

## The Parallel Architecture Approach, Single Program Multiple Data (Spmd) Implementation on Clusters of Terminals Using Java Rmi

Sudhir Kumar Meesala<sup>1</sup>, Dr. Pabitra Mohan Khilar<sup>2</sup>, Dr. A. K. Shrivastava<sup>3</sup>

<sup>1</sup>ph.D. Scholar, Dept. Of Computer Science & Engineering, Dr. C.V. Raman University, Kota, Bilaspur(Cg), India

<sup>2</sup>assistant Professor, Department Of Computer Science And Engineering, Nit, Rourkela(Orrissa), India

<sup>3</sup>professor & Head, Department Of Physics, Dr. C. V. Raman University, Kota, Bilaspur(Cg), India

### Abstract

The Complexity Of Computation Computer Power Is Unexpectedly Increasing Day By Day. Today's High Level Computer And Its High Level Utility Is Already Effected Each And Every Part Of Of Our Real Life. We All Know That Computer Power Is Effected From Astrophysics To Rural Areas And It Covers All Internal Subareas Of Each And Every Organization Either It Is A Related National Level Government Project Or International Level Projects. Many Scientific, Economic, And Research Areas Need A Specific Power To Solve Their Unsolved, Large And Complex Problems, But Maximum Solution Are Highly Economic Effective And Expensive. The Numeric Simulation Of Complex Systems Like Molecular Biology, Weather Forecast, Climate Modeling, Circuit Design, Biometric, Re-Engineering, Recycling Engineering And Many More Are Some Of Such Problems. There Are Many Approaches To Solve Them. But Two Major Effective Solutions Are Either An Expensive Parallel Supercomputer Has To Be Used [First], Or The Computer Power Of Workstations In A Net Can Be Bundle To Computer The Task Distributed [Second]. The Second Approach Has The Advantage That We Use The Available Hardware Cost-Effective. This Paper Describes The Architecture Of A Heterogeneous, Concurrent, And Distributed System, Which Can Be Used For Solving Large Computational Problems. Here We Present The Basic Solution By Single Program Stream And Multiple Data Stream (SPMD) Architecture For Solving Large Complex Problem. We Present A Concurrent Tasks Distributed Application For Solving Complex Computational Tasks In Parallel. The Design Process Is Parallel Processing Implementation On Clusters Of Terminals Using Java RMI.

**Keywords:** Single Program Multiple Data (SPMD), Remote Method Invocation (RMI), Parallel Processing, Distributed Technology, Molecular Biology, Weather Forecast, Climate Modeling, Circuit Design, Biometric, Re-Engineering, Recycling Engineering

### I. Introduction

In Computer Calculations, SPMD (Single Program, Multiple Data) Is A Technique Employed To Achieve Parallelism; It Is A Subcategory Or One Of The Basic Type Of MIMD. Tasks Are Split Up And Run Simultaneously On Multiple Processors With Different Input In Order To Obtain Results Faster. SPMD Is The Most Common Style Of Parallel Programming.[1] It Is Also A Essential For Research Concepts Such As Active Messages And Distributed Shared Memory.

#### Deference Between SPMD & SIMD

In SPMD, Multiple Autonomous Processors Simultaneously Execute The Same Program At Independent Points, Rather Than In The Lockstep That SIMD Imposes On Different Data. With SPMD, Tasks Can Be Executed On General Purpose Cpus; SIMD Requires Vector Processors To Manipulate Data Streams. Note That The Two Are Not Mutually Exclusive.

#### Concept Of Distributed Memory

SPMD Usually Refers To Message Passing Programming On Distributed Memory Computer Architectures. A Distributed Memory Computer Consists Of A Collection Of Independent Computers, Called Nodes. Each Node

Starts Its Own Program And Communicates With Other Nodes By Sending And Receiving Messages, Calling Send/Receive Routines For That Purpose. Barrier Synchronization May Also Be Implemented By Messages. The Messages Can Be Sent By A Number Of Communication Mechanisms, Such As TCP/IP Over Ethernet, Or Specialized High-Speed Interconnects Such As Myrinet And Supercomputer Interconnect. Serial Sections Of The Program Are Implemented By Identical Computation On All Nodes Rather Than Computing The Result On One Node And Sending It To The Others.

Nowadays, The Programmer Is Isolated From The Details Of The Message Passing By Standard Interfaces, Such As PVM And MPI.

Distributed Memory Is The Programming Style Used On Parallel Supercomputers From Homegrown Beowulf Clusters To The Largest Clusters On The Teragrid.

### **Concept Of Shared Memory**

On A Shared Memory Machine (A Computer With Several Cpus That Access The Same Memory Space), Messages Can Be Sent By Depositing Their Contents In A Shared Memory Area. This Is Often The Most Efficient Way To Program Shared Memory Computers With Large Number Of Processors, Especially On NUMA Machines, Where Memory Is Local To Processors And Accessing Memory Of Another Processor Takes Longer. SPMD On A Shared Memory Machine Is Usually Implemented By Standard (Heavyweight) Processes. Unlike SPMD, Shared Memory Multiprocessing, Also Called Symmetric Multiprocessing Or SMP, Presents The Programmer With A Common Memory Space And The Possibility To Parallelize Execution By Having The Program Take Different Paths On Different Processors. The Program Starts Executing On One Processor And The Execution Splits In A Parallel Region, Which Is Started When Parallel Directives Are Encountered. In A Parallel Region, The Processors Execute A Single Program On Different Data. A Typical Example Is The Parallel DO Loop, Where Different Processors Work On Separate Parts Of The Arrays Involved In The Loop. At The End Of The Loop, Execution Is Synchronized, Only One Processor Continues, And The Others Wait. The Current Standard Interface For Shared Memory Multiprocessing Is Openmp. It Is Usually Implemented By Lightweight Processes, Called Threads.

### **Combination Of Levels Of Parallelism**

Current Computers Allow Exploiting Of Many Parallel Modes At The Same Time For Maximum Combined Effect. A Distributed Memory Program Using MPI May Run On A Collection Of Nodes. Each Node May Be A Shared Memory Computer And Execute In Parallel On Multiple Cpus Using Openmp. Within Each CPU, SIMD Vector Instructions (Usually Generated Automatically By The Compiler) And Superscalar Instruction Execution (Usually Handled Transparently By The CPU Itself), Such As Pipelining And The Use Of Multiple Parallel Functional Units, Are Used For Maximum Single CPU Speed.

### **SPMD History**

SPMD Was Proposed First In 1983 By Michel Auguin (University Of Nice Sophia-Antipolis) And François Larbey (Thomson/Sintra) In The OPSILA Parallel Computer[2] And Next In 1984 By Frederica Damera At IBM For Highly Parallel Machines Like The RP3 (The IBM Research Parallel Processor Prototype), In An Unpublished IBM Memo.[3] By The Late 1980s, There Were Many Distributed Computers With Proprietary Message Passing Libraries. The First SPMD Standard Was PVM. The Current De Facto Standard Is MPI.

### **Parallel & Distributed Architecture**

Distributed Computing Is Method Of Computer Processing In Which Different Parts Of A Program Run Simultaneously On Two Or More Computers That Are Communicating With Each Other Over A Network. Distributed Computing Is A Type Of Parallel Computing.[7] But The Latter Term Is Most Commonly Used To Refer To Processing In Which Different Parts Of A Program Run Simultaneously On Two Or More Processor That Are Part Of The Same Computer. While Both Types Of Processing Require That A Program Be Parallelized - Divided Into Sections That Can Run Simultaneously, Distributed Computing Also Requires That The Division Of The Program Take Into Account The Different Environments On Which The Different Sections Of The Program Will Be Running. For Example, Two Computers Are Likely To Have Different File Systems And Different Hardware Components.[3]

Distributed Computing Is A Natural Result Of The Use Of Network To Allow Computers To Efficiently Communicate. But Distributed Computing Is Distinct From *networking*. The Latter Refers To Two Or More Computers Interacting With Each Other, But Not, Typically, Sharing The Processing Of A Single Program. The World Wide Web Is An Example Of A Network, But Not An Example Of Distributed Computing.[14]

There Are Numerous Technologies And Standards Used To Construct Distributed Computations, Including Some Which Are Specially Designed And Optimize For That Purpose, Such As Remote Procedure Calls (RPC), Remote Method Invocation (RMI) Or Net Remoting.[5]

Organizing The Interaction Between Each Computer Is Of Prime Importance. In Order To Be Able To Use The Widest Possible Range And Types Of Computers, The Protocol Or Communication Channel Should Not Contain Or Use Any Information That May Not Be Understood By Certain Machines. Special Care Must Also Be Taken That Messages Are Indeed Delivered Correctly And That Invalid Messages Are Rejected Which Would Otherwise Bring Down The System And Perhaps The Rest Of The Network.

Various Hardware And Software Architectures Are Used For Distributed Computing. At A Lower Level, It Is Necessary To Interconnect Multiple Cpus With Some Sort Of Network, Regardless Of Whether That Network Is Printed Onto A Circuit Board Of Made Up Of Loosely -Coupled Devices And Cables. At A Higher Level, It Is Necessary To Interconnect Processes Running On Those Cpus With Some Sort Of Communication System.

Distributed Programming Typically Falls Into One Of Several Basic Architecture Or Categories:

- Client-Server
- 3-Tier Architecture
- N-Tier Architecture
- Distributed Objects
- Loose Coupling Or Tight Coupling.

1) Client-Server-- Smart Client Code The Server For Data, Then Formats And Displays It To The User. Input At The Client Is Committed Back To The Server When It Represents A Permanent Change.

2) 3-Tier Architecture:- Three Tier Systems Move The Client Intelligence To A Middle Tier So That Stateless Clients Can Be Used. This Simplifies Application Deployment. Most Web Applications Are 3-Tier.

3)N-Tier Architecture:- N-Tier Refers Typically To Web Application Which Further Forward Their Request To Other Enterprise Services. This Type Of Application Is The One Most Responsible For The Success Of Application Servers.

1. Tightly Coupled (Clustered):- Refers Typically To A Set Of Highly Integrated Machines That Run The Same Process In Parallel, Subdividing The Task In Part That Are Made Individually By Each One, And Then Put Back Together To Make The Final Result.

2. Peer-To-Peer:- Architecture Where There Is No Special Machine Of Machines That Provide A Service Or Manage The Network Resources. Instead All Responsibilities Are Uniformly Divided Among All Machines, Known As Peers.

A Multi Computer System Is A System Made Up Of Several Independent Computers Interconnected By A Telecommunication Network. Multi Computer System Can Be Homogeneous Or Heterogeneous: A Homogeneous Distributed System Is One Where All Cpus Are Similar And Are Connected By A Single Type Of Network. They Are Often Used For Parallel Computing.[11][12].

A Heterogeneous Distributed System Is Made Up Of Different Kind Of Computers, Possibly With Vastly Differing Memory Sizes, Processing Power And Even Basic Underlying Architecture. They Are In Widespread Use Today, With Many Companies Adopting This Architecture Owing To The Speed With Which Hardware Goes Obsolete And The Cost Of Upgrading A Whole System Simultaneous.

The Types Of Distributed Systems Are Based On Flynn's Taxonomy Of Systems: -

- Single Instruction Single Data (SISD)
- Single Instruction Multiple Data (SIMD)
- Multiple Instruction Single Data (MISD)
- Multiple Instruction Multiple Data (MIMD)
- Single Program Multiple Data (SPMD)

We Are Implementing Client-Server Architecture And Single Program Multiple Date (SPMD) Taxonomy.

### **Remote Method Invocation(RMI)**

Remote Method Invocation (RMI) Allows A Java Object That Executes On One Machine To Invoke A Method Of A Java Object That Executes On Another Machine. This Is An Important Feature, Because It Allows You To Build Distributed Application. While A Complete Discussion Of RMI Is Outside The Scope Of This Paper, The Following Paper Describes The Basic Principles Of Java RMI.[22]. The RMI Implementation Is Essentially Built From Three Abstraction Layers

#### **A. The Stub/Skeleton Layer**

This Layer Intercepts Method Calls Made By The Client To The Interface Reference And Redirects These Calls To A Remote Object. Stubs Are Specific To The Client Side, Whereas Skeletons Are Found On The Server Side. To Achieve Location Transparency, RMI Introduces Two Special Kinds Of Objects Known As Stubs And Skeletons That Serve As An Interface Between An Application And Rest Of The RMI System. This Layer's Purpose Is To Transfer Data To The Remote Reference Layer Via Marshalling And Unmarshalling. Marshalling Refers To The Process Of Converting The Data Or Object Being Transferred Into A Byte Stream And

Unmarshalling Is The Reverse – Converting The Stream Into An Object Or Data. This Conversion Is Achieved Via Object Serialization.

The Stub/ Skeleton Layer Of The RMI Lies Just Below The Actual Application And Is Based On The Proxy Design Pattern. In The RMI Use Of The Proxy Pattern, The Stub Class Plays The Role Of The Proxy For The Remote Service Implementation. The Skeleton Is A Helper Class That Is Generated By RMI To Help The Object Communicate With TheStub; It Reads The Parameters For The Method Call From The Link, Makes The Call To The Remote ServiceImplementation Object, Accepts The Return Value And Then Writes The Return Value Back To The Stub.In Short, The Proxy Pattern Forces Method Calls To Occur Through A Proxy That Acts As A Surrogate, Delegating AllCalls To The Actual Object In A Manner Transparent To The Original Caller.

**Stub**

The Stub Is A Client-Side Object That Represents (Or Acts As A Proxy For) The Remote Object. The Stub Has The SameInterface, Or List Of Methods, As The Remote Object. However When The Client Calls A Stub Method, The StubForwards The Request Via The RMI Infrastructure To The Remote Object (Via The Skeleton), Which Actually ExecutesIt.

**Sequence Of Events Performed By The Stub:**

- Step 1. Initiates A Connection With The Remote VM Containing The Remote Object.
- Step2. Marshals (Writes And Transmits) The Parameters To The Remote.
- Step3 VM Waits For The Result Of The Method Invocation.
- Step4 Unmarshals (Reads) The Return Value Or Exception Returned.
- Step5 Return The Value To The Caller.

In The Remote VM, Each Remote Object May Have A Corresponding Skeleton[16].

**Skeleton**

On The Server Side, The Skeleton Object Takes Care Of All The Details Of “Remoteness” So That The Actual Remote Object Does Not Need To Worry About Them. In Other Words We Can Pretty Much Code A Remote Object The Same Way As If It Were Local; The Skeleton Insulates The Remote Object From The RMI Infrastructure.

**Sequence Of Events Performed By The Skeleton**

- Unmarshals (Reads) The Parameters For The Remote Method (Remember That These Were Marshaled By The Stub On The Client Side)
- Invokes The Method On The Actual Remote Object Implementation.
- Marshals (Writes And Transmits) The Result (Return Value Or Exception) To The Caller (Which Is Then Unmarshalled By The Stub)

The Diagram Shows The RMI Architecture (Fig 3 & 4)

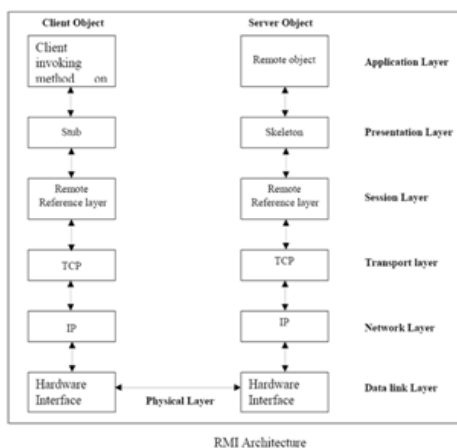


Fig.3 : RMI Architecture

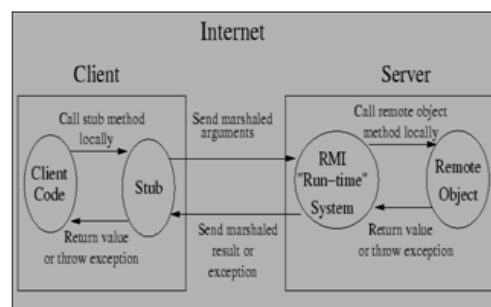


Fig 4: RMI Architecture

**B. The Remote Reference Layer :** The Remote Reference Layer Defines And Supports The Invocation Semantics Of The RMI Connection. This Layer Maintains The Session During The Method Call.



### **C. The Transport Layer**

The Transport Layer Makes The Stream-Based Network Connections Over TCP/IP Between The Jvms, And Responsible For Setting And Managing Those Connections. Even If Two Jvms Are Running On The Same Physical Computer, They Connect Through Their Host Computers TCP/IP Network Protocol Stack. RMI Uses A Protocol Called JRMP (Java Remote Method Protocol) On Top Of TCP/IP (An Analogy Is HTTP Over TCP/IP).

#### **Single Program Multiple Data(SPMD) Algorithm**

We Are Implementing Remote Method Invocation From JAVA Language As Platform To Apply Parallel Processing Concept Single Program Stream Multiple Data Stream(SIMD) In Distributed Network; Here We Are Using Client /Server Architecture. Server Is The Class Where The Distribution Process Occurs. We Are Having A Set Of Randomly Generated Numbers. Here As We Have Single Client We Retrieve Three Numbers From Client And Give Them To Server For Factorial Calculation And Summation. Client Has Job Of Distribution Of Numbers. There Can Be Many Servers And They Can Have Different Methods, Which Can Be Applied Concurrently, Result Will Be Returned To Client For Further Operations.

As Many Server Are Present In This Application. So, We Have To Implement Thread To Bring Access Of Server To One Server At One Time. This Will Not Cause Corruption Of Data And Thus The Work Produce Satisfactorily Results.

RMI Is A Simple Method Used For Developing And Deploying Distributed Object Application In A Java Environment. Creating Distributed Object Application Using RMI Is A Simple As Writing A Stand-Alone Java Application.

RMI Enables A Programmer To Create Distributed Java Application, In Which The Methods Of Remote Java Object Can Be Called From Other Java Virtual Machines Running Either On The Same Host Or On Different Hosts Scattered Across A Network.

A Call To Remote Object Using RMI Is Identical To A Call Made To A Local Object With The Following Exceptions:

1. An Object Passed As A Parameter To A Remote Method Or Returned From The Method Must Be Serialization Or Be Another Remote Object.
2. An Object Passed As A Parameter To A Remote Method Or Returned From The Method Called Is Passed By Value And Not By Reference.
3. A Client Always Refers To A Remote Object Through One Of The Remote Interface Those Implements. A Remote Object Can Be Typecast To Any Of The Interfaces That A Client Implements.

When A Client Application Makes A Remote Call, The Call Passes To The Stub And Then On To The Remote Reference Layer, If Then Passes It Via The Network Layer From The Client To The Server, Where The Remote Reference Layer, On The Sever Side, Unpacks The Arguments And Passes Them To The Skeleton And Then To The Server. Class File. The Return Value Of The Method Call Then Takes The Reverse Trip Back To The Client Side.

When A Client Makes A Call To A Remote Method, That Client Receives A Reference To The Remote Object, Which Implements The Remote Method. All Interactions By The Client Are Performed With The Stub Is Responsible For Data Transfer Between The Local System And The Remote System.

Stub Object On The Client Does Not Interact Directly With The Remote Object On The Server. There Exists A Sever Side Proxy Object Called The Skeleton, Which Is Responsible For Transferring Data Between A Stub And The Actual Object Being Reference On The Server.

In Any Distributed Application, For The Client Side Of The Application To Make The Call To Remote Object, That Client Object Would First Be Able To Locate The Remote Object RMI Provide The Registry Services N Or The Name Services To Make This Possible.

We Register Any Remote Object That It Is Exporting With A Name Server Called A Registry. We Can Maintain A Registry Server That Is Running On A Well-Known Pre Defined Port Number. An Application Can Register With The Registry If It Is On Same Physical Machine.

#### **Steps For Creating RMI Applications: -**

- Define An Interface Of The Remote Classes.
- Implement The Interface In Server-Side Application.
- Bind Objects To Registry Service.
- Create Stubs And Skeleton Classes.
- Create And Compile Client Program To Access The Remote Objects.
- Install Files On Client And Server Machines.
- Start The RMI Registry

**Steps Involved In Running The RMI Application:**

In case The Server Application And Client Application Is Run In The Same Machine: -

- Run The RMI Registry At Specified Port, If Not Specified, It Runs At The Default Port 1099.
- Run The Server Application In Another DOS Window.
- Run The Client Application From The Same Machine.

In case The Server Application And Client Application Is Run On The Separate Machine: -

- Run The RMI Registry At Specified Port, If Not Specified, It Runs At The Default Port 1099.
- Run The Server Application In Another DOS Window.
- Run The Client Application From A Separate Machine.

Following These Steps RMI Application Can Be Implemented.

**Algorithm for Developing and Running the RMI Application for Distributed System.**

Step 1: Enter And Compile The Source Code

Enter The Source Code For Addserverintf.java, Addserverimpl.java, Addserver.java, Addclient.java Then Compile All Above Java Files.

Step 2: Generate Stubs And Skeletons

Compile The Remote Method Invocation (Rmic) From Addserverimpl.java File. The Rmic Addserverimpl Generates Two New Files: Addserverimpl\_Skel.class (Skeleton) And Addserverimpl\_Stub.class (Stub). When Using Rmic, Be Sure That CLASSPATH Is Set To Include The Current Directory.

Step 3: Install Files On The Client And Server Machines.

Copy Addclient.class, Addserverimpl\_Stub.class, And Addserverintf.class To A Directory On The Client Machine. Copy Addserverintf.class, Addserverimpl.class, Addserverimpl\_Skel.class, Addserverimpl\_Stub.class And Addserver.class To A Directory On The Server Machines.

Step 4: Start The RMI Registry On The Server Machine. Start Rmiregistry

Step 5: Start The Server

Java Addserver

Step 6: Start The Client

For Calculating Serially (Run At Each And Individual Machine). The Addclient Software Requires Four Arguments: The Name Or IP Address Of The Server Machine And The Three Numbers That Are To Be Summed Together Of First Two Number And Factorial Of Third Number. You May Invoke It From The Command Line By Using One Of The Two Formats Shown Here. (Ex: Java Addclient 172.16.16.14 458 475 5 Or Java Addclient Server1 485 475 5) For Calculating Parallel (Run All At Same Time) The Addclient Software Requires Arguments: Three Numbers That Are To Be Summed Together Of First Two Number And Factorial Of Third Number. In This Process We Never Use The IP Address Because We Already Use All IP Address In Addclient.java. You May Invoke It From The Command Line By Using One Of The Two Formats Shown Here Ex: Java Addclient 458 475 5).

**II. Results and Conclusion**

We Succeeded In Implementing Remote Method Invocation From JAVA Language As A Platform To Apply Single Program Stream Multiple Data Stream (SPMD) On Clusters Of Terminal's (COT's). Here We Are Using Client/Server Architecture. Server Is The Class Where The Distribution Process Occurs. We Are Having A Set Of Randomly Generated Numbers. Here As We Have Single Client We Retrieve Nth Tasks And Give Them To Nth Server For Various Complex Calculations. Client Has Job Of Distribution Of Numbers. There Can Be Many Nodes As A Servers And They Can Have Different Methods, Which Can Be Applied Concurrently, Result Will Be Returned To Client For Further Operations.

To Estimate The Performance Of The Distributed System The Time For The Computation Of The Task Solved By Different Servers Has To Be Measured In The Sequential And Parallel Case. The Result Was Examined Only Within The Area 1 To 20 Terminal's (See Table 1)

**Table I: Serial And Parallel Time In Seconds On Number Of Terminal**

No Of Terminals	Serial Time(Sec.)	Parallel Time(Sec.)
2	3.856	2.382
4	5.785	2.437
6	7.754	2.492
8	9.723	2.579
10	11.842	2.592
12	13.142	2.719
14	15.752	2.827
16	17.625	2.843
18	19.679	2.906

Following Charts Are Representing The Graphically Performance Of Serial And Parallel Distribution And Compare Between Both Process (See Fig 3,4,5 And 6)

### III. Conclusion

An Advantage Of Using Parallel Processing Instead Of Serial Processing Is Low Cost, High Efficiency Resulting From Use Of Multiprocessing Technique. Using Parallel Processing With Distributed Network Provides Additional Advantage Of Flexibility And Speed Up In Complex Calculations. Using RMI We Can Further Enhance The Application By Performing File Transfer Remotely. We Can Also Use The Output Given By The Server To Client For Further Calculations

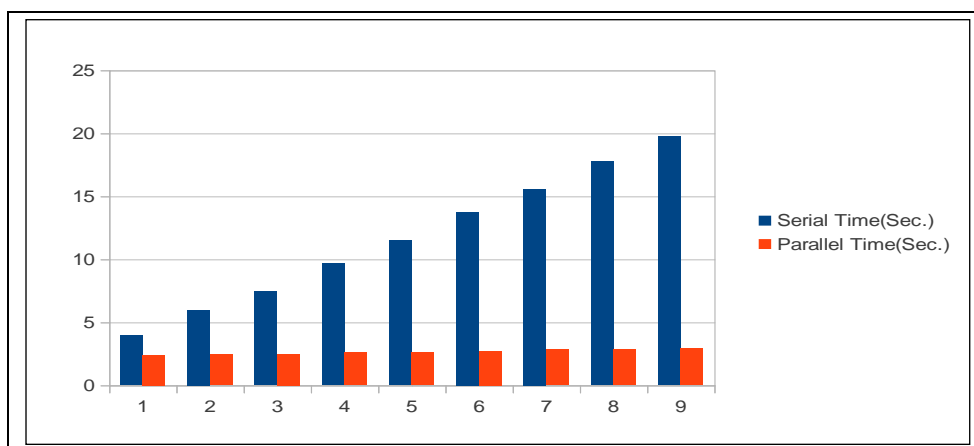


Fig 5: - Comparison Chart Of Parallel And Serial Time Using MIMD

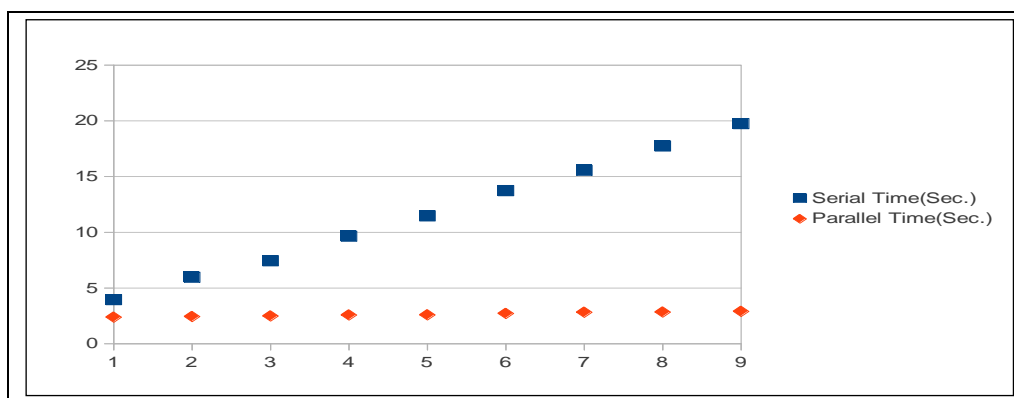


Fig 6: - Comparison Chart Of Parallel And Serial Time Using MIMD

### IV. Acknowledgement

My Express Thanks And Gratitude To All The Departments' Personals And Sponsors Who Give Me A Opportunity To Present And Express My Paper On This Level. I Wish To Place On My Record My Deep Sense Of Gratitude To All Reference Papers Authors For Them Valuable Help Through Their Papers, Books, Websites Etc.

#### Author's Profile

**Sudhir Kumar Meesala** Has Received His Master Of Technology Degree In Computer Technology From National Institute Of Technology, Raipur(CG) The Year 2007. At Present He Is Pursuing Ph.D.. With The Specialization Of Computer Science And Engineering College. His Area Of Interest Parallel Processing, Distributed Technology, Compiler Design, Image Processing, Operating System, Network Programming And Structured Computer Engineering Etc.

**Dr. Pabitra Mohan Khilar** Has Received His Ph.D. In Computer Science And Engineering From IIT Kharagpur(WB) India In The Year 2009. At Present He Is Working As An Associate Professor At Department Of Computer Science And Engineering, National Institute Of Technology, Rourkela(Orissa). His Areas Of Interests Are Parallel And Distributed Computing, Cloud And Grid Computing, Distributed Wireless Network, Distributed Embedded Network Etc.

**Dr. A. K. Shrivastava** Has Received His Ph.D. In Physics(Microwave Propagation) From B. R. Ambedkar Bihar University, Muzafferpur(Bihar) In The Year 2003. At Present He Is Working As An Professor And Head At Dr. C. V. Raman University, Kota, Bilaspur(CG) In The Department Of Physics. His Areas Of Interests Are Power Electronics, Electrical Drives, Power Systems, Renewable Energy Sources And Custom Power Devices, .

### References

- [1] Algorithms And Theory Of Computation Handbook, CRC Press LLC, 1999, "Single Program Multiple Data", In Dictionary Of Algorithms And Data Structures [Online], Vreda Pieterse And Paul E. Black, Eds. 17 December 2004. (Accessed TODAY) Available From: [Http://Www.Nist.Gov/Dads/HTML/Singleprogrm.Html](http://www.nist.gov/dads/html/singleprogrm.html)
- [2] M. Auguin, F. Larbey, OPSILA : An Advanced SIMD For Numerical Analysis And Signal Processing, In Microcomputers : Developments In Industry, Business, And Education / Ninth EUROMICRO Symposium On Microprocessing And Microprogramming, Pp 311-318 Madrid, September 13–16, 1983
- [3] F. Darema, SPMD Model: Past, Present And Future, Recent Advances In Parallel Virtual Machine And Message Passing Interface: 8th European PVM/MPI Users' Group Meeting, Santorini/Thera, Greece, September 23–26, 2001. [Lecture Notes In Computer Science](#) 2131, P. 1, 2001.
- [4] Batcher, K. E., "Sorting Networks And Their Applications", Proc~ AFIPS 1968 SJCC, Vol. 32, Montvale, NJ: AFIPS Press, Pp. 307-314.
- [5] Dina Bitton , David J. Dewitt , David K. Hsaio , Jaishankar Menon,"A Taxonomy Of Parallel Sorting, ACM Computing Surveys (CSUR)", V.16 N.3, P.287-318, Sept. 1984 [Doi>10.1145/2514.2516]
- [6] Dongarr; J. J.; And Eisenstat; "Squeezing The Most Out Of Algorithms In Cray Fortran", Argone National Laboratory, May 1983.
- [7] Dongarra, J. J.; And Hiromoo, Robert E.; "A Collection Of Parallel Linear Equation Routines For The Denelcor HEP", Parallel Computing, Vol. 1, No. 2, December 1984.
- [8] "A Resource Estimation And Call Admission Algorithm For Wirelessmultimedia Networks Using The Shadow ... - All 7 Versions "»DA Levine, IF Akyildiz, M Naghshineh - Networking, IEEE/ACM Transactions On, 1997- Ieeexplore.Ieee.Org.
- [9] Spinodal-Type Dynamics In Fractal Aggregation Of Colloidal Clusters- All 4 Versions »M Carpineti, M Giglio - Physical Review Letters, 1992.
- [10] Implementing Global Memory Management In A Workstation Cluster - All 7 Versions »MJ Feeley, WE Morgan, EP Pighin, AR Karlin, HM ... - ACM SIGOPS Operating Systems Review, 1995 - Cs.Ubc.Ca
- [11] Supporting Parallel Applications On Clusters Of Workstations: The Virtual Communication Machine- ...-All 6 Versions »MC Rosu, K Schwan, R Fujimoto - Cluster Computing, 1998
- [12] Supporting Parallel Applications On Clusters Of Workstations: Theintelligent Network Interface ... - All 3 Versions » M Rosu, K Schwan, R Fujimoto - High Performance Distributed Computing, 1997. Proceedings. ..., 1997 - Ieeexplore.Ieee.Org
- [13] [PS] Iterative Solution Of General Sparse Linear Systems On Clusters Of Workstations - All 16 Versions »GC Lo, Y Saad - Report Umsi-96-117, Minnesota Supercomputer Institute, ..., 1996 - Cs.Umn.Edu
- [14] Javaparty– Transparent Remote Objects In Java - All 21 Versions » M Philippsen, M Zenger - Concurrency Practice And Experience, 1997 - Doi.Wiley.Com
- [15] Gosling, J.; Joy, B.; Steele, G.; Bracha, G.: "The Java Language Specification", Second Edition. Addison-Wesley Publishing Company,1999.
- [16] Lindholm, T.; Yellin, F.: "The Java Virtual Machine Specification", Second Edition. Addison-Wesley Publishing Company, 1999.
- [17] Downing, T.: "Java RMI: Remote Method Invocation". IDG Books Worldwide, 1998.
- [18] Herbert Schildt : "Java 2: The Complete Reference" Fifth Edition 2002 . Tata Mcgraw-Hill Publishing Company Limited New Delhi.
- [19] Lea, D.: "Concurrent Programming In Java-Design Principles And Patterns". Addison-Wesley Publishing Company, 1998.
- [20] Liang, S.: The Java Native Interface: Programmer's Guide And Specification. Addison-Wesley Publishing Company, 1999.
- [21] "A Survey Of Parallel Computer Architectures", Duncan, Ralph, IEEE Computer, Feb 1990, Pp 5-16
- [22] [Http://Www.Geekmantra.Com/Subsection.Php?Section= MI&Subsection = RMI+Architecture.](http://www.geekmantra.com/subsection.php?section=MI&subsection=RMI+Architecture)