Parallelization of Annotated Java code in a Distributed Network

Pratibha S. Yalagi
Assistant Professor
Department of Information Technology
Walchand Institute of Technology
Solapur, India.

Sulabha S. Apte
Professor & Head
Department of Computer Science & Engg.
Walchand Institute of Technology
Solapur, India.

ABSTRACT
Parallelism has been employed for many years, mainly in high-performance computing. The work focuses towards a new parallel execution technique in a distributed network in which the java code is parallelized and independent code is executed on different system in accordance with the availability of the system resource in a distributed network. It speeds up the execution of a particular application to a great extent. Dependencies among the code are detected. The proposed system can be used for parallel computation of the java program, which can be used in industry for executing large java codes. For execution of large java codes, time required will be large. The proposed system can harness the power of nearby all Java enabled machines. The aim is to turn a normal computer into a Super-Computer without extra hardware or space. It is designed for Parallel Computing using both wired & wireless network connections for achieving good speed of execution with the help of distributed network. The annotations are used in the code as indicators for parallel execution. Based on the annotations provided in the code it is parallelized by rebuilding the code for further execution in the network.

Keywords
Parallelism in Java, Annotations, High performance computing, load balancing, distributed networks.

1. INTRODUCTION
Increasing the speed of the program execution is always the research area in the era of high performance computing. Scientific computing, interactive, virtual reality & data mining applications are totally based on increasing performance for the development of advanced computing approaches. Parallelism is used in the execution of the program for providing better performance and to increase the computing speed. Many calculations are carried out simultaneously in parallel computing which is operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently.[12] The use of distributed networks and resources can be used to achieve parallelism in great extent. Java is the language which provides many features to facilitate parallelism and it is suitable for distributed computing [5, 6]. It is suitable for distributed applications in which many components are running on different computers cooperatively using distributed networks. It provides APIs for socket communications[7], RMI[8], threads & synchronization for distributed application development. Annotations can be used in the program to provide the functionality of parallel execution through the program. One of the approach called Parallel Abstraction Layer (PAL) which is working as a bridge between a programming model and the parallel computer architectures, such as clusters of computer resources. [9] The work focuses on the technique of providing annotations in the program for parallel execution in a distributed environment. Use of the annotations in a program for parallelizing the execution of a sequential code is already done by many methods. In the proposed system annotations are used for Java program execution. A server reads the program & rebuilds the annotated program so that it can execute in parallel on different client machines in a network. It can be used for parallel computation of the java program for executing large java codes in the application domain of scientific computation, interactive environments, virtual reality or data mining. For execution of large java codes, time required will be large.

2. LITERATURE REVIEW
Sequential applications are easier to develop than developing parallel applications. Java supports parallelism through many APIs supporting socket programming, RMI, threads & synchronization for distributed applications [7, 8]. Usually programmers write applications directly interacting with the middleware. Several efforts have been spent to face the problem of synchronization and inter thread communication. One of the technique provided is the use of Parallel Abstraction Layer (PAL) [9]. In this the programmer is responsible to choose which parts of code have to be computed in parallel through the use of annotations. Using the information provided by programmers PAL transforms the program code into a parallel one [1]. Method level parallelism can be achieved by method speculation for data-parallel applications with less compiler & programmer efforts [2]. The zJava project aims at automatic parallelization technology for Java programs that use pointer-based dynamic data structures. It exploits parallelism among methods by creating an asynchronous thread of execution for each method invocation in a program. Methods are analyzed at compile-time to determine the data they access. It executes sequential Java programs, automatically extracting, packaging and synchronizing parallelism among methods[13]. Main method in the program is it starts executing sequentially and for each method invocation, an independent thread is created to asynchronously execute the body of the method for concurrent execution.[3] AdJava [4] uses the underutilized computers in a distributed network by automatically distributing the user application using load balancing & migration of objects[11]. This system designs parallel distributed objects using software agents for handling collaboration of objects for interaction among hosts. The ADAJ (Adaptive Distributed Application in Java) project provides a platform for irregular applications. The object location is achieved by an intelligent adaptive redistribution strategy exploiting dynamically information about the platform states. ADAJ is designed as a programming and execution environment for distributed and parallel object oriented applications [6].

3. SYSTEM DESIGN

The proposed system parallelizes the java code in a distributed environment using the annotations in a program provided by the programmer. The programmer needs to specify the dependencies between the codes. This can be done using the concept of annotation. Here, the developer or the programmer can annotate the method that is independent of the other. And on the basis of these annotations, the system rebuilds the program and adds remote methods to the original program. Along with this, it preserves the original copy of the program. In the proposed system one of the computer resource in a distributed network acts as a server & others as client by finding the CPU usage of the clients. Only, if the CPU usage of clients is less than threshold then and then it sends the independent code at client side for execution. Thus, the java code is executed at client side & the result will be send back to server. The time required for the execution of the Java program in the proposed system is less than that on an individual computer system. The wired and wireless computer resources (if required) can be used for parallelization. Basically the work focuses on the deep study of multi-threading in Java, program compilation, dependencies in the program, inter process communication, synchronization, and socket programming.

3.1 System Architecture

The system consists of the following modules as shown in Fig. 1.

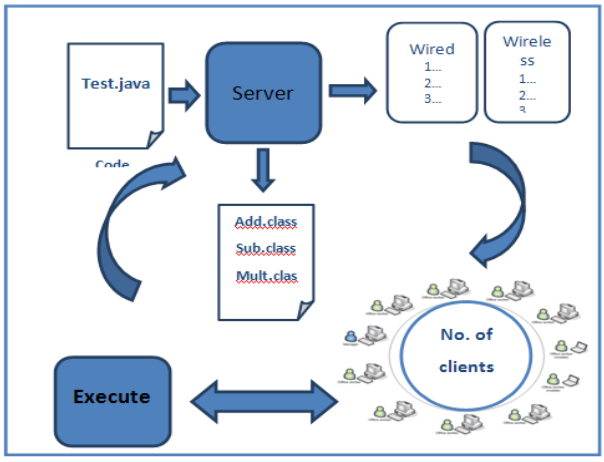


Fig. 1. System architecture.

Here the source program which is written in Java is first compiled at server side for recording the methods and checking the dependency. At the same time, it calculates load on the server and as per the need it divides the code/program into modules. Concurrently it checks for devices in the network having JVM's and lists them as wired and wireless. Then these divided small modules are sent to client for execution. And if the load on client machine reaches threshold then another client is selected for execution and the process continues. After execution, the individual results are sent back to server wherein server records all the readings. These readings are sent back to server.

3.2 Server Architecture

Here the server searches for the devices within the network and waits for the device response. As shown in Fig.2. Depending on the response, it prioritizes the devices as:

1: Wired

2: Wireless

Also it checks for the load at server side.

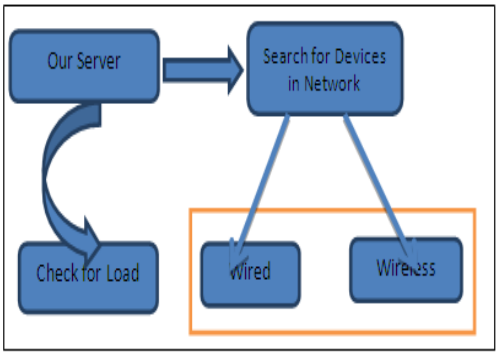


Fig. 2. Server architecture.

4. IMPLEMENTATION

4.1 Algorithm for Server Side Implementation

1. Read the annotated java program
2. Check the program; if the program is small execute it on its host computer.
3. If it is large then the annotations are considered for parallelism.
4. Check for dependencies between the various parts of program.
5. Resolve the dependencies or connect few smaller code parts.
6. Check the number of LAN (wired) connections & number of Wi-Fi (non-wired) connections.
7. Decide whether or not to use the wireless connections. If the number of LAN connections is equal or more than number of pieces of programs then we do not use the wireless connections.

4.2 Sender & Receiver Phase

1. Sender sends the small data to all machines that are under-utilized using sockets.
2. The need for high speed LAN networks arises now as we have to send data very quickly.
3. The receiver receives the data & sends it to its processer for computing.

4.3 Synchronization& Error Handling Phase

1. The synchronizer handles all the messages that are passed between the main server & all its clients.
2. It keeps a record of all messages (codes) & is the deciding agent who determines which result is of which part of the program.
3. The error handler handles all the errors that may arise viz. PC Shutdown, Heavy Load, etc.

4.4 Working Model

The system has mainly the components called ObjectManager and ObjectRegistry as shown in Fig. 3.

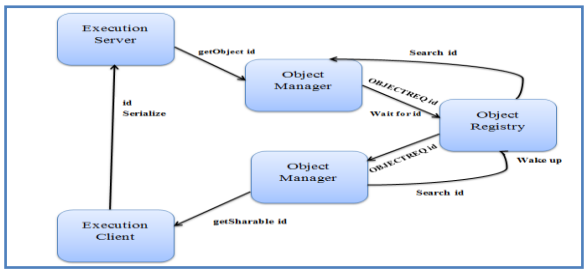


Fig. 3. Working Model.

4.4.1 Object Registry:

It stores the table of containing object ID and ID holder. It listens the thread which listens all the requests regarding objects from Object Manager and then pass the control to service thread where it registers all the commands and check them against the various commands listed in command.java file and accordingly provides services to them.

4.4.2 Object Manager:

The working of Object Manager starts with establishing connection with object Registry and it then starts Receiver Thread(). Here it reads the command and check it against the commands of command.java and accordingly provide service, where the different functionality for Read-only objects as well as Sharable Objects are provided. For Read Only Objects, the system does not wait for accessing objects if it is already in use i.e. Read-only objects can be accessed simultaneously by all the clients e.g. account no.: objects of this kind can be accessed simultaneously as account no. of a person is always the same. For sharable objects, the system provides lock and unlock functionality i.e. when an object is requested and if it is available in Object Registry then access is permitted to the client to hold that object and then it locks the object and after accessing that object it unlocks it so that other clients that are waiting for the object can get a chance to hold it. The need of lock and unlock mechanism of operating system is to ensure data consistency so that updated data exists in Object Registry.

5. RESULTS AND ANALYSIS

The sample input program Adder.java with the annotations is shown in Fig. 4. The Adder.java.org: A backup file of original program is shown in Fig. 5. The annotated program is converted as a parallelized code which is shown in Fig. 6. This can be executed in parallel in a distributed network. The Table I shows the results generated for time in centi seconds required for the execution and the number of processes.

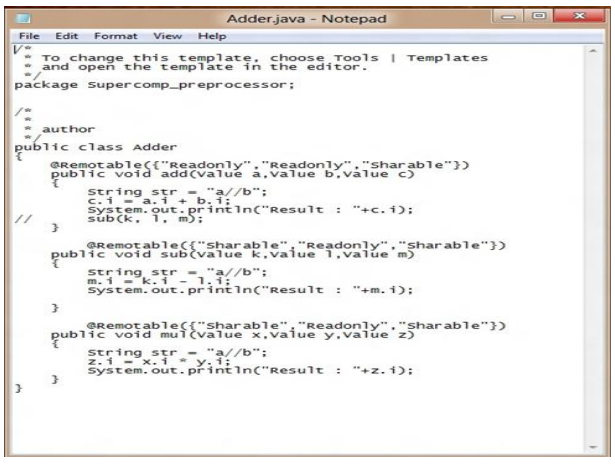


Fig. 4. Sample Input Program.

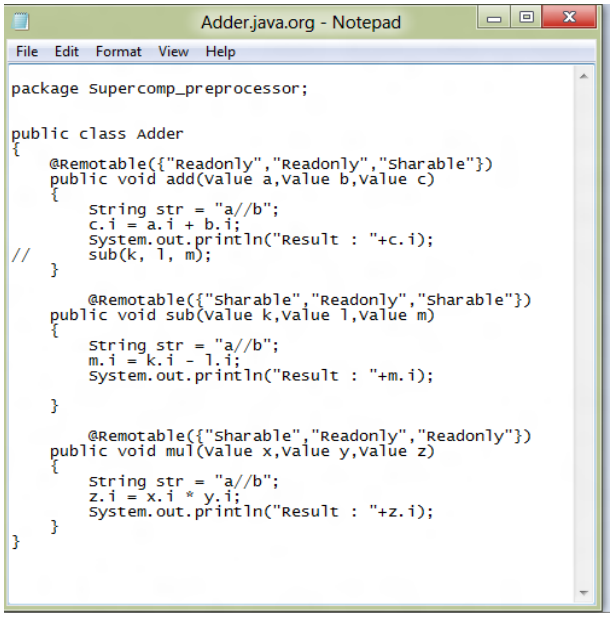


Fig. 5. Back up program

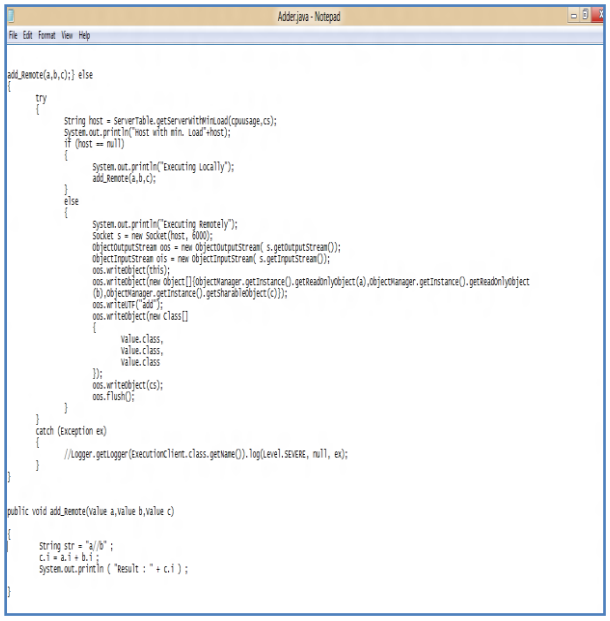
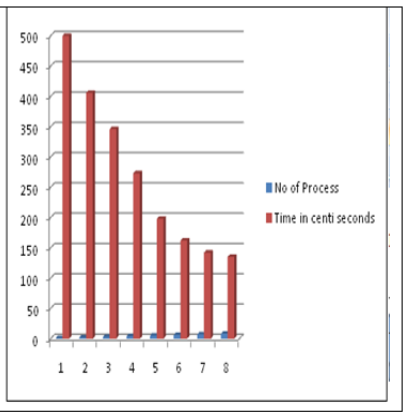


Fig 6 Remote method generation in Program using annotations

Table I & graph comparing time v/s number of processes

No of Process	Time in centi seconds
1	500
2	406
3	346
4	273
5	198
6	162
7	142



6. CONCLUSION

In the proposed system the Java code is structured using annotations. The annotations are provided by the programmer for achieving parallelism in the execution using the distributed networks. The wired & wireless system resources are recognized and they can be used for the execution of large codes in a distributed network. The code is rebuilt using the annotations to execute them in parallel. The advantages of this system are:

Faster processing: Multiple processors work together for executing a program.

Easy to use: The system can be used by any programmer for the development of Java applications

Load Balancing: The network load is balanced by the distribution of load from highly loaded system to less loaded systems.

7. FUTURE SCOPE

It has been observed that parallelization and the high performance computing is the highly studied area. Use of annotations in a program is a programmer's task which can be modeled by the direct use of annotation. The system can be enhanced by the automatic parallel code generation in which the programmer need not to write a parallel program with the annotations rather they can automatically generate through the dependency finders/

8. REFERENCES

[1] Patrizio Dazzi, "Let's Annotate to Let Our Code Run in Parallel," arXiv:1306.2267v1 [cs.PL] 6 Jun 2013.

[2] Michael K. Chen, Kunle Olukotun, "Exploiting Method-Level Parallelism in Single-Threaded Java Programs," IEEE. Proceedings of PACT'98, 12-18 October 1998 in Paris, France.

[3] Bryan chan and tarek s. Abdelrahman. (2004) "Run-Time Support for the Automatic Parallelization of Java Programs," Journal of Supercomputing, 28, 91–117, Kluwer Academic Publishers.

[4] Mohammad M. Fuad and Michael J. Oudshoorn, "AdJava – Automatic Distribution of Java Applications," Twenty-Fifth Australasian Computer Science Conference (ACSC2002), Melbourne, Australia.Conferences in Research and Practice in Information Technology,2001, Vol. 4.

[5] Mark A. Baker, Matthew Grove and Aamir Shafi. (2006) Parallel and Distributed Computing with Java. Proceedings of The Fifth International Symposium on Parallel and Distributed Computing (ISPDC'06) 0-7695-2638-1/06 IEEE.

[6] Alan Kaminsky. (2007) Parallel Java: A Unified API for Shared Memory and Cluster Parallel Programming in 100% Java. 1-4244-0910-1/07/, IEEE.

[7] Sun products - jdk1.2. api & language documentation. http://java.sun.com/products/jdk/1.2/docs/api/overviewsummary.html.

[8] Sun products - jdk1.2. remote method invocation. http://java.sun.com/products/jdk/1.2/docs/guide/rmi/index.html.

[9] M. Danelutto, P. Dazzi, D. Laforenza, M. Pasin, L. Presti, and M. Vanneschi." PAL: Exploiting Java Annotations for Parallelism," Achievements in European Research on Grid Systems, Springer, (November 2007)

[10] Steve J. Chapin, Jon B. Weissman. Distributed and multiprocessor scheduling. handbook

[11] R. Olejnik, a. Bouchi, b. Toursel,"An Java Object observation policy for load balancing".

[12] Brayan Chan, Tarek S. Abdelrahman, "Run-Time Support for the Automatic Parallelization of Java Programs", The Journal of Supercomputing, Volume 28 , Issue 1 (April 2004), pp 91 - 117

[13] Pratibha S. Yalagi, Sulabha S. Apte, "Exploiting Parallelism For A Java Code With An Efficient Parallelization Technique,"International Journal Of Computer Engineering & Technology (IJCET) Volume 3, Issue 3, October - December (2012), pp. 484-489.