E0-245: ASP

Lecture 11: Multithreading and Synchronization

Dipanjan Gope

Module 1: OOPs and DS

JAVA and C++ mainly: (others C# and Objective-C)

- Object oriented programing: Classes and objects
- Inheritance, polymorphism, abstract class, const
- Templates and generics
- Data structures
- Standard library, JCF, STL
- Complexity analysis
- **Multithreading and synchronization**
- Good programming styles

Highway Driving

Single Thread : One Lane Road
CPU Frequency: Speed Limit
Instructions : Vehicles
Through-put : Vehicles per min

Speed Limit = 30kmph

Through-Put = $\frac{\text{Speed}}{\text{Car To Car Length}}$

50 cars per minute

How do you increase the through-put?

- Increase speed-limit: 30-60-80---- SORRY
- Decrease car to car length: SORRY
- ???

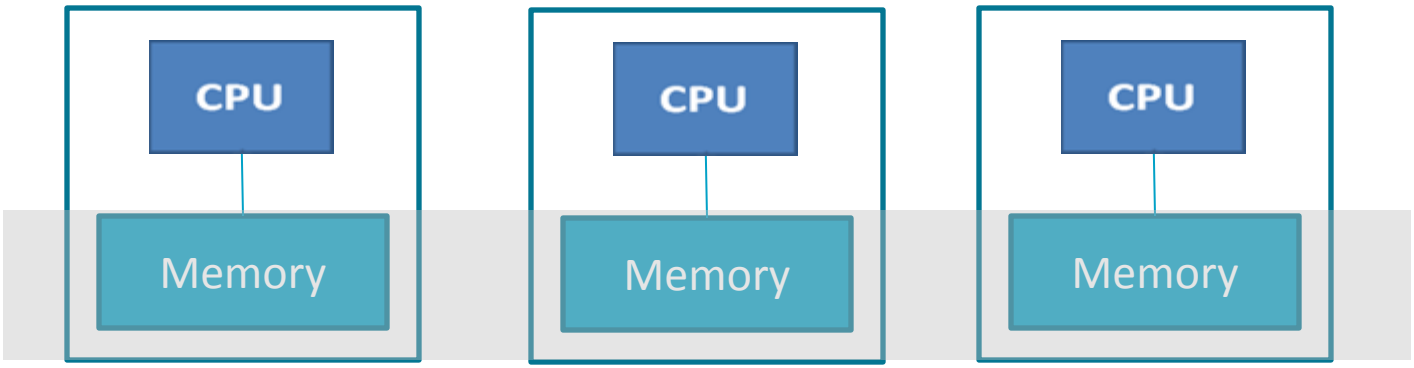
Highway Driving

Through-Put = Through-Put x Lanes

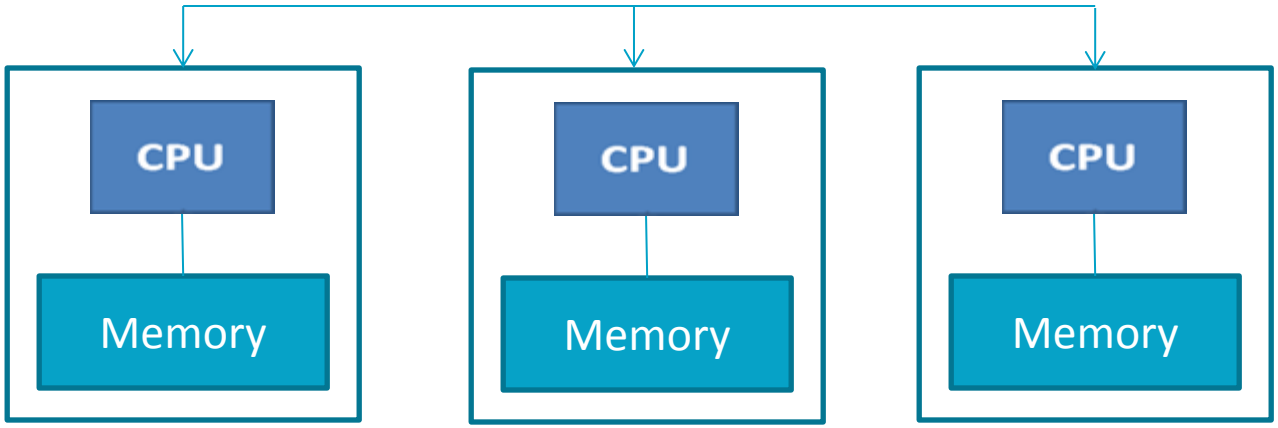
Scaling = Lanes = n?

10 years back …

MPI

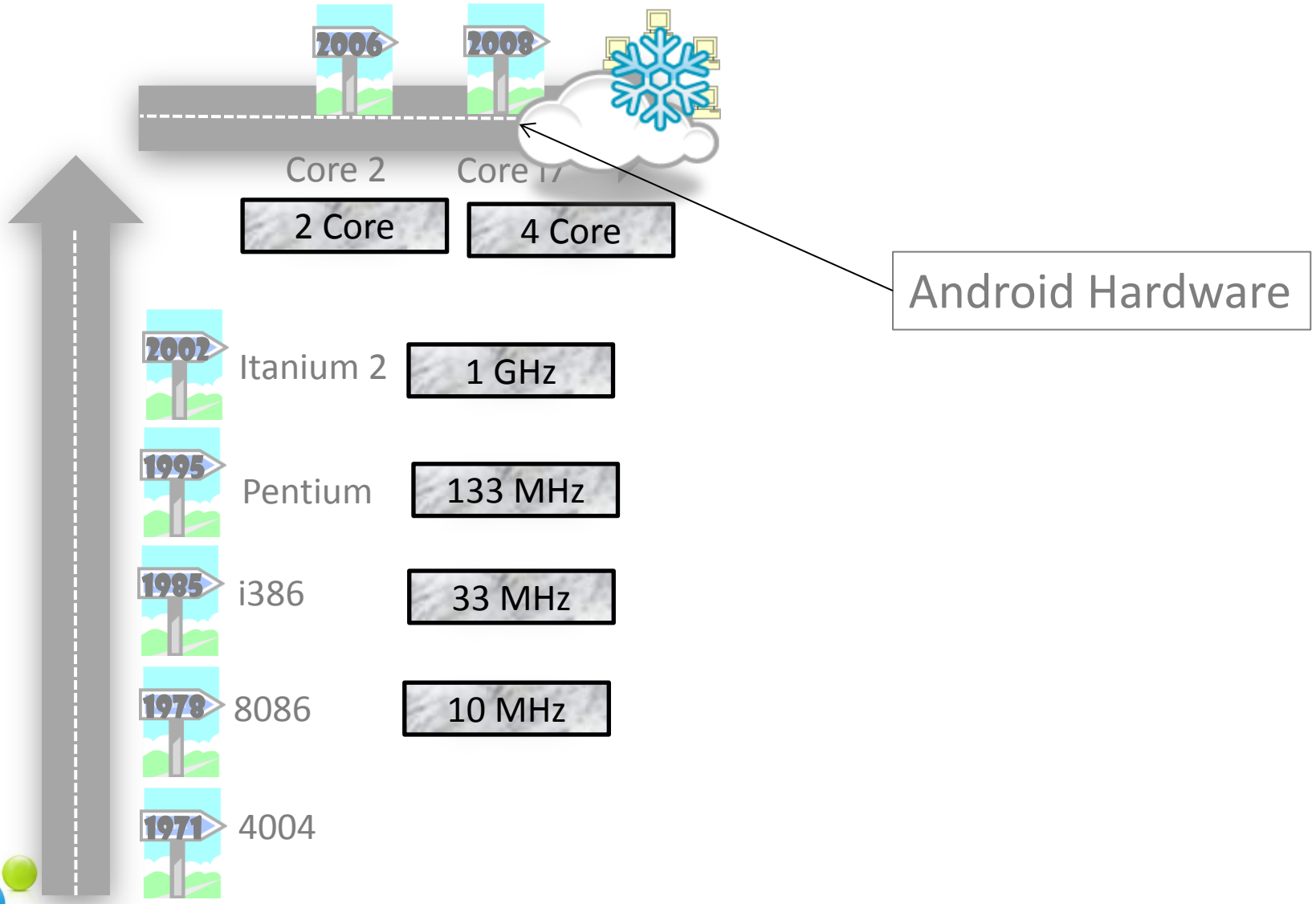


SGE/
LSF



Parallel Programming restricted to large MNCs and national labs

Hardware Roadmap



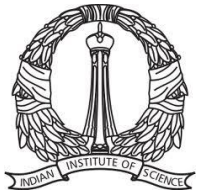
CPU-Parallelization Today

- **Message Passing:**
  - MPI (Cluster)

- **Explicit Threading:**
  - Java: Threads
  - C++: Pthreads

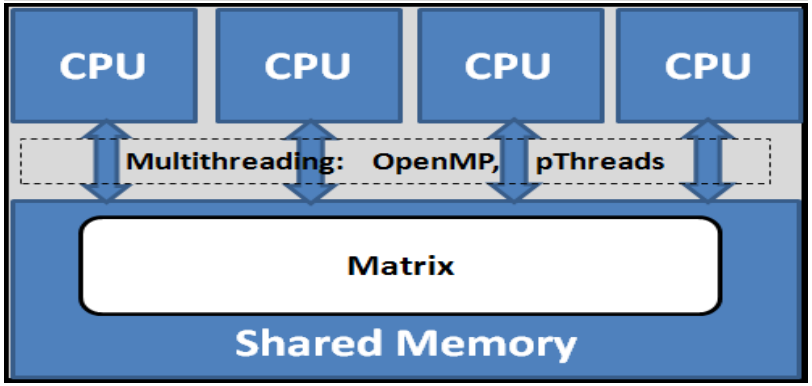
- **Compiler-directed:**
  - Open-MP

- **Libraries:**
  - PARDISO, ScaLapack, PLAPACK, PETSc

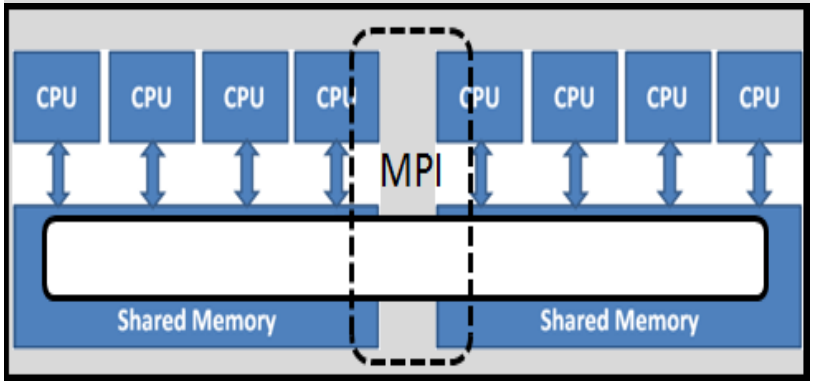


Parallelization Paradigms

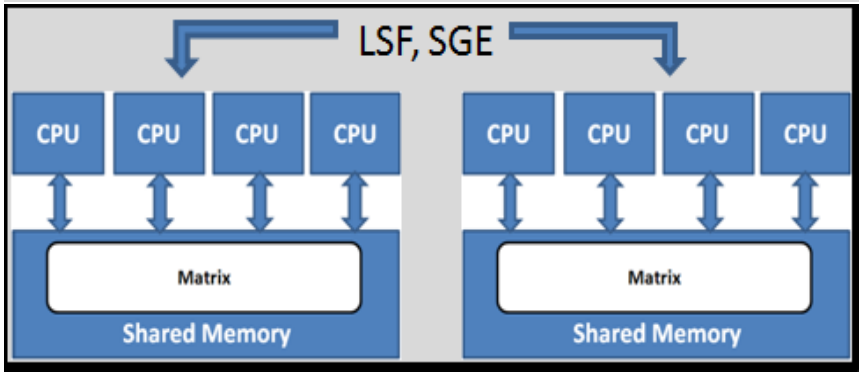
Multithreading



MPI



Distribution



Multithreading: Easy Way

Tool.exe

TestCase 1

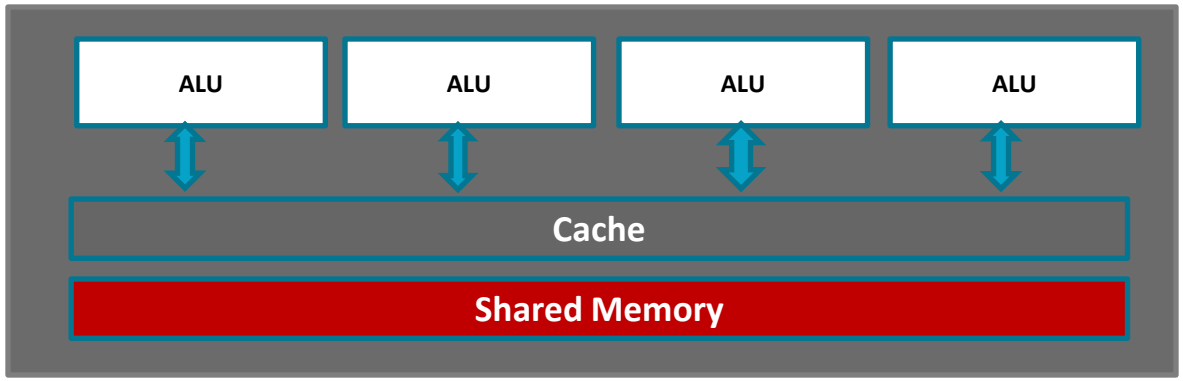
TestCase 2

TestCase 3

TestCase 4

TestCase ..

TestCase ..



"Embarrassingly Parallel" Class of Problems

Lets code …

Example 1: Hello World

```
public class testHelloWorldMT {
    public static void main(String[] args)
    {
        Runnable runnable = new Runnable()
        {
            public void run()
            {
                System.out.println("Hello, World!");
            }
        };

        Thread thread = new Thread(runnable);
        thread.start();
    }
}
```

http://www.diva-portal.org/smash/get/diva2:427682/FULLTEXT01.pdf

Example 2: Multiple Threads

```
class RunnableDemo implements Runnable {
   private Thread t;
   private String threadName;

   RunnableDemo( String name){
       threadName = name;
       System.out.println("Creating " +  threadName );
   }
   public void run() {
      System.out.println("Running " +  threadName );
      try {
         for(int i = 4; i > 0; i--) {
            System.out.println("Thread: " + threadName + ",
            // Let the thread sleep for a while.
            Thread.sleep(5000);
         }
      } catch (InterruptedException e) {
         System.out.println("Thread " +  threadName + " int
      }
      System.out.println("Thread " +  threadName + " exiting
   }

   public void start ()
   {
      System.out.println("Starting " +  threadName );
      if (t == null)
      {
         t = new Thread (this, threadName);
         t.start ();
      }
   }
```

```
public class testMT {
      public static void main(String args[]) {

         RunnableDemo R1 = new RunnableDemo( "Thread-1");
         R1.start();

         RunnableDemo R2 = new RunnableDemo( "Thread-2");
         R2.start();
      }
}
```

Problems @ Javadoc Declaration Console

```
<terminated> testMT [Java Application] C:\Program Files\Java\
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 4
Thread: Thread-1, 3
Thread: Thread-2, 3
Thread: Thread-2, 2
Thread: Thread-1, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
```

http://www.tutorialspoint.com/java/java_multithreading.htm

Multithreading in JAVA

- *java.lang.Threads executes java.lang.Runnable*
- *JOMP (JAVA OpenMP)*
- *PJ (Parallel JAVA)*
- *DPJ (Deterministic Parallel JAVA)*
- *JConqurr*
- *JaMP (JAVA Open-MP)*

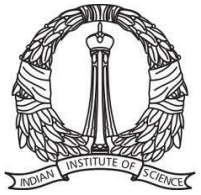
Multithreading in JAVA

http://www.diva-portal.org/smash/get/diva2:427682/FULLTEXT01.pdf

	JOMP	PJ	DPJ	JConqurr	JaMP
Pure Java	Yes	Yes	Yes	Yes	Yes
Active Project	No	Yes	Yes	Yes	Yes
Design Time Checking	No	Yes	No	Yes	No
Source Code Available	No	Yes	Yes	Yes	Yes
Existing Code can run without using API	Yes	No	No	No	Yes
Easy to use	Yes	No	No	Yes	Yes
Any other Issue?	Does not support Java 1.5.	Using a lot of memory. Got out of memory error in our Matrix multiplication experiment.	No	Toolkit not working properly.	No

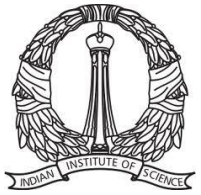
Open MP Contents

- Runtime functions/ environment variables
- Constructs:
  - Parallel regions
  - Work-sharing
- Data environment
- Synchronization



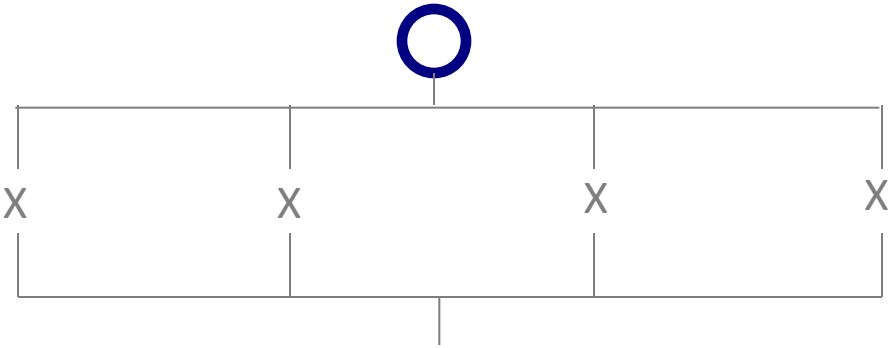
OpenMP: Runtime Functions

- **Modify/Check Number of Threads**
  - omp [set/get] num_threads()
  - omp_get_thread_num()
  - omp_get_max_threads()
- **Parallel Region**
  - omp_in_parallel()
- **How many Processors**
  - omp_get_num_procs()
- **Explicit Locks**
  - omp_[set/unset]_lock()
- **More.. + Environment Variables**
  - Consult manual



OMP: Parallel Regions

```
#pragma omp parallel
{
        X
}
```

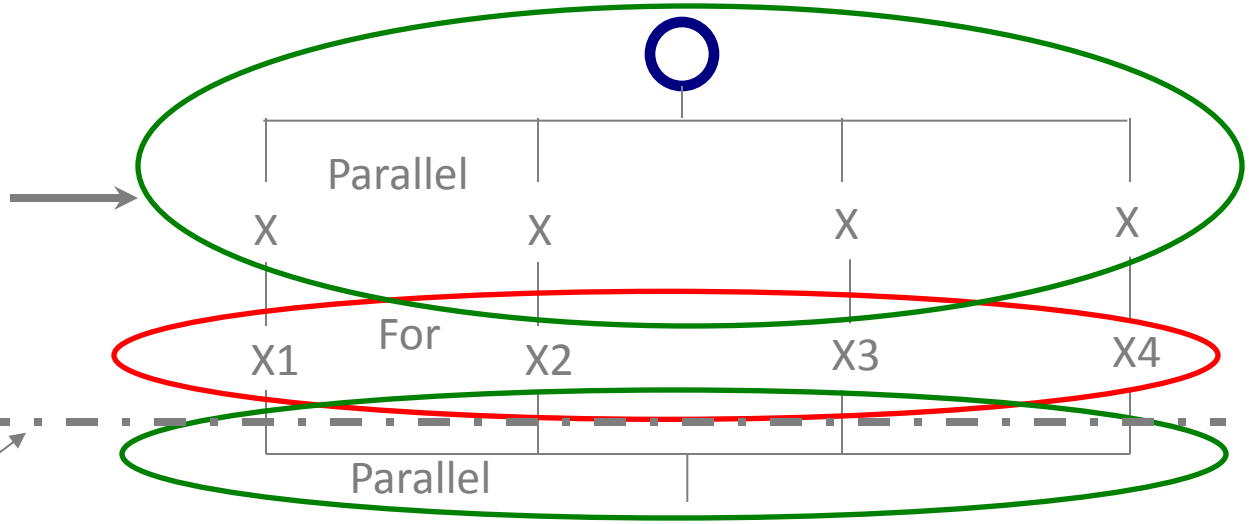



```
        omp_set_num_threads(4);
#pragma omp parallel
        {
          myID = omp_get_thread_num ();
          a[myID] = myID;
        }
#endif
```

a[0]=0
a[1]=1
a[2]=2
a[3]=3

OMP: Work sharing

```
#pragma omp parallel
{
#pragma omp for
    {
        X
    }
}
```




```
#pragma omp parallel
{
#pragma omp for
        for (i=0;i<4;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
}
```

a[0]=0
a[1]=1
a[2]=2
a[3]=3

Dipanjan Gope

OMP: Worksharing

```
#pragma omp parallel
{
#pragma omp for
        for (i=0;i<N;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
}
#endif
```

a[0]=0
a[1]=1
a[2]=2
a[3]=3

a[0]=3
a[1]=3
a[2]=3
a[3]=3

```
#pragma omp parallel
{
//#pragma omp for
        for (i=0;i<N;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
}
#endif
```

OMP: Worksharing

```
#pragma omp parallel
{
#pragma omp for
        for (i=0;i<N;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
}
#endif
```

```
Time 4.797000 seconds
Press any key to continue_
```

```
Time 20.031000 seconds
Press any key to continue_
```

```
#pragma omp parallel
{
//#pragma omp for
        for (i=0;i<N;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
}
#endif
```

OMP: Worksharing

```
#pragma omp parallel
{
#pragma omp for
        for (i=0;i<N;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
}
#endif
```

```
#pragma omp parallel for
        for (i=0;i<N;i++)
        {
          myID = omp_get_thread_num ();
          a[i] = myID;
        }
#endif
```

An Example Code

```
#pragma omp parallel for
for(i=0; i<N; i++)
{
   int type = i%2;
   switch (type)
   {
   case 0:
      evenSquare += (i*i);
      break;

   case 1:
      oddSquare += (i*i);
      break;
   }
}
```


Terminologies

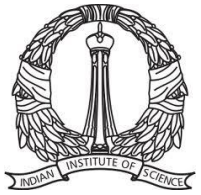
- **Correctness Concept:**

  - Race Condition, Deadlock, Synchronization

- **Performance Concept:**

  - Speedup<>Memory, Efficiency

  - Granularity, Load Imbalance



Storage Attributes

- Shared

```
#pragma omp parallel
for (i = 0; i < IMAX; i++)
{
        ii = i * i;
        isquared[i] = ii;
}
```

- Private

```
#pragma omp parallel for private(ii)
for (i = 0; i < IMAX; i++)
{
        ii = i * i;
        isquared[i] = ii;
}
```

- Firstprivate

```
iinit = 0;
#pragma omp parallel for
for (i = 0; i < IMAX; i++)
{
        if (i > 0) iinit = i * i;
        work[i] = iinit;
}
```

```
iinit = 0;
#pragma omp parallel for firstprivate(iinit)
for (i = 0; i < IMAX; i++)
{
        if (i > 0) iinit = i * i;
        work[i] = iinit;
}
```

- Lastprivate

```
#pragma omp parallel for
for (i = 0; i < IMAX; i++)
{
        if (i > 0) iinit = i * i;
        work[i] = iinit;
}
printf ("Final value of loop index was %d\n", i);
```

```
#pragma omp parallel for lastprivate(i)
for (i = 0; i < IMAX; i++)
{
        if (i > 0) iinit = i * i;
        work[i] = iinit;
}
printf ("Final value of loop index was %d\n", i);
```

- Reduction

```
isqsum = 0;
#pragma omp parallel for
for (i = 0; i < IMAX; i++)
isqsum += isquared[i];
```

```
isqsum = 0;
#pragma omp parallel for reduction (+: isqsum)
for (i = 0; i < IMAX; i++)
isqsum += isquared[i];
```

- Threadprivate

NO YES

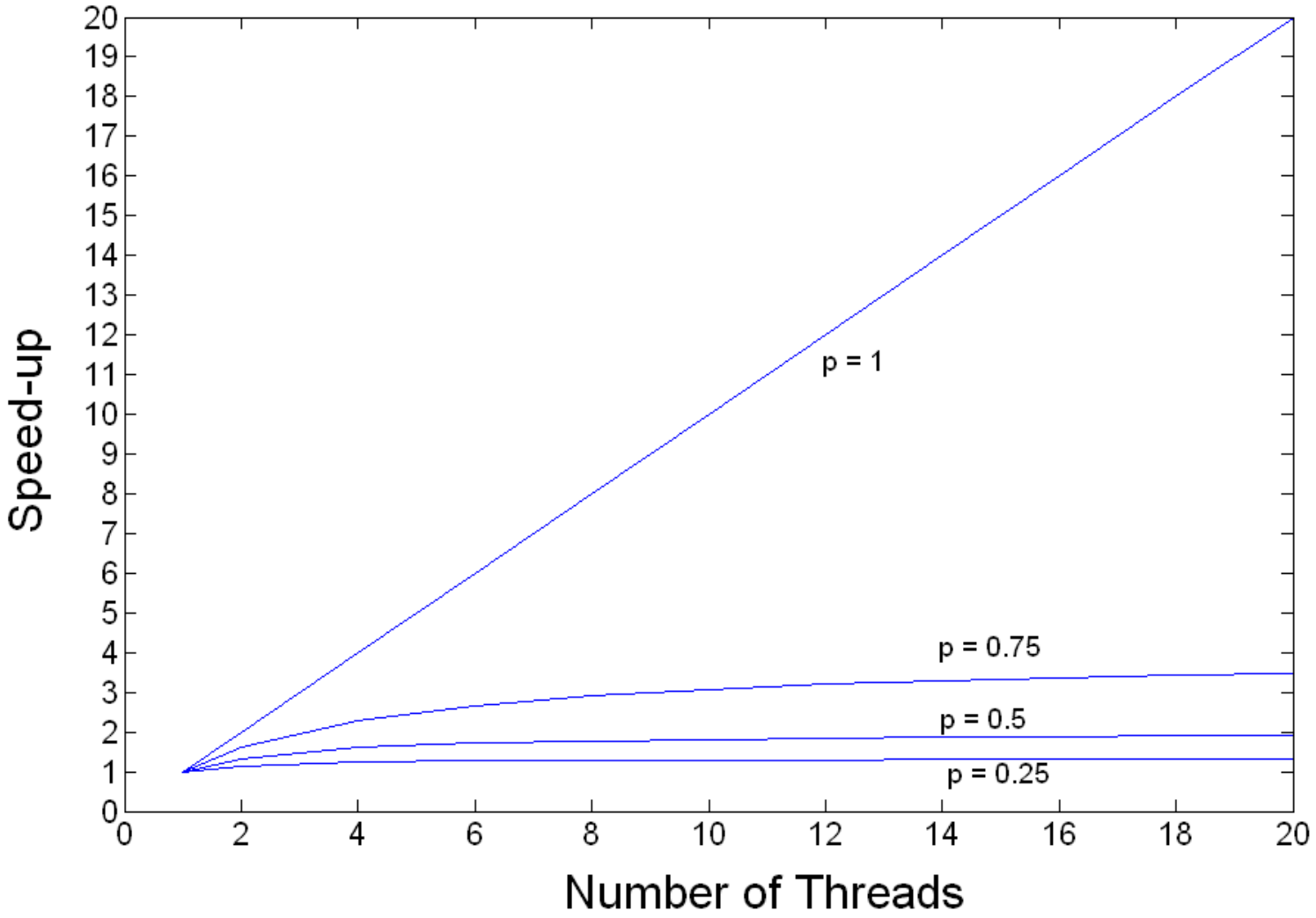
Challenge 1: Serial Content

Minimize Serial Code: Amdahl's Law

- $T_{parallel} = \{(1-P) + P/n\} T_{serial} + O$
- Scaling = $T_{serial} / T_{parallel}$

Challenge 1: Make P = 1 Such That Scaling = n

Speedup vs. N


Speed-up
Number of Threads
p = 1
p = 0.75
p = 0.5
p = 0.25
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
0
2
4
6
8
10
12
14
16
18
20

Challenge 2: Shared/Private Memory

Private Memory

- Expensive Memory
- Cheap Run-Time
- Accuracy OK
- Data Environment

Shared Memory

- Cheap Memory
- Oops Run-Time
- Accuracy Issue
- Synchronization

Synchronization

Synchronization:

- **Shared Memory Read-Write**

  - Signaling of Threads:

Critical Region

Problem

Solution

Synchronization

- Critical
- Atomic
- Barrier
- Flush
- Ordered
- Master
- Single

Signaling of Threads

Synchronization Terminology

CAUSE

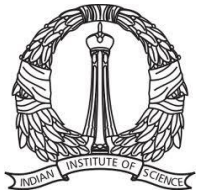
- Critical Region
  - Global Data
  - I/O

EFFECT

- Race Condition
- Deadlock

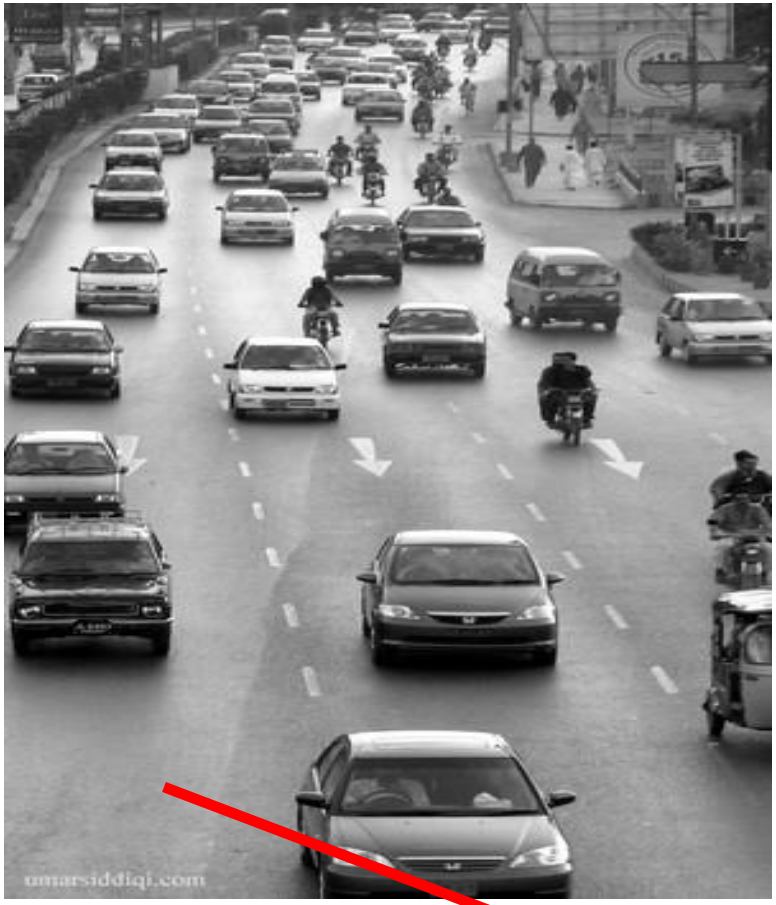
SOLUTION

- Critical Section: x
- Mutex: Mutual Exception: 200x
- Synchronization Barriers: 200x



Challenge 3: Load Balancing

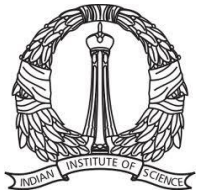
Load Balancing:



Challenge 4: Granularity

Granularity

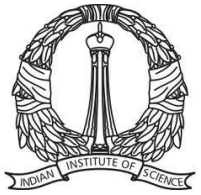
- Actual Work / Synchronization Work
- Make sure you have enough work to parallelize



Challenge 5: Memory Related

Memory Related

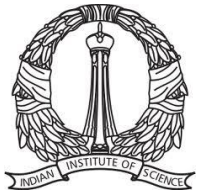
- NUMA
- cache architecture
- false sharing
- shared cache



Challenge 6: Parallel Overhead

Parallel Overhead

- Creation and deletion of threads take time
- Can use thread pool



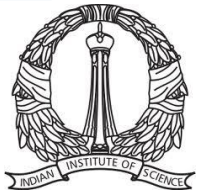
Benefits of Proper MT

Perfect Linear Scaling (100% parallel content)

	Time	Utility Computing Cost
1 Compute Unit	10 hours	10h x Rs1Unit = 10
10 Compute Units	1 hours	1h x Rs10Unit = 10
Speedup	**10x**	**Same**

50% parallel content

	Time	Utility Computing Cost
1 Compute Unit	10 hours	10h x Rs1Unit = 10
10 Compute Units	5.5 hours	5.5h x Rs10Unit = 55
Speedup	**1.8x**	**5.5x**



Debugging

- Print Statements
  - Can make bugs seem to “go-away”
  - Include thread ID in print message
  - Protect I/O with a critical section