CS-251: Parallel and Distributed Computing

Lecture 02 – Introduction to the Parallel and Distributed Computing

Dr. Zeeshan Rafi

PhD MIS, MPhil IT, BS Software Engineering
Former Software Engineer, Database Administrator
System Analyst, Big Data Analyst
Member Turkish Scientific & Technological Research Council

Department of Computing and Information Technology

Istanbul University, TR

KHAS University, TR

University of Gujrat, PK

GCF University, Pk

Why Use Parallel And Distributed Systems? Why Not Use Them?

Introduction

- Parallel Computing and Distributed Computing are two important models of computing that have important roles in today's high-performance computing.
- Both are designed to perform a large number of calculations breaking down the processes into several parallel tasks; however, they differ in structure, function, and utilization.
- Therefore, in the following article, there is a dissection of Parallel Computing and Distributed Computing, their gains, losses, and applications.

Birth of Parallel and Distributed Computing

Before taking a toll on Parallel Computing, first, let's take a look at the background of computations of computer software and why it failed for the modern era.

Computer software was written conventionally for serial computing. This meant that to solve a problem, an algorithm divides the problem into smaller instructions. These discrete instructions are then executed on the Central Processing Unit of a computer one by one. Only after one instruction is finished, next one starts.

A real-life example of this would be people standing in a queue waiting for a movie ticket and there is only a cashier. The cashier is giving tickets one by one to the persons. The complexity of this situation increases when there are 2 queues and only one cashier.

So, in short, Serial Computing is following:
1.In this, a problem statement is broken into discrete instructions.
2.Then the instructions are executed one by one.
3.Only one instruction is executed at any moment of time.

Look at point 3. This was causing a huge problem in the computing industry as only one instruction was getting executed at any moment of time. This was a huge waste of hardware resources as only one part of the hardware will be running for particular instruction and of time. As problem statements were getting heavier and bulkier, so does the amount of time in execution of those statements. Examples of processors are Pentium 3 and Pentium 4.

Now let's come back to our real-life problem. We could definitely say that complexity will decrease when there are 2 queues and 2 cashiers giving tickets to 2 persons simultaneously. This is an example of Parallel Computing.

Parallel and Distributed Computing

- What is a parallel computer?
  - A collection of processing elements that communicate and coorperate to solve large problems fast.
- What is a distributed system?
  - A collection of independent computers that appear to its users as a single coherent system.
  - A parallel computer is implicitly a distributed system.

History of computing

Four decades of computing:

- Batch Era
- Time sharing Era
- Desktop Era
- Network Era

Batch Era

- Batch processing
  - Is execution of a series of programs on a computer without manual intervention
  - The term originated in the days when users entered programs on punch cards

Time-sharing Era

- **time-sharing** is the sharing of a computing resource among many users by means of multiprogramming and multi-tasking

- Developing a system that supported multiple users at the same time

Desktop Era

- Personal Computers (PCs)
- With WAN

Network Era

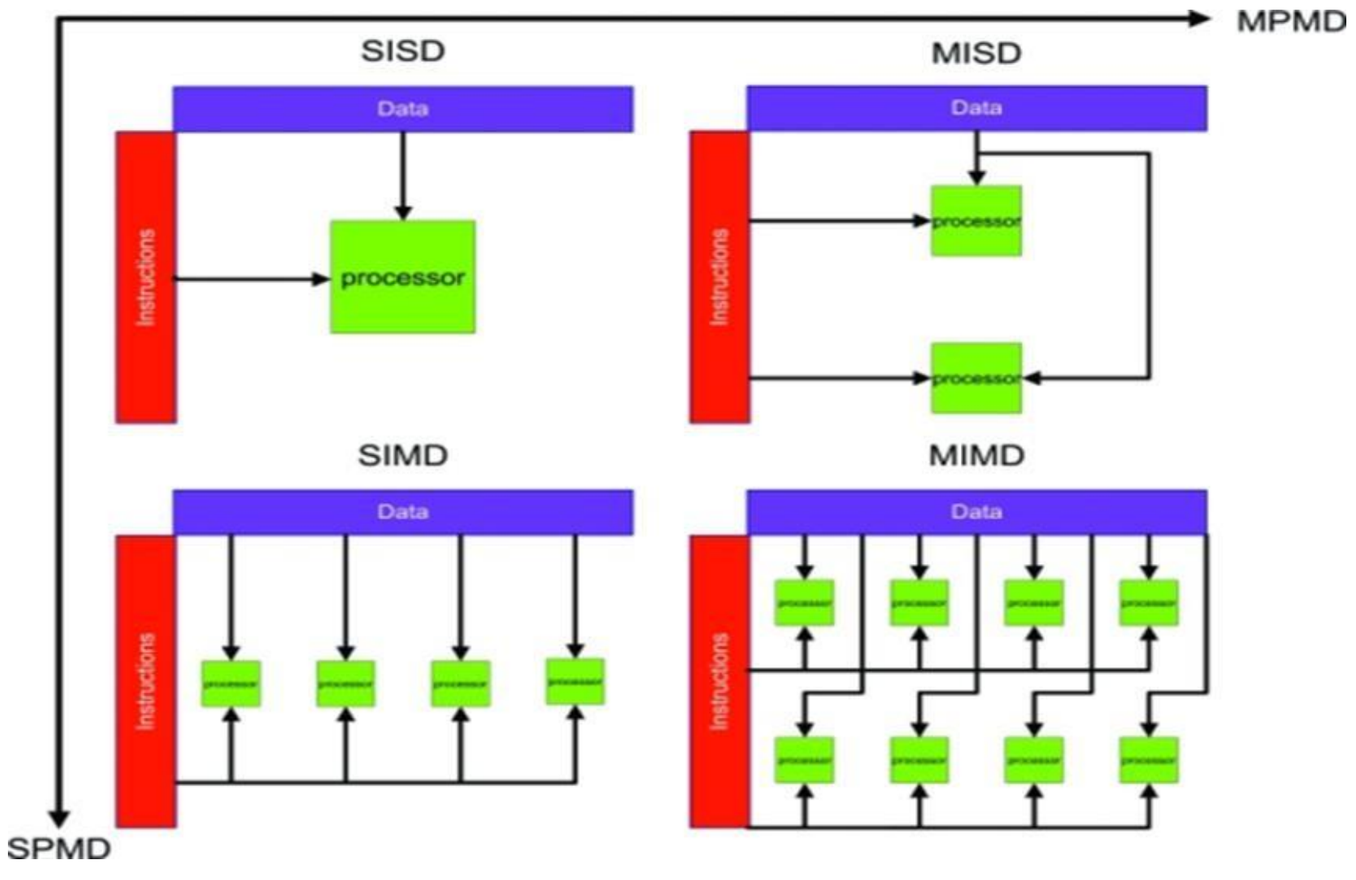
- Systems with:
  - Shared memory
  - Distributed memory
  - Example for parallel computers: Intel iPSC, nCUBE

Flynn's Taxonomy Of Computer Architecture

Two types of information flow into processor:

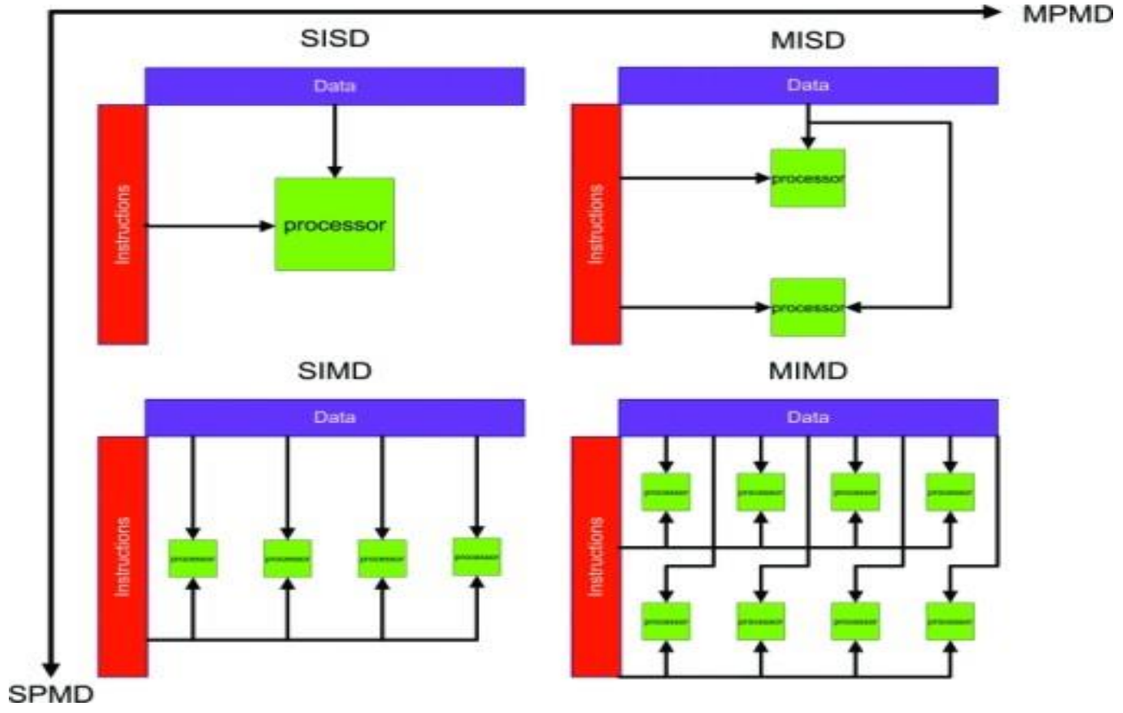
- Instructions
- Data

what are instructions and data?



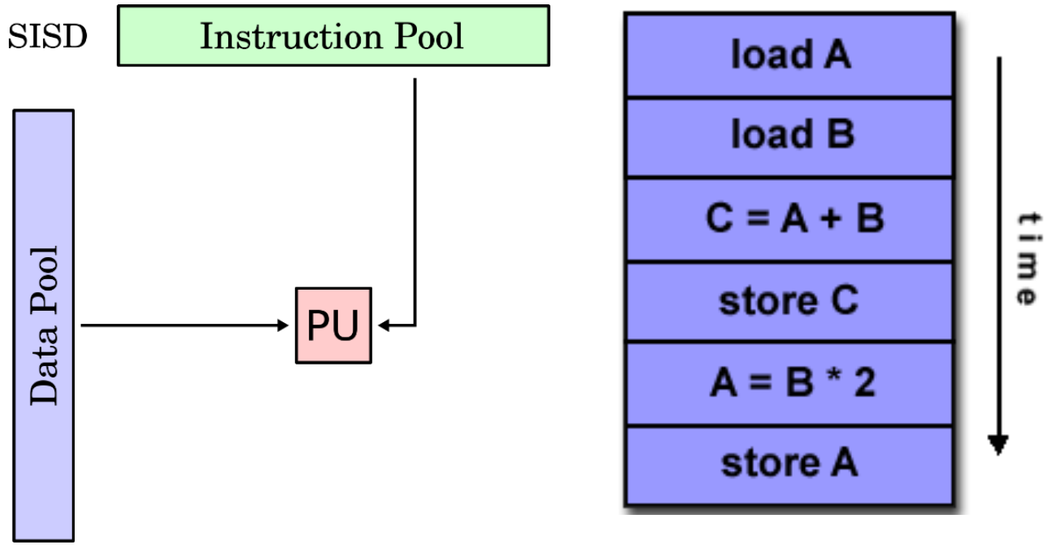
Flynn's Taxonomy Of Computer Architecture

1. single-instruction single-data streams (SISD)
2. single-instruction multiple-data streams (SIMD)
3. multiple-instruction single-data streams (MISD)
4. multiple-instruction multiple-data streams (MIMD)



Single Instruction Single Data (SISD)

- A serial (non-parallel) computer
- This is the oldest type of computer



UNIVAC1

IBM 360

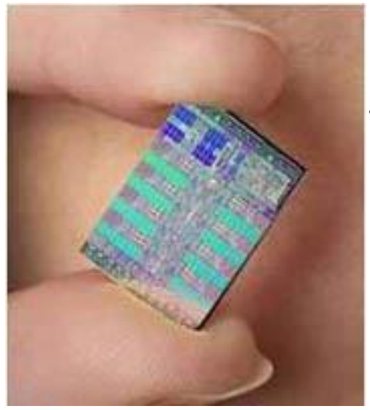
CRAY1

CDC 7600

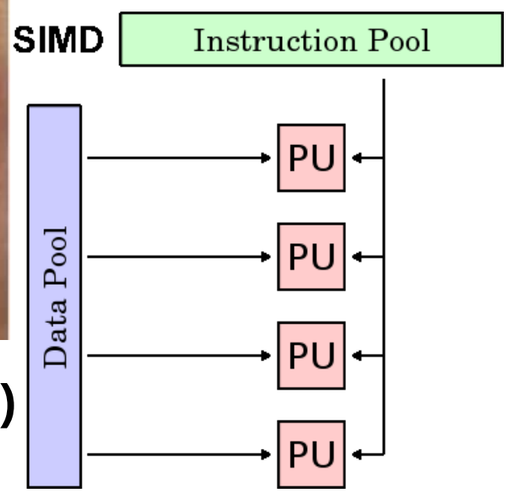
PDP1

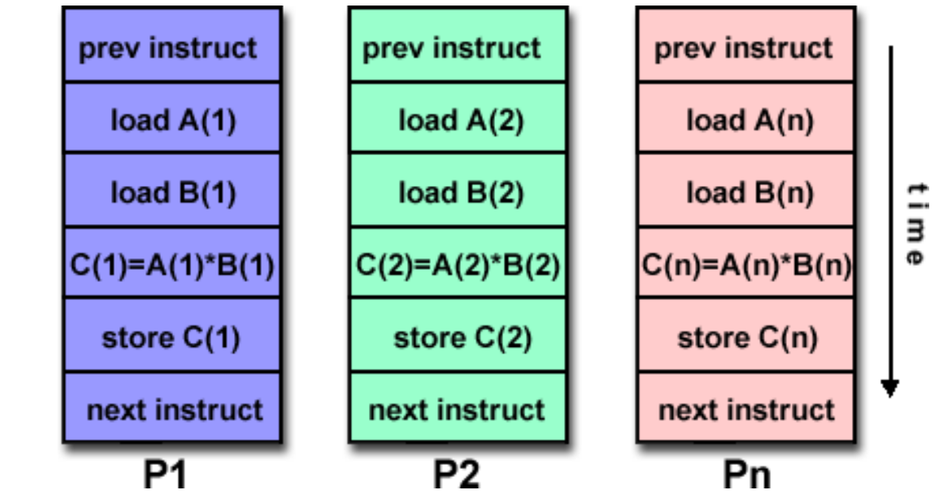
Single Instruction Multiple Data (SIMD)

ILLIAC IV



Cell Processor (GPU)

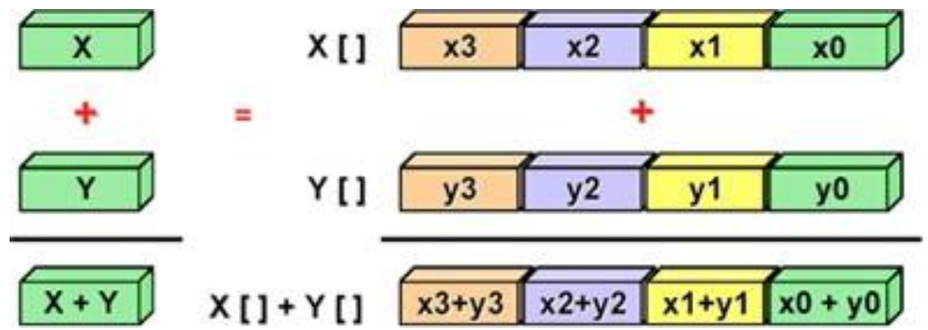




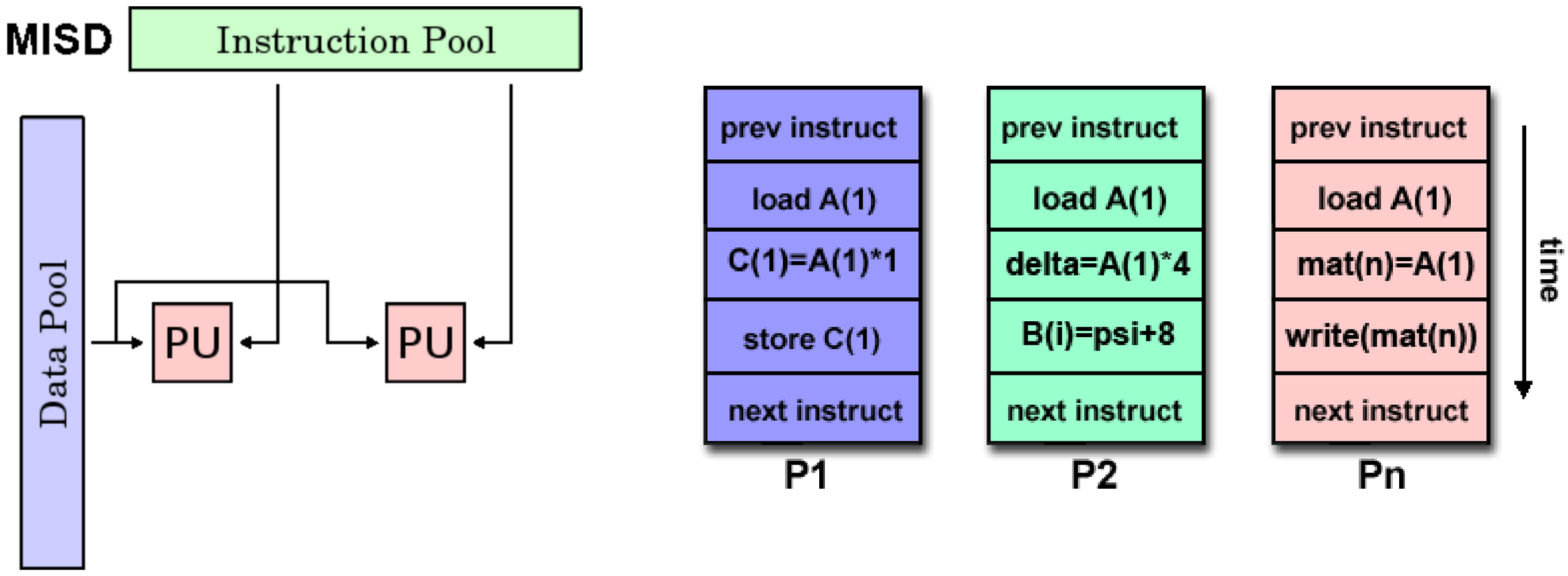
MasPar

Cray X-MP

Cray Y-MP

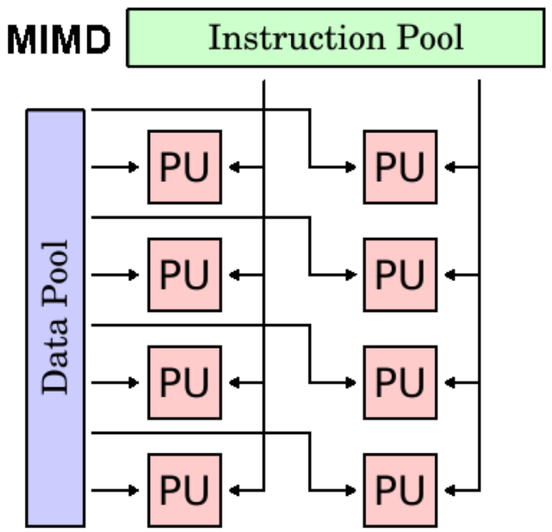


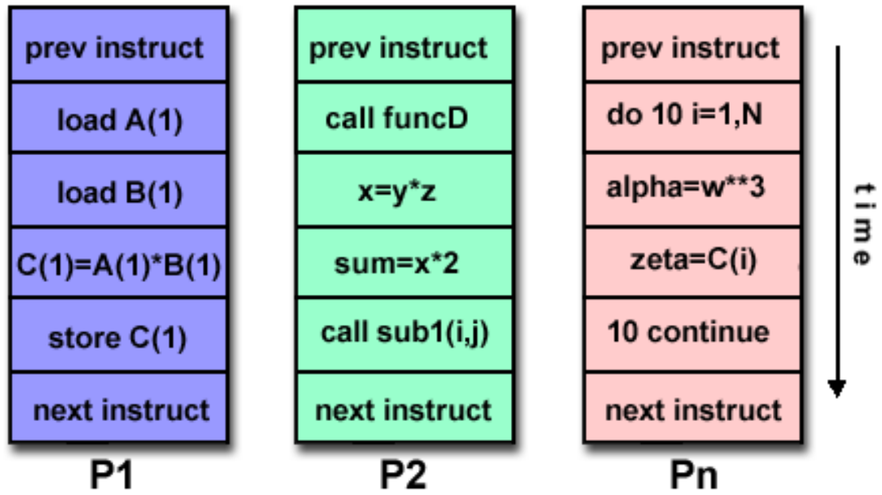
Multiple Instruction Single Data



The Space Shuttle flight control computers

Multiple Instruction Multiple Data (MIMD)





IBM POWER5

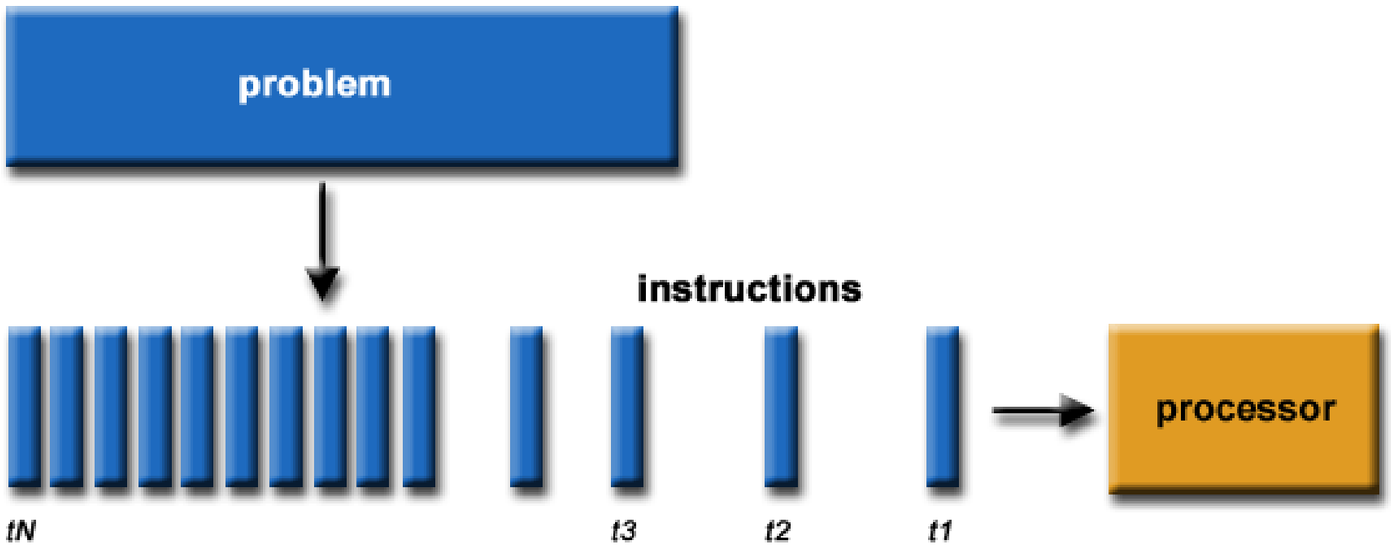
HP/Compaq Alphaserver

Intel IA32

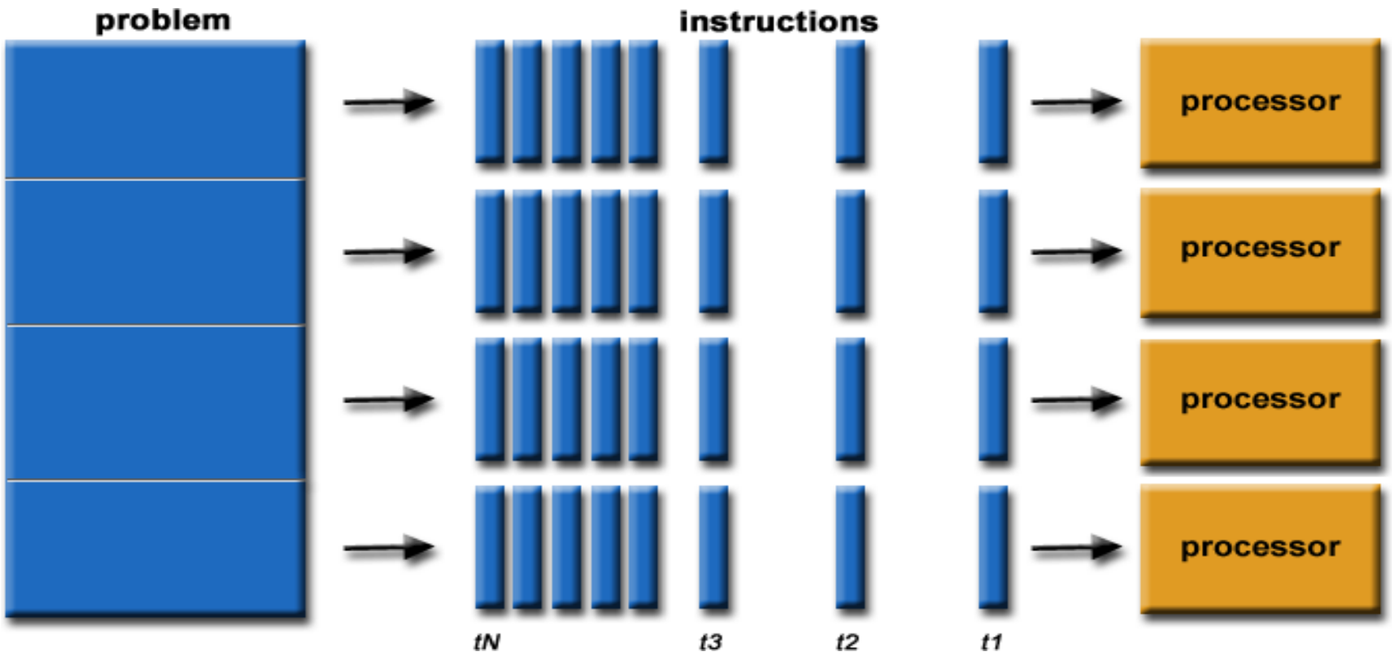
AMD Opteron

Serial VS Parallel Computing

Serial Computing



Parallel computing



What is Parallel Computing?

- In parallel computing multiple processors performs multiple tasks assigned to them simultaneously.
- Memory in parallel systems can either be shared or distributed.
- Parallel computing provides concurrency and saves time and money.

Examples

Blockchains, Smartphones, Laptop computers, Internet of Things, Artificial intelligence and machine learning, Space shuttle, Supercomputers are the technologies that uses Parallel computing technology.

Advantages of Parallel Computing

•**Increased Speed:** In this technique, several calculations are executed concurrently hence reducing the time of computation required to complete large scale problems.

•**Efficient Use of Resources:** Takes full advantage of all the processing units it is equipped with hence making the best use of the machine's computational power.

•**Scalability:** Also the more processors built into the system, the more complex problems can be solved within a short time.

•**Improved Performance for Complex Tasks:** Best suited for activities which involve a large numerical calculation like, number simulation, scientific analysis and modeling and data processing.

Disadvantages of Parallel Computing

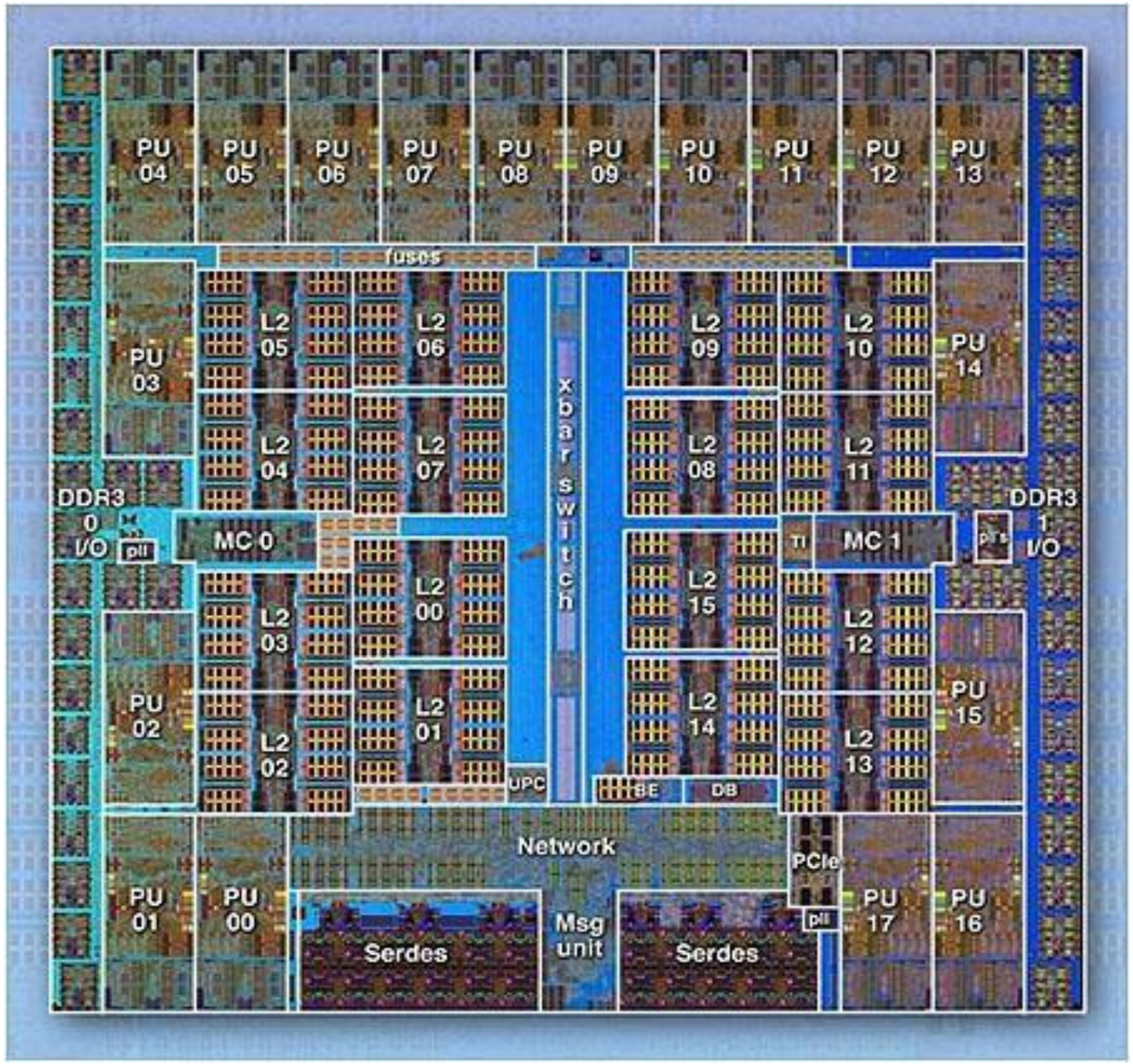
•**Complexity in Programming:** Parallel writing programming that is used in organizing tasks in a parallel manner is even more difficult than that of serial programming.

•**Synchronization Issues:** Interaction of various processors when operating concurrently can become synchronized and result in problem areas on the overall communication.

•**Hardware Costs:** The implementation of parallel computing does probably involve the use of certain components such as multi-core processors which could possibly be costly than the normal systems.

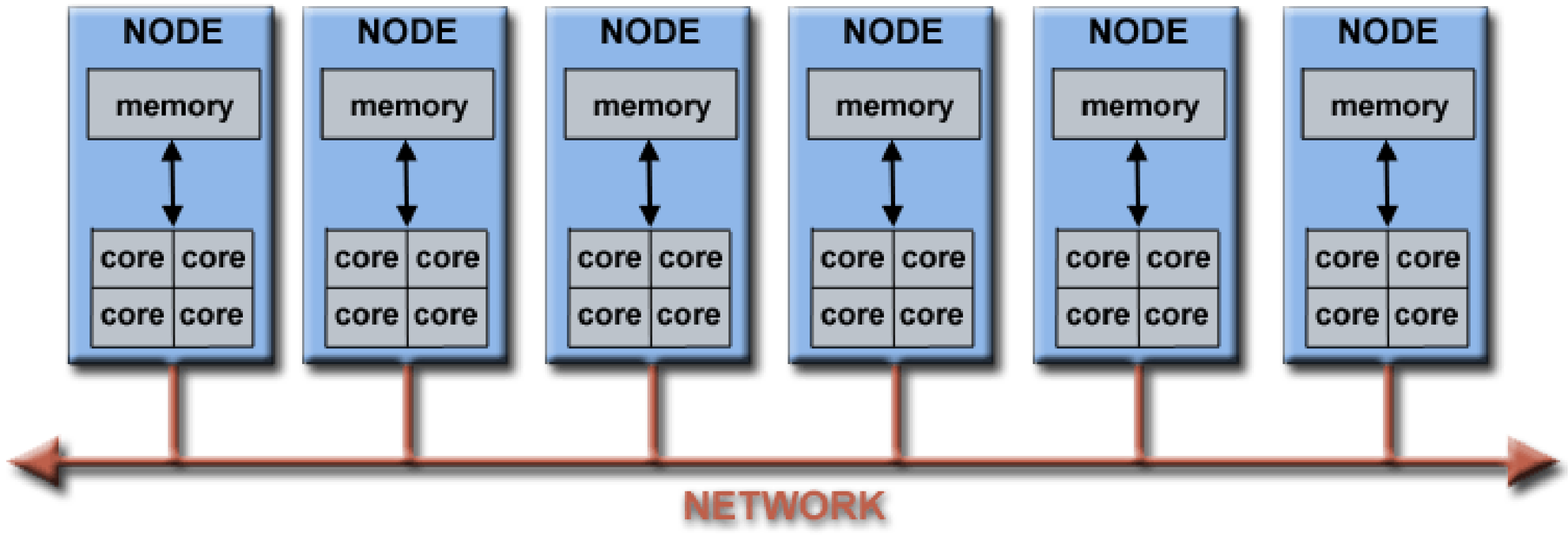
Parallel Computers

- all stand-alone computers today are parallel from a hardware perspective



Parallel Computers

- Networks connect multiple stand-alone computers (nodes) to make larger parallel computer clusters.



Types of Parallelism:

1. **Bit-level parallelism -**
   It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.
   Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. **Instruction-level parallelism -**
   A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.
3. **Task Parallelism -**
   Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently.
4. **Data-level parallelism (DLP) -**
   Instructions from a single stream operate concurrently on several data – Limited by non-regular data manipulation patterns and by memory bandwidth

Why parallel computing?

- The whole real-world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
- Real-world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.
- Parallel computing provides concurrency and saves time and money.
- Complex, large datasets, and their management can be organized only and only using parallel computing's approach.
- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computation only some part of the hardware was used and the rest rendered idle.
- Also, it is impractical to implement real-time systems using serial computing.

Applications of Parallel Computing:

- Databases and Data mining.
- Real-time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality, and virtual reality.

Limitations of Parallel Computing:

- It addresses such as communication and synchronization between multiple sub-tasks and processes which is difficult to achieve.
- The algorithms must be managed in such a way that they can be handled in a parallel mechanism.
- The algorithms or programs must have low coupling and high cohesion. But it's difficult to create such programs.
- More technically skilled and expert programmers can code a parallelism-based program well.

What is Distributed Computing?

In distributed computing we have multiple autonomous computers which seems to the user as single system. In distributed systems there is no shared memory and computers communicate with each other through message passing. In distributed computing a single task is divided among different computers.

Examples

Artificial Intelligence and Machine Learning, Scientific Research and High-Performance Computing, Financial Sectors, Energy and Environment sectors, Internet of Things, Blockchain and Cryptocurrencies are the areas where distributed computing is used.

Advantages of Distributed Computing

•**Fault Tolerance:** The failure of one node means that this node is no longer part of the computations, but that is not fatal for the entire computation since there are other computers participating in the process thereby making the system more reliable.

•**Cost-Effective:** Builds upon existing hardware and has flexibility in utilizing commodity machines instead of the need to have expensive and specific processors for its use.

•**Scalability:** The distributed systems have the ability to scale and expand horizontally through the addition of more machines in the networks and therefore they can take on greater workloads and processes.

•**Geographic Distribution:** Distributed computing makes it possible to execute tasks at different points thereby eliminating latencies.

Disadvantages of Distributed Computing

•**Complexity in Management:** The task of managing a distributed system itself can be made more difficult since it may require dealing with the latency and/or failure of a network as well as issues related to synchronizing the information to be distributed.

•**Communication Overhead:** Inter node communication requirements can actually hinder the package transfer between nodes that are geographically distant and hence the overall performance is greatly compromised.

•**Security Concerns:** In general, distributed systems are less secure as compared to centralized system because distributed systems heavily depend on a network.

What Are They

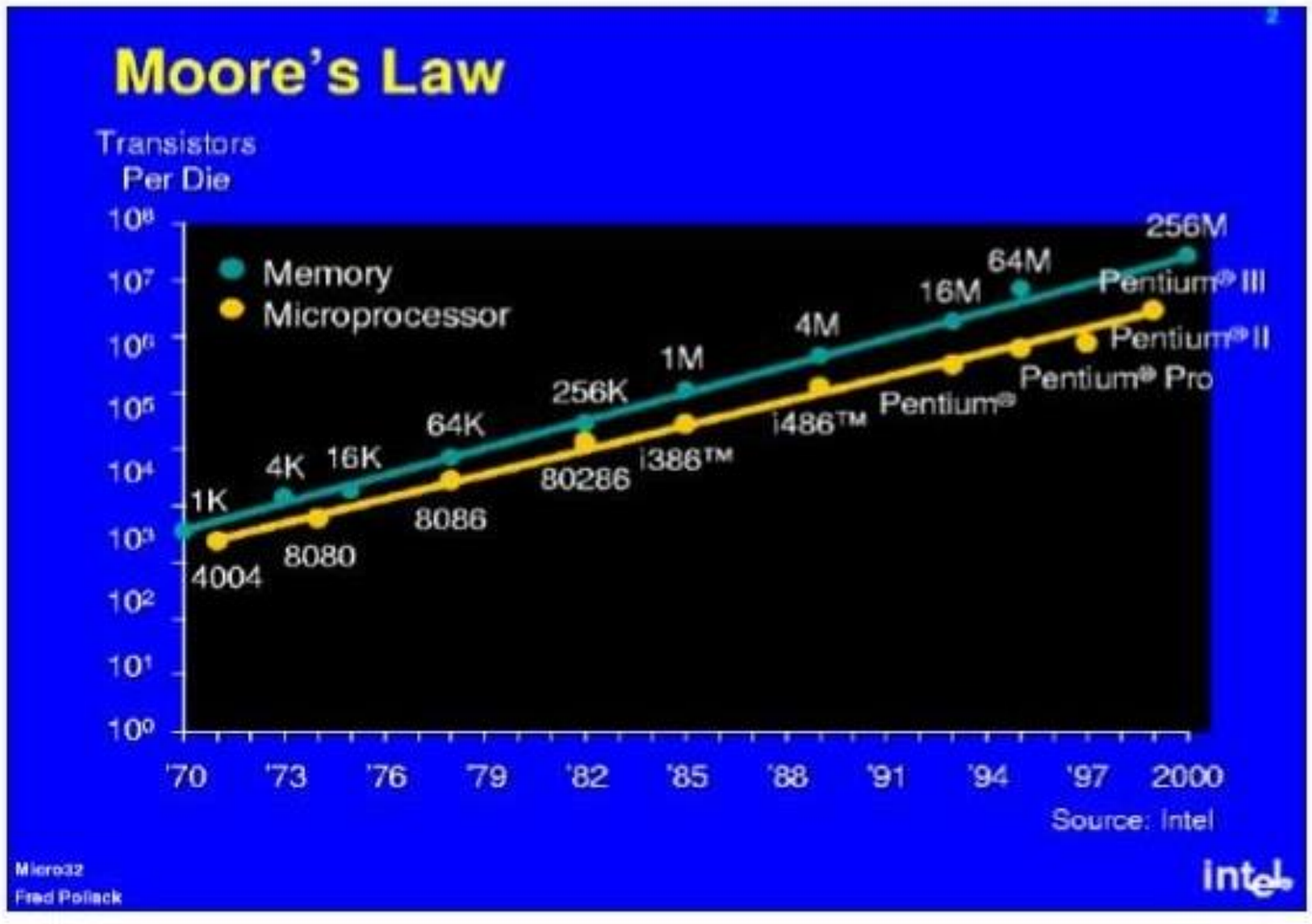
- More than one processing element
- Elements communicate and cooperate
- Appear as one system

- PDS hardware is everywhere.
  - diablo.cs.fsu.edu (2-CPU SMP)
  - Multicore workstations/laptops
  - linprog.cs.fsu.edu (4-node cluster, 2-CPU SMP, 8 cores per node)
  - Hadoop clusters

  - IBM Blue Gene/L at DOE/NNSA/LLNL (262992 processors).
  - Data centers
  - SETI@home.

Why Distributed and Parallel Computing?

- Moore's law: the number of transistors double every 18 months.



Why Distributed and Parallel Computing?

- How to make good use of the increasing number of transistors on a chip?
  - Increase the computation width (70's and 80's)
    - 4bit->8bit->16bit->32bit->64bit->….
  - Instruction level parallelism (90's)
    - Pipeline, LIW, etc
  - ILP to the extreme (early 00's)
    - Out of order execution, 6-way issues, etc
  - Sequential program performance keeps increasing.
    - The clock rate keeps increasing, clock cycles get smaller and smaller.

Why Distributed and Parallel Computing?

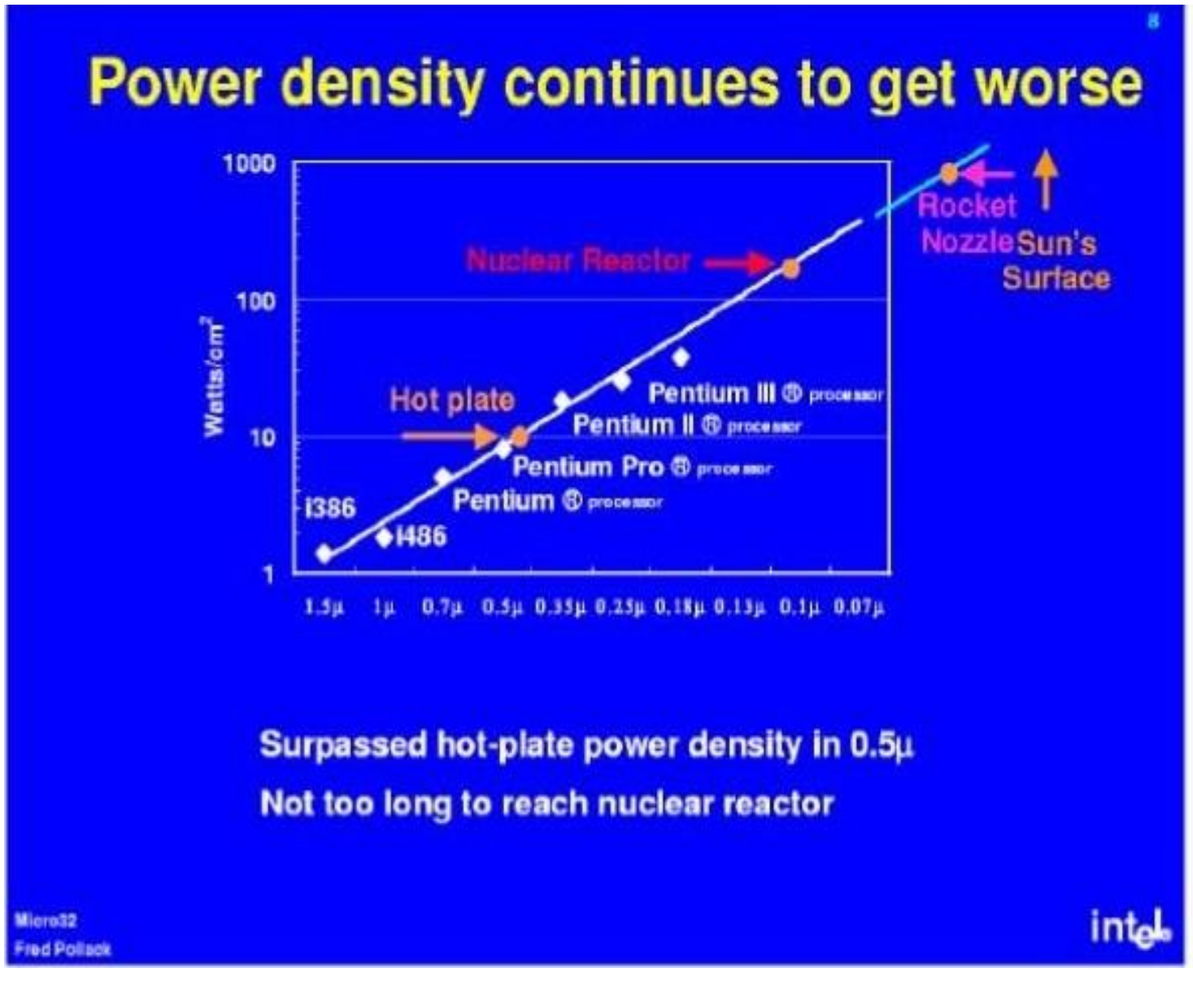
- The fundament limit of the speed of a sequential processor.
  - Power wall (high frequency results in heat)
  - Latency wall (speed of light does not change)

Power Wall

- In computer architecture, a power wall refers to the point where power consumption becomes a limiting factor in a processor's performance.
- This occurs when the power needed to increase clock speed or add more processing cores exceeds the available power budget.
- Essentially, you hit a wall in performance gains due to power limitations.

What will happen if we keep pushing ILP

- Instruction-Level Parallelism (ILP) refers to the ability of a processor to execute multiple instructions simultaneously within a single core
- ILP allows modern processors to work on multiple instructions concurrently, leading to faster execution speeds and improved overall performance.



Latency Wall

- **Latency:**
- In computer science, latency is the time delay between an input and a response. It's the time it takes for data to travel from one point to another and back.
- **"Latency Wall" as a concept:**
- The "latency wall" refers to the limitations or barriers imposed by latency in achieving desired performance or user experience.

- Speed of light = 300000000m/s
- One cycle at 4Ghz = 0.00000000025s
- The distance that the light can move at one cycle:
  - 0.00000000025 * 300000000 = 7.5cm

Intel chip dimension = 1.47 in x 1.47 in
= 3.73cm x 3.73cm

Why Distributed and Parallel Computing?

- Power wall and latency wall indicate that the era of single thread performance improvement through Moore's law is ending/has ended.
- More transistors on a chip are now applied to increase system *throughput*, the total number of instructions executed, but not latency, the time for a job to finish.
  - Improving ILP improves both.
  - We see a multi-core era

Why Distributed and Parallel Computing?

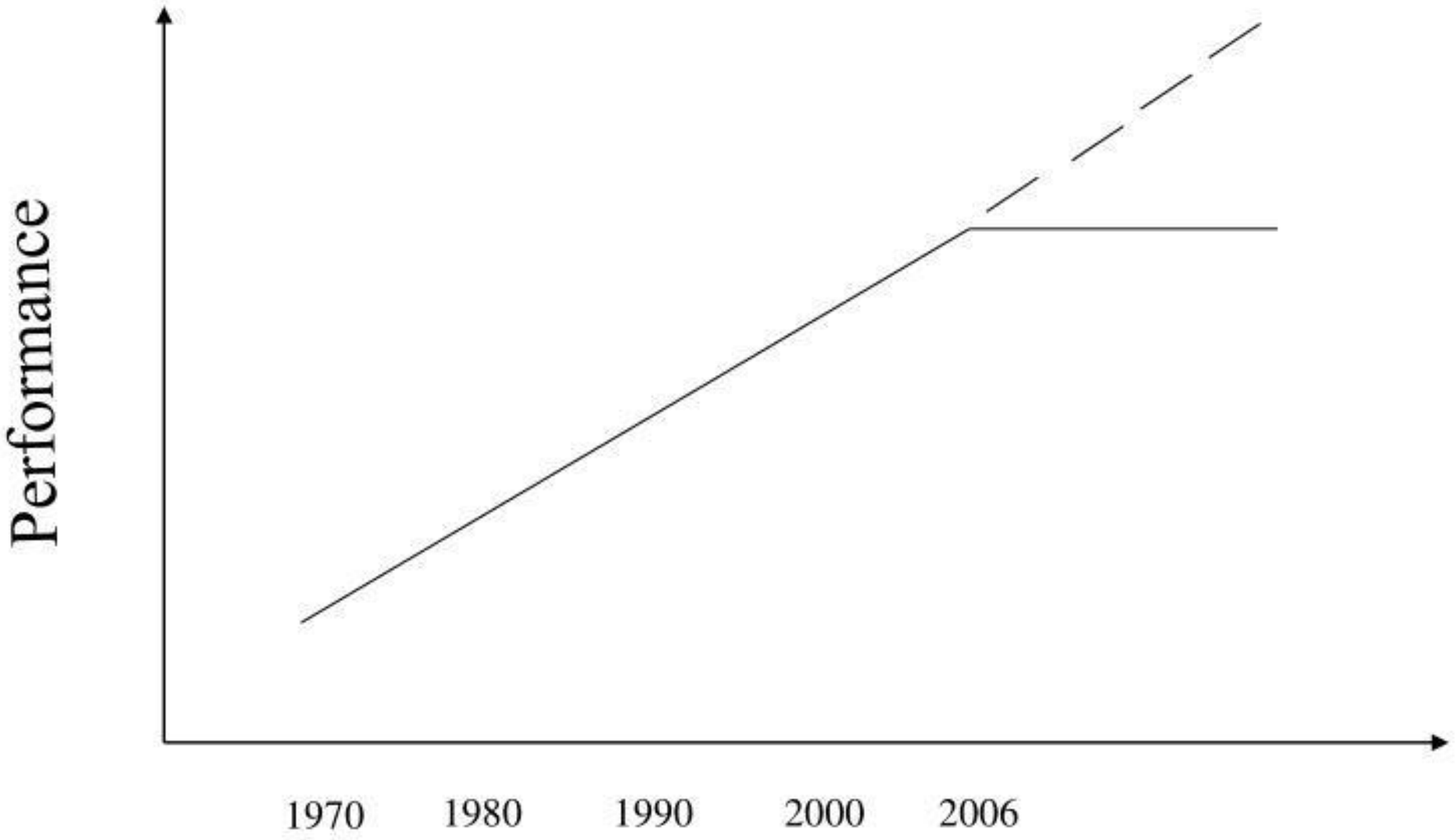
- The marching of multi-core (2004-now)
  - Mainstream processors:
    - INTEL Quad-core Xeon (4 cores),
    - AMD Quad-core Opteron (4 cores),
    - SUN Niagara (8 cores),
    - IBM Power6 (2 cores)
  - Others
    - Intel Tflops (80 cores)
    - CISCO CSR-1 (180 cores) (network processors)
  - Increase the throughput without increasing the clock rate.

Why Distributed and Parallel Computing?

- Programming multi-core systems is fundamentally different from programming traditional computers.
  - Parallelism needs to be explicitly expressed in the program.
- This is traditionally referred to as parallel computing.
  - PDC is not really a choice anymore.

```
For (I=0; I<500; I++)
  a[I] = 0;
```

GO PARALLEL



Why Distributed and Parallel Computing?

- In the foreseeable future, parallel systems with multi-core processors are going mainstream.
  - Lots of hardware, software, and human issues in mainstream parallel/distributed computing
    - How to make use a large number of processors simultaneously
    - How to write parallel programs easily and efficiently?
    - What kind of software supports is needed?
    - Analyzing and optimizing parallel programs is still a hard problem.
    - How to debug concurrent program?

Issues With Parallel Computing

- Data sharing – single versus multiple address space.
- Process coordination – synchronization using locks, messages, and other means.
- Distributed versus centralized memory.
- Connectivity – single shared bus versus network with many different topologies.
- Fault tolerance/reliability.

Distributed System Issues

- Transparency
  - Access (data representation), location, migration, relocation, replication, concurrency, failure, persistence
- Scalability
  - Size, geographically
- Reliability/fault tolerance