ICAWMSCS ’24 May 27-31, 2024

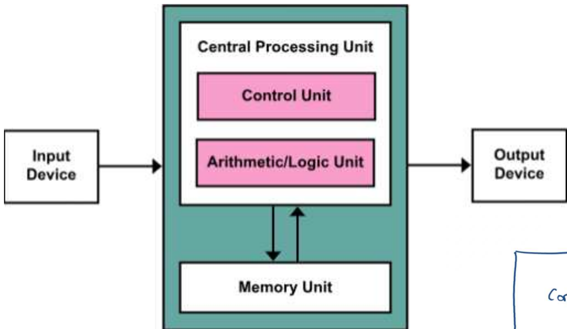
High Performance Computing Workshop - Day 2

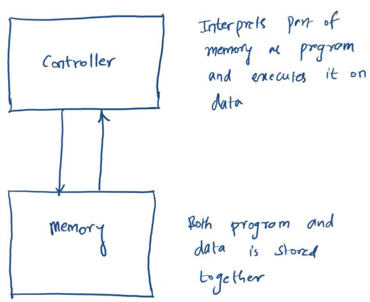
By
Prof. Clement Onime
onime@ictp.it
ICTP,
Trieste, Italy

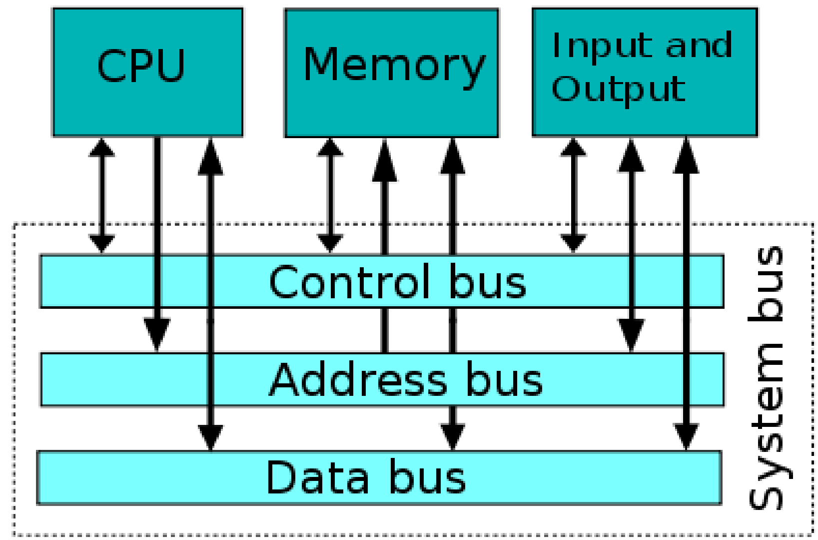
File operations

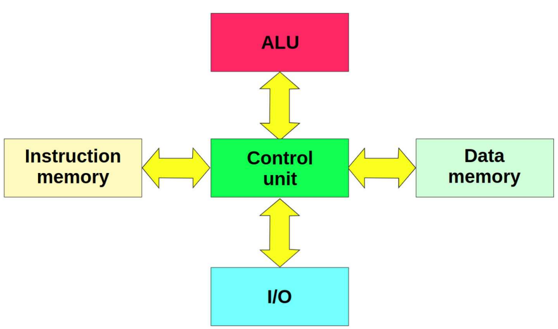
- Input
  - Sometimes can differentiate between text and binary files
  - Sequential or random
- Output
  - Create non existing
  - overwrite or clobber existing
  - Or append
- Fopen, printf, scanf, fgetc, fclose, fputc, ungetc

Answer: Von Neumann vs Harvard vs modern architecture









CONCEPTS OF PARALLELISM

Section outline

- Role of computing in Science
- Parallel computing
  - Why is it important?
  - Accessing High Performance Computing
    - Build your own cluster
    - Others
    - Cloud based
- OpenMP parallel programming
- Examples of algorithms

What some challenges you would like to overcome?

Scientific might be..

A problem as it can be…

- … too **HARD**
  - e.g. building large wind tunnels
- … too **EXPENSIVE**
  - building a throw-away passenger jet
  - Simulate lasers behavior
- … too **SLOW**
  - waiting for climate or galactic evolution
- … too **DANGEROUS** or **CONTROVERSIAL**
  - Research on nuclear or radioactive material
  - stem cell research

Science with computers

Paradigm of Science

- Observe
- Theorize
- Experimentation

Now, less expensive

- Observe
- Theorize
- Computation
- Modeling & simulation

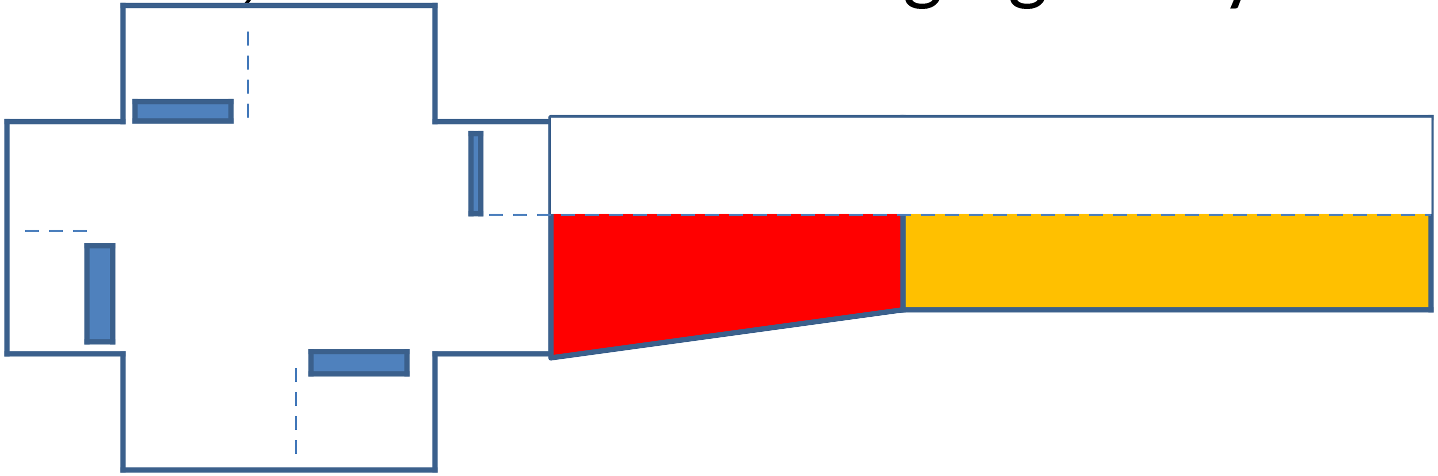
Also, in certain fields, the observe phase is replaced by Simulation. E.g: Study of the early Universe...

What is parallel computing

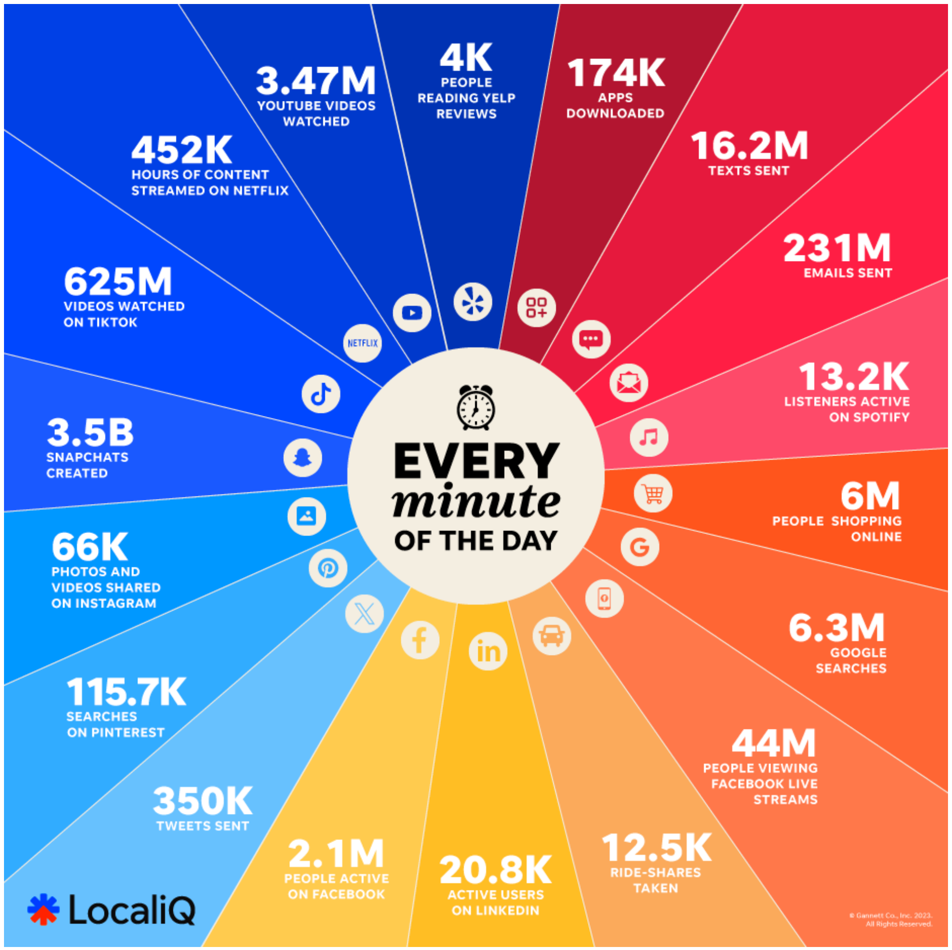
- Serial computing
  - Single program broken into parts, parts broken into single instructions, single instructions run one at a time on single CPU
- Parallel computing
  - Program broken into parts, each part broken into single instructions that maybe run concurrently on different CPU(s)
- *Simultaneous use of multiple compute resources to solve a computational problem*

Why Parallel computing modeling real world

- Useful as it can closely model the real world situations:- Many complex interrelated events happening at the same time within a temporal sequence. E.g: rush hour traffic at big junction, and rain… or changing car tyre



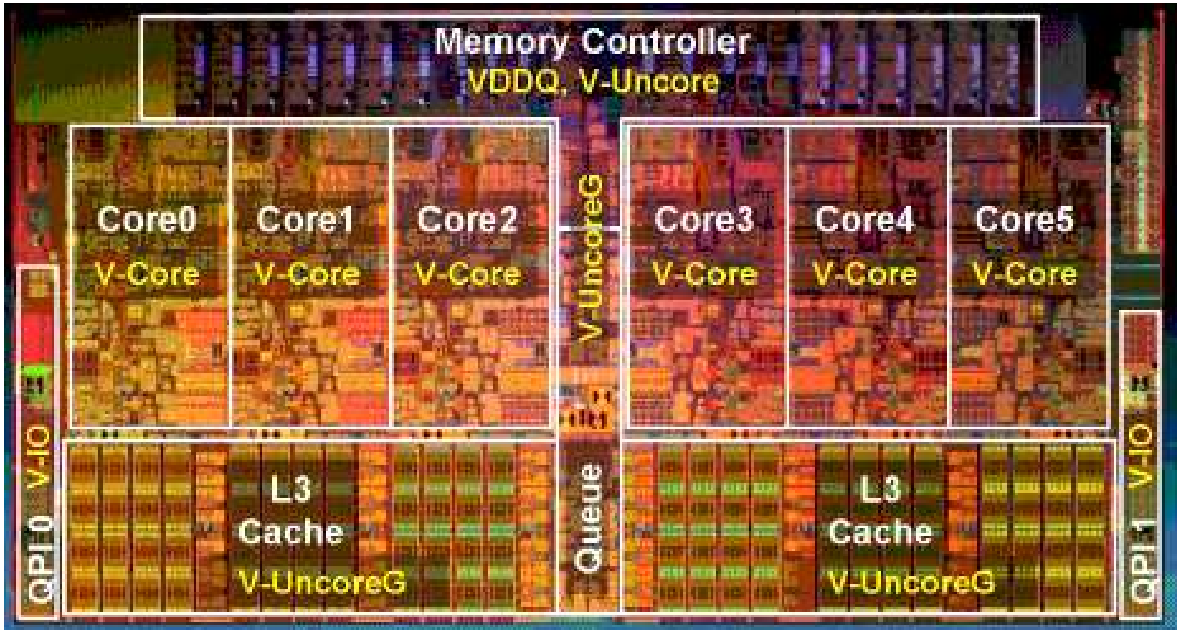
What’s happening right now?



- Tweeter
  - 5833 tweets per second or 171µs per tweet
  - Directed graph model
    - $G = (V, E)$
    - Inter user relationship
      - $(v_a, v_j) \in E$
    - Shortest distance
      - $d(v_a, v_j)$
    - Sub-graphs
      - Single source shortest path

Why Parallel computing

- Only way to make optimal use of new/ evolving generations of CPU processors. E.g: dual core and beyond.

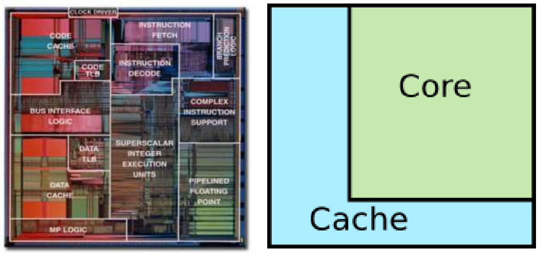


Intel Westmere CPU

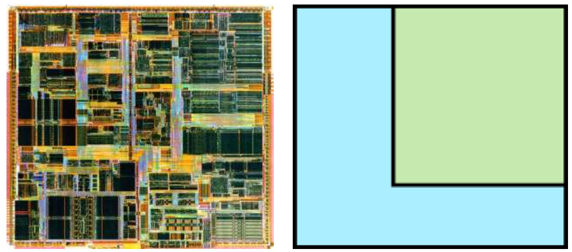
Microprocessor Evolution:

from single- to multi-core

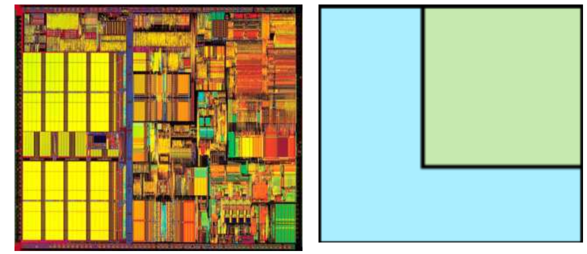
Pentium I



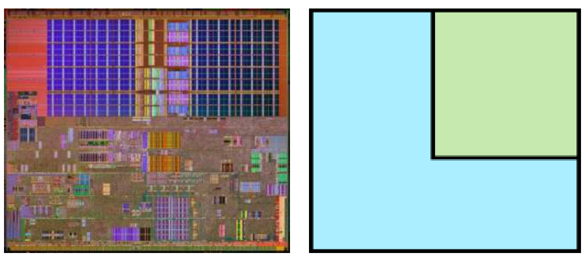
Pentium II



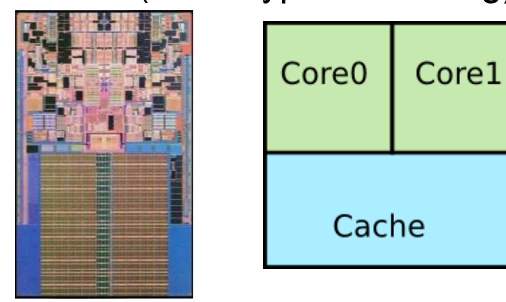
Pentium III



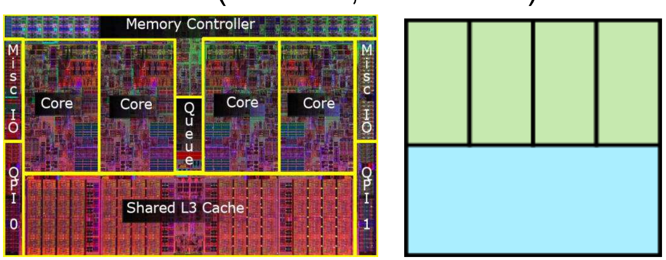
Pentium IV (Hyperthreading)



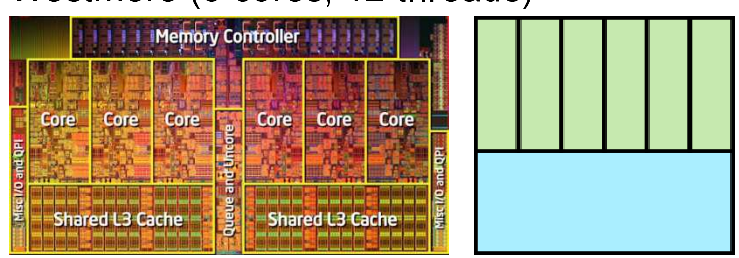
Core 2 (New HyperThreading)



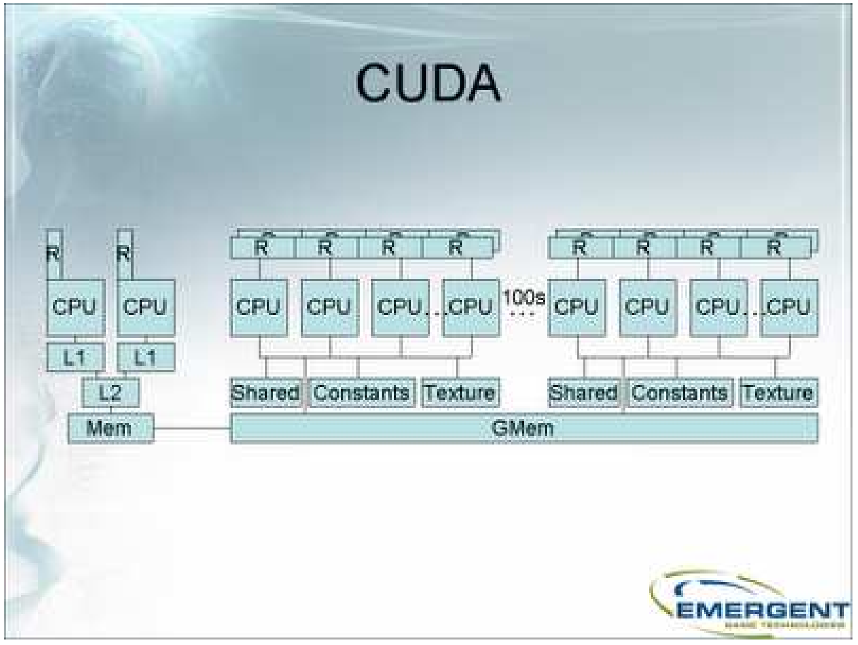
Core i7- 9xx (4 cores, 8 threads)



Westmere (6 cores, 12 threads)



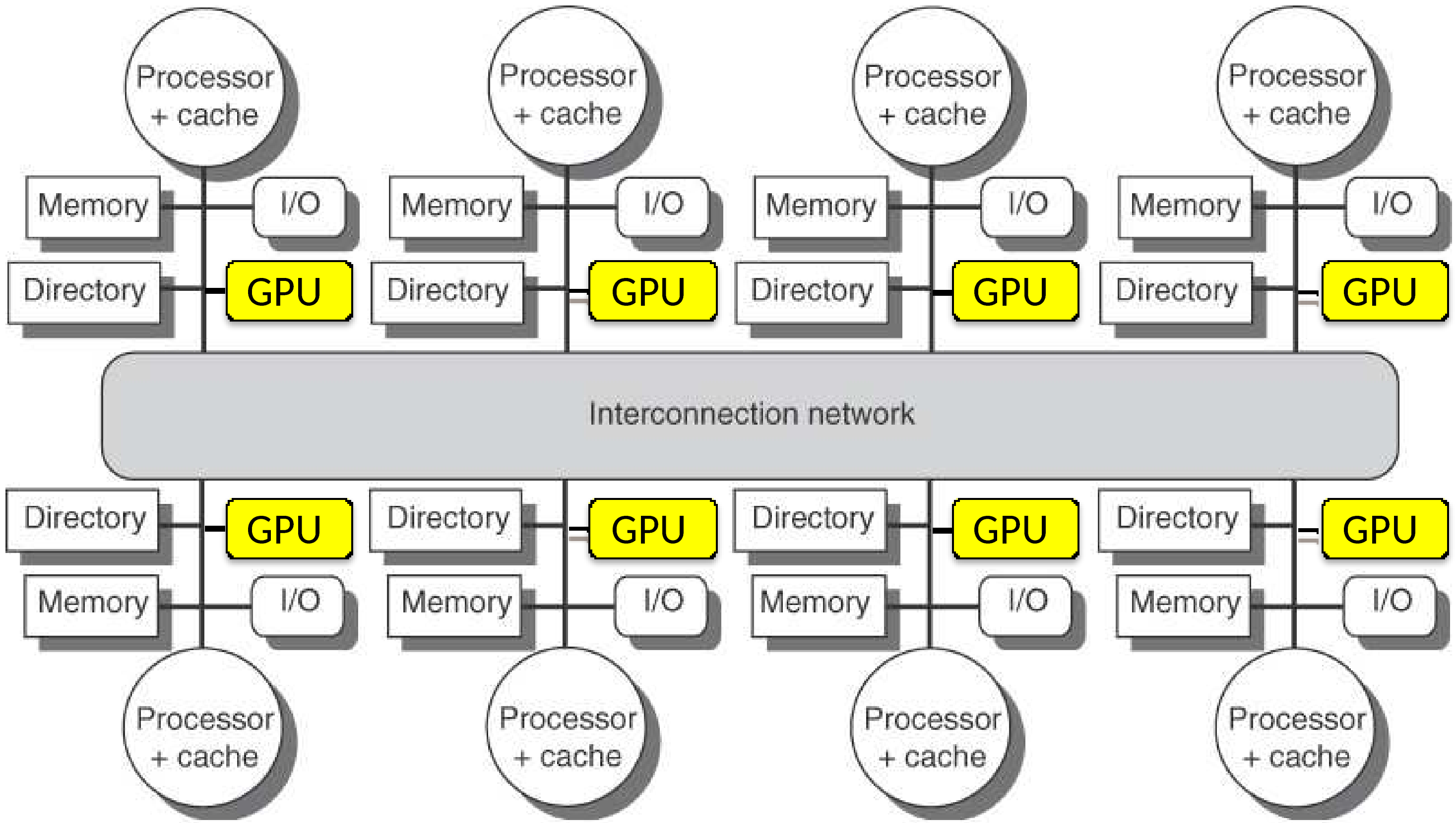
Now many-cores



- Single computer
  - 2483 Nvidia cores
- CUDA
  - Nvidia specific
- OpenCL
  - Same executable software can run on cpu and/or gpu
  - Distributed/networked systems

HPC clusters and Super-Computer systems

standard hardware plus accelerators



Slide from Introduction to HPC by ICHEC, Ireland

A full hybrid system

pros & cons

ADVANTAGES:

- Accelerators (GPUs) can speed-up the calculation up to 100x times!

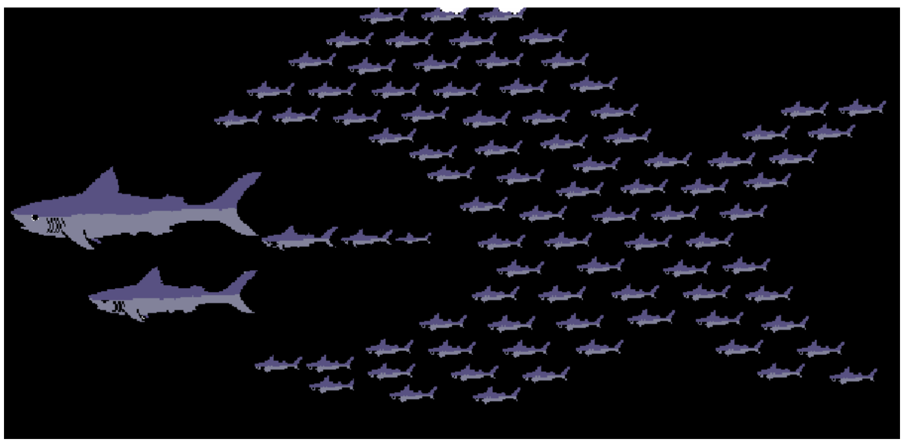
DISADVANTAGES:

- Accelerators require their own programming environment, need to learn..

→ A little bit of work for potentially huge advantages in speed and performance. Think about that.

Slide from Introduction to HPC by ICHEC, Ireland

Scaling your scientific research work



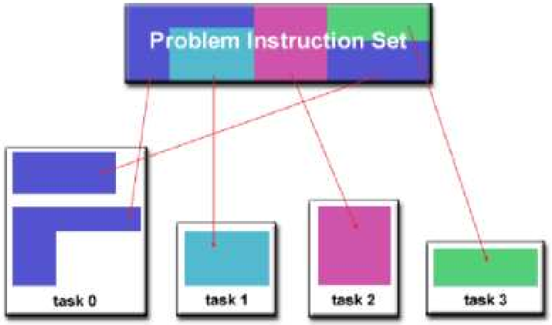
- Many computers (nodes) interconnected by high speed network.
- Commodity clusters
- Hybrid supports both shared & distributed architecture & programming

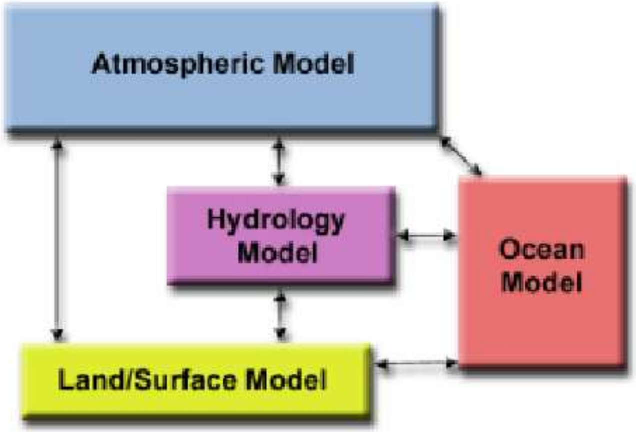
THINKING IN PARALLEL

Break your work into tasks

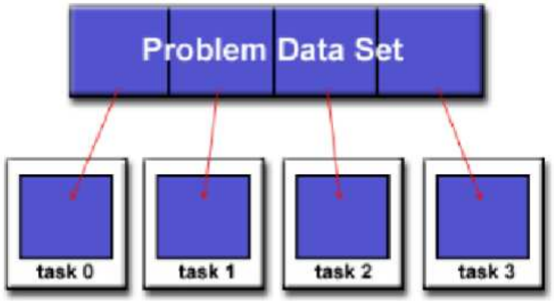
Step 1: Decomposition

- Breaking the problem into tasks (discrete chunks of work)
  - Functional decomposition is based on tasks to be done.
  - Domain decomposition is based on data partitioning

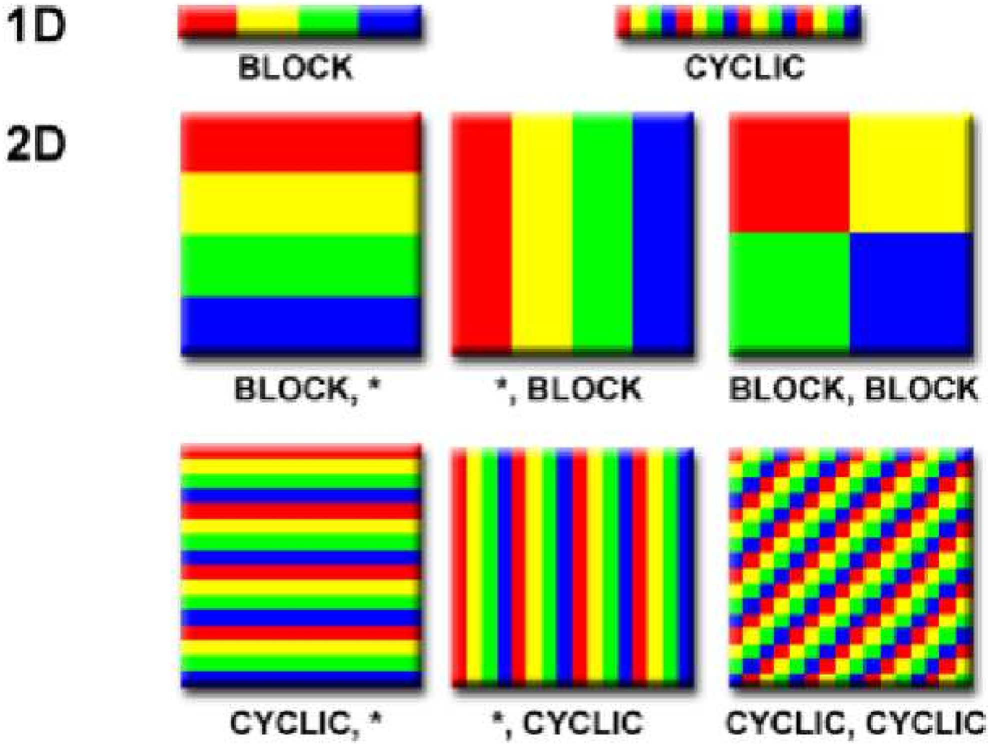




Preparing your data



- There are different ways to partition data:



Data Parallel approach

- Parallelism is focused around operations on data
- Data is organised in common structure such as array (1, 2 or 3D)
- Tasks work collectively on same data structure but each task has a different range or portion.
- All tasks perform same operation on data.
- Can be carried on shared or distributed memory architecture systems

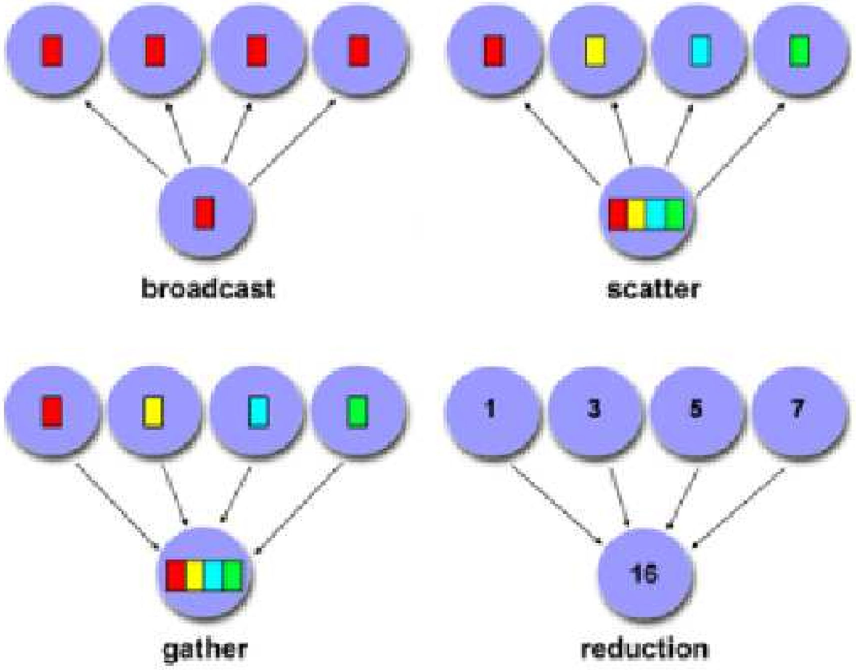
Data parallel implementation

- Program usually have to be written from the ground up as data parallel.
- Requires a data parallel compiler(or library)
  - Included in Fortran 95.
- Compiler directives used to specify distribution and alignment of data

Data exchange b/w tasks

STEP 3: Communication

- Is communication required?
  - Embarrassingly parallel (no communication needed)
  - If yes then consider
    - cost, synchronous, latency/ bandwidth
    - scope



Collating outputs

STEP 3: Synchronization

- Is synchronization required
- Which one
  - Barriers:
    - Starting or stopping together
  - Locks/semaphores
    - Critical regions (code or data)
  - Acknowledgments
    - MPI

STEP 4: Data dependencies

- Does the order of statements affect results?
  - Check especially in loops
  - A(J) = A(J-1) * 2.0
  - Y= X**3
- ***Other STEPS***
  - Load Balancing
  - Granularity (ratio of computation to communication)
  - I/O

Exercise:

- Select an existing computer problem and apply the 4 steps..

HANDS-ON EXAMPLES

Basic implementation blocks

- Iterative
  - Fixed size iterations
    - “FOR” loops
- Recursive
  - Variable sized iterations
    - “WHILE” loops
- Conditions
  - “IF”
- Parallelism
  - Divide and conquer
    - by tasks/functions
      - Split tasks into smaller independent units for execution
    - by data
      - Split data into smaller independent blocks for processing

Vector addition

- Basic equation

  c = a + b

  Where a and b are array vectors with 1000 element each.

- Serial implementation:

```
for (i = 0; i < 1000; i++)
        c[i] = a[i] + b[i];
```

Try to write a serial recursive implementation?

- Parallel implementation

```
#pragma omp doschedule
for (i = 0; i < 1000; i++)
        c[i] = a[i] + b[i];
```

  – What is the optimal number of threads that should be used?
    - Explain your answer
  – Is this task or data parallel?

Matrix – Vector multiplication

- Algorithm

  c[i][j] += a[i][k] * b[k][j]

  - *Where i is no of rows in a, j is number of cols in b , k is no of cols in a*

- Serial implementation
  - Iterative (how many loops are needed?)
  - Can you develop the recursive solution for this?

- Notes for Parallel implementation
  - Iterative is always easier
  - Shared memory using openmp
  - Alternative distributed memory using openmpi
    - What communications?

Factorial

n!=n(n-1)(n-2)..1

- Algorithm

$$\text{For } n \geq 0,\ n! = \begin{cases} 1, & \text{if } n = 0 \\ n * (n-1)!, & \text{otherwise} \end{cases}$$

- Serial implementation

```
/* Factorial */
Recursive:
int factorial(int n) {
        if (n == 0)
                return 1;
        else
                return (n * factorial(n-1));

}

Iterative:
int factorial(int n) {
        int i;
        int factorial;

        factorial = 1
        for (i=n; i>=2; i--)
                factorial *= i;

        return (factorial);
}
```

- Parallel implementation
  - Hint:

Fibonacci numbers

- Algorithm

$$\text{fib(n)} = \begin{cases} 0, & \text{if n} = 0 \\ 1, & \text{if n} = 1 \\ \text{fib(n-1)} + \text{fib(n-2)}, & \text{otherwise} \end{cases}$$

- Serial implementation
  - Recursive
  - Iterative
  - *Memory based speed up?*

- Parallel implementation

GCD

- Algorithm

For $m \geq n > 0$

$$\text{gcd(m, n)} = \begin{cases} \text{n}, & \text{if m \% n} = 0 \\ \text{gcd(n, m\%n)}, & \text{otherwise} \end{cases}$$

- Serial implementation
  - Recursive solution
  - Iterative solution

- Parallel implementation

Calculation of PI (π)

- Equation

$$\int_0^1 \frac{1}{1+x^2}dx = arctan(x)\Big|_0^1 = arctan(1) - arctan(0) = arctan(1) = \frac{\pi}{4}$$

$$\pi = 4\int_0^1 \frac{1}{1+x^2}dx$$

Integration: determining the numerical area under a function may be approximated from the summation of fixed width slices of f(x) at midpoints as shown in the diagram. Accuracy is inversely proportional to the width.