Introduction to Distributed and Parallel Computing
CS-401

Dr. Sanjay Saxena

Visiting Faculty, CSE, IIIT Vadodara

Assistant Professor, CSE, IIIT Bhubaneswar

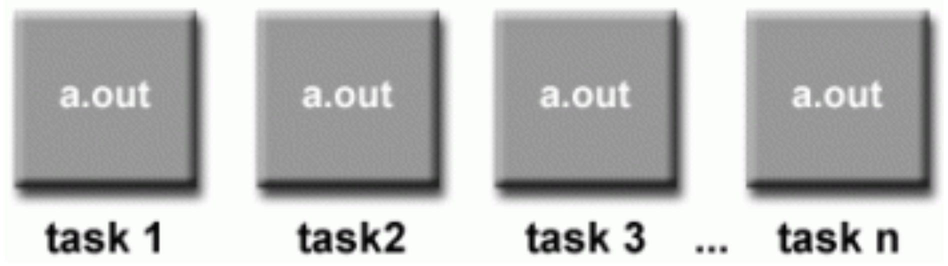
Post doc – University of Pennsylvania, USA

PhD – Indian Institute of Technology(BHU), Varanasi

SPMD and MPMD

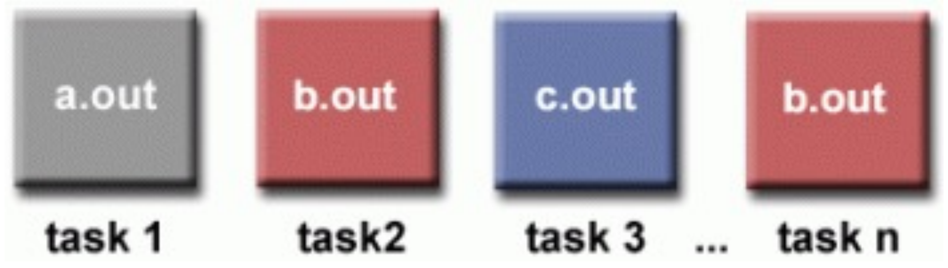
Single Program Multiple Data (SPMD)

- SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- SINGLE PROGRAM: All tasks execute their copy of the same program simultaneously. This program can be threads, message passing, data parallel or hybrid.
- MULTIPLE DATA: All tasks may use different data
- SPMD programs usually have the necessary logic programmed into them to allow different tasks to branch or conditionally execute only those parts of the program they are designed to execute. That is, tasks do not necessarily have to execute the entire program - perhaps only a portion of it.
- The SPMD model, using message passing or hybrid programming, is probably the most commonly used parallel programming model for multi-node clusters.



Multiple Program Multiple Data (MPMD)

- Like SPMD, MPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- MULTIPLE PROGRAM: Tasks may execute different programs simultaneously. The programs can be threads, message passing, data parallel or hybrid.
- MULTIPLE DATA: All tasks may use different data
- MPMD applications are not as common as SPMD applications, but may be better suited for certain types of problems, particularly those that lend themselves better to functional decomposition than domain decomposition (discussed later under Partitioning).



GPU (Graphics Processing Unit)

- GPUs are also known as video cards or graphics cards.
- In order to display pictures, videos, and 2D or 3D animations, each device uses a GPU.
- A GPU performs fast calculations of arithmetic and frees up the CPU to do different things.
- A GPU has lots of smaller cores made for multi-tasking, while a CPU makes use of some cores primarily based on sequential serial processing.
- Originally intended for sending an image to the pixels on the screen, someone then figured out that by treating data as a texture map you could perfom lots of calculations simultanously, and the age of the GPGPU (general-purpose graphics processing unit) began.

CPU strengths	GPU strengths
Very large main memory	High bandwidth main memory
Very fast clock speeds	Latency tolerant via parallelism
Latency optimized via large caches	Significantly more compute resources
Small number of threads can run very quickly	High throughput
	High performance/watt

CPU weaknesses	GPU weaknesses
Relatively low memory bandwidth	Relatively low memory capacity
Low performance/watt	Low per-thread performance

What's better?

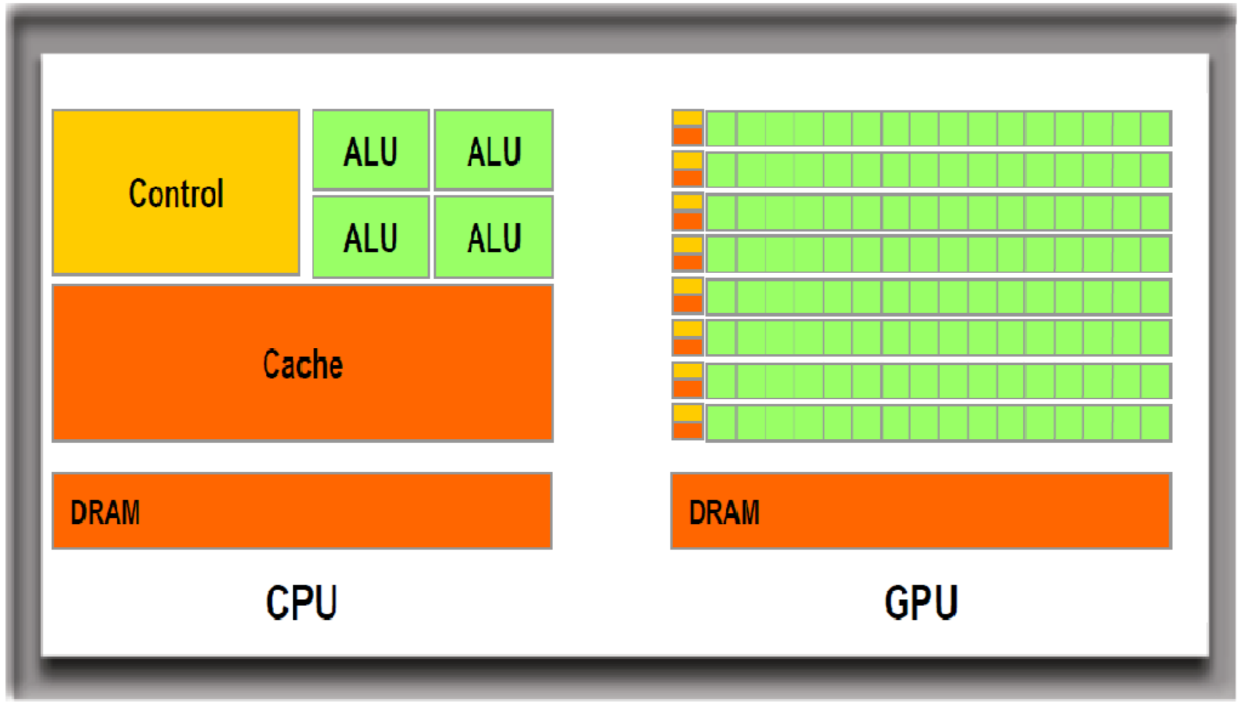
What's better?

Deliver many packages within a reasonable timescale.

Deliver a package as soon as possible

How CPU and GPU Work Together

- A CPU works together with a GPU to increase the throughput of data and the number of concurrent calculations within an application.
- GPUs were originally designed to create images for computer graphics and video game consoles, but since the early 2010's, GPUs can also be used to accelerate calculations involving massive amounts of data.
- A CPU can never be fully replaced by a GPU: a GPU complements CPU architecture by allowing repetitive calculations within an application to be run in parallel while the main program continues to run on the CPU.



- The CPU can be thought of as the taskmaster of the entire system, coordinating a wide range of general-purpose computing tasks, with the GPU performing a narrower range of more specialized tasks (usually mathematical).

Why is a GPU suitable for image processing?

- **Shared Memory:** Almost every modern GPU has shared memory, making it way better and exponentially faster than the CPU's cache. It works best with algorithms that have a high degree of locality.
- **Managing Load:** In contrast to a CPU, a GPU can considerably reduce the load on a subsystem by modifying the number of registers.
- **Speed:** The parallel processing aspect of GPU makes it much faster than a CPU because of the better memory and processing power bandwidth. GPUs are almost 100x quicker in processing than a CPU.
- **Embedded Applications:** GPUs are a better alternative to embedded applications like ASICs and FPGAs as they offer more flexibility. Parallel execution of various tasks: The different hardware modules of a GPU allow entirely diverse tasks to be executed simultaneously—for instance, tensor kernels for neural networks.

Why is GPU's used?

GPU's for gaming

GPU for Machine Learning

GPU for Video Editing and Content Creation

Designing Parallel Programs

Automatic vs. Manual Parallelization

- Designing and developing parallel programs has characteristically been a very manual process. The programmer is typically responsible for both identifying and actually implementing parallelism.
- Very often, manually developing parallel codes is a time consuming, complex, error-prone and iterative process.
- For a number of years now, various tools have been available to assist the programmer with converting serial programs into parallel programs. The most common type of tool used to automatically parallelize a serial program is a parallelizing compiler or pre-processor.
- A parallelizing compiler generally works in two different ways:

Fully Automatic

- The compiler analyzes the source code and identifies opportunities for parallelism.
- The analysis includes identifying inhibitors to parallelism and possibly a cost weighting on whether or not the parallelism would actually improve performance.
- Loops (do, for) are the most frequent target for automatic parallelization.

Programmer Directed

- Using "compiler directives" or possibly compiler flags, the programmer explicitly tells the compiler how to parallelize the code.
- May be able to be used in conjunction with some degree of automatic parallelization also.
- The most common compiler generated parallelization is done using on-node shared memory and threads (such as OpenMP).
- If you are beginning with an existing serial code and have time or budget constraints, then automatic parallelization may be the answer. However, there are several important caveats that apply to automatic parallelization:
    - Wrong results may be produced
    - Performance may actually degrade
    - Much less flexible than manual parallelization
    - Limited to a subset (mostly loops) of code
    - May actually not parallelize code if the compiler analysis suggests there are inhibitors or the code is too complex
- The remainder of this section applies to the manual method of developing parallel codes.

What is Java?

- Java is a high-level, general-purpose, object-oriented, and secure programming language developed by James Gosling at Sun Microsystems, Inc. in 1991.
- It is formally known as OAK. In 1995, Sun Microsystem changed the name to Java. In 2009, Sun Microsystem takeover by Oracle Corporation.

Features of Java

- **Simple:** Java is a simple language because its syntax is simple, clean, and easy to understand. Complex and ambiguous concepts of C++ are either eliminated or re-implemented in Java. For example, pointer and operator overloading are not used in Java.

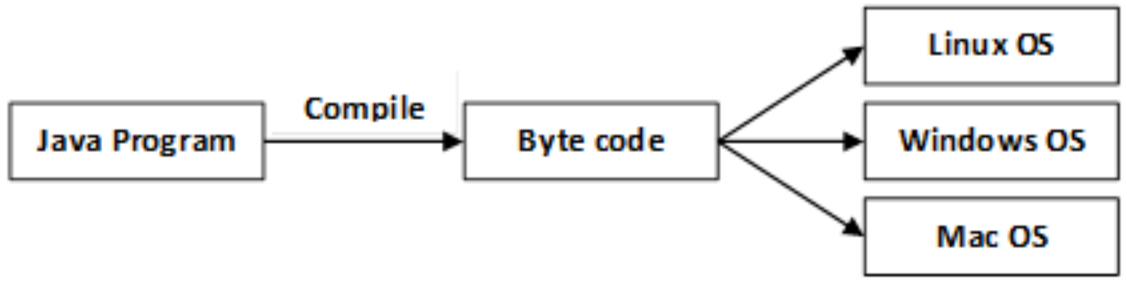
- **Object-Oriented:** In Java, everything is in the form of the object. It means it has some data and behavior. A program must have at least one class and object.

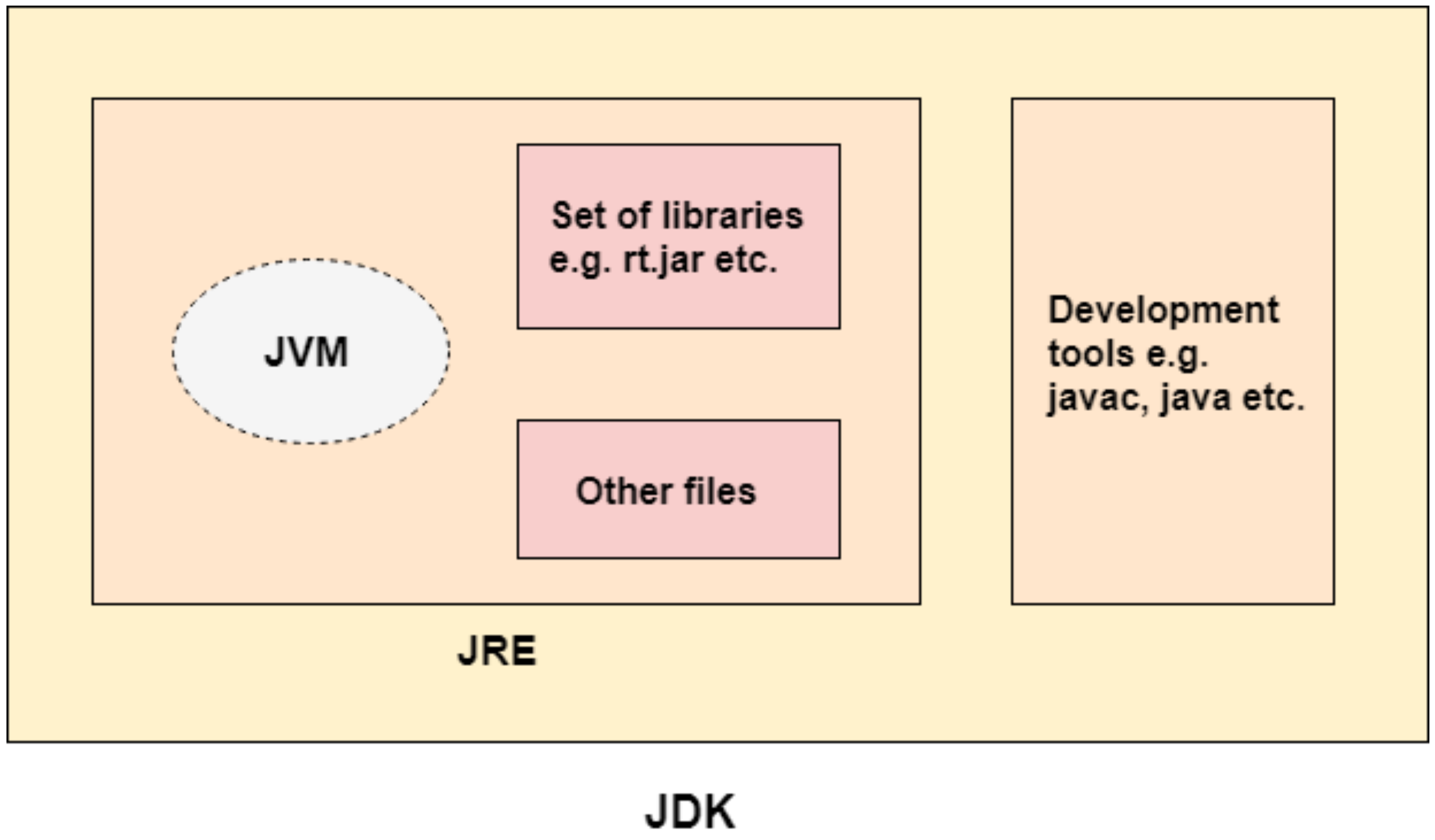
- **Robust:** Java makes an effort to check error at run time and compile time. It uses a strong memory management system called garbage collector. Exception handling and garbage collection features make it strong.

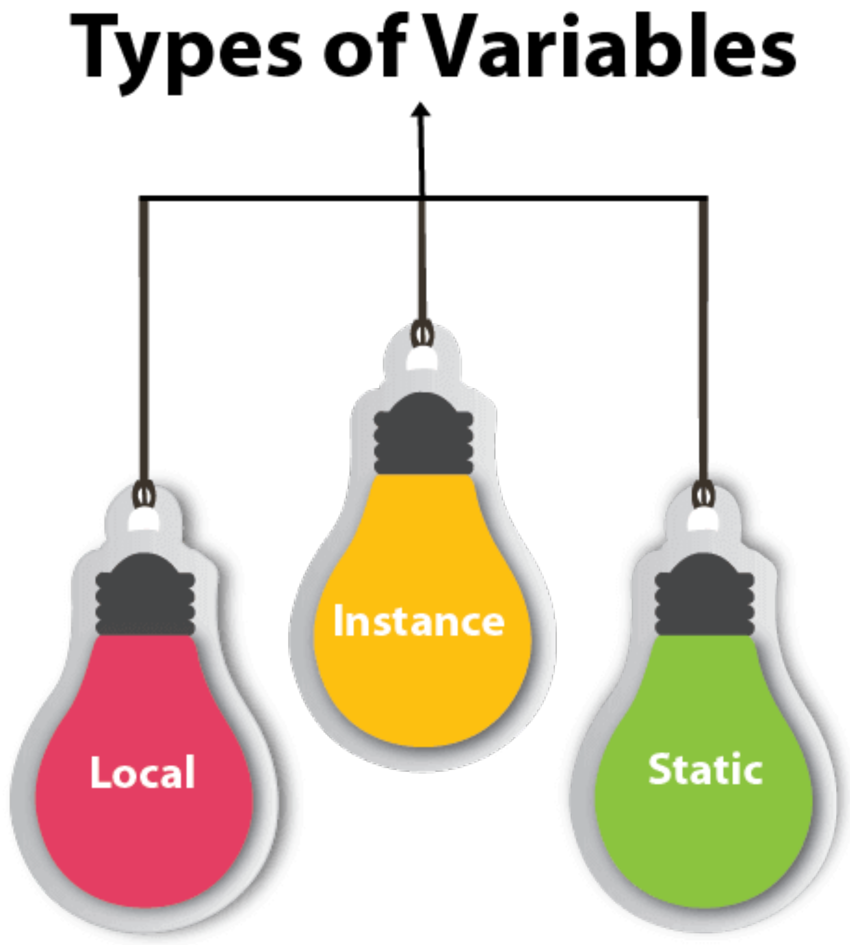
- **Secure:** Java is a secure programming language because it has no explicit pointer and programs runs in the virtual machine. Java contains a security manager that defines the access of Java classes.

- **Platform-Independent:** Java provides a guarantee that code writes once and run anywhere. This byte code is platform-independent and can be run on any machine.

- **Portable:** Java Byte code can be carried to any platform. No implementation-dependent features. Everything related to storage is predefined, for example, the size of primitive data types.
- **High Performance:** Java is an interpreted language. Java enables high performance with the use of the Just-In-Time compiler.
- **Distributed:** Java also has networking facilities. It is designed for the distributed environment of the internet because it supports TCP/IP protocol. It can run over the internet. EJB and RMI are used to create a distributed system.
- **Multi-threaded:** Java also supports multi-threading. It means to handle more than one job a time.




Set of libraries e.g. rt.jar etc.
JVM
Other files
Development tools e.g. javac, java etc.
JRE
JDK



- **Local Variable**

A variable declared inside the body of the method is called a local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with a "static" keyword.

- **Instance Variable**

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as static.

It is called an instance variable because its value is instance-specific and is not shared among instances.

- **Static Variable**

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

```
public class Simple
{
        public static void main(String[] args)
        {
                int a=10;
                float f=a;
                System.out.println(a);
                System.out.println(f);
        }
}
```

Thanks & Cheers!!

Small aim is a crime; have great aim.

Bharat-Ratan A. P. J. Abdul Kalam