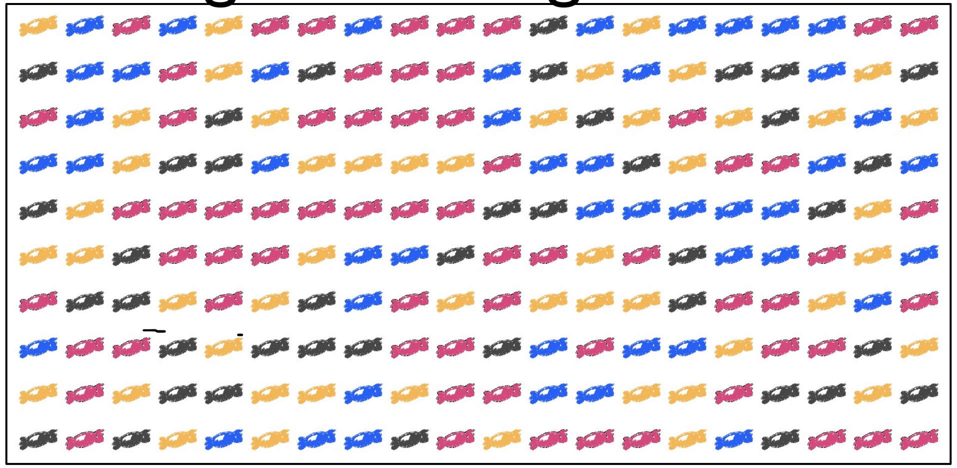
 =

 =



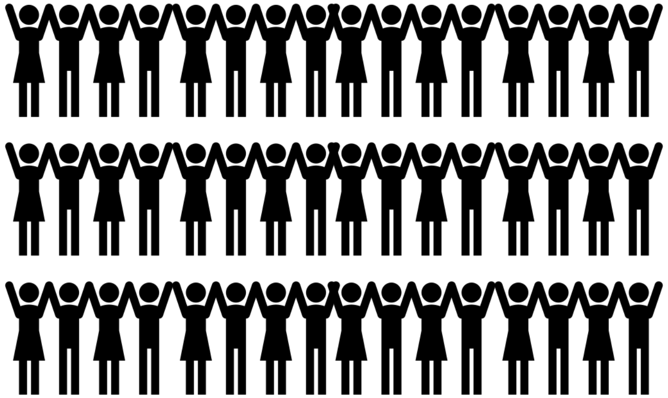
# What do we need to parallelize?

Large data, Large # tasks

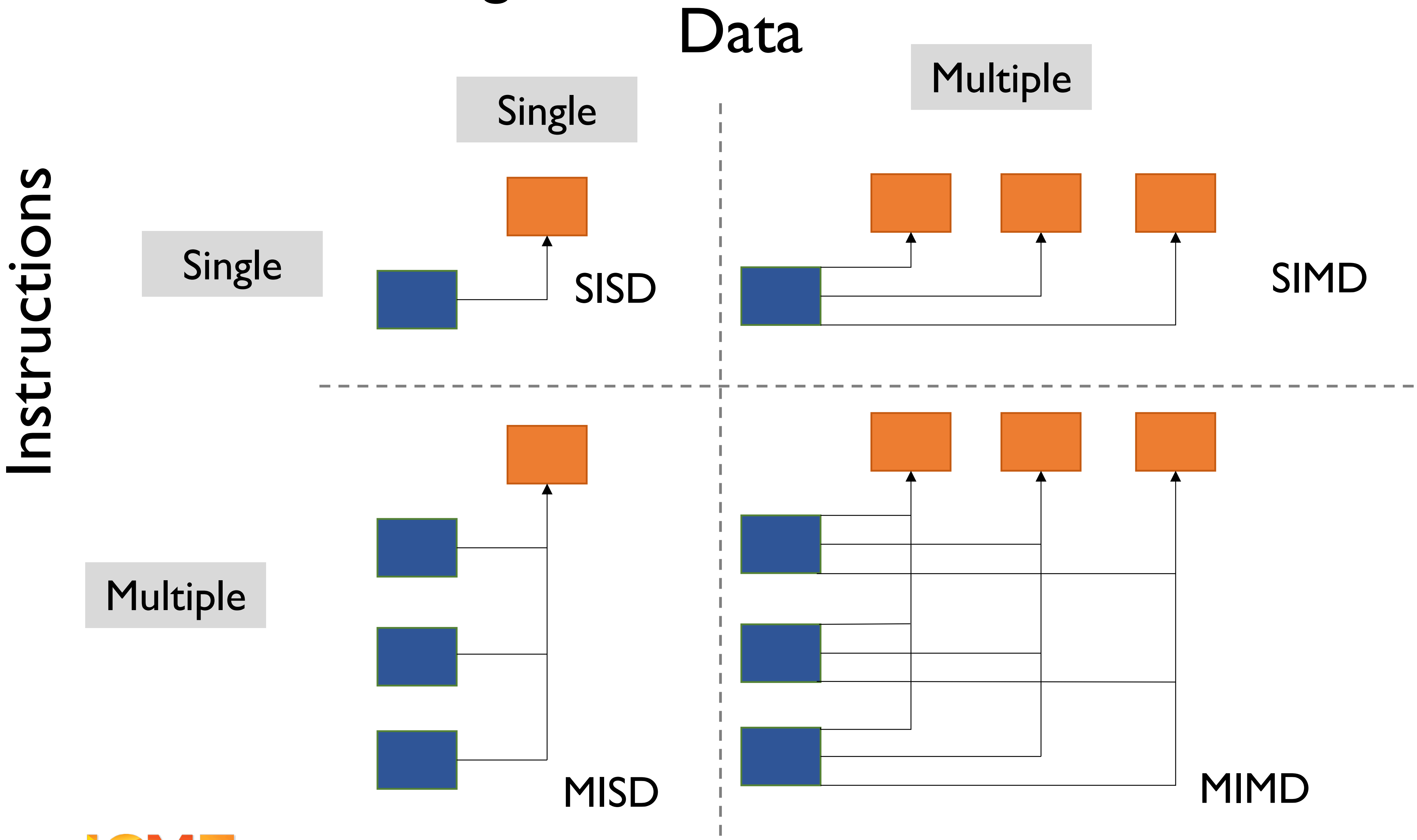


+

Large computing power



# How does distributing Data and Tasks work?

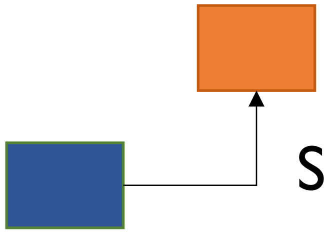


# How does distributing Data and Tasks work?

Data

Instructions

| | Single | Multiple |
|---|---|---|
| **Single** | SISD | SIMD |
| **Multiple** | MISD | MIMD |



SISD

SIMD

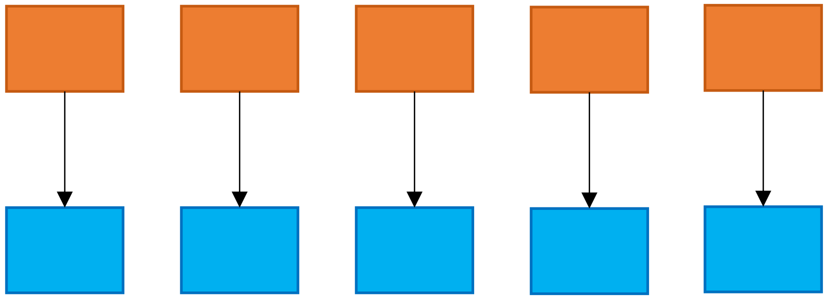
MISD

MIMD

# SIMD challenge: Distribute data (a.k.a. Data Parallelism)

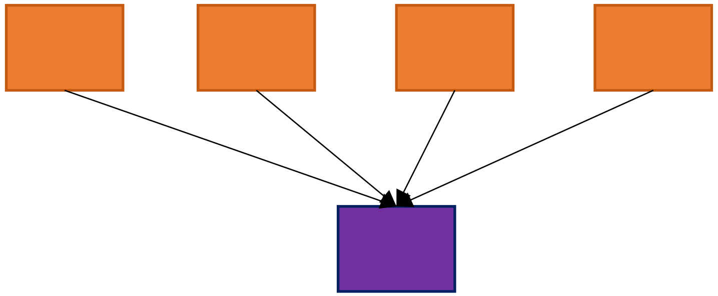
## Transform Data

## Map

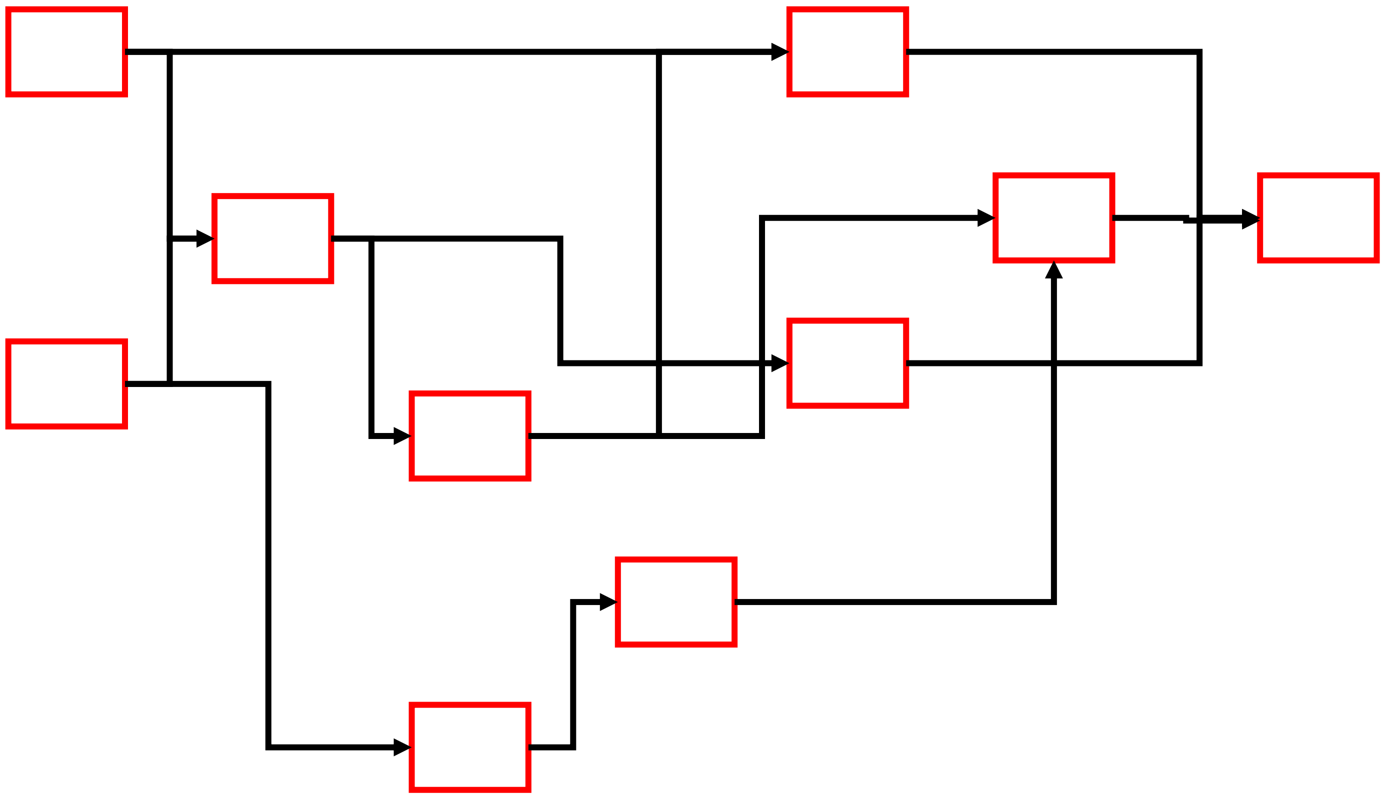


## Aggregate results

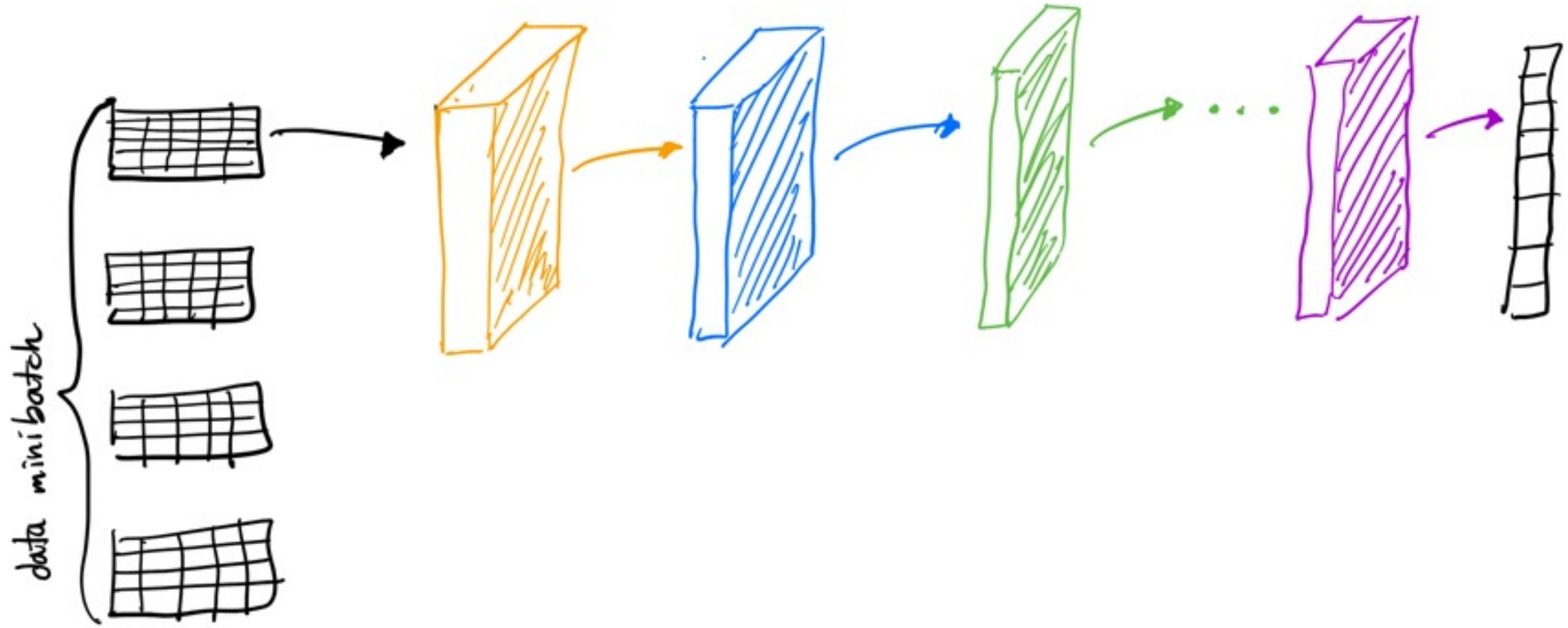
## Reduce



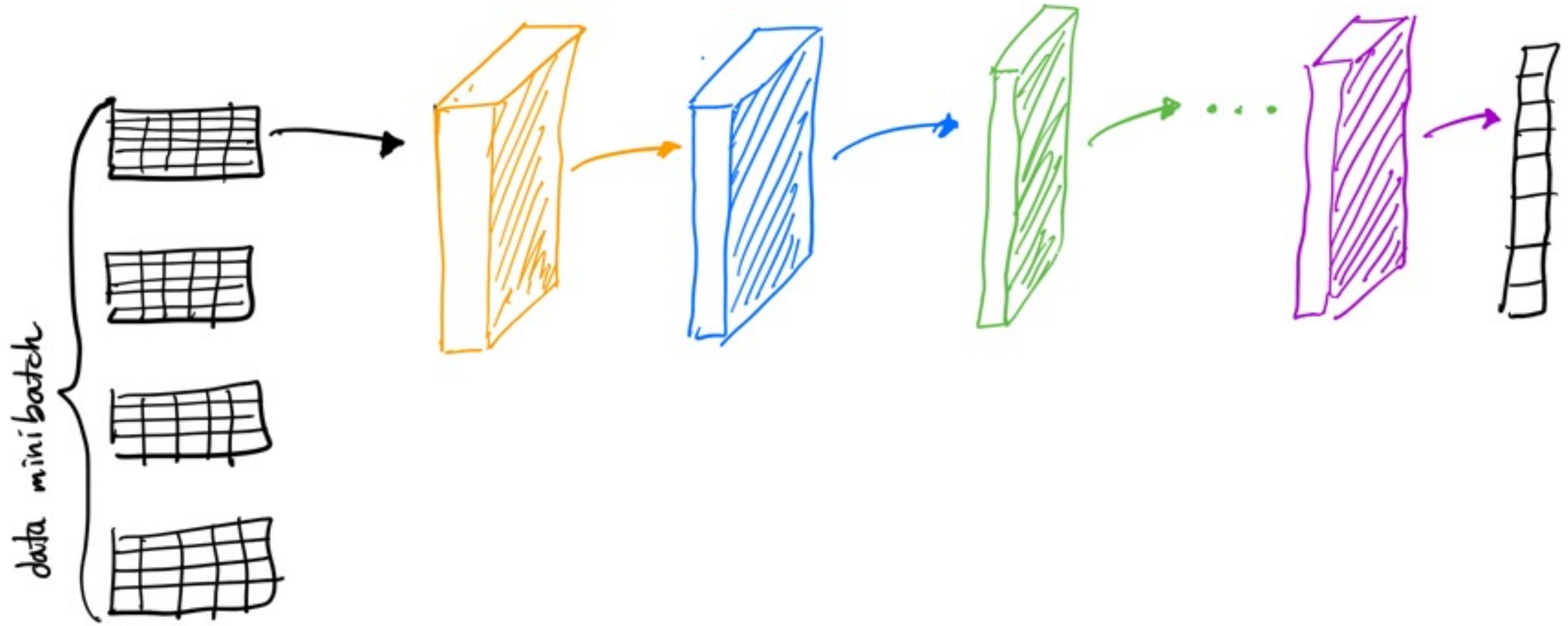
# MISD challenge: Manage tasks dependencies (a.k.a. Task Parallelism)



# Example: Inference with Deep Neural Networks

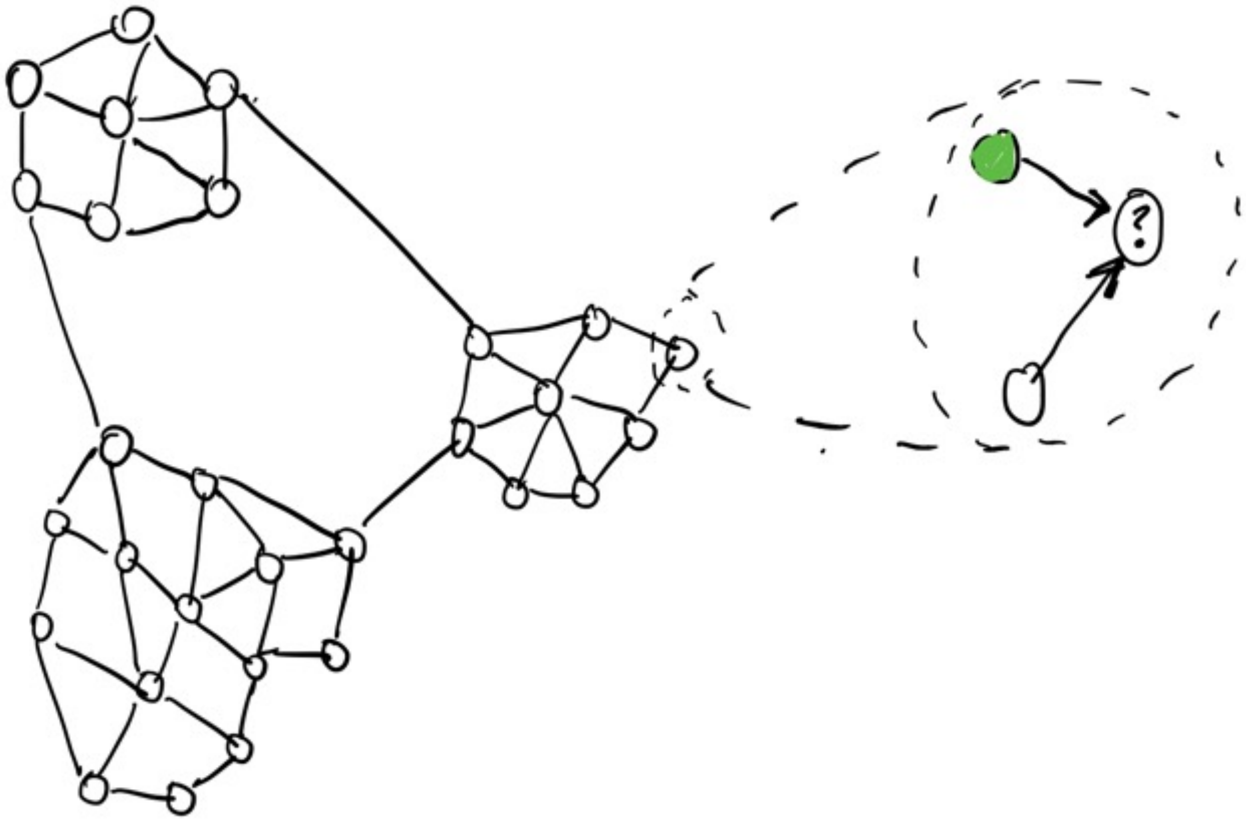


# Example: Inference with Deep Neural Networks



# Example: Neighbor effects in a sparse network

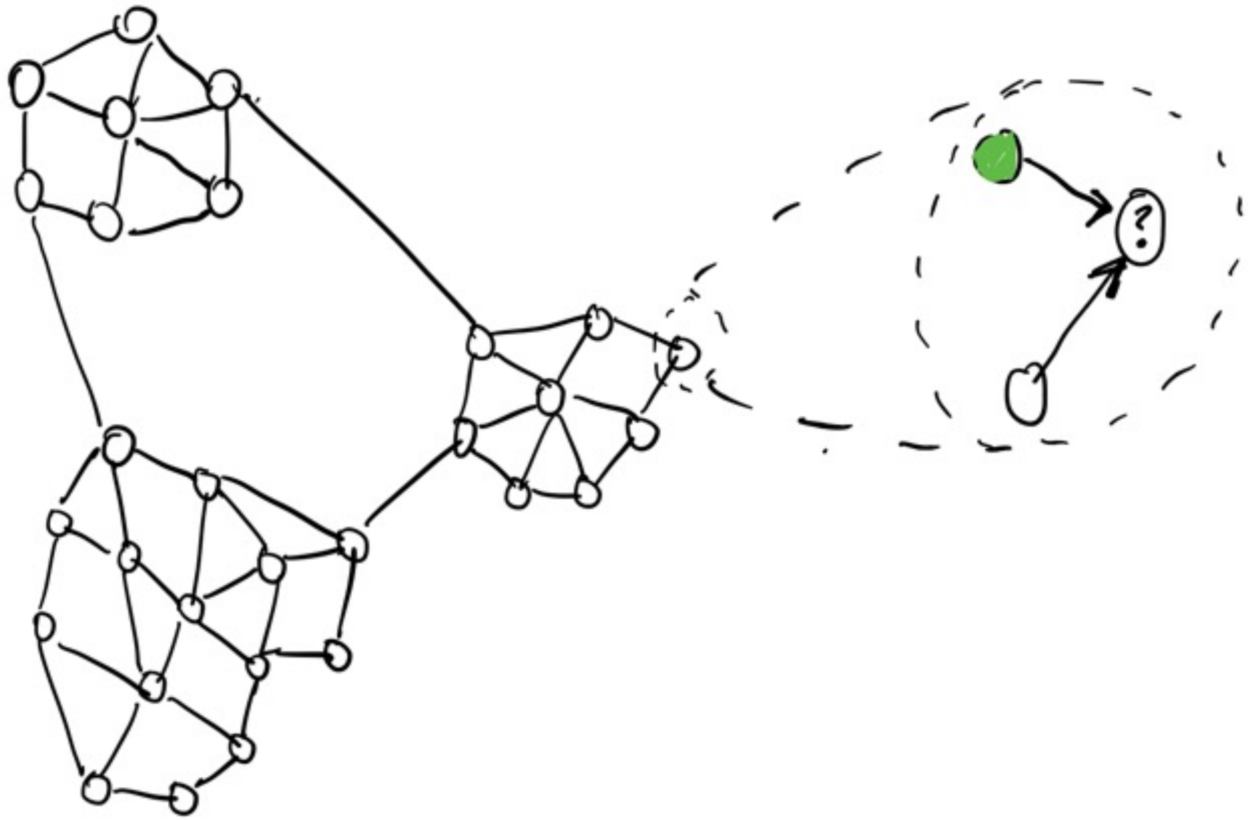
## (a.k.a. sparse matrix-vector product)




Heat equation, Page rank, Contact tracing …

# Example: Neighbor effects in a sparse network

## (a.k.a. sparse matrix-vector product)




Heat equation, Page rank, Contact tracing …

# Recap

✓ Parallelism = Large data/tasks + Large number of resources

Data Parallelism

How to distribute, transform, and aggregate data?

Task Parallelism

How to distribute and execute tasks given data dependencies?

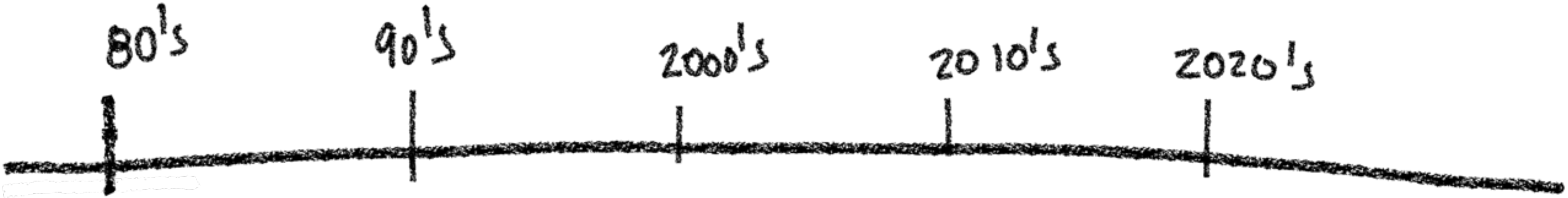
✓ Constrains: Problem based + Resources based: Memory & Communication

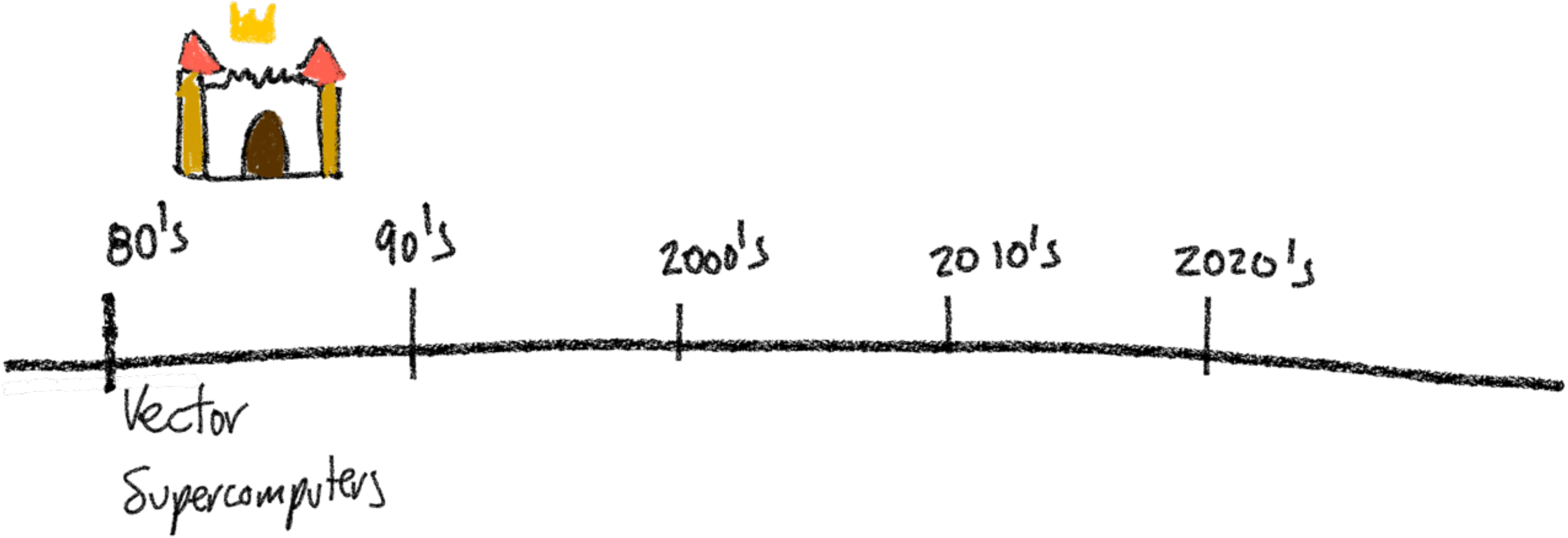
# 2)Architecture

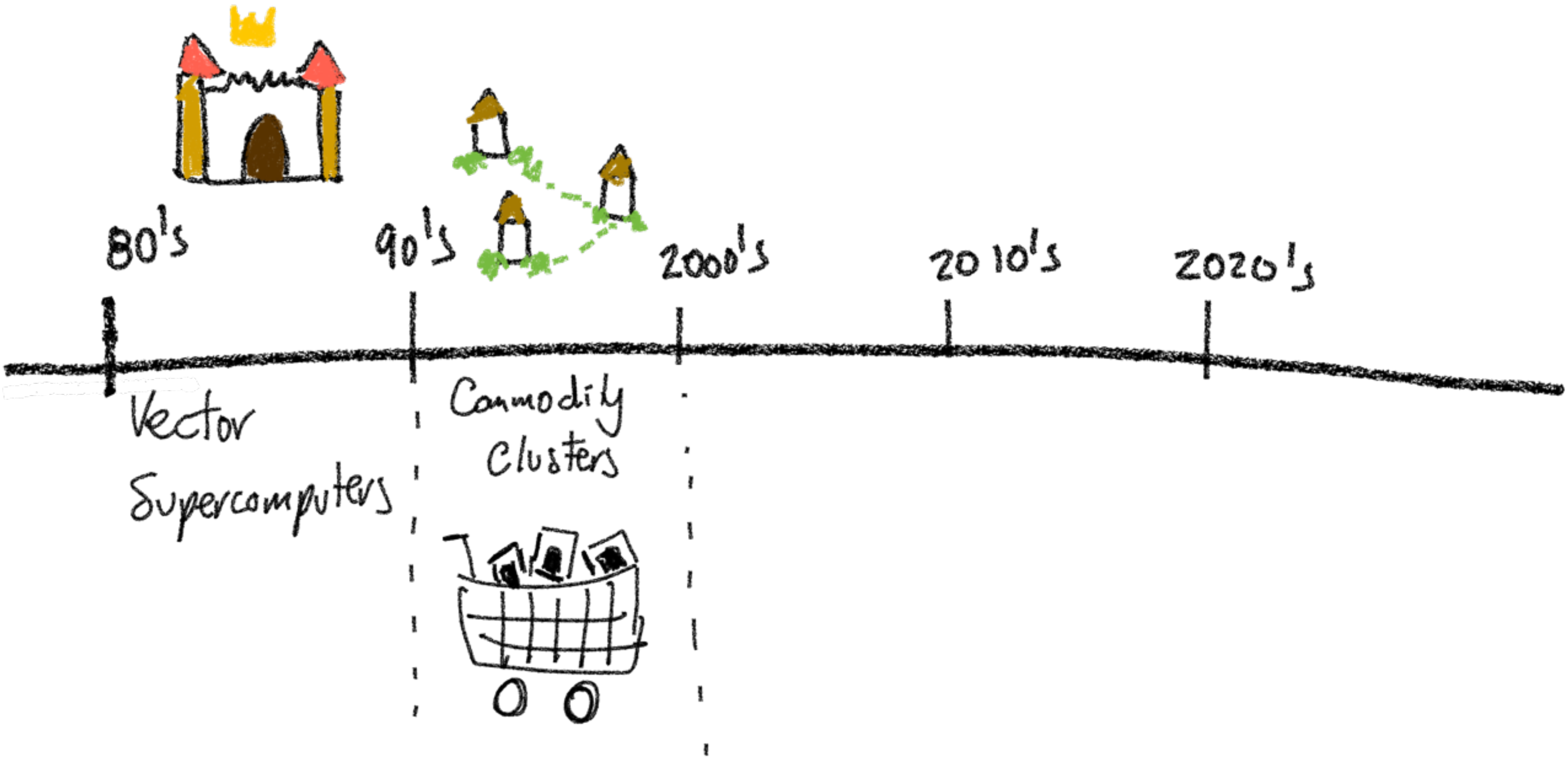
*From the Chip to the Supercomputer*

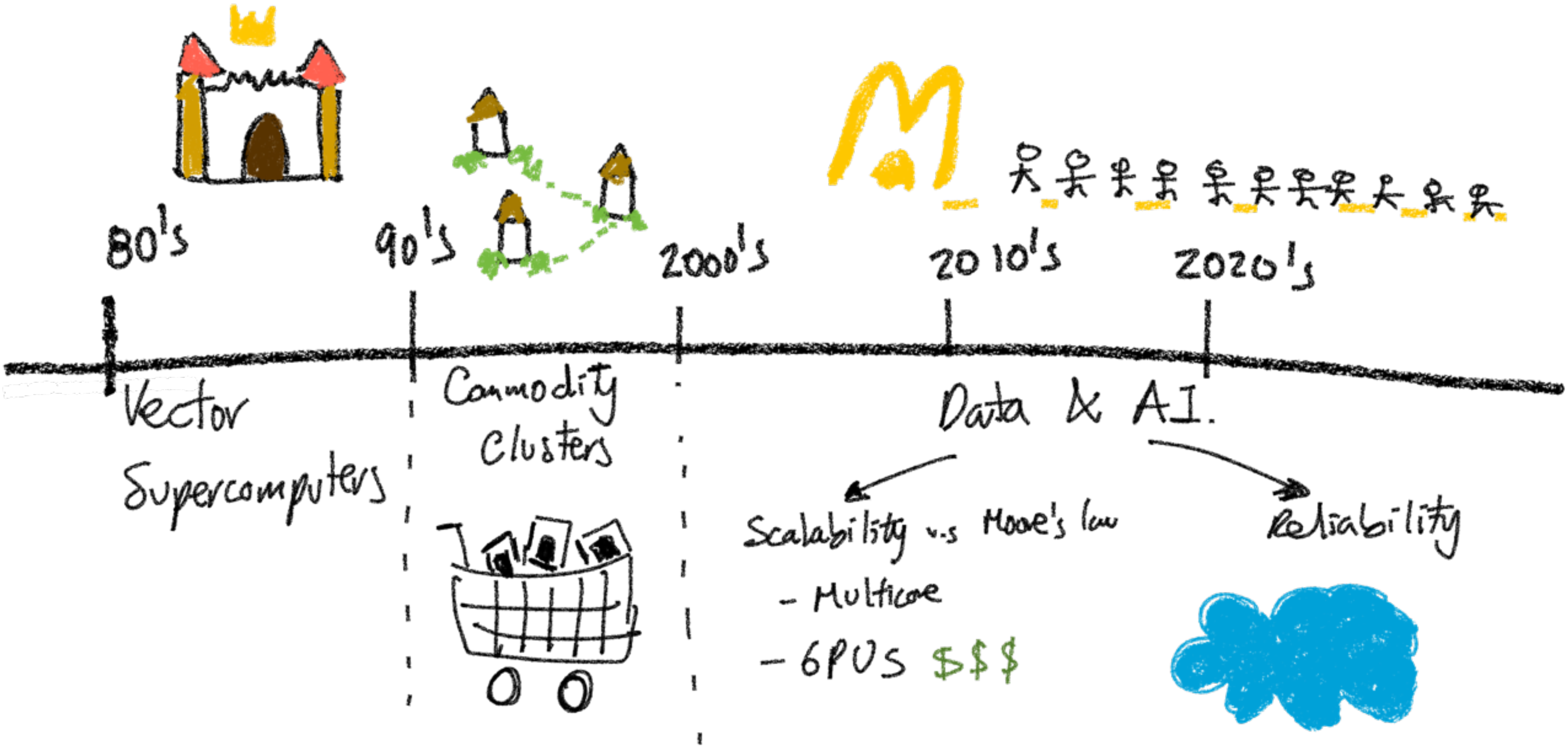
Inspired by: [The Future of Scientific Computation by Bruce Hendrickson]

# When you think about hardware for parallel computing, what comes to mind?


80's
90's
2000's
2010's
2020's


80's
90's
2000's
20 10's
2020's
Vector
Supercomputers


80's
90's
2000's
2010's
2020's
Vector Supercomputers
Commodity Clusters


80's
90's
2000's
2010's
2020's
Vector
Supercomputers
Commodity
Clusters
Data & AI.
Scalability v.s Moore's law
- Multicore
- GPUs $$$
Reliability

# Reality of Scientific Computation today

Heterogenous

Continuous change

Local & In demand

Portability & Reliability

# Reality of Scientific Computation today

Heterogenous

Continuous change

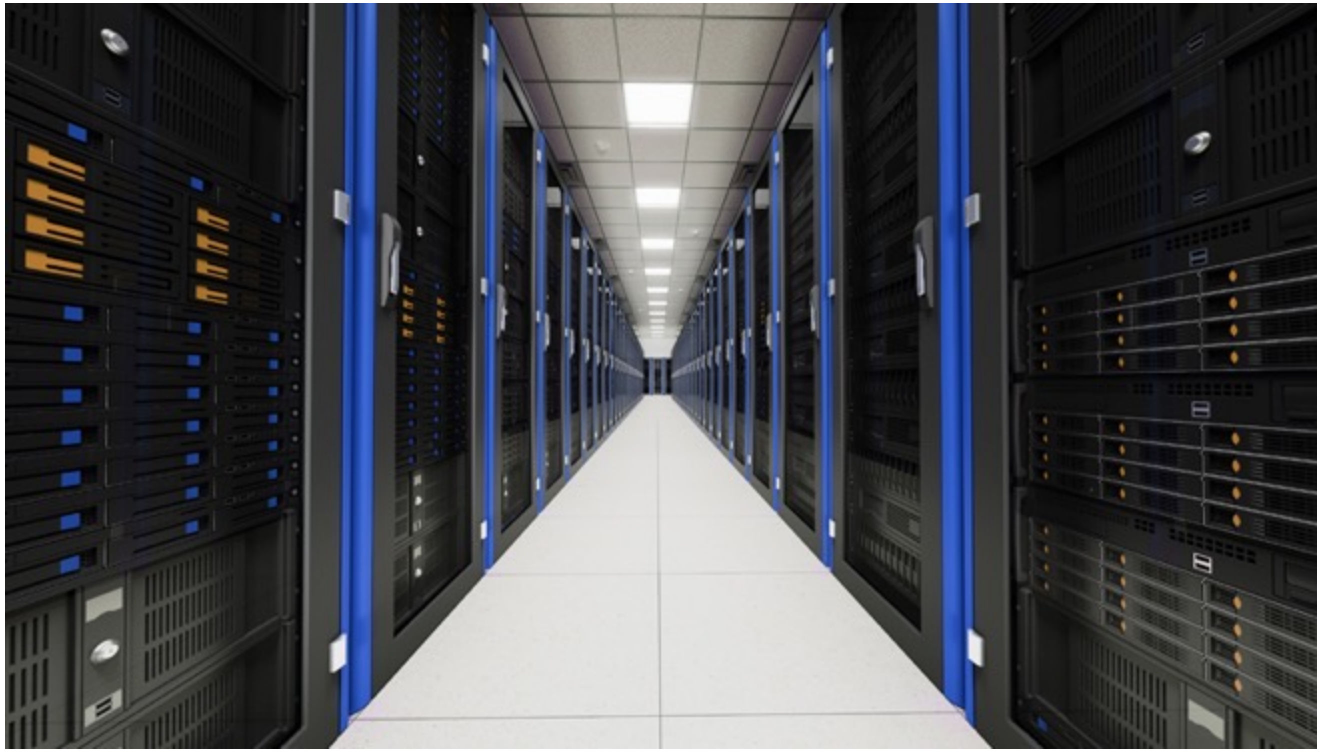
Local & In demand

Portability & Reliability

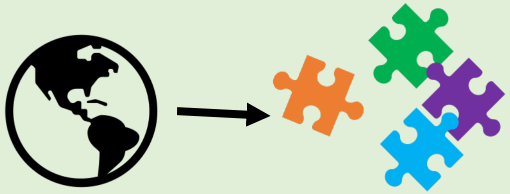
Balanced communication is more challenging than ever!

# Communication decisions from Laptop to HPC cluster

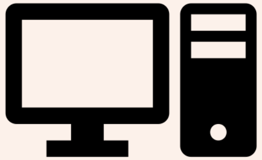
¿… ?



HPC



1)Algorithms



2)Architecture

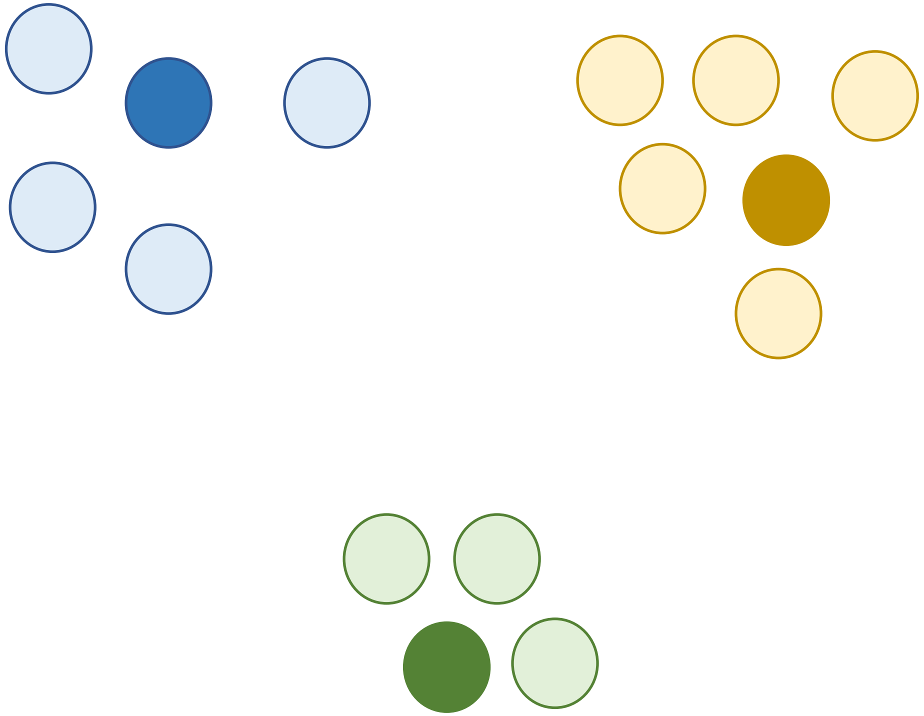
## Solutions

**3)Shared Memory**
OpenMP

**4)Distributed Memory**
MPI

**5) Unified Engine**
Spark

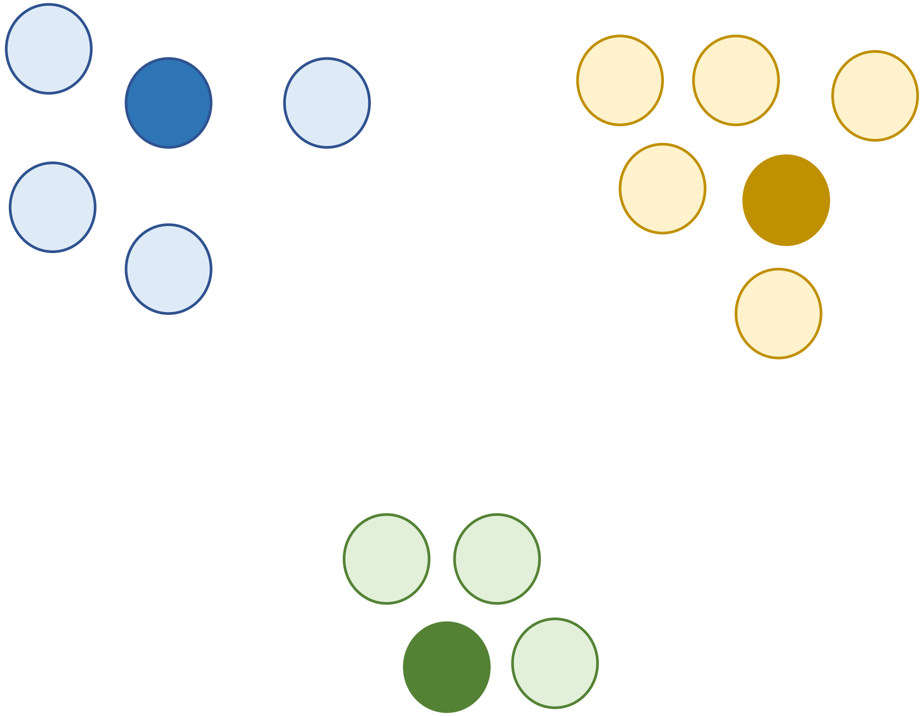
# Exercise: K-means algorithm



Find prototypes and clusters that minimize

$$J = \sum_{l=1}^{L} \sum_{i \in C_l} d(x^{(i)}, \tilde{x}_l)$$

# Exercise: K-means algorithm



(0) Initialize centroids $\tilde{x}_1, \dots, \tilde{x}_k$ at random

(1) Iterate until clusters do not change

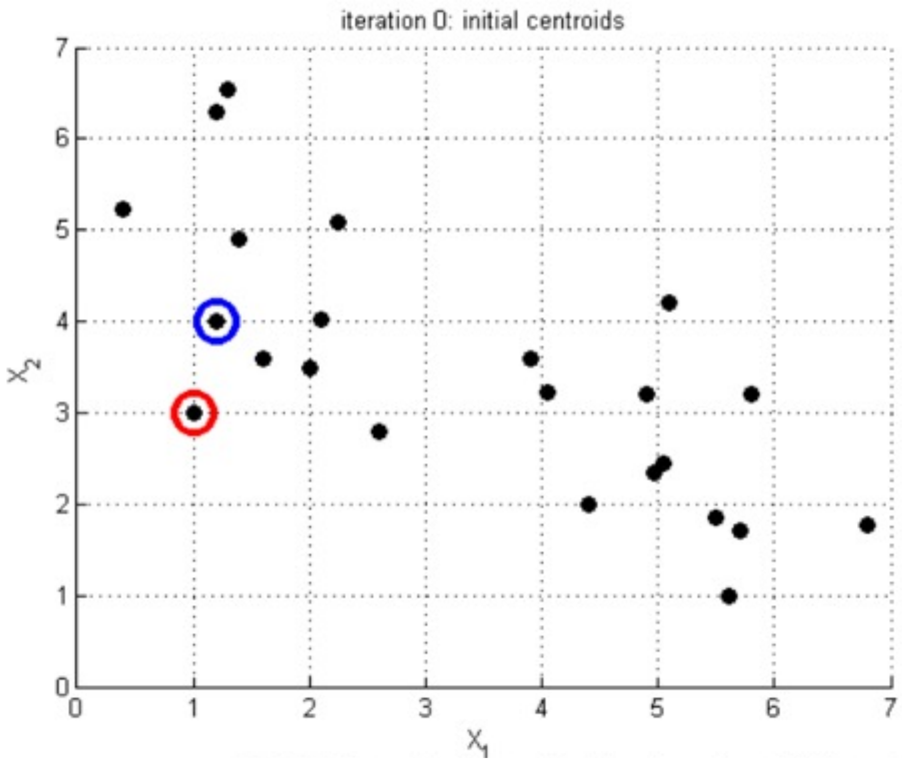
(a) Find best cluster for each point $x^{(i)}$

$$\text{cluster}\left(\mathrm{x}^i\right) = \underset{1,\dots,k}{\text{argmin}}\, d(x^i, \tilde{x}_l)$$

(b)Update centroid $\tilde{x}_l$ for each cluster $C_l$

$$\tilde{x}_l = \frac{1}{|C_l|} \sum_{i \in C_l} x^i$$

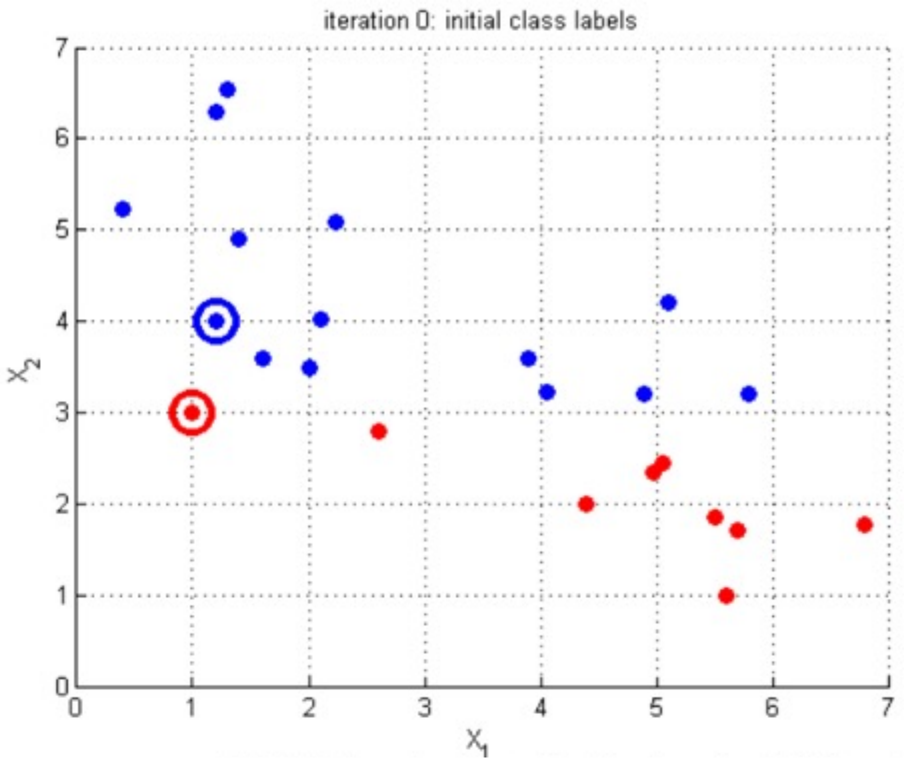
# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)

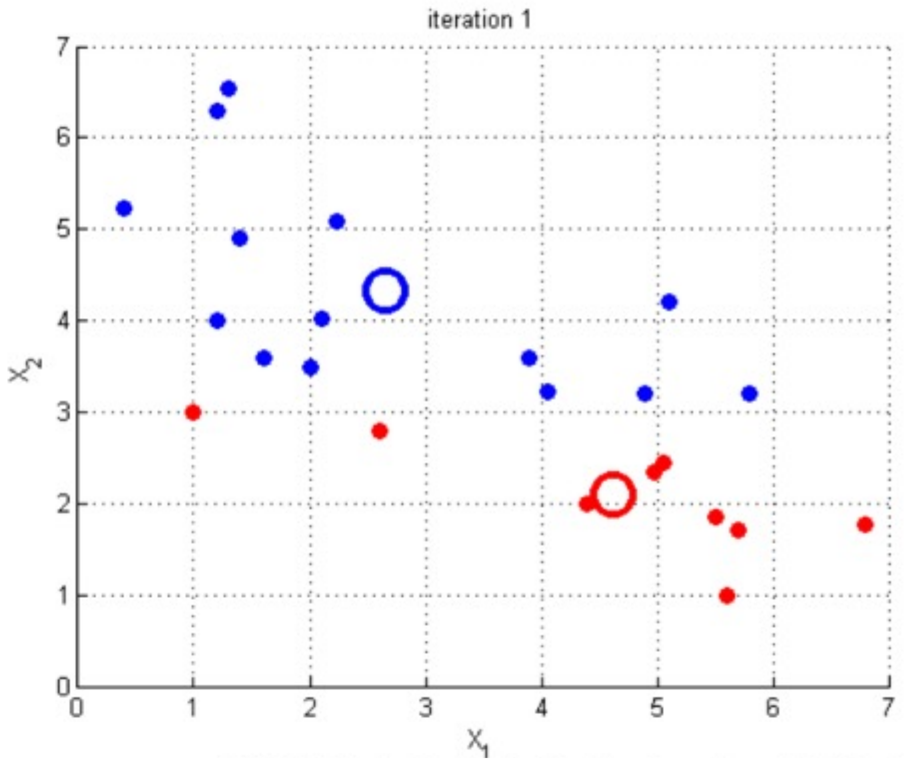



Pick initial centroids

Assign initial clusters

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)



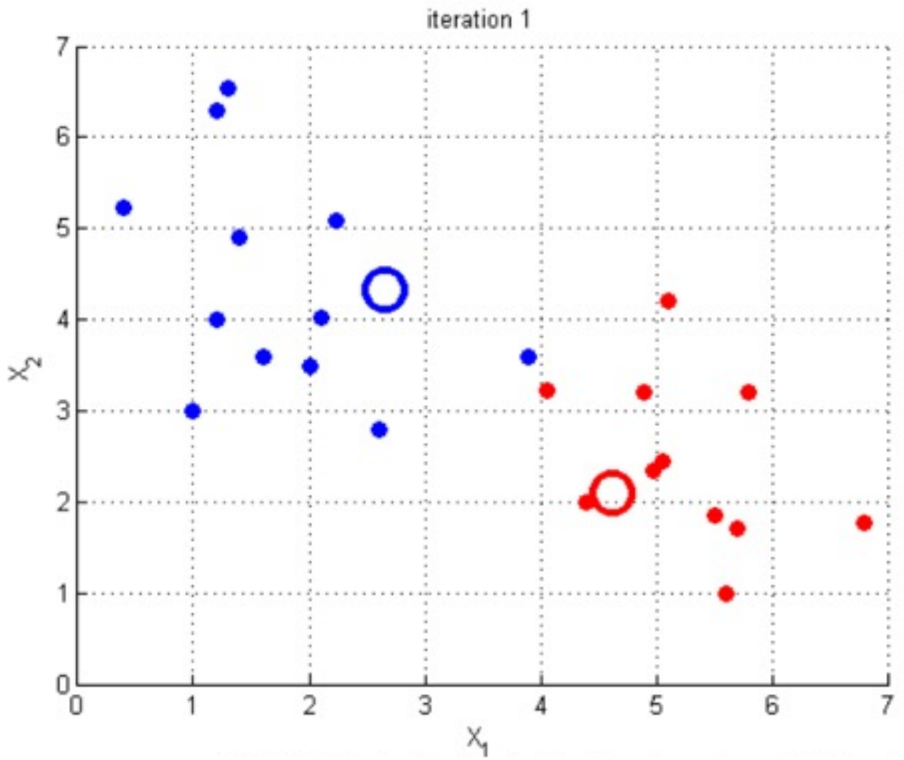

Pick initial centroids

Assign initial clusters

Update centroids

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

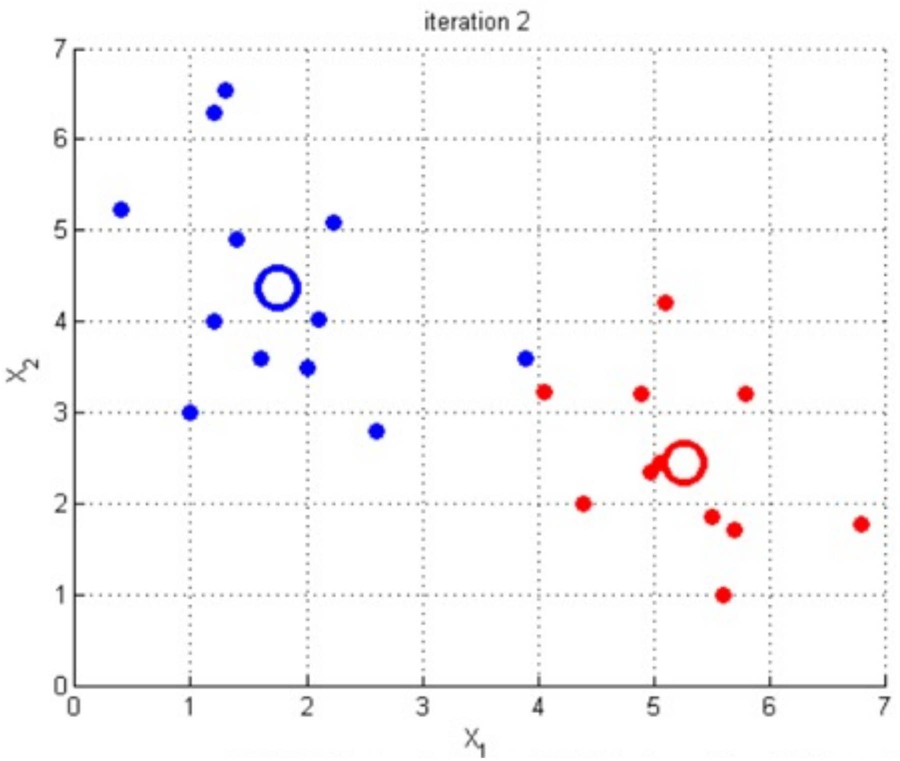
Assign initial clusters

Update centroids

Reassign clusters

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

Assign initial clusters

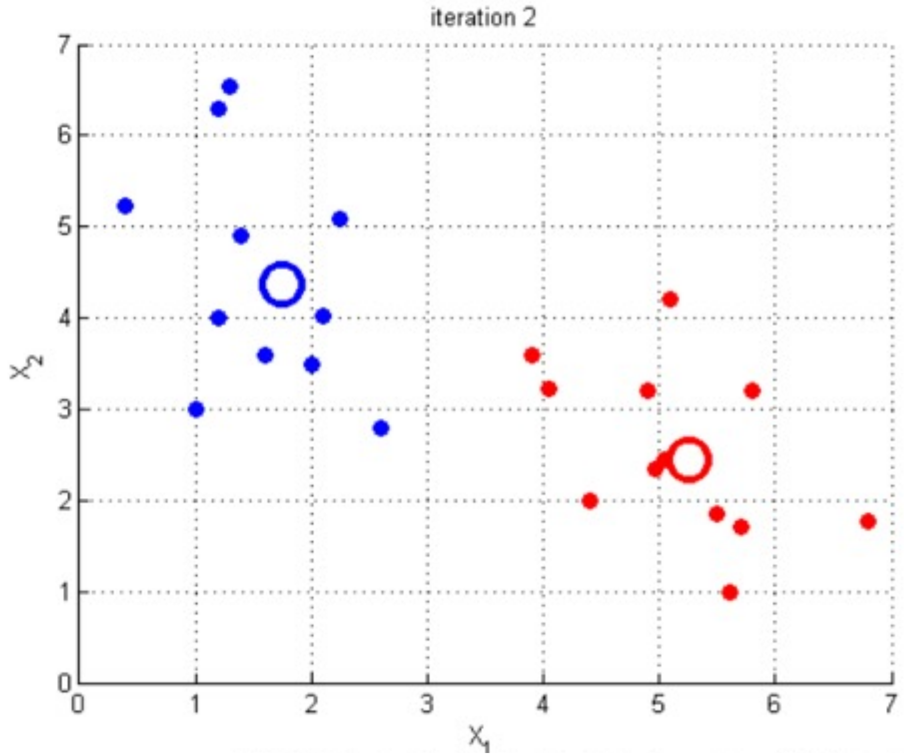
Update centroids

Reassign clusters

Update centroids

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

Assign initial clusters

Update centroids

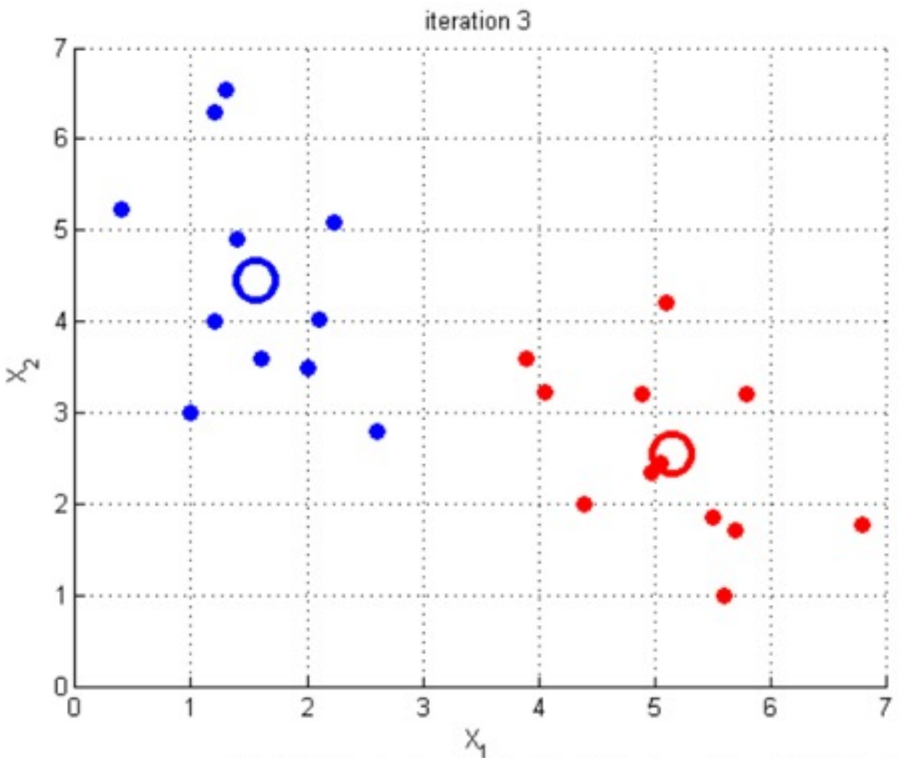
Reassign clusters

Update centroids

Reassign clusters

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

Assign initial clusters

Update centroids

Reassign clusters

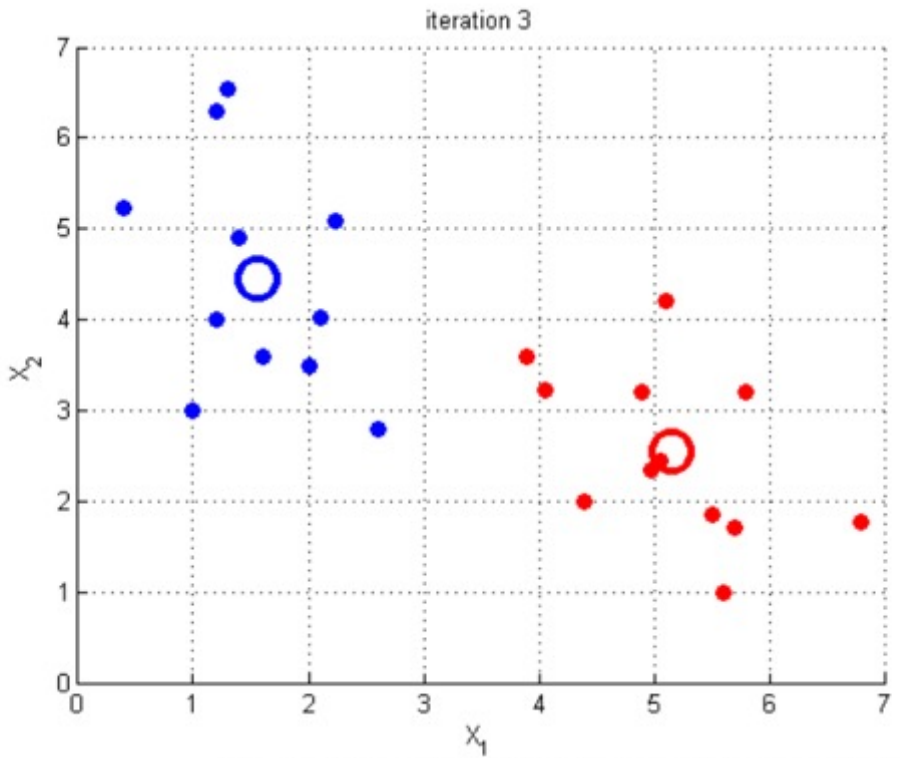
Update centroids

Reassign clusters

Update centroids

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

Assign initial clusters

Update centroids

Reassign clusters

Update centroids

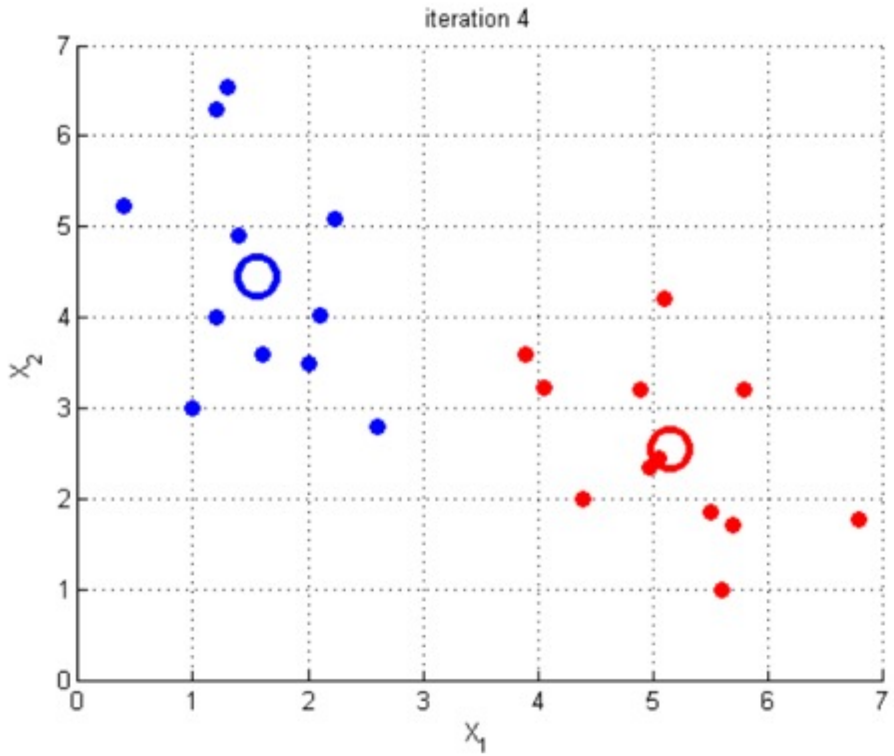
Reassign clusters

Update centroids

Reassign clusters

* Simulation done by Karianne Bergen

# K-means algorithm (Lloyd's Algorithm)




Pick initial centroids

Assign initial clusters

Update centroids

Reassign clusters

Update centroids

Reassign clusters

Update centroids

Reassign clusters

Converged

* Simulation done by Karianne Bergen

# Exercise: K-means algorithm

To do:

1. Identify regions of the algorithm that can be parallelizable

2. How would you divide the work in each parallelizable region?

3. What dependencies do you find?

Given a set of points $x^1, \dots, x^n$

(0) Initialize centroids $\tilde{x}_1, \dots, \tilde{x}_k$ at random

(1) Iterate until clusters do not change

(a) Find best cluster for each point $x^i$

$$\mathrm{cluster}\left(\mathrm{x}^i\right) = \underset{1,\dots,k}{\operatorname{argmin}}\, d(x^i, \tilde{x}_l)$$

(b)Update centroid $\tilde{x}_l$ for each cluster $C_l$

$$\tilde{x}_l = \frac{1}{|C_l|} \sum_{i \in C_l} x^i$$