CS677: Lecture 3

Introduction to Parallelization

August 6, 2024

Logistics

- Office hours
  - Instructors – by appointment or after class
  - TAs - Webpage https://www.cse.iitk.ac.in/users/cs677/pages/tas.html
- Group formation
  - Email by August 9 (hard deadline)
  - Include names, roll numbers, email-ids
  - Send to nitesht@cse

Parallelism Everywhere

Source: amazon.com

Why Parallel?

Task: Find the average age (or any statistics) of Indians

Current population of India is estimated at 1,442,945,809 people at mid year according to UN data.

Time (1 human): > 40 years

Time (1 CPU): 10 s

Time (2 CPUs): 5 s

Time (4 CPUs): 3 s

Why Fast?

Investing Guide

China's stock market crash...in 2 minutes

by Charles Riley and Sophia Yan @CNNMoneyInvest

August 27, 2015: 10:09 AM ET

Why China's market meltdown affects the rest of th

THE ECONOMIC TIMES | Markets

English Edition ▾ | Today's Paper

Subscribe Sign In

Freedom Month Offer is Live

Home ETPrime Markets News Industry Rise Politics Wealth Mutual Funds Tech Careers Opinion NRI Panache ET NOW Spotlight

Stocks ▾ IPOs/FPOs Expert Views Markets Data Investment Ideas Cryptocurrency ▾ Commodities ▾ Forex ▾ Live Stream! Technicals ▾ More ▾

Business News › Markets › Stocks › News › Stock market crash: Investors lose Rs 4 lakh crore in wealth in 5 minutes

Stock market crash: Investors lose Rs 4 lakh crore in wealth in 5 minutes

ETMarkets.com • Last Updated: Oct 11, 2018, 05:31 PM IST

SHARE FONT SIZE SAVE PRINT COMMENT 27

Disaster Prediction

Source: thehindu.com

Forecast & Warnings (Annexure II)

- **West, Central, East and South Peninsular India**
  - ✓ Fairly widespread to widespread light to moderate rainfall accompanied with thunderstorm & lightning very likely over the region during next 5 days.
  - ✓ **Isolated extremely heavy rainfall very likely over Konkan & Goa, Madhya Maharashtra, Telangana during 18th-20th; South Interior Karnataka, Coastal Karnataka, Gujarat State, Coastal Andhra Pradesh & Yanam on 18th & 19th; Tamil Nadu on 18th; Vidarbha, south Chhattisgarh on 19th & 20th and south Odisha on 19th July.**
  - ✓ **Heavy to very heavy rainfall very likely at isolated/some places over Konkan & Goa, Madhya Maharashtra, Gujarat State, Coastal & South Interior Karnataka during next 5 days; Coastal**

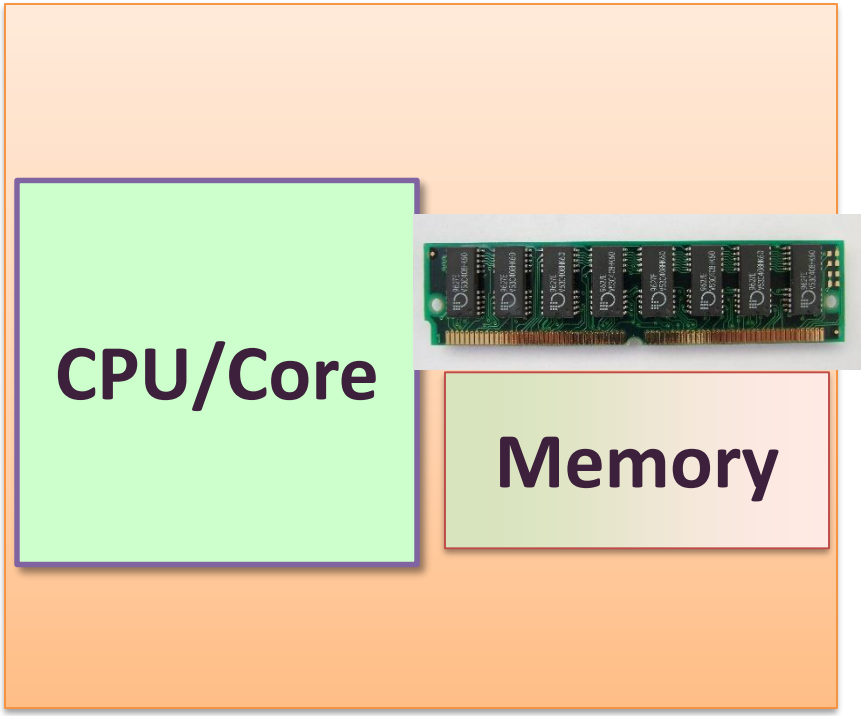
Source: imd.gov.in

Parallelism

A parallel computer is a collection of processing elements that communicate and cooperate to solve large problems fast.

– Almasi and Gottlieb (1989)

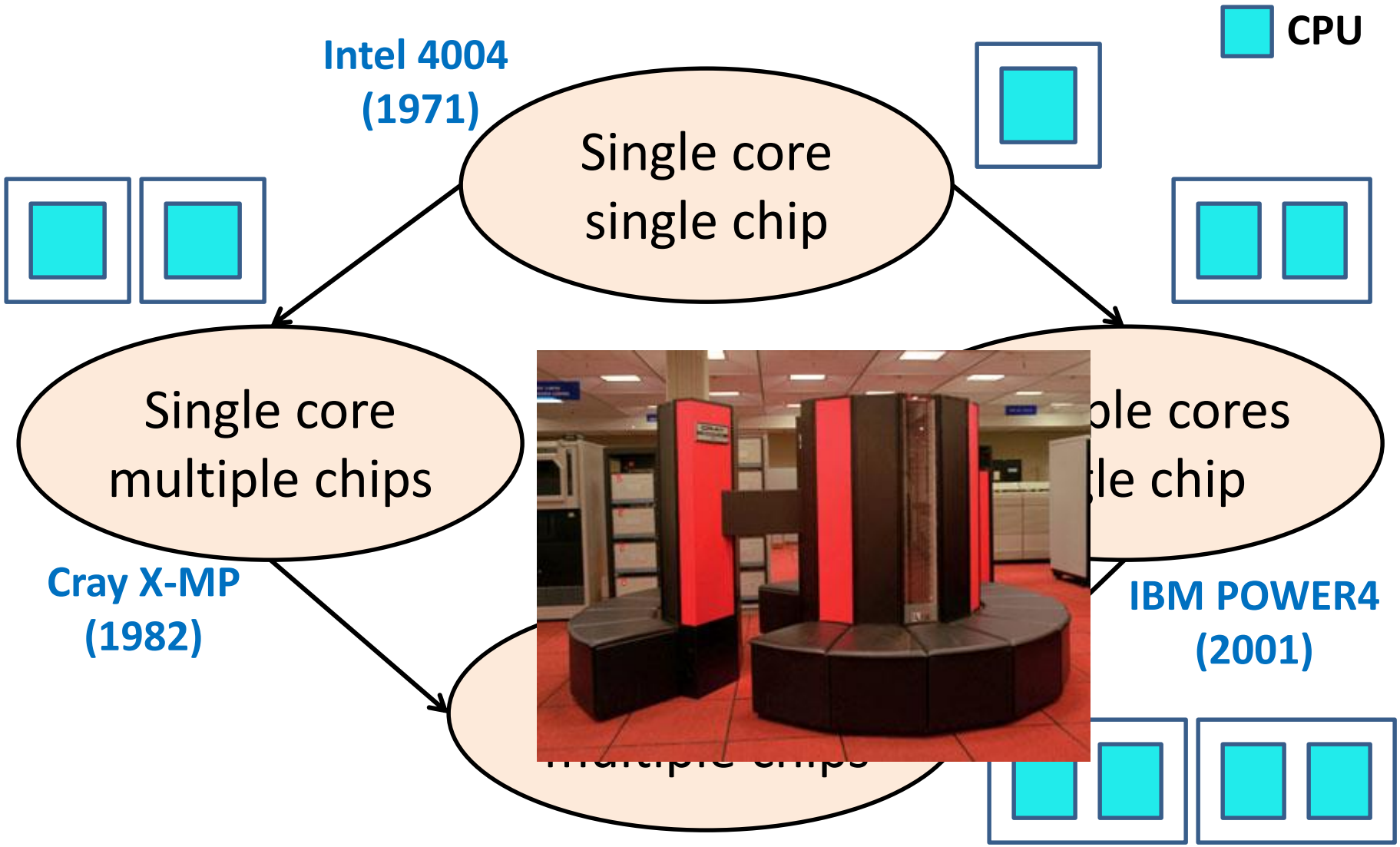
Basic Computing Unit



Processing unit

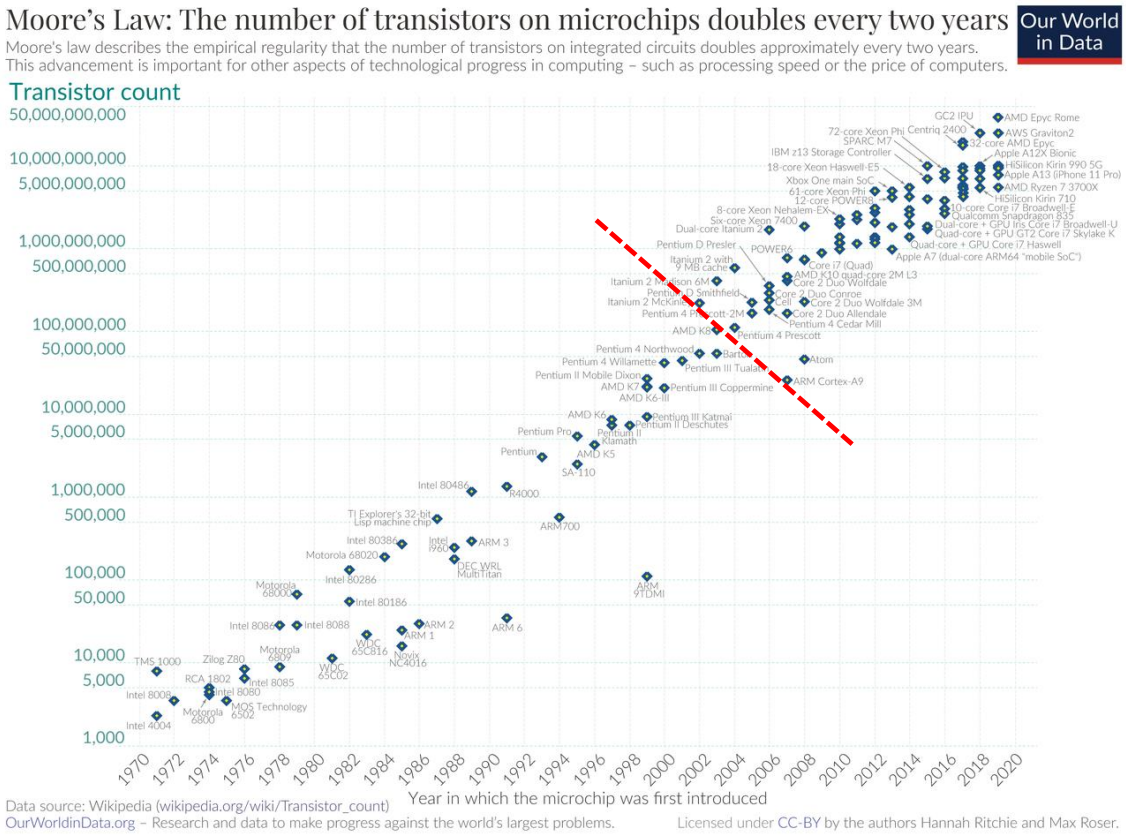
Intel Core i7
(Courtesy: www.intel.com)

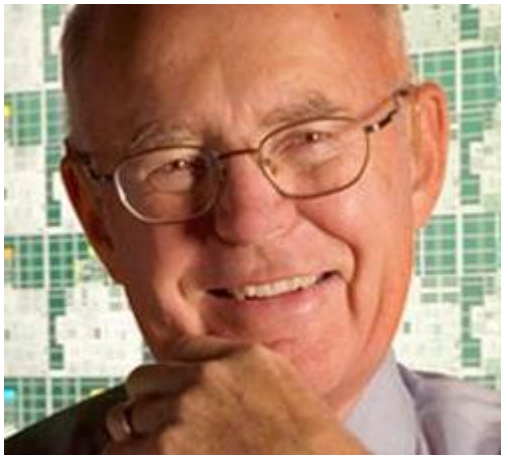
Multicore Era



Moore’s Law

Number of CPU cores per node increased

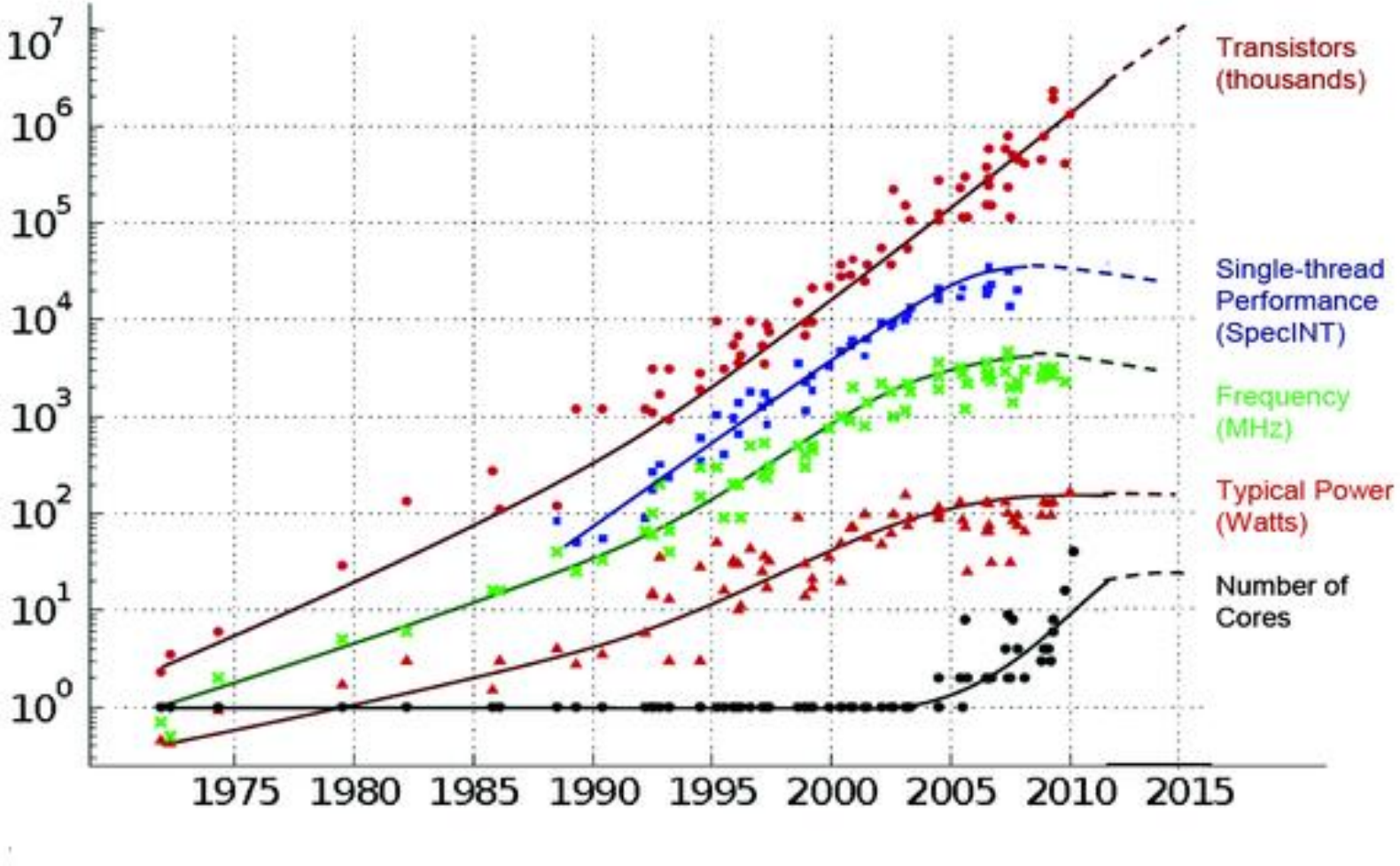




Gordon Moore

[Source: Wikipedia]

35 YEARS OF MICROPROCESSOR TREND DATA

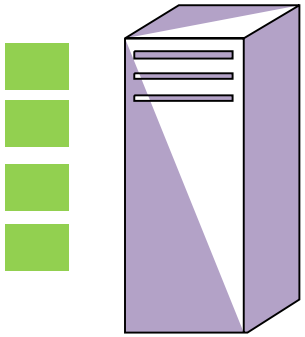


Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

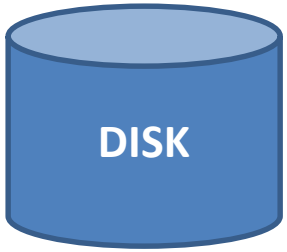
System – Simplified View

Memory

Fast Memory



CPU cores

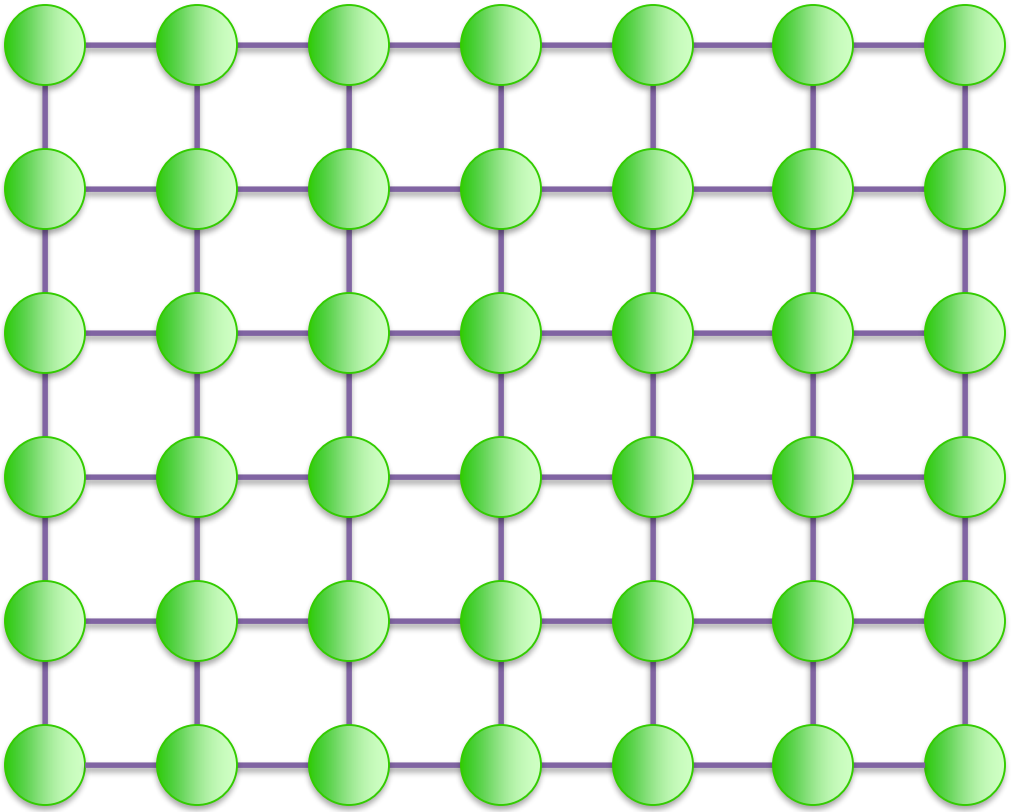


Parallel Computing

Supercomputer/Cluster/Data Center

Network is the backbone for data communication

Parallel Computer



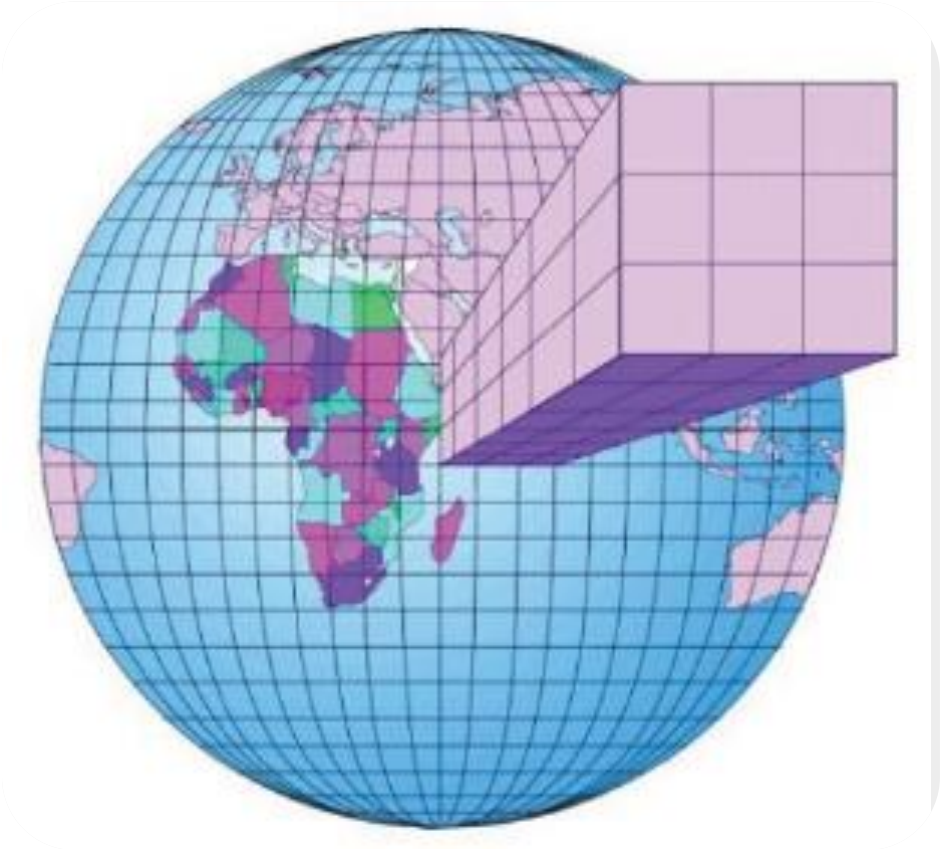
Compute nodes

top500.org (Jun'24)

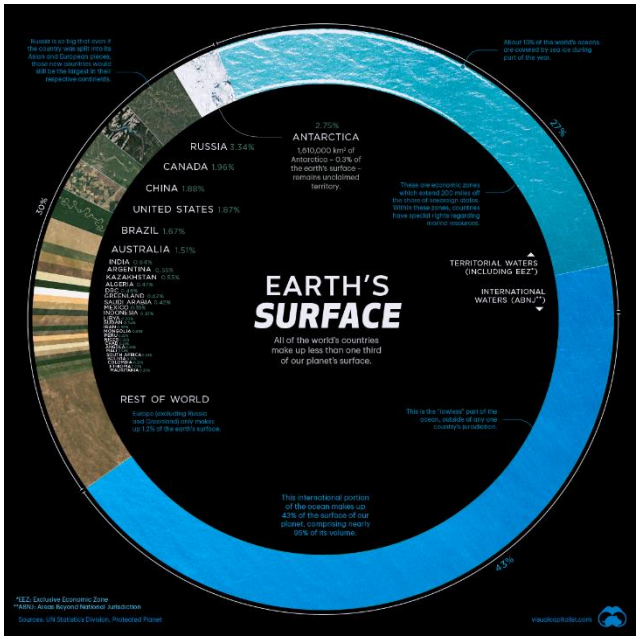
Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,206.00	1,714.81	22,786
2	**Aurora** - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory	9,264,128	1,01		
			561.20		
			442.01	537.21	29,899
			379.70	531.51	7,107
			270.00	353.75	5,194

~ $600 million
~ 7300 sq. ft.
~ 22 MW power
~ 23000 L water

Discretization

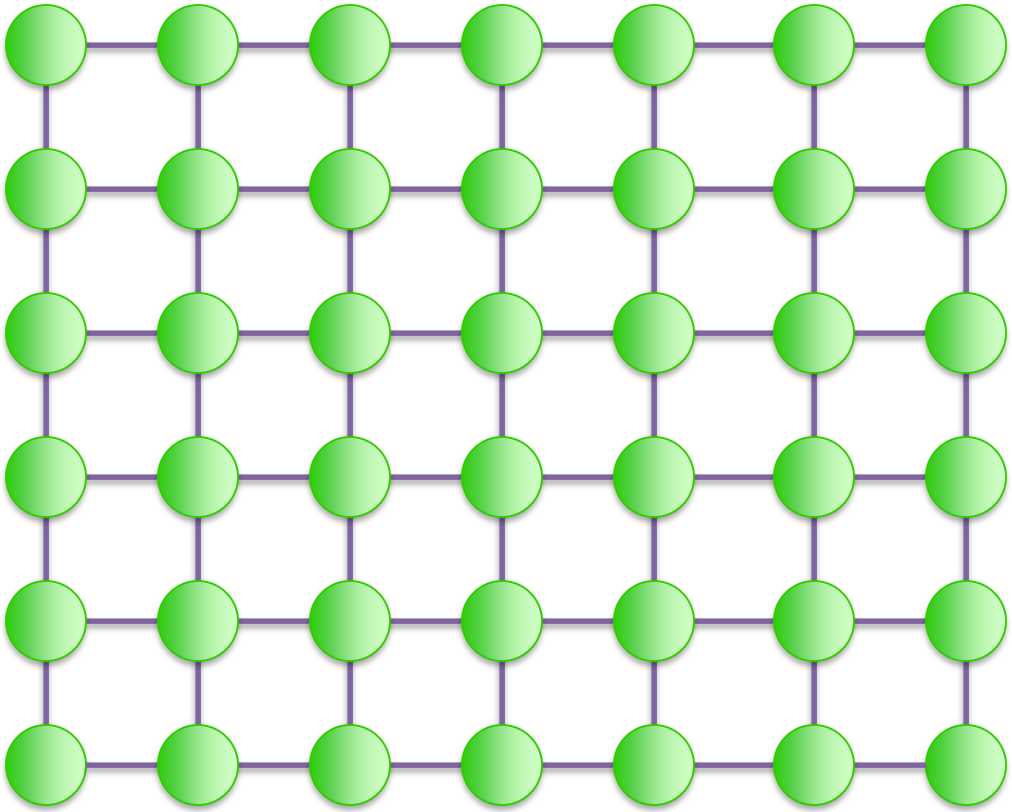


Gridded mesh for a global model
[Credit: Tompkins, ICTP]

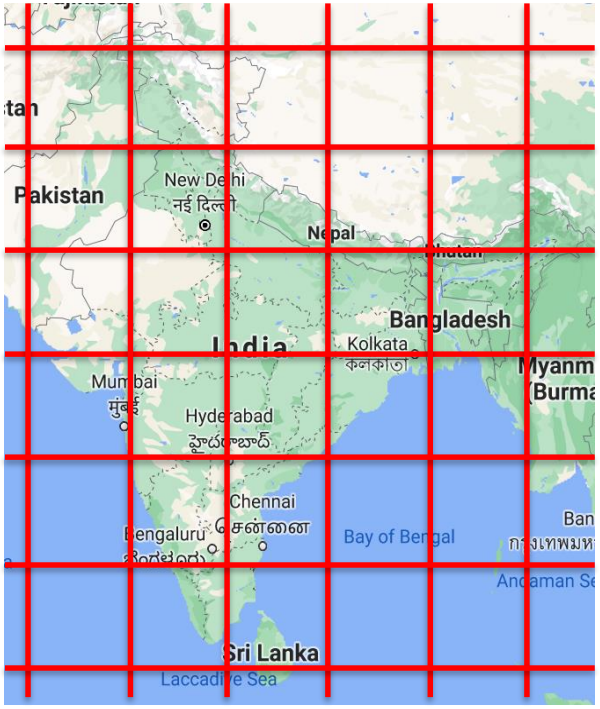


Surface area of Earth: ~501 million sq. km.
Credit: World Economic Forum

Domain Decomposition



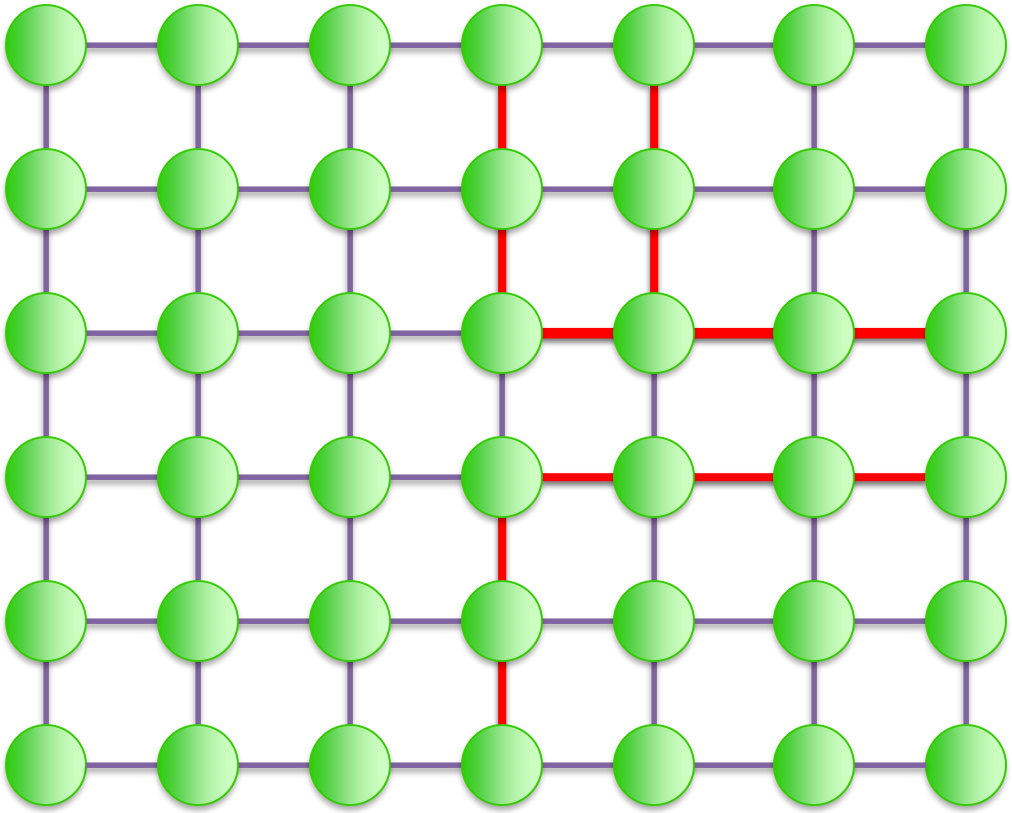
Compute nodes



Surface area of India: ~3.2 million sq. km.

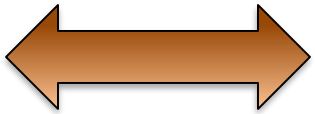
Data Bottleneck

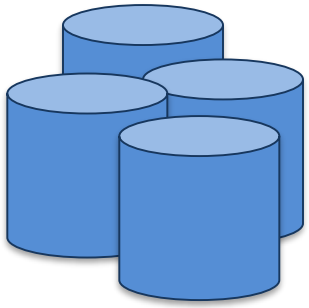
Congestion



Compute nodes

Read/write





Storage

Example: ‘Age’ is data

Average – Serial vs. Parallel

Serial

for i = 1 to N

sum += a[i]

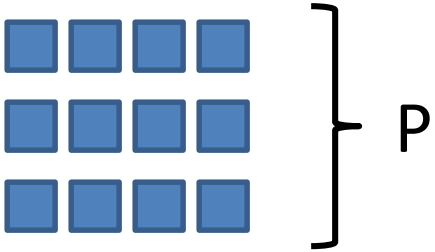
avg = sum/N

Parallel

for i = 1 to N/P

sum += a[i]

collect sums and compute

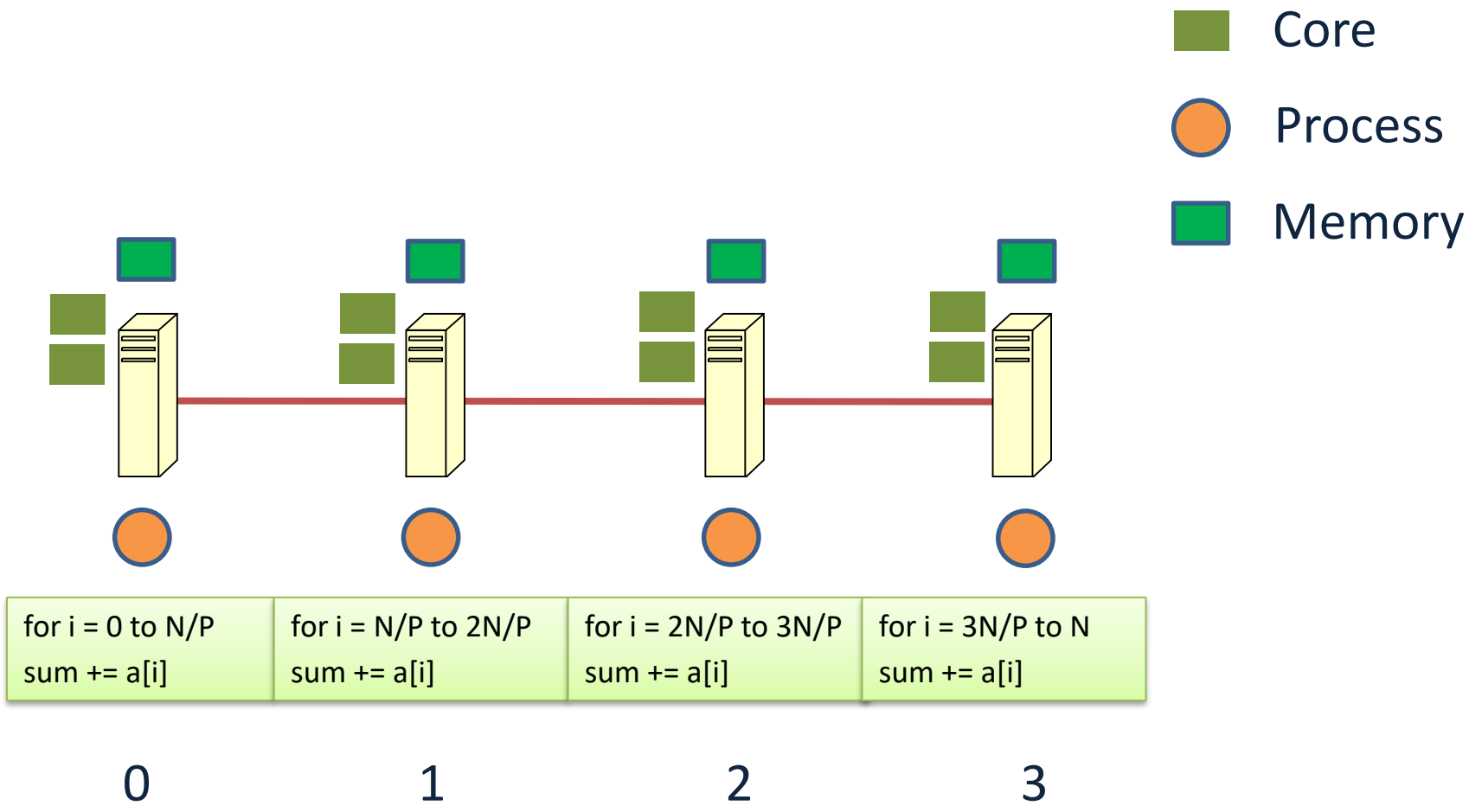


Parallel Computer

A parallel computer is a collection of processing elements that communicate and cooperate to solve large problems fast.

– Almasi and Gottlieb (1989)

Parallel Average



Parallel Code Example

```
// local computation at every process/thread
for i = N/P * id ; i < N/P * (id+1) ; i++
    localsum += a[i]

// collect localsum, add up in one of the ranks
  and compute average
```

Performance Measure

- Speedup

$$S_P = \frac{\text{Time (1 processor)}}{\text{Time (P processors)}}$$

- Efficiency

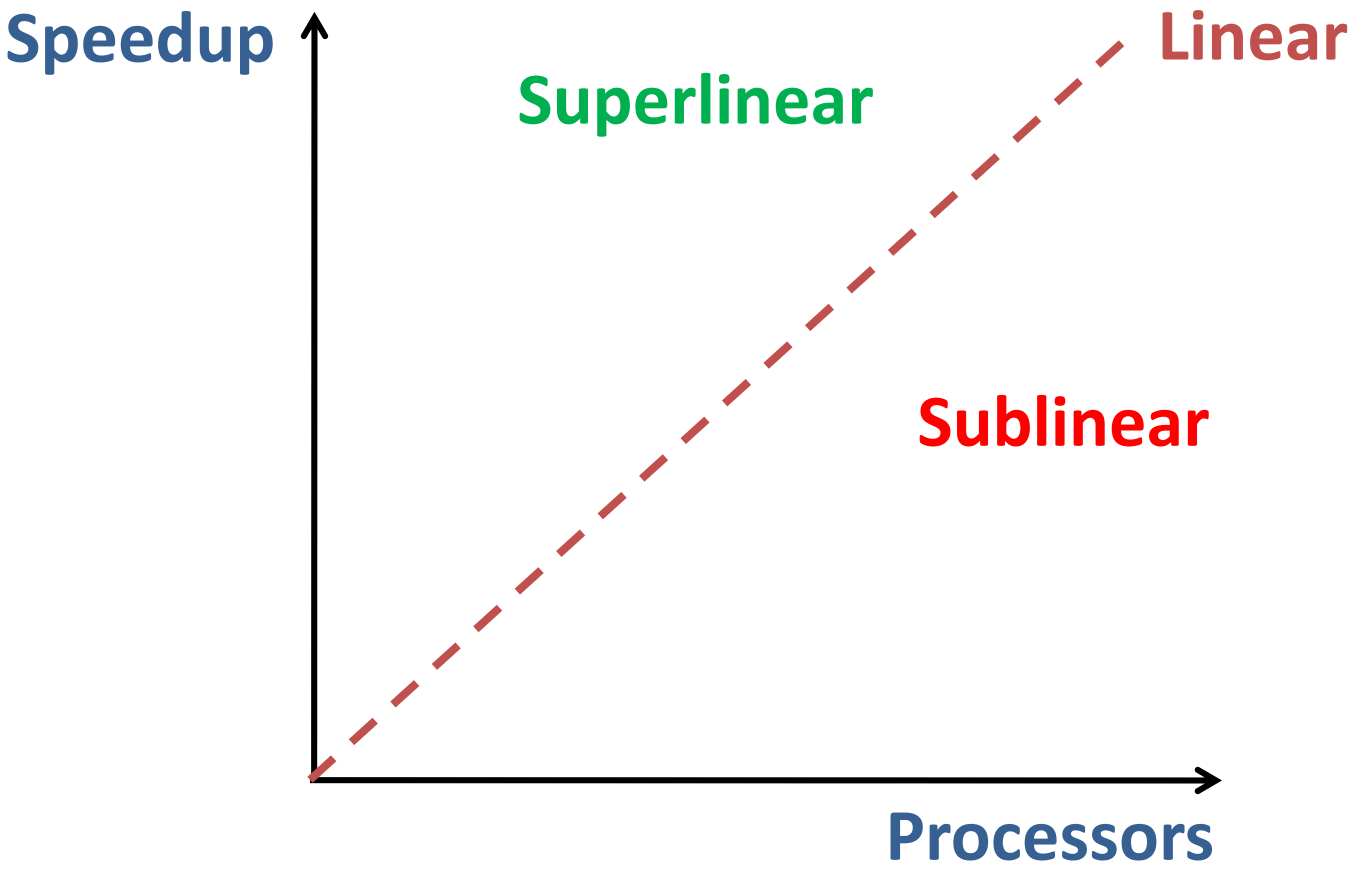
$$E_P = \frac{S_P}{P}$$

Parallel Performance (Parallel Sum)

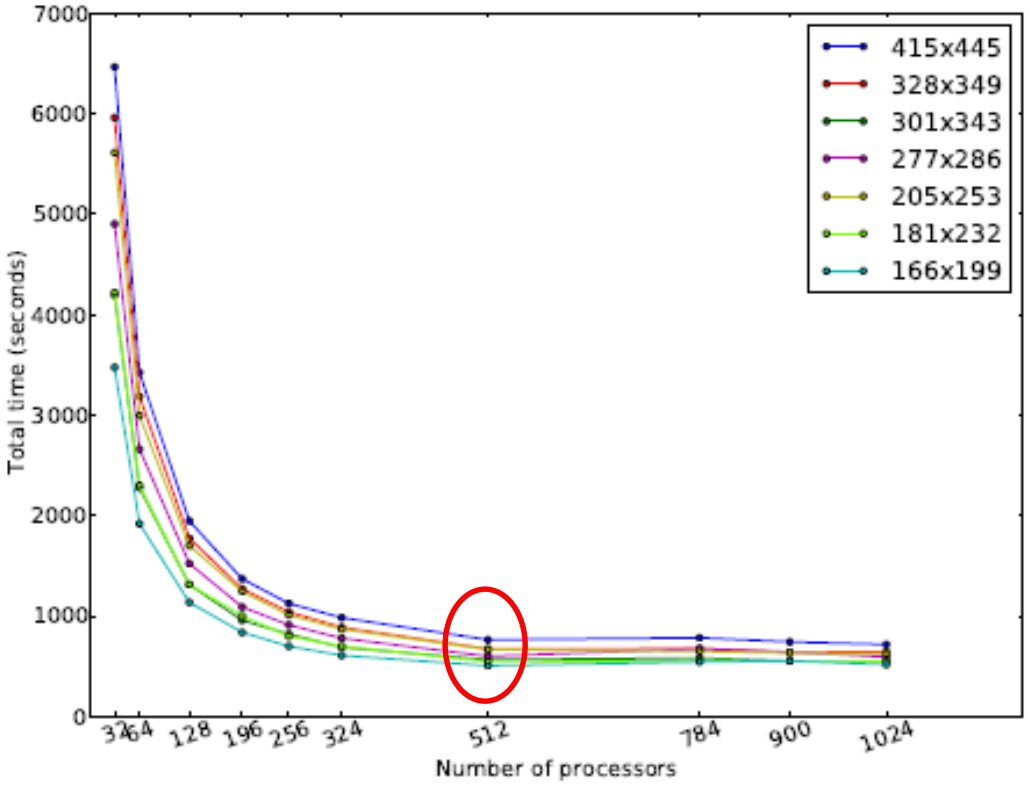
Parallel efficiency of summing 10^7 doubles

#Processes	Time (sec)	Speedup
1	0.025	1
2	0.013	1.9
4	0.010	2.5
8	0.009	2.8
12	0.007	3.6

Ideal Speedup



Scalability Bottleneck



Performance of weather simulation application

Programming

Parallel Programming Models

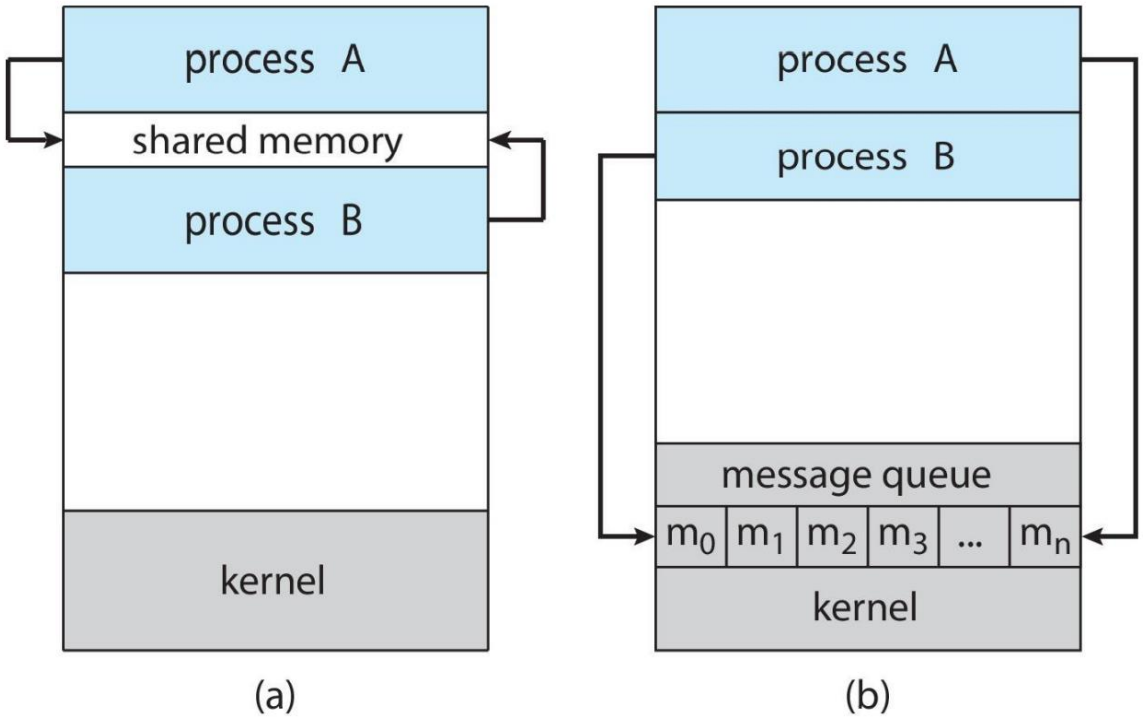
Libraries	MPI, TBB, Pthread, OpenMP, …
New languages	Haskell, X10, Chapel, …
Extensions	Coarray Fortran, UPC, Cilk, OpenCL, …

- Shared memory
  - OpenMP, Pthreads, CUDA, …
- Distributed memory
  - MPI, UPC, …
- Hybrid
  - MPI + OpenMP, MPI + CUDA

Sharing Data

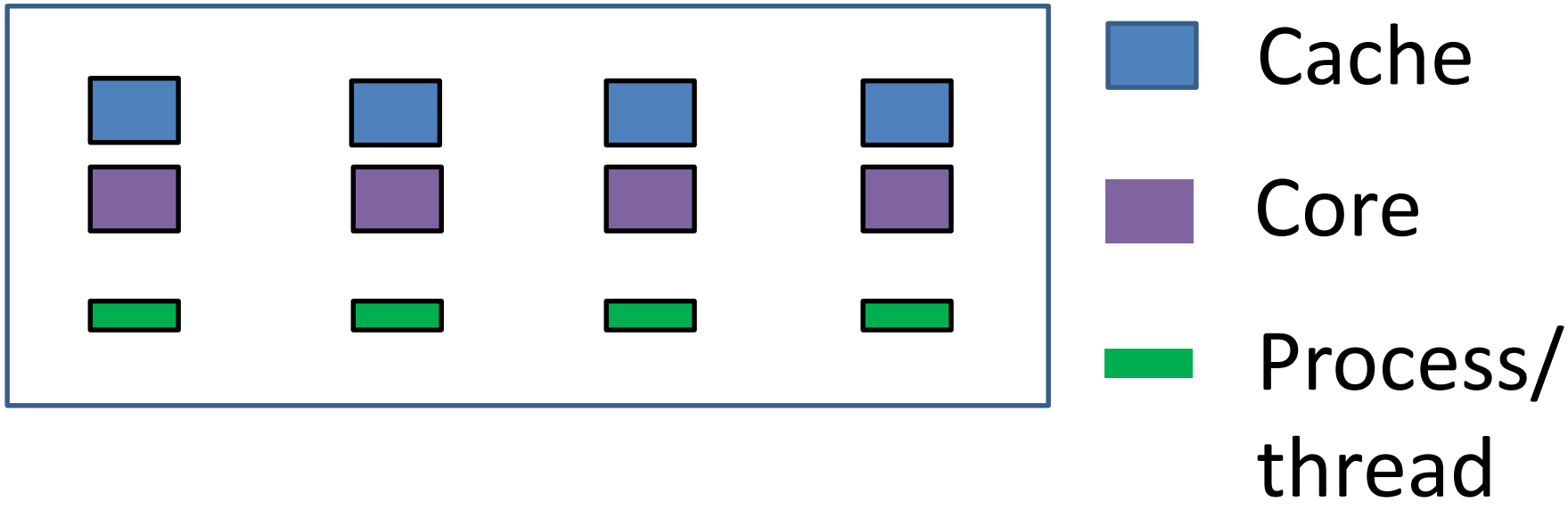
(a) Shared memory.

(b) Message passing.



Parallel Programming Models

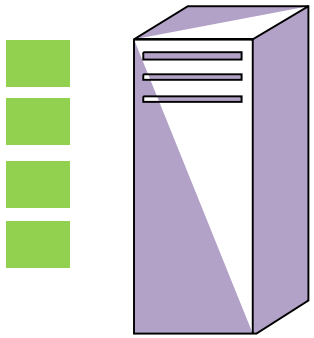
Shared memory programming – OpenMP, Pthreads
Distributed memory programming – MPI



Shared Memory Programming

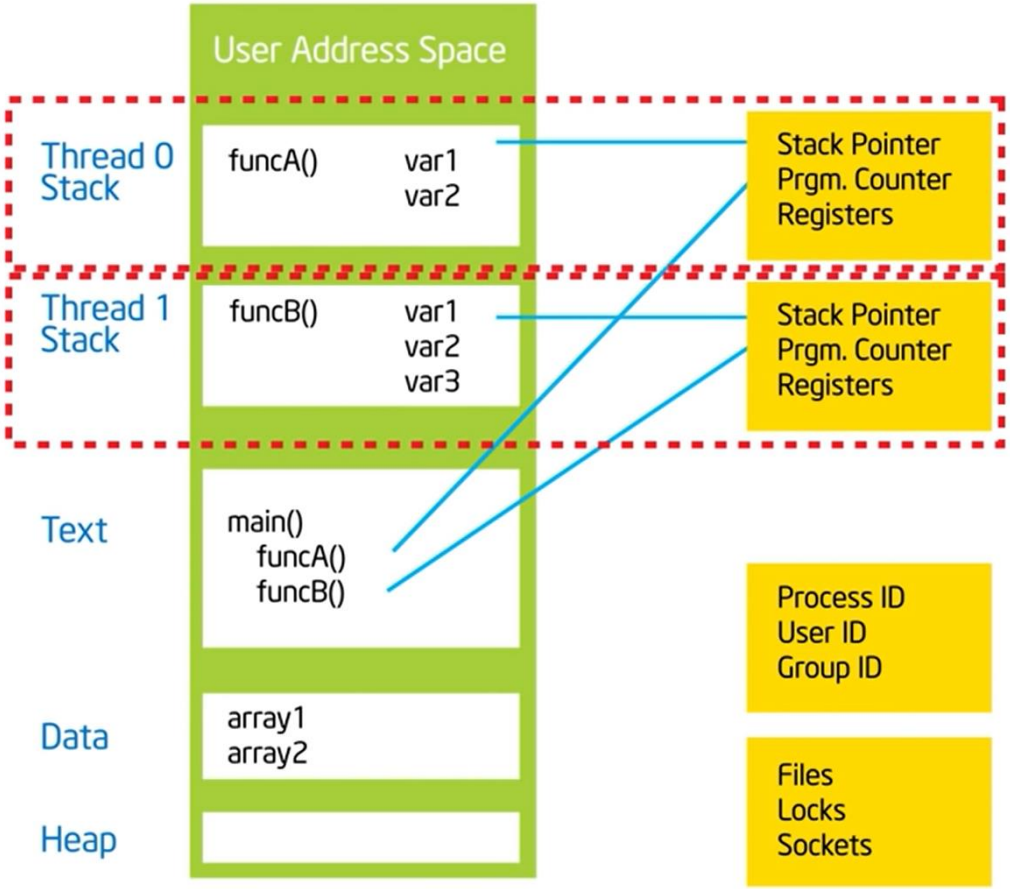
- Shared address space
- Time taken to access certain memory words is longer (NUMA)
- Programming paradigms – Pthreads, OpenMP
- Need to worry about concurrent access

Memory



CPU cores

Threads



Threads:

* Threads are "light weight processes"
* Threads share Process state among multiple threads. This greatly reduces the cost of switching context.

From Tim Mattson's slides

OpenMP (Open Multiprocessing)

- Standard for shared memory programming
  - Compiler directives
  - Runtime routines
  - Environment variables
- OpenMP Architecture Review Board
- First released in Nov’97
- Current version 5.1 (Nov’20)

OpenMP Example

- Thread-based
- Fork-join model

```
#pragma omp parallel  //fork
{



} //join
```

Spawn a default number of threads

OpenMP

```c
#include <stdio.h>
#include <omp.h>

int main() {

#pragma omp parallel num_threads(4)
{
    int num_threads = omp_get_num_threads();

    printf ("Hello world %d\n", num_threads);
}

  return 0;

}
```

$ gcc –fopenmp –o foo foo.c

OpenMP

```c
#include <stdio.h>
#include <omp.h>

int main() {

    omp_set_num_threads(4);

#pragma omp parallel
{
    int num_threads = omp_get_num_threads();

    printf ("Hello world %d\n", num_threads);
}

  return 0;

}
```

OpenMP

```c
#include <stdio.h>
#include <omp.h>

int main() {

#pragma omp parallel
{
    int num_threads = omp_get_num_threads();
    int thread_id = omp_get_thread_num();

    printf ("Hello from thread %d of %d\n", thread_id, num_threads);
}

  return 0;

}
```

Output

```
Hello from thread 8 of 12
Hello from thread 6 of 12
Hello from thread 0 of 12
Hello from thread 5 of 12
Hello from thread 10 of 12
Hello from thread 11 of 12
Hello from thread 4 of 12
Hello from thread 2 of 12
Hello from thread 1 of 12
Hello from thread 9 of 12
Hello from thread 3 of 12
Hello from thread 7 of 12
```

```
Hello from thread 11 of 12
Hello from thread 10 of 12
Hello from thread 5 of 12
Hello from thread 8 of 12
Hello from thread 9 of 12
Hello from thread 1 of 12
Hello from thread 2 of 12
Hello from thread 3 of 12
Hello from thread 0 of 12
Hello from thread 4 of 12
Hello from thread 6 of 12
Hello from thread 7 of 12
```

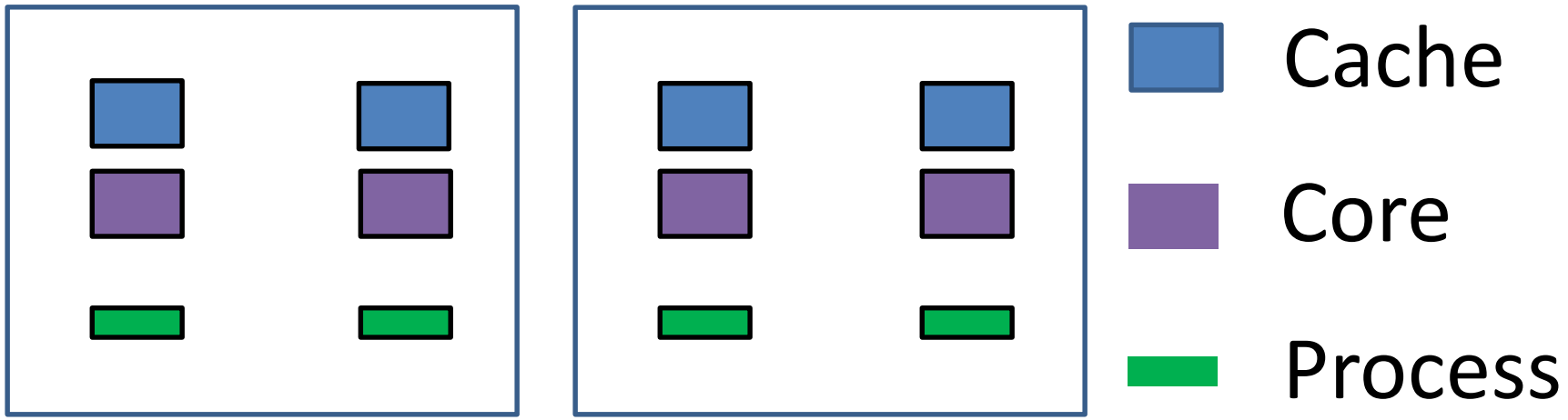
OpenMP – Parallel Sum

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

int main(int argc, char **argv) {

    int thread_id, num_threads, data_per_thread;
    int N = atoi (argv[1]);
    int x[N];
    time_t t;

    srand((unsigned) time(&t));

    for (int i=0; i<N; i++) x[i] = rand()%10;

#pragma omp parallel private (thread_id)
{
    num_threads = omp_get_num_threads();
    thread_id = omp_get_thread_num();









}

  return 0;

}
```

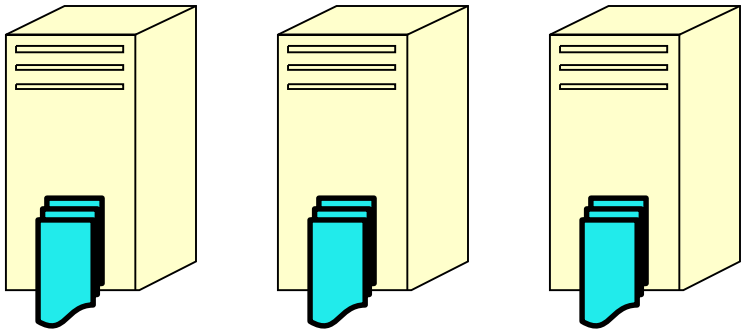
OpenMP Timing

```
double stime = omp_get_wtime();
#pragma omp parallel
{
   …
}
double etime = omp_get_wtime();
```

Multiple Systems



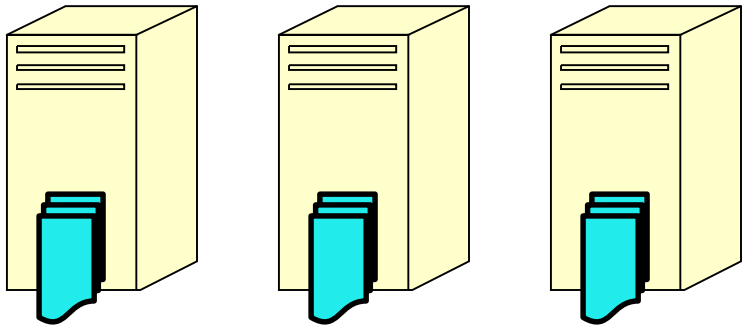
Distributed Memory Systems



Node

64 – 192 GB RAM/node

- Networked systems
- Distributed memory
  - Local memory
  - Remote memory
- Parallel file system

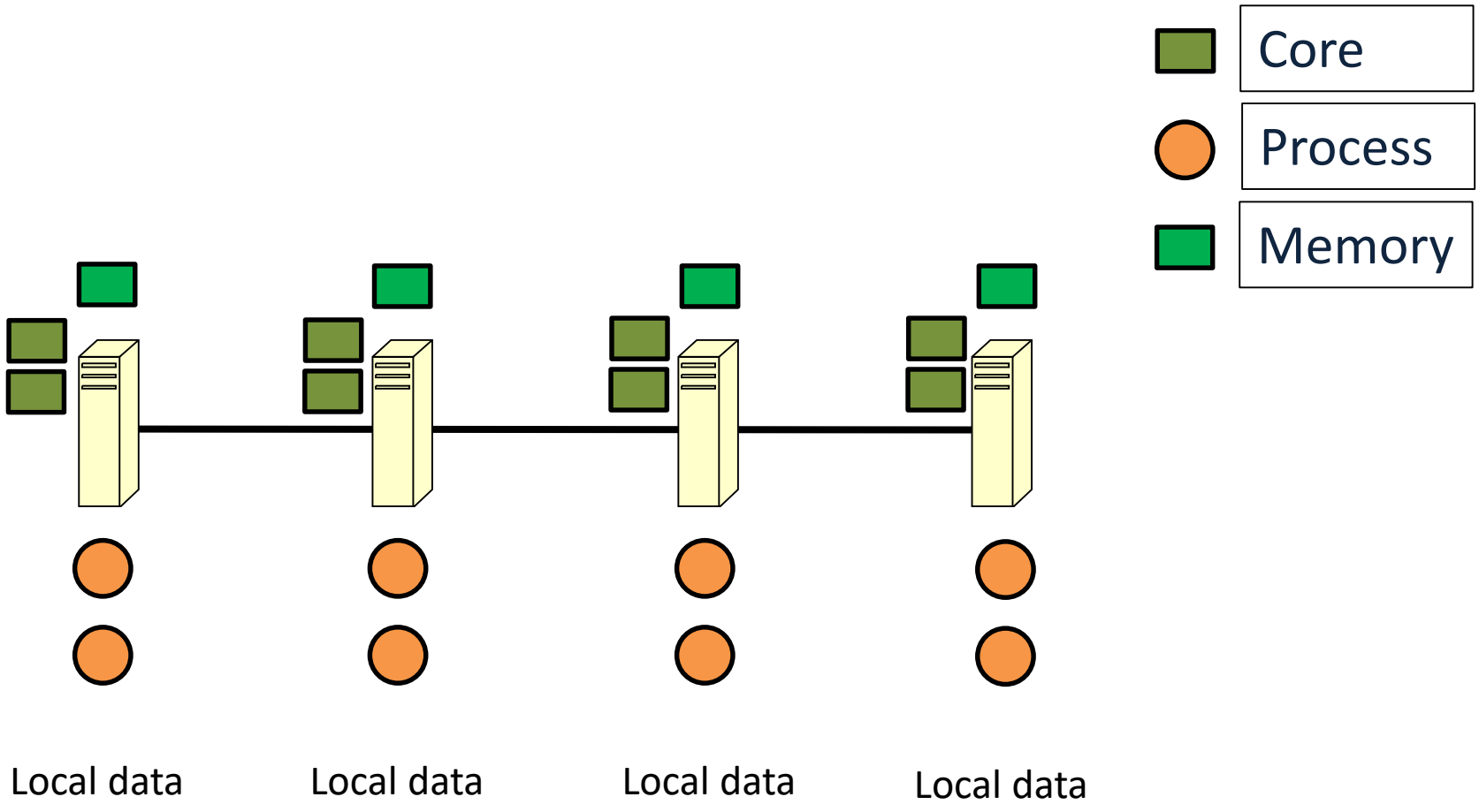


Cluster

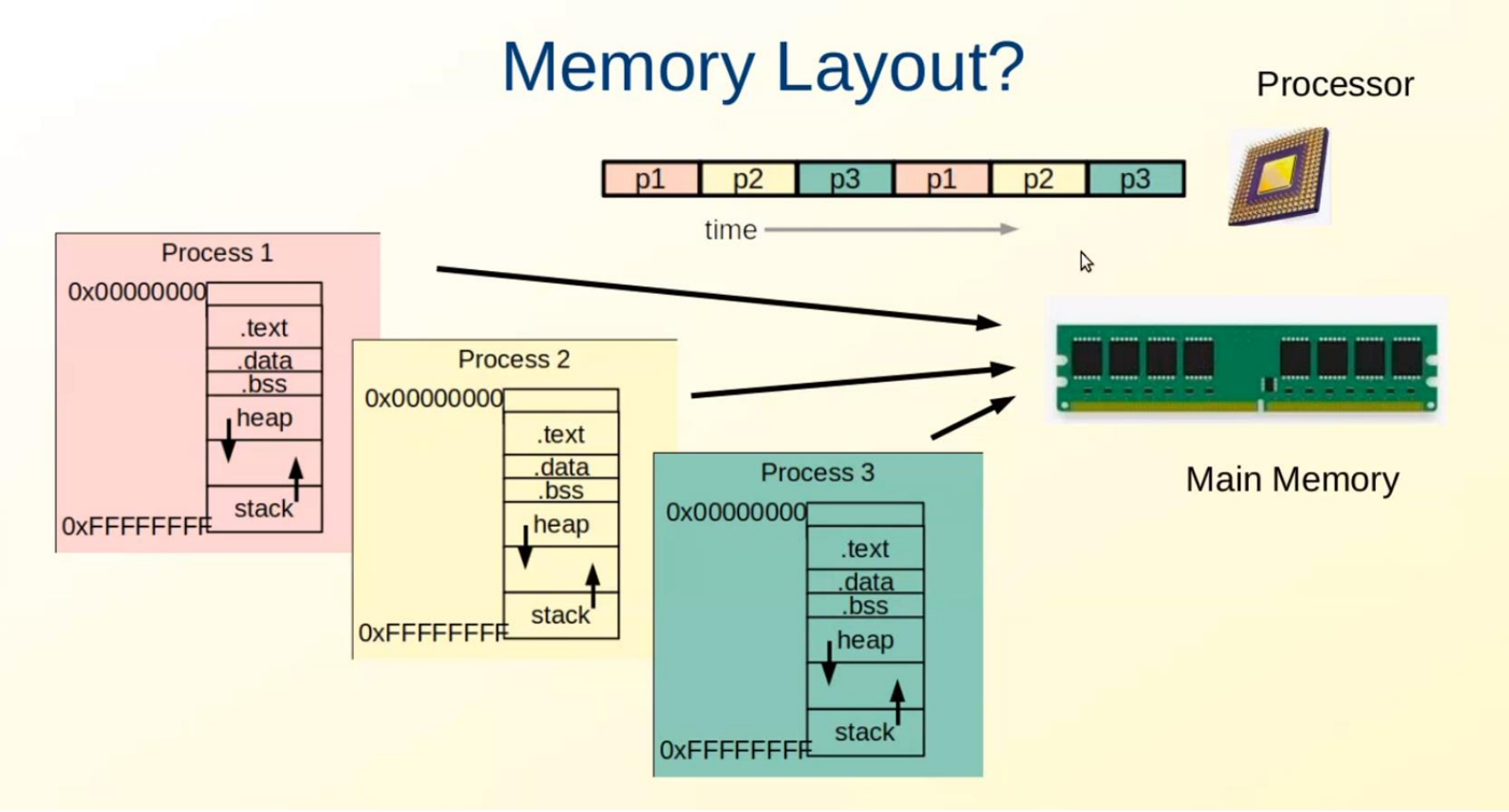
MPI (Message Passing Interface)

- Standard for message passing in a distributed memory environment (most widely used programming model in supercomputers)
- Efforts began in 1991 by Jack Dongarra, Tony Hey, and David W. Walker
- MPI Forum formed in 1993
  - Version 1.0: 1994
  - Version 4.0: 2021

Process - Distinct Address Space

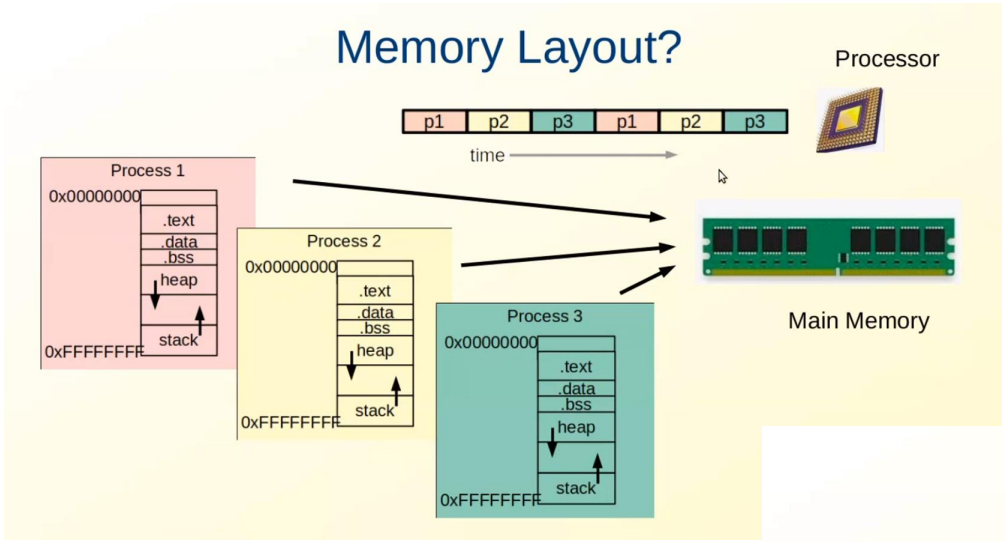


Multiple Processes on a Single Node

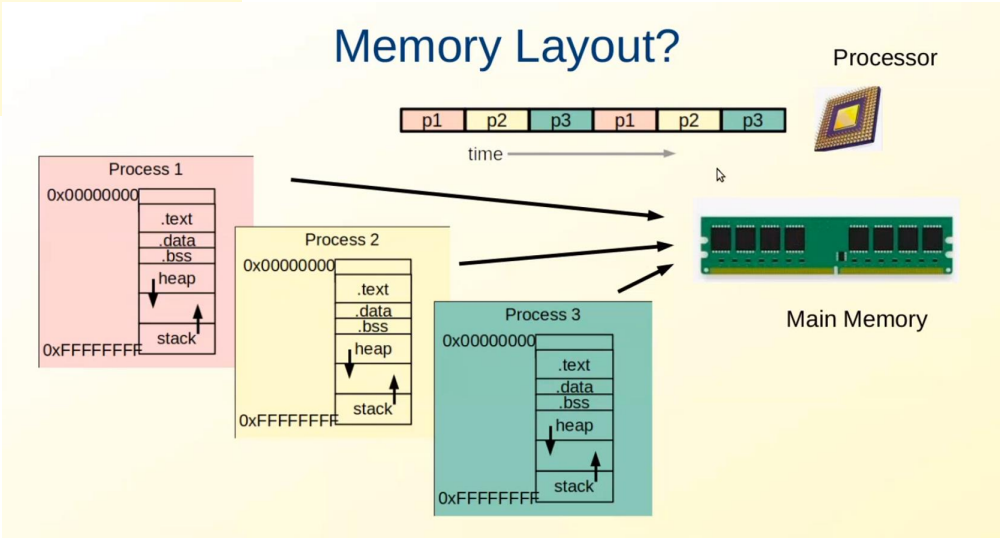


From N. Karanjkar's slides

Multiple Processes on Multiple Nodes

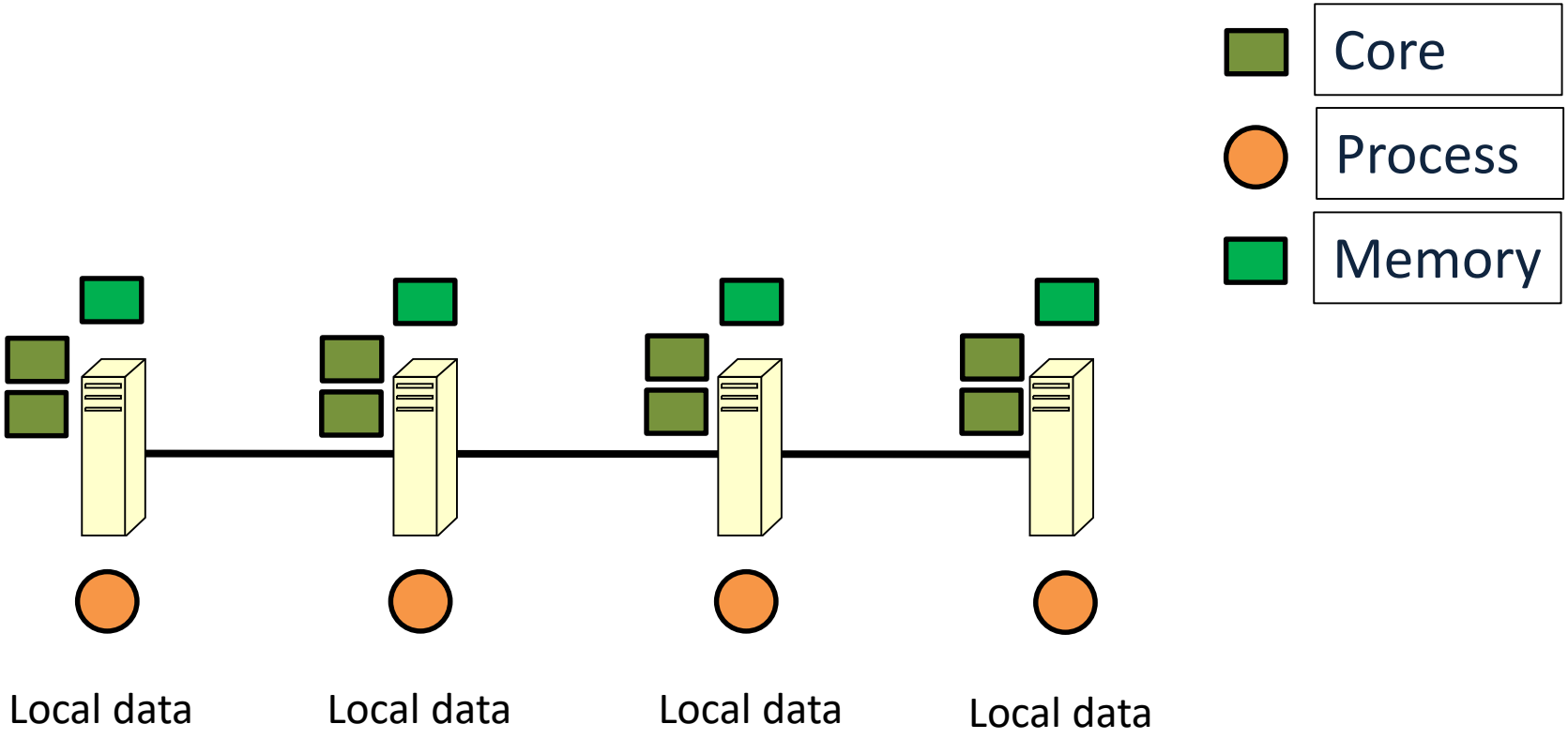


Node 1

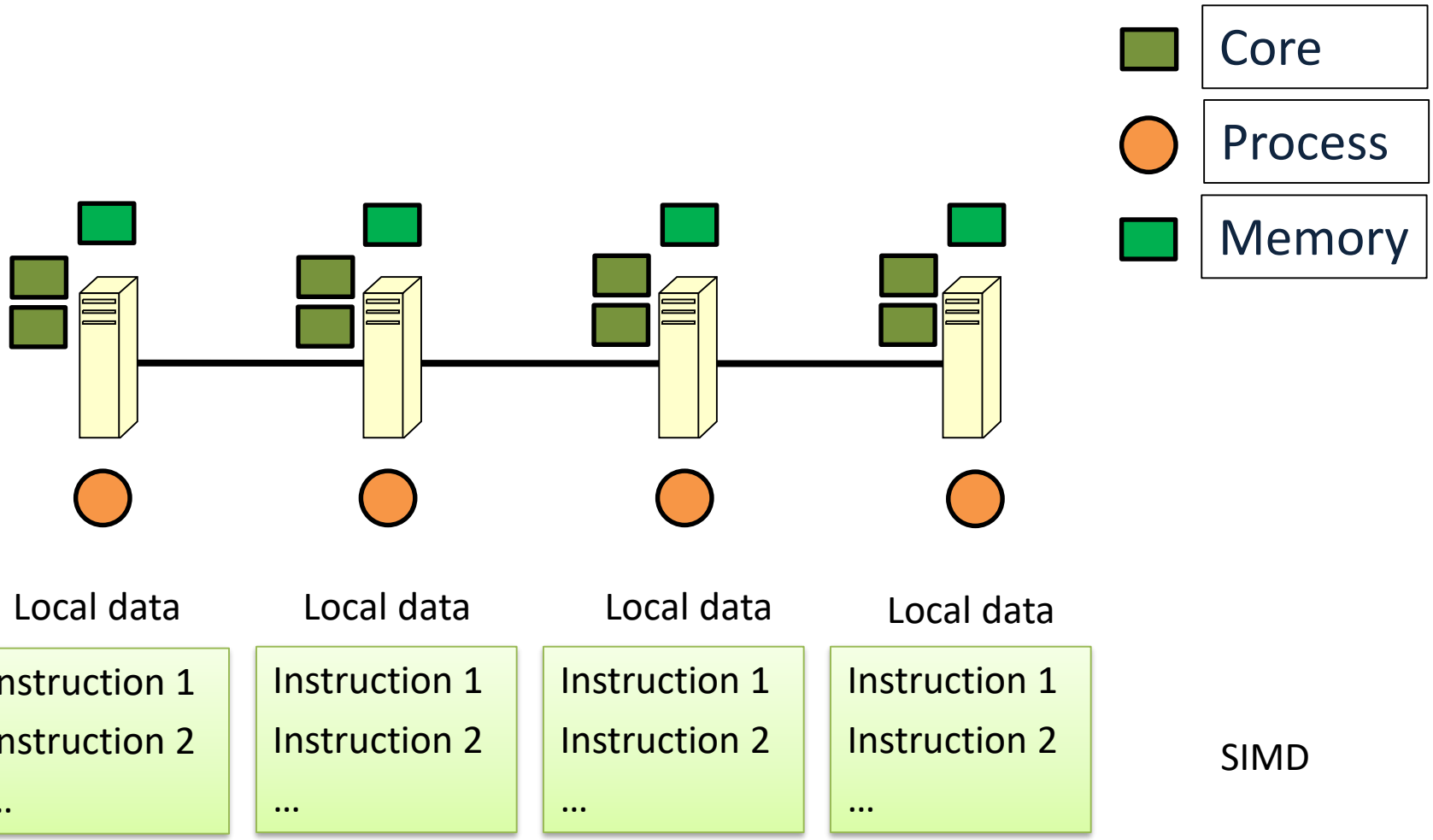


Node 2

Communication using Messages

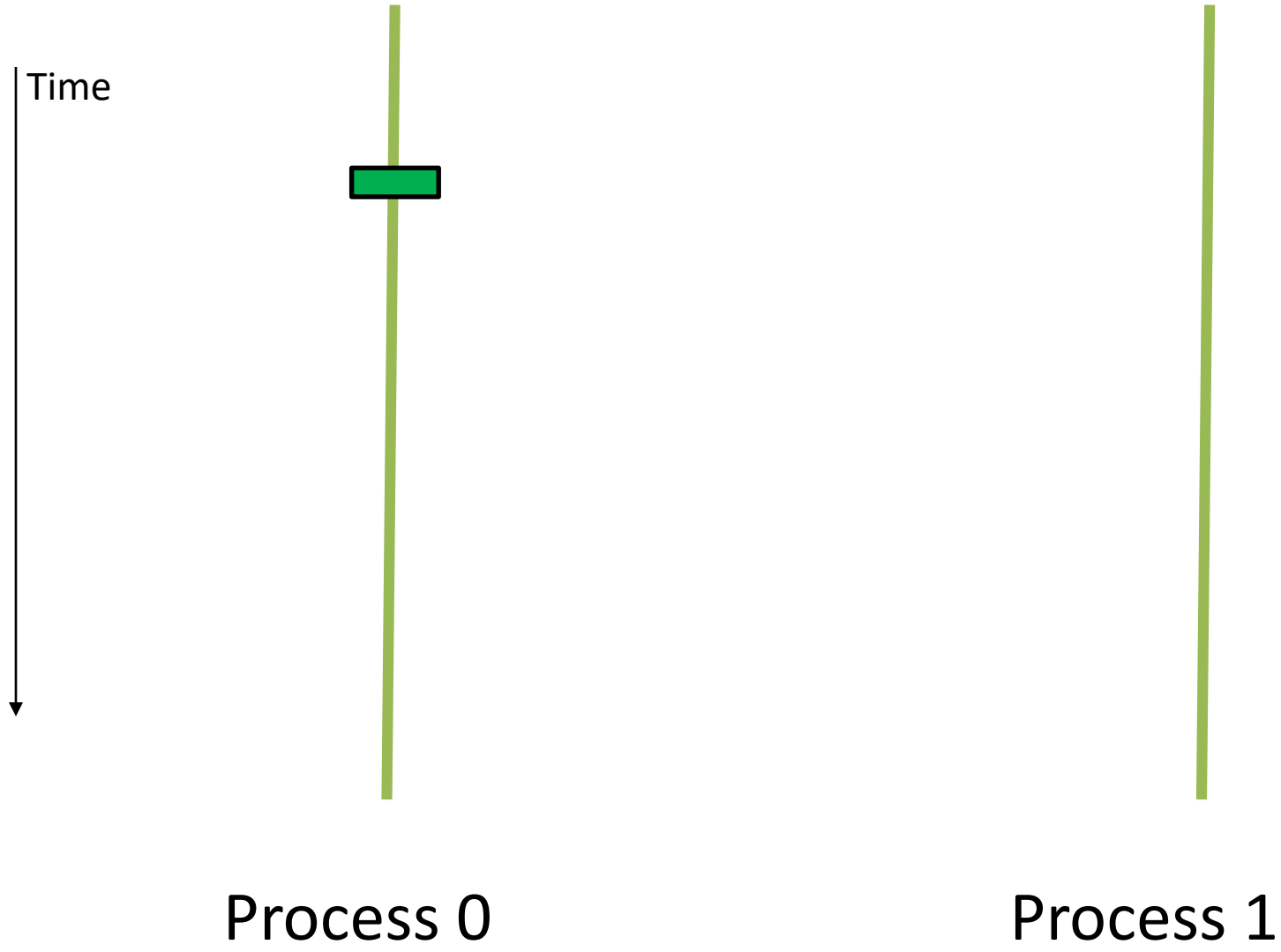


Communication using Messages



SIMD

Message Passing



MPI Programming

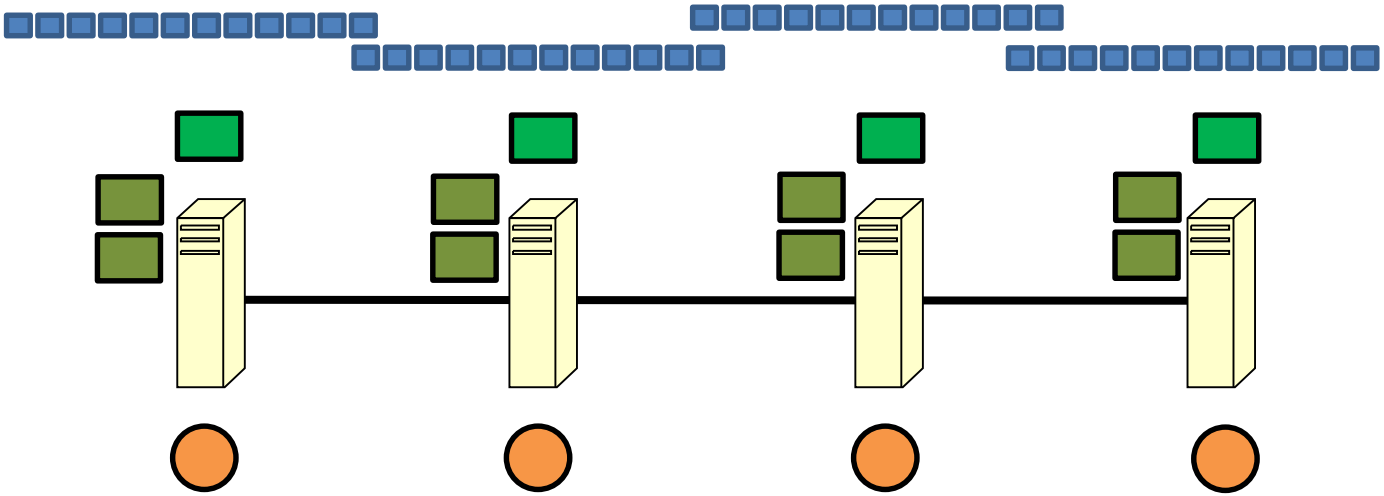
```
#include <stdio.h>
#include "mpi.h"

int main(int argc, char *argv[])
{
  // initialize MPI
  MPI_Init (&argc, &argv);

  printf ("Hello, world!\n");

  // done with MPI
  MPI_Finalize();
}

~
~
```

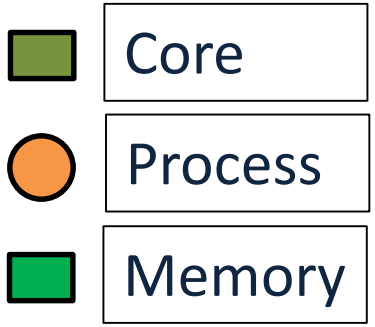
MPI Programming

```c
#include <stdio.h>
#include "mpi.h"

int main(int argc, char *argv[])
{
  int numtasks, rank, len;
  char hostname[MPI_MAX_PROCESSOR_NAME];

  // initialize MPI
  MPI_Init (&argc, &argv);

  // get number of tasks
  MPI_Comm_size (MPI_COMM_WORLD, &numtasks);

  // get my rank
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);

  // this one is obvious
  MPI_Get_processor_name (hostname, &len);

  printf ("Number of tasks=%d My rank=%d Running on %s\n", numtasks, rank, hostname);

  // done with MPI
  MPI_Finalize();
}
```

mpicc -o program.x program.c

Communication using Messages



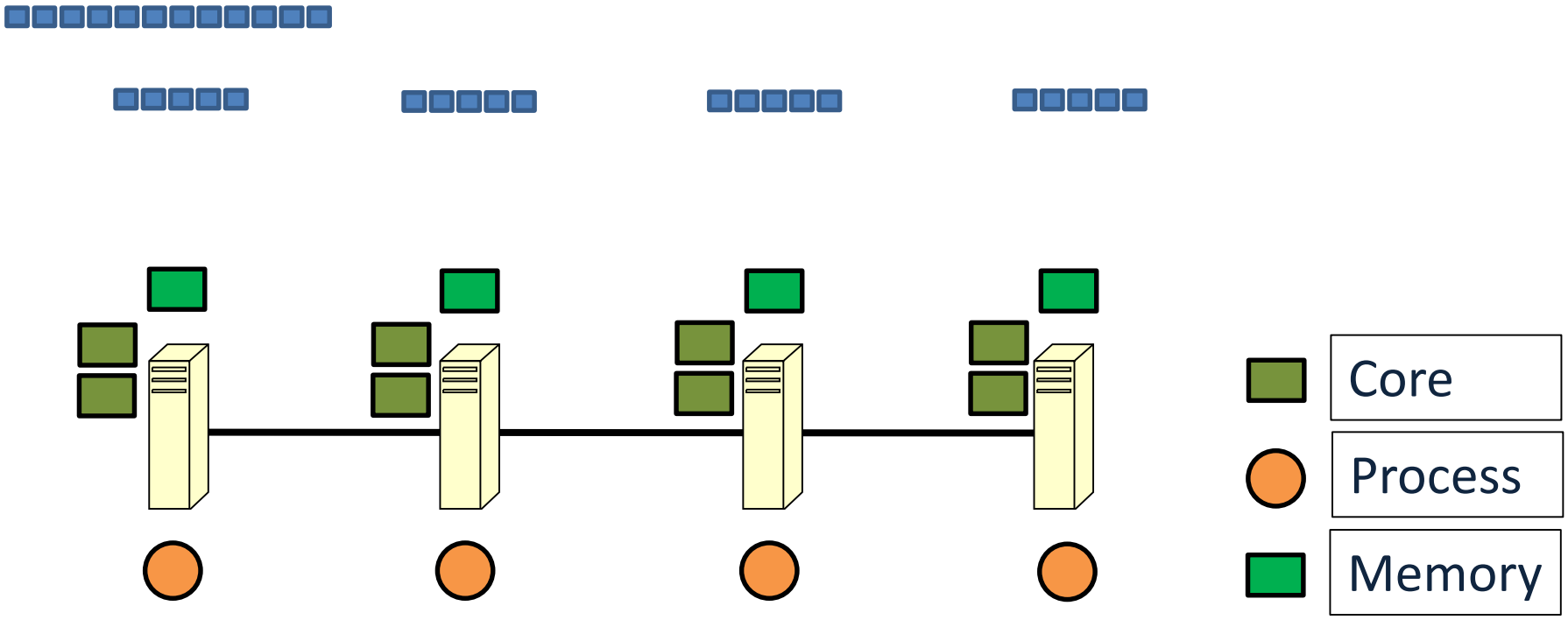


for i = 0 to N/P sum += a[i]	for i = N/P to 2N/P sum += a[i]	for i = 2N/P to 3N/P sum += a[i]	for i = 3N/P to N sum += a[i]

```
for i = N/P * rank ; i < N/P * (rank+1) ; i++
        localsum += a[i]
```

Collect localsum, add up at one of the ranks

Communication using Messages

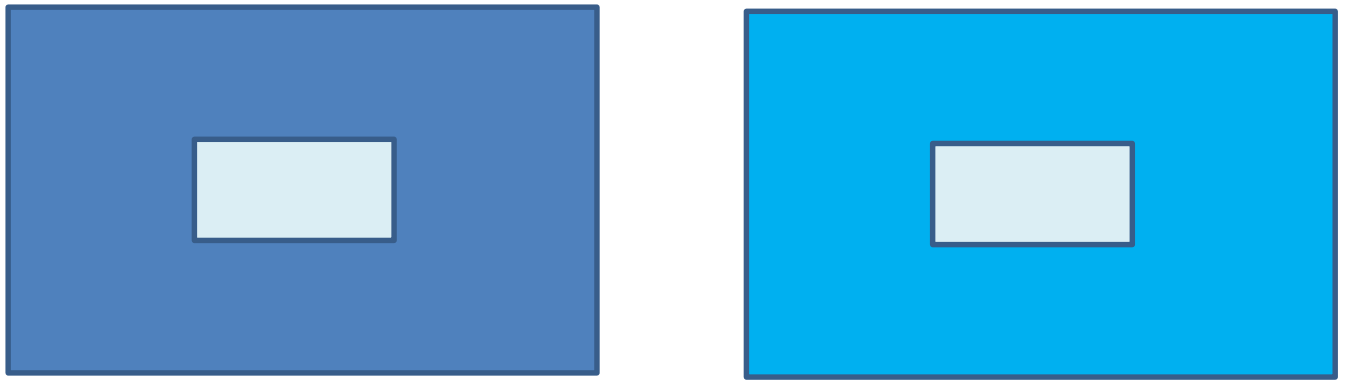


Simplest Communication Primitives

- MPI_Send
- MPI_Recv

MPI Programming

int MPI_Send (const void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)



int MPI_Recv (void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)

MPI Programming

```
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);

// Sender process
if (myrank == 0)    /* code for process 0 */
{
  strcpy (message,"Hello, there");
  MPI_Send (message, strlen(message)+1, MPI_CHAR, 1, 99,
MPI_COMM_WORLD);
}

// Receiver process
else if (myrank == 1)  /* code for process 1 */
{
  MPI_Recv (message, 20, MPI_CHAR, 0, 99, MPI_COMM_WORLD,
&status);
  printf ("received :%s\n", message);
}
```

MPI – Parallel Sum

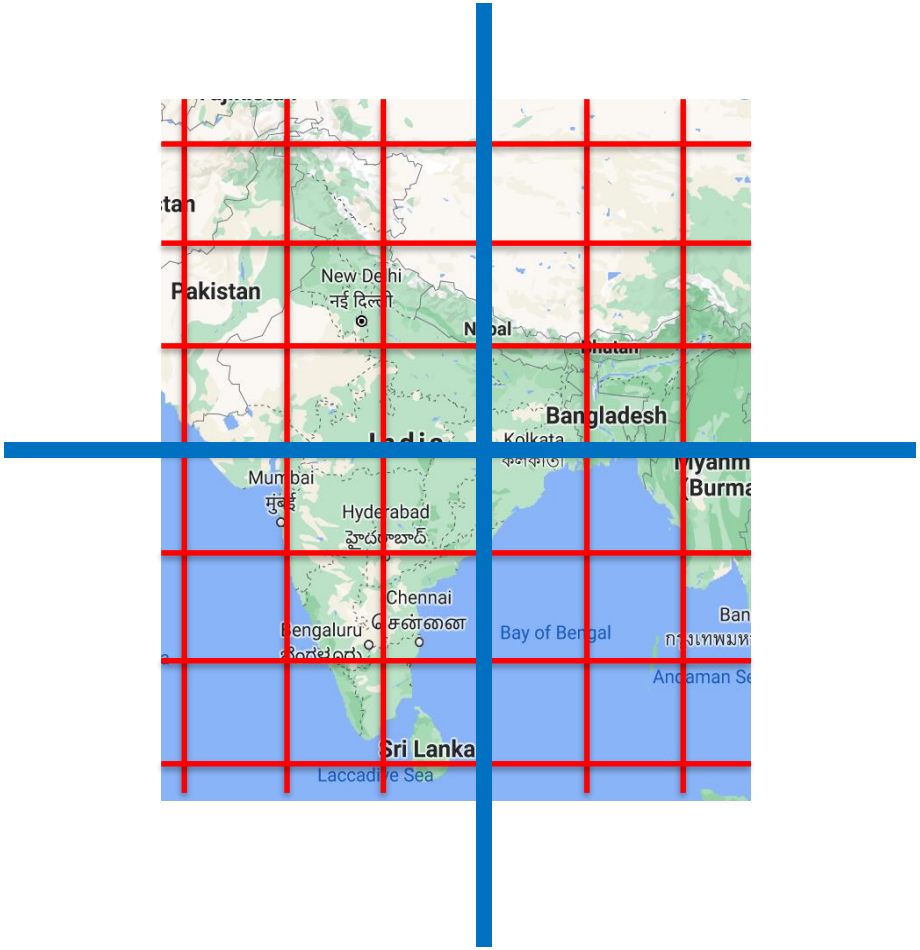
Assume the data array resides in the memory of process 0 initially

```
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);

// Sender process
if (myrank == 0)  /* code for process 0 */
{
 for (int rank=1; rank<SIZE ; rank++) {
   start = rank*N/size*sizeof(int);
   MPI_Send (data+start, N/size, MPI_INT, rank, 99, MPI_COMM_WORLD);
 }
}
else /* code for processes 1 … SIZE */
{
  MPI_Recv (data, N/size, MPI_CHAR, 0, 99, MPI_COMM_WORLD, &status);
}
```

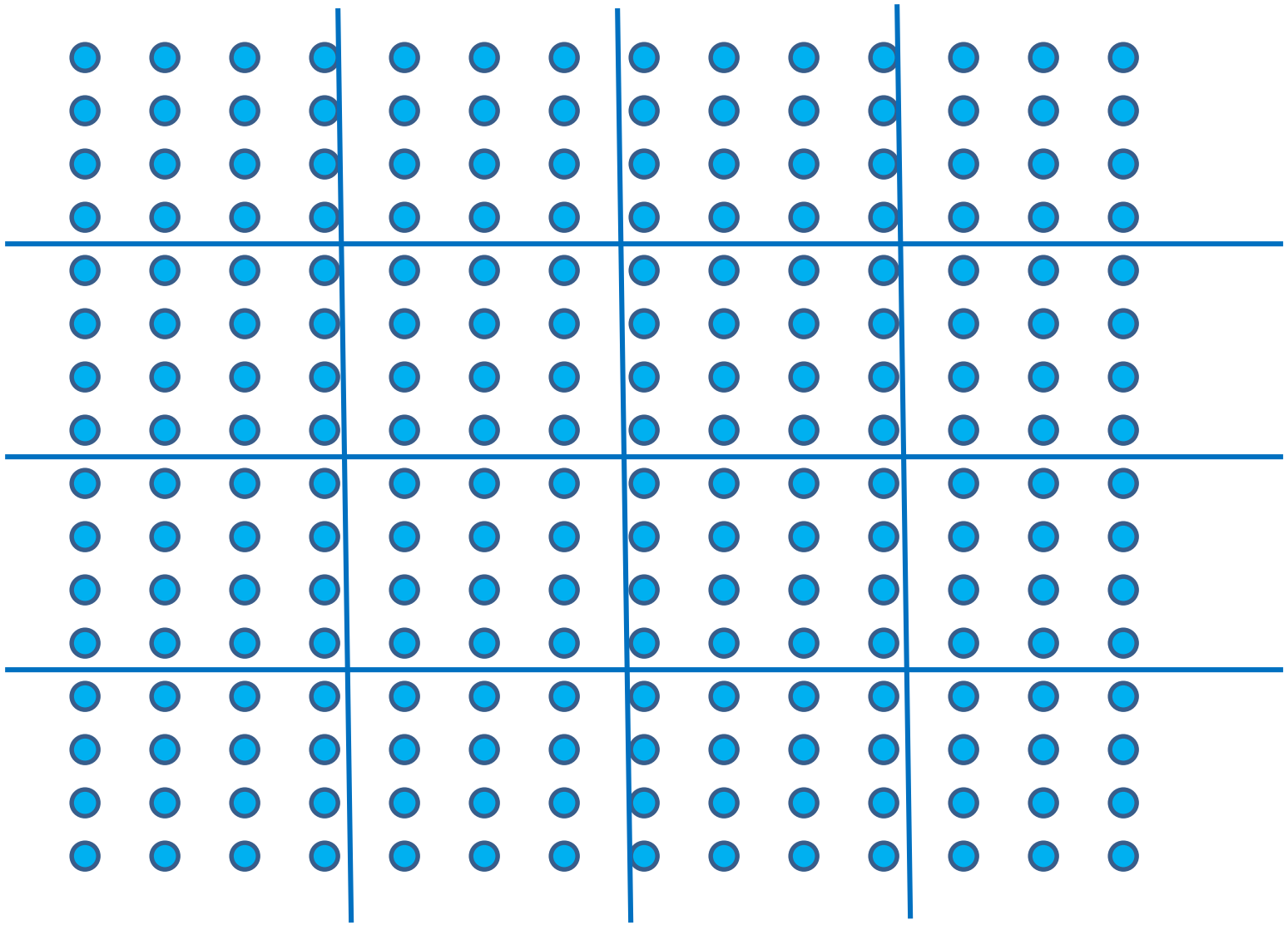
MPI Timing

```
double stime = MPI_Wtime();
  …
  …
  …
double etime = MPI_Wtime();
```

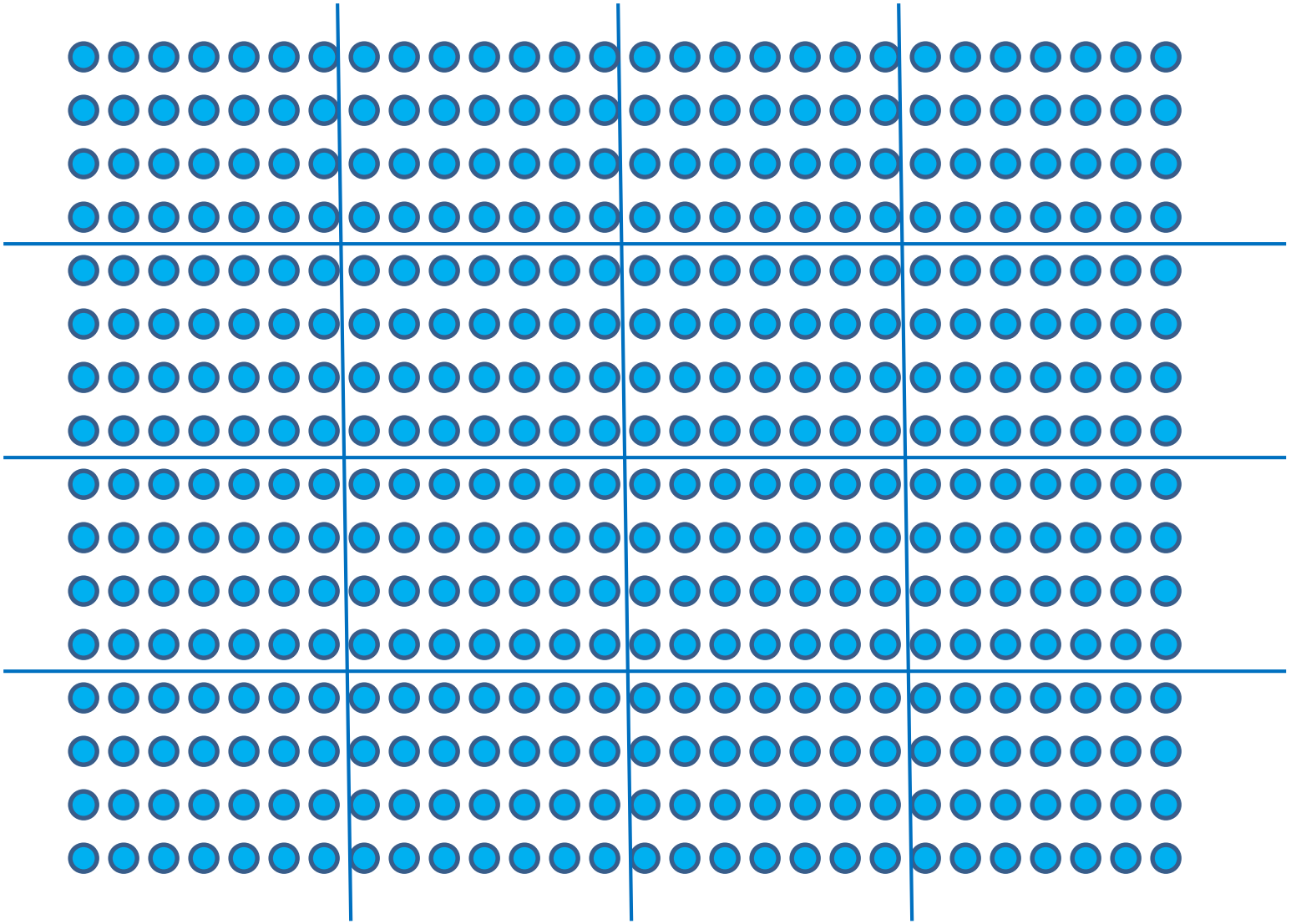
Performance vs. Accuracy



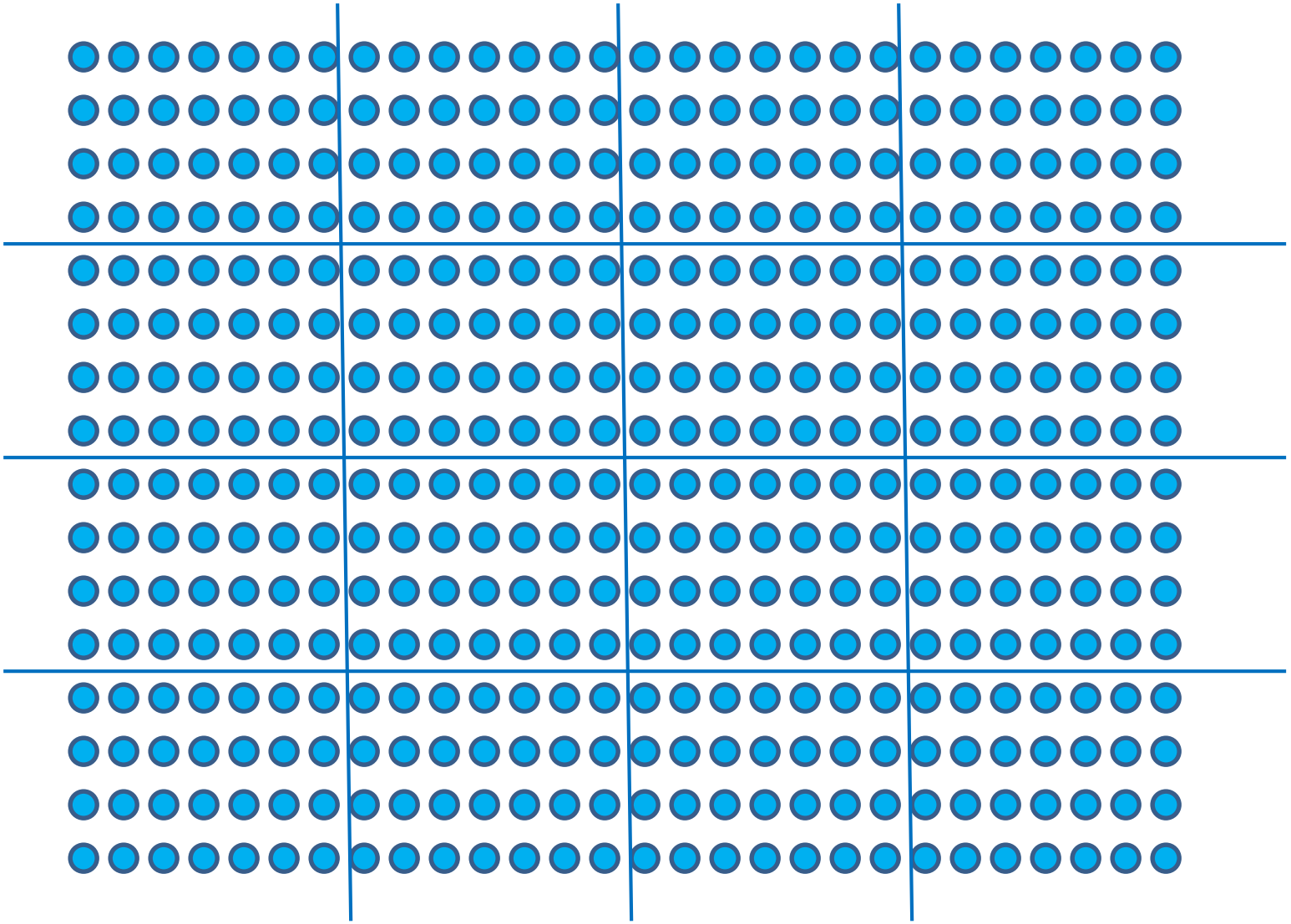
Interpolation



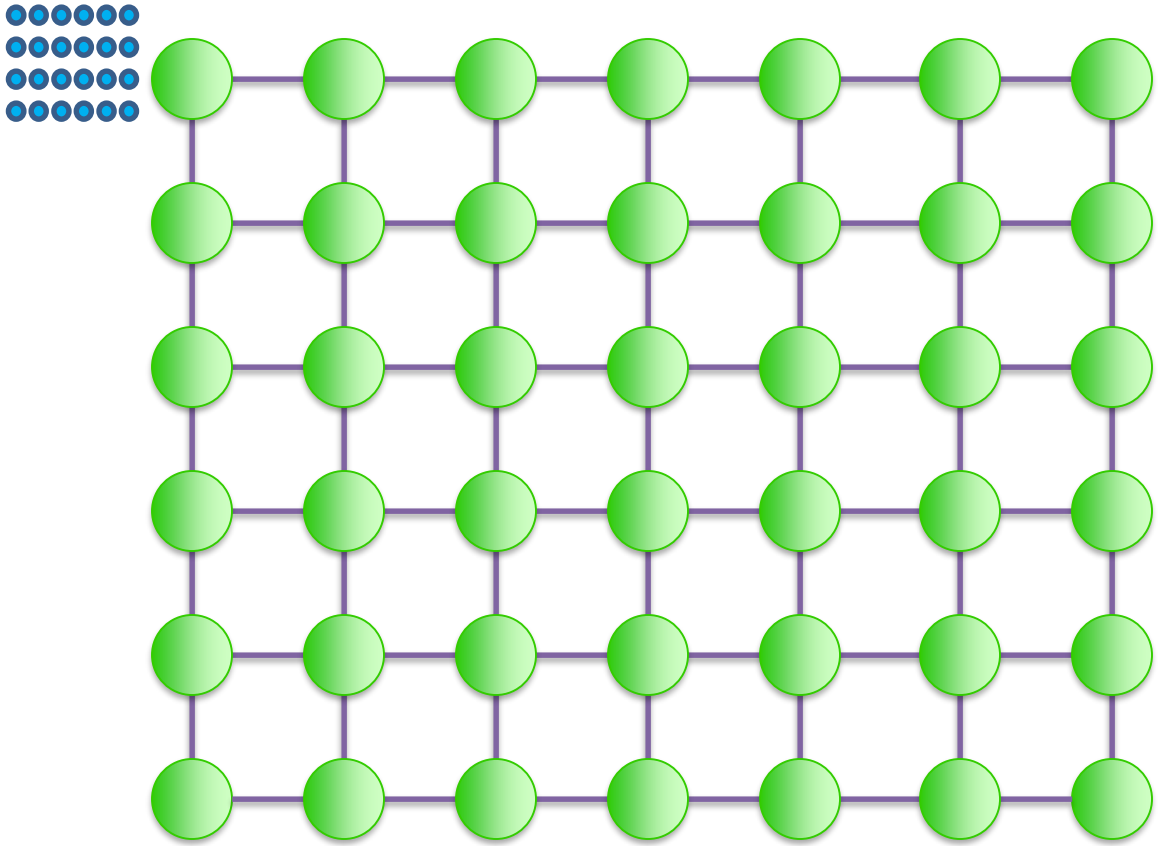
Interpolation



Range/Query

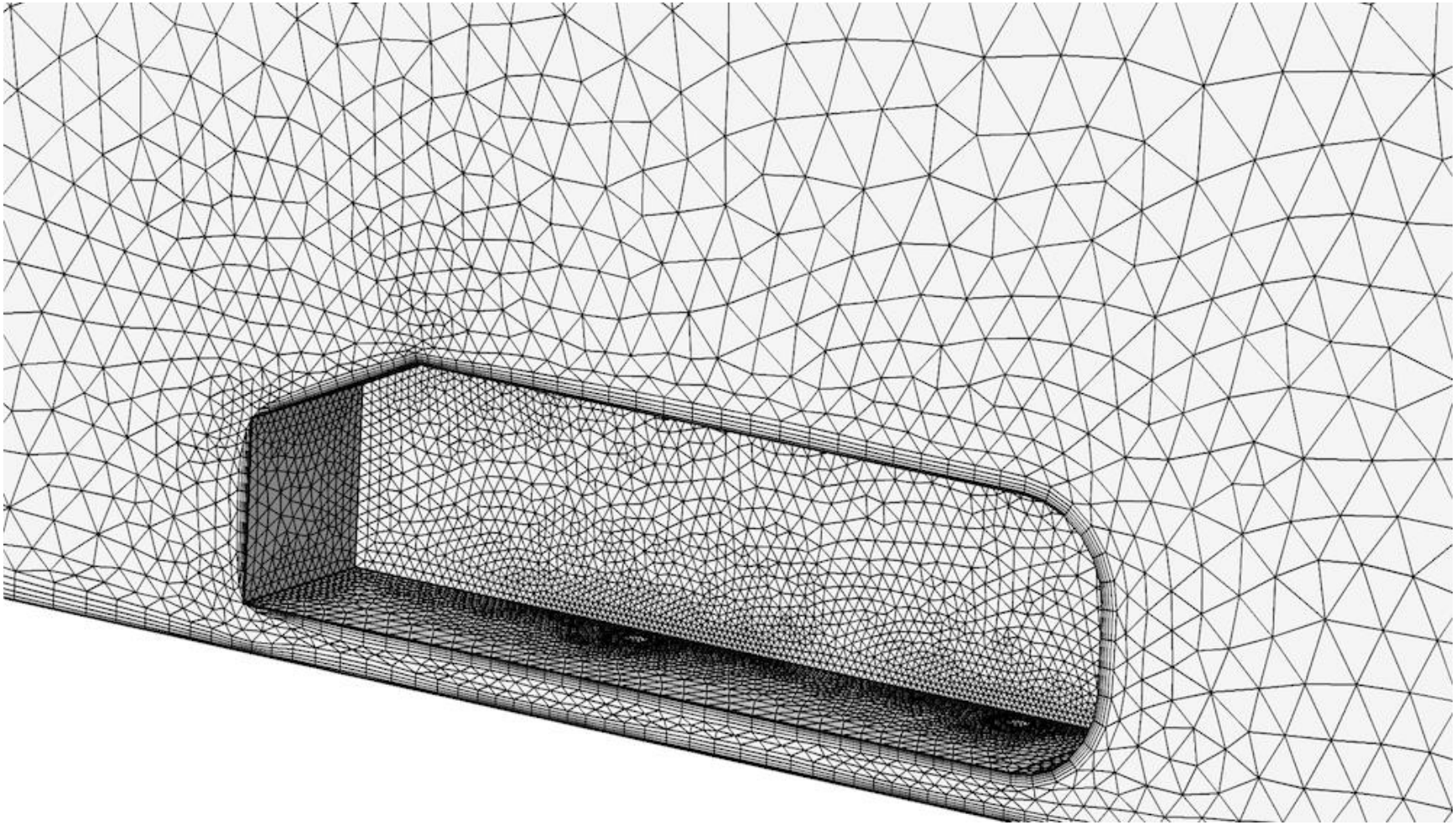


Query on a Million Processes



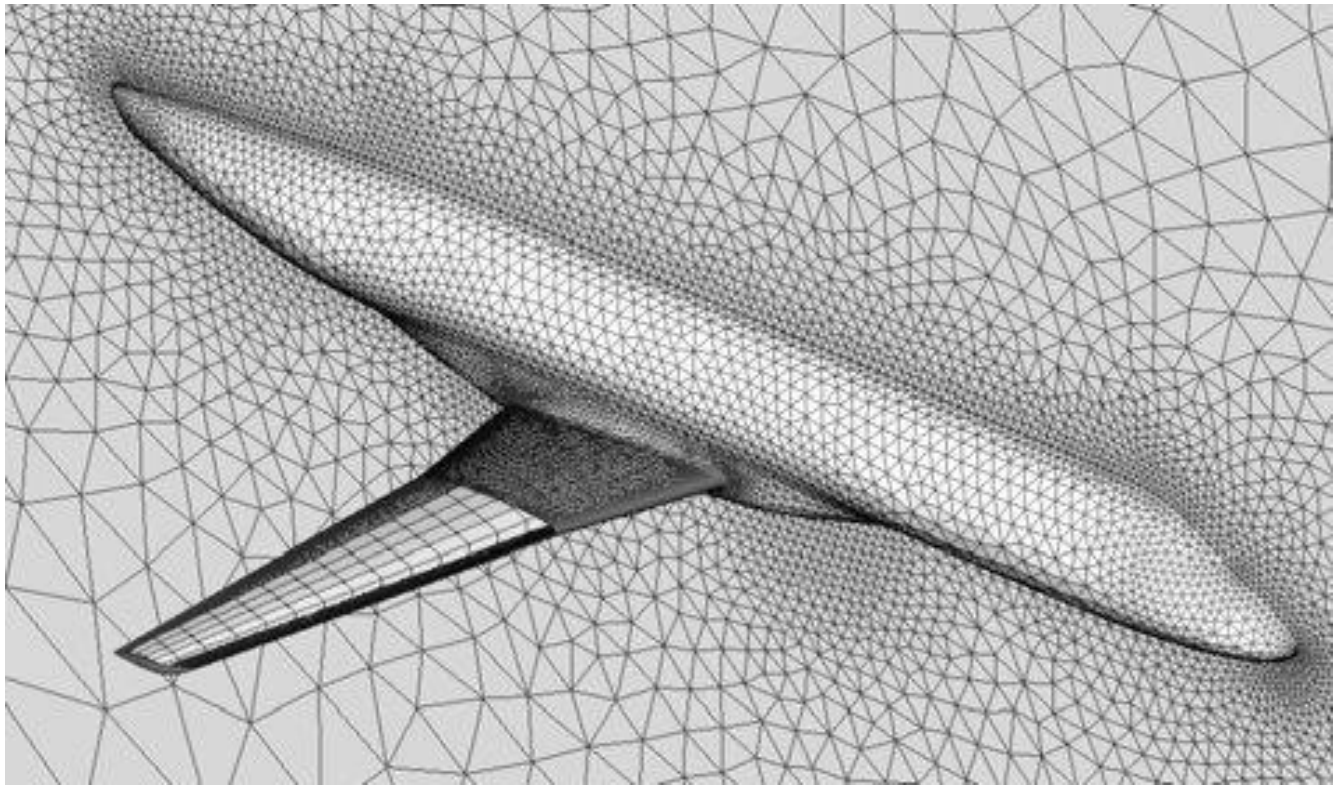
Compute nodes

Unstructured Mesh



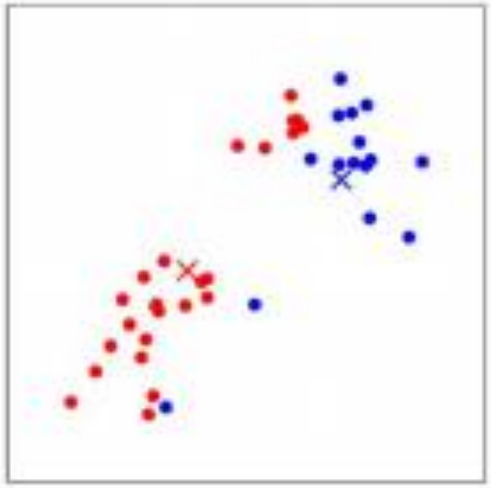
Source: COMSOL

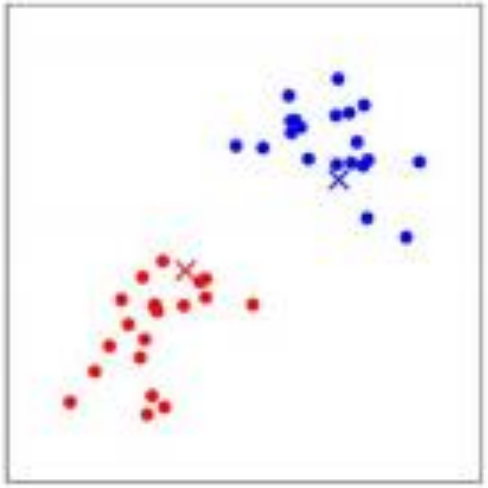
Unstructured Mesh



Obayashi et al., Multi-objective Design Exploration Using Efficient Global Optimization

Clustering Example





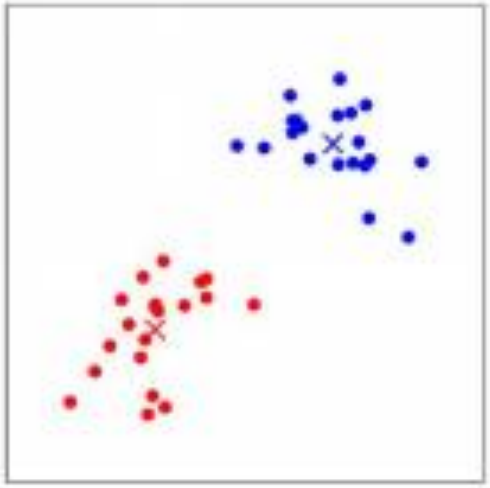
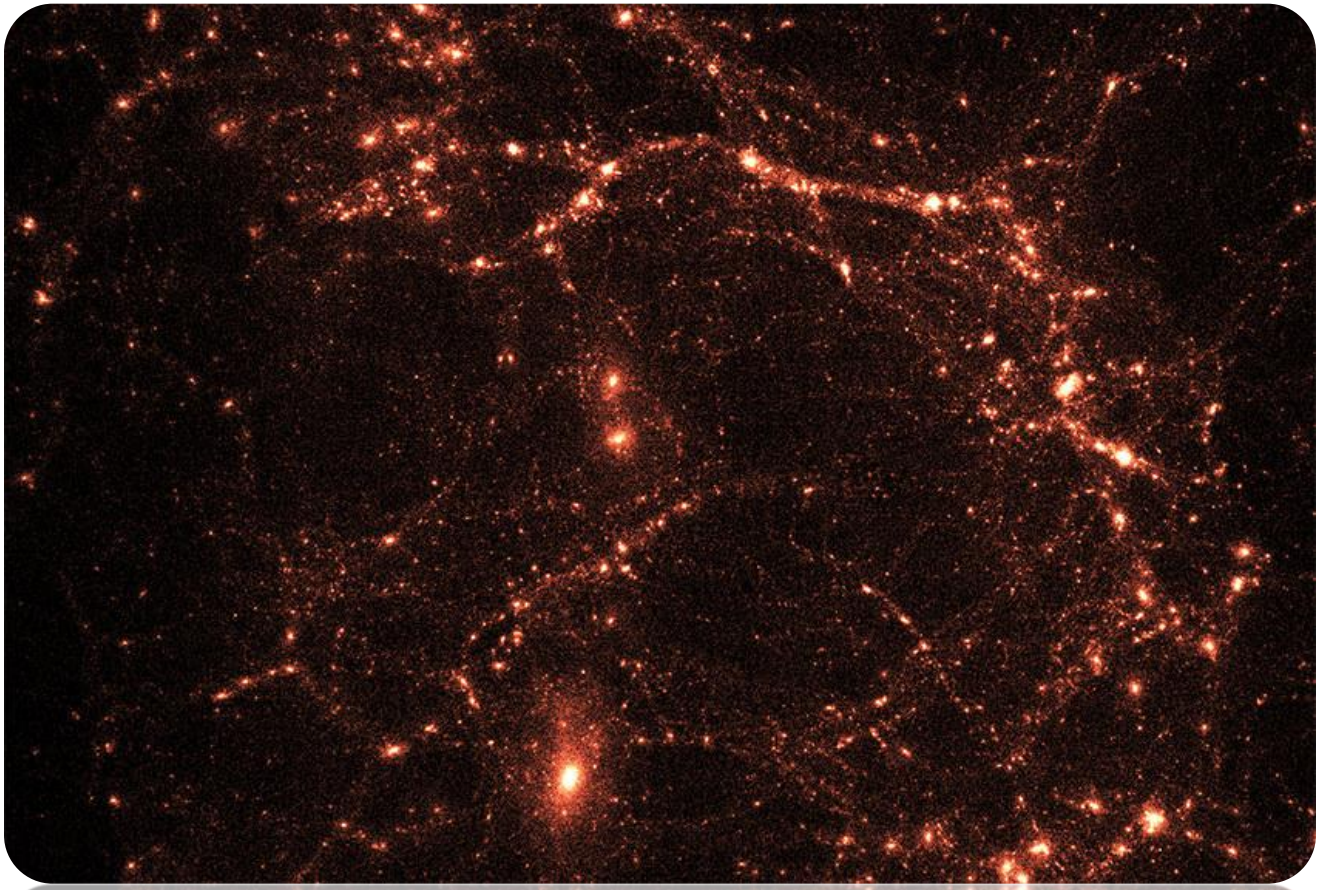


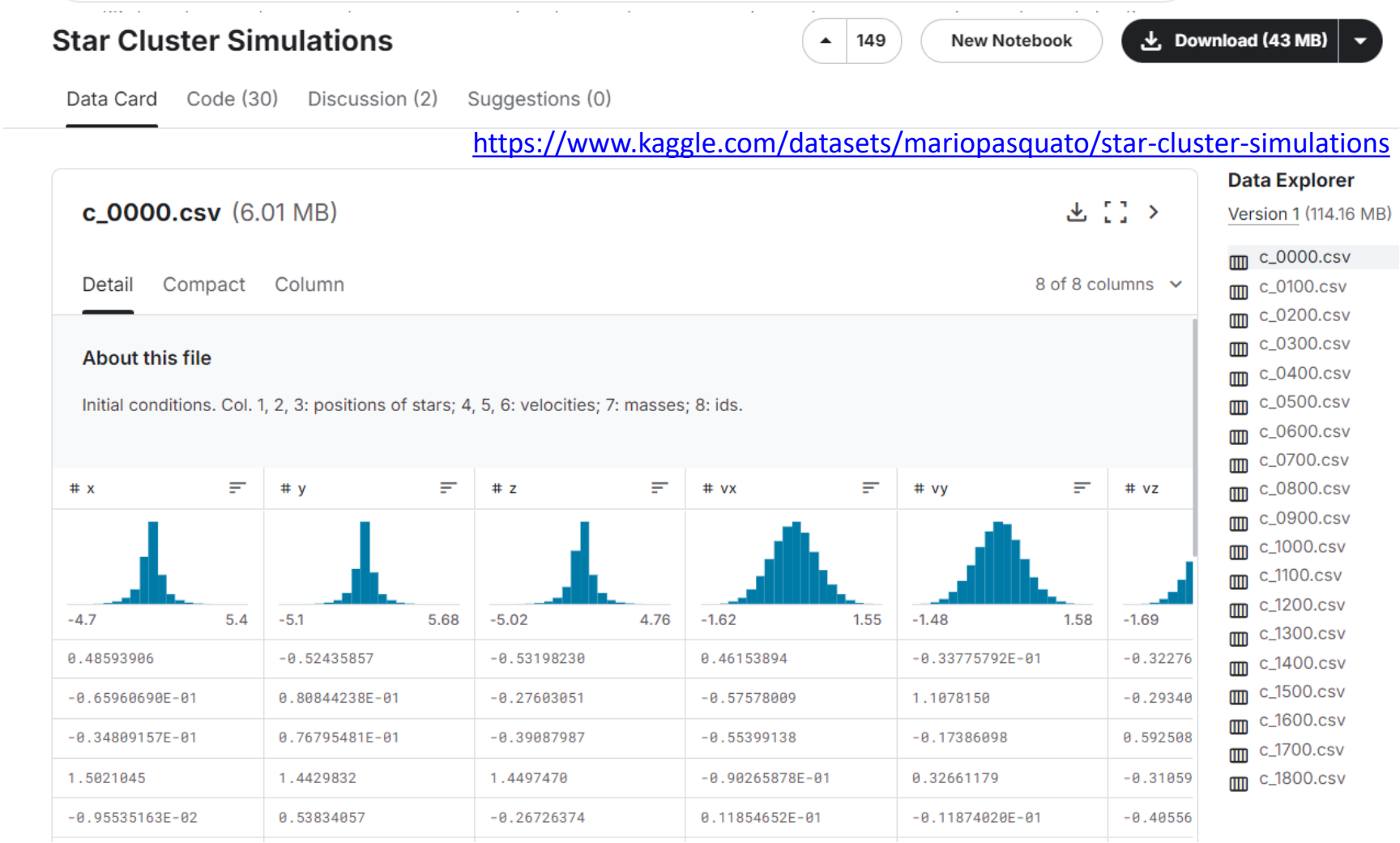
Image source: stanford.edu

Cosmological Simulation



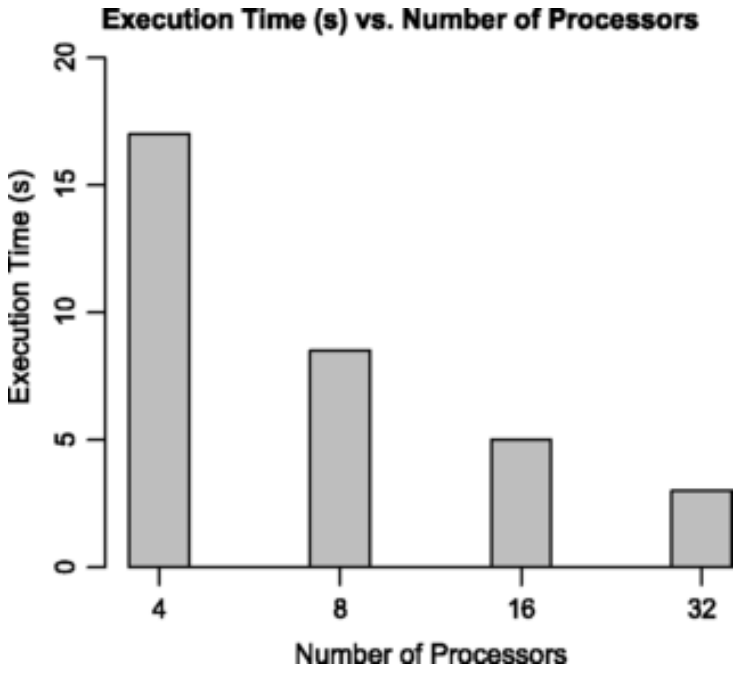
[Credit: ANL]

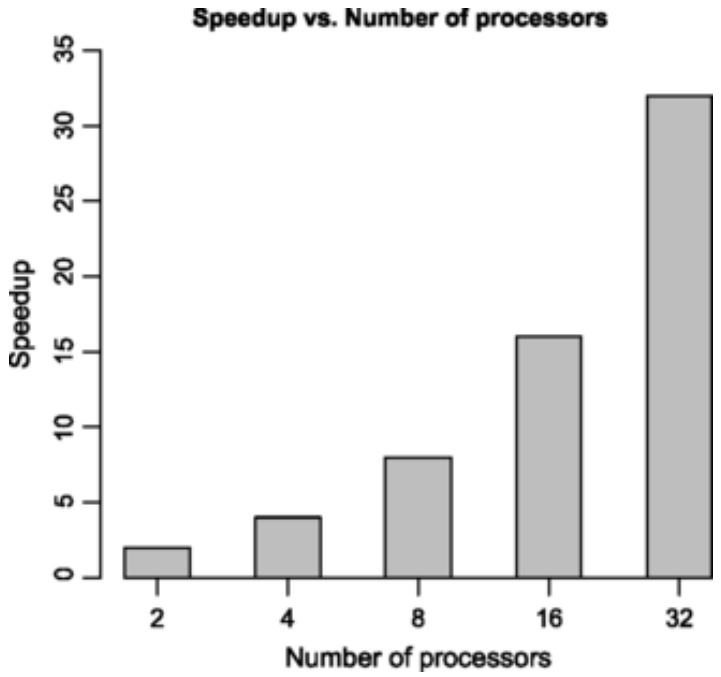
Kaggle.com



There are initially 64000 particles. At end of the simulation there are 63970.
This is because some particles escape the cluster.

Performance and Speedup





Yang et al., High performance data clustering: a comparative analysis of performance for GPU, RASC, MPI, and OpenMP implementations, Journal of Supercomputing 2013.

MPI Implementations

"The MPI standard includes point-to-point message-passing, collective communications, group and communicator concepts, process topologies, environmental management, process creation and management, one-sided communications, extended collective operations, external interfaces, I/O, some miscellaneous topics, and a profiling interface." – MPI report

- MPICH (ANL)
- MVAPICH (OSU)
- OpenMPI
- Intel MPI
- Cray MPI

Programming

- Shell scripts (e.g. bash)
- ssh basics
  - E.g. ssh –X
  - …
- Mostly in C/C++
- Compilation, Makefiles, ...
- Linux environment variables
  - PATH
  - LD_LIBRARY_PATH
  - …

H.W.: Install MPI on your Laptop

- Linux or Linux VM on Windows
  - apt/snap/yum/brew
- Windows
  - No support

- https://www.mpich.org/documentation/guides/

References for MPI

- (CSA) DE Culler, JP Singh and A Gupta, Parallel Computer Architecture: A Hardware/Software Approach Morgan-Kaufmann, 1998.
- (GGKK) A Grama, A Gupta, G Karypis, and V Kumar, Introduction to Parallel Computing. 2nd Ed., Addison-Wesley, 2003.
- (MPI) Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker and Jack Dongarra, MPI - The Complete Reference, Second Edition, Volume 1, The MPI Core.
- (GLS) William Gropp, Ewing Lusk, Anthony Skjellum, Using MPI: portable parallel programming with the message-passing interface, 3rd Ed., Cambridge MIT Press, 2014.
- (PP) Peter S Pacheco, An Introduction to Parallel Programming, Morgan Kaufmann, 2011.

Thank You