

Experimental Performance Analysis of Ada Programs in Cluster System

Korochkin Alexandr
National Technical
University of Ukraine –
Kiev Polytechnic Institute
Kiev, Ukraine
(044) 4549338

cora@comsys.ntu-kpi.kiev.ua

Salah Imad
King Abdullah II School for
Information Technology
University of Jordan
Amman, Jordan P.O.13919
imad@wanadoo.ja

Korochkin Dmitry
National Technical
University of Ukraine –
Kiev Polytechnic Institute
Kiev, Ukraine
(044) 4549338
cora@comsys.ntu-kpi.kiev.ua

ABSTRACT

In this paper, we describe the use of the Ada language for programming in modern cluster systems. The results of experimental performance analysis of Ada and Java programs in a real cluster systems based on dual SMP nodes are presented.

Categories and Subject Descriptors

B.8.2 Performance Analysis and Design Aids.

C.2.4 Distributed systems.

D.1.3 Concurrent Programming – *Parallel programming, distributed programming*

General Terms: Performance. Experimentation.

Keywords: Ada, Java, SMP, distributed (cluster) systems, remote procedure call, monitor, process, mutual exclusion, process synchronization, client-server model.

1. INTRODUCTION

Cluster systems are a kind of distributed system dedicated to solutions of large computing problems [4]. There are two requirements for advanced cluster systems: an organization which must provide for high speed node communication and effective computing in each node.

The first requirement is implemented through the use of modern network facilities (switches and network adapters) such as Gigabit, Myrinet, SCI. The second requirement is implemented by nodes with power multiprocessors computer systems. (SMP systems are used most often). Such advance cluster systems provide high performance based on two levels of parallel computing: in each node and between nodes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGAda'05, November 13–17, 2005, Atlanta, Georgia, USA.

Copyright 2005 ACM 1-59593-185-6/05/0011...\$5.00.

Cluster programming requires efficient exchanges information for both computing levels. Nodes in cluster system exchange information through their network (a message-based synchronization and communication). In node processors exchange information through shared memory (a shared-variable synchronization and communication).

Traditionally communications libraries PVM and MPI are used for programming in cluster systems [4]. Modern programming languages often have their own effective features for development of parallel and distributed applications [5].

Ada83 was one the first language in which concurrency was an integral part. Ada95 supports both communications within a node and between nodes. Therefore Ada95 seems most suitable for programming advanced cluster systems.

The aim of this work is an analysis of Ada95 features for programming an advanced cluster based on SPM systems and performance analysis of Ada95 distributed programs in real cluster systems.

In particular, we plan to

- analyze Ada tools for programming in SPM systems
- analyze Ada tools for internodes communication in cluster systems
- develop a set of Ada distributed applications
- create a cluster system based on dual SMP systems
- test the Ada distributed application on the real cluster system.

2. PROGRAMMING FOR CLUSTER

In this section we present a structure of real created cluster system and analyze of Ada facilities for programming in this system

2.1 Cluster System

A cluster system (named CS-SMP8) was built on Beowulf technology [4]. It consists of 8 nodes and 8-ports switch (Fig. 1). Dual SMP system with two Xeon III processors was used in each node of system. Fast Ethernet was used in switch and network adapters for node communication.

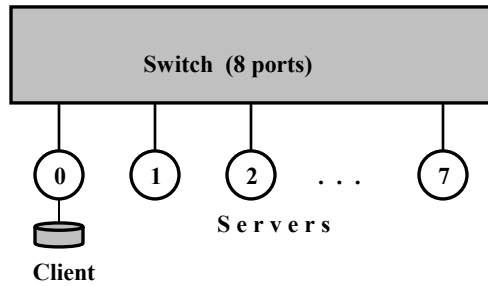


Figure 1. Cluster system CS-SMP8

Programming the CS-SMP8 requires solution for two problems: node communication and calculation within each node of the system.

2.2 Programming in Nodes

As each CS-SMP8 node is a dual SMP systems, effective facilities for process creation and process interaction are required. Ada possesses a good process model based on classic Hoare's concept of communication sequential processes and implemented through the task units and task types with discriminates [1, 3, 5].

As found in all concurrent programming environments, programming SMP systems requires solutions for the problems of mutual exclusion and event synchronization. Mutual exclusion is the exclusive access to shared resources. Synchronization is the satisfaction of constraints on the interleaving of the actions of different processes. Ada's model, based on shared variables, is used for both mutual exclusion and synchronization,

2.2.1 Mutual Exclusion

Ada provides an effective feature for mutual exclusion upon access to common (shared) resources: atomic and volatile variables, semaphores, protected object/type [2, 3].

Atomic operations provide synchronized access to shared resources via atomic (volatile) variables and types specified via the pragmas `Atomic` and `Volatile`.

Semaphore primitives provide an efficient facility for mutual exclusion. Ada95 supports a binary (Boolean) semaphore through the package `Synchronous_Task_Control` (Annex D "Real Time Systems"). This package includes the type `Suspension_Object` for semaphore creation and operations `Suspend_Until_True()` and `Set_True()` for work with semaphores.

Monitors provide higher-level methods for mutual exclusion. Ada95 has an advanced implementation of condition monitors via protected objects/types. A protected object encapsulates a shared variable as a protected element. It provides coordinated (synchronized) access to shared data, through calls protected operations (protected entries and protected subprograms). Protected operations solve the problem of mutual exclusion automatically. Protected functions provide a concurrent read-only access to the protected elements and can be used for fast access to shared data from processes.

2.2.2 Process synchronization

In Ada95 there are different ways to implement condition (event) synchronization: rendezvous, protected entries, semaphores. The

`Suspension_Object` type achieves simple event-based synchronization between tasks. Procedures `Suspend_Until_True()` and `Set_True()` implement the `Wait` and `Signal` operations for process blocking and unblocking before and after event.

Protected objects provide more efficient mechanisms for process synchronization based on monitors. Condition synchronization is supported by barriers in protected entries. Protected entry is guarded by the barrier inside the body of the protected object. If this barrier evaluates to false when the entry call is made, the calling task is suspended until barrier evaluates to true no other tasks are currently active inside the protected object.

2.3 Node communication

Programming for CS-SMP8 requires programming of nodes communication. It is based on client-server model and can be implemented by sockets or remote procedure calls in Ada95.

2.3.1 Sockets

A socket abstraction consists of the data structure holding the information needed for communication, and the systems calls manipulating the socket structure. Once a socket is created, it can be used to wait for an incoming connection (passive socket), or it can be used to initiate connection (active socket).

There are a two ways to use sockets in Ada95:

- use sockets from Win32 via Ada95 compilers bound with Win32 libraries;
- use special Ada95 packages supported the sockets mechanism.

Experimental performance analysis of Ada95 distributed application based on sockets is presented in [6].

2.3.2 Remote Procedure Call

In Ada95 remote procedures calls mechanism (RPC) is based on Distributed Systems Annex (DSA) [1, 5]. DSA defines facilities for supporting the implementation of distributed systems using multiple partitions working cooperatively as a part of single Ada program. Annex E defines a distributed system, distributed program and the process of configuring the partition of the program.

A distributed program is one or more partitions that execute independently. Inter partition communication is implemented by RPC mechanism. Partitions are built on categorized library units. For categorization of library units are used the categorization pragmas: `Shared_Passive`, `Remote_Types`, `Remote_Call_Interface`.

A remote call interface library unit can be used as an interface for remote procedure call between active partitions. A remote procedure (subprogram) call invokes the execution of a procedure in another partition and may be in another node of CS-SMP. DSA defines Partition Communication Systems (PCS) based on package `System.RPC`. PCS provides a facility for supporting communication between the partitions.

3. EXPERIMENTS

We used an example of matrix multiplication to evaluate the use of Ada in programming our CS-SMP8 cluster system.

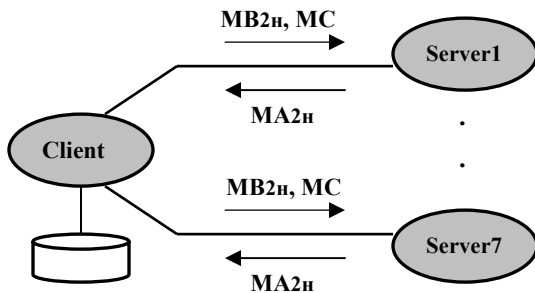
3.1 Matrix Multiplication

Parallel algorithm for matrix multiplication $MA = MB * MC$ can be presented as

$$MA_H = MB_H * MC \quad (3.1)$$

MB_H - H - rows (columns) of MB matrix.

Calculation of expression (3.1) can be assigned to each processor of a server node. Therefore client node sends a part matrix MB_{2H} and complete matrix MC to each server node. (Fig. 2).



MA, MB, MC

Figure 2. Server-Client communication

Algorithms of client and server nodes:

Client

1. Input matrixes MB and MC
2. Send MB_{2H} and MC to each server
3. Receive result MA_{2H} from each server
4. Output result MA

Server

1. Receive MB_{2H} and MC from client
2. Calculate A_{2H}
3. Send A_{2H} to client

Distributed program for matrix multiplication includes two forms of partitions for the server and client implementations. In our implementation of the client algorithm, we create seven tasks named CS . Each of these tasks calls the remote procedure $Culc()$ (Fig. 3).

Server partition includes package Box with remote interface and remote procedure

```
procedure Culc(MB: in Matrix2H, MC: in
                Matrix, MA: out Matrix2H);
```

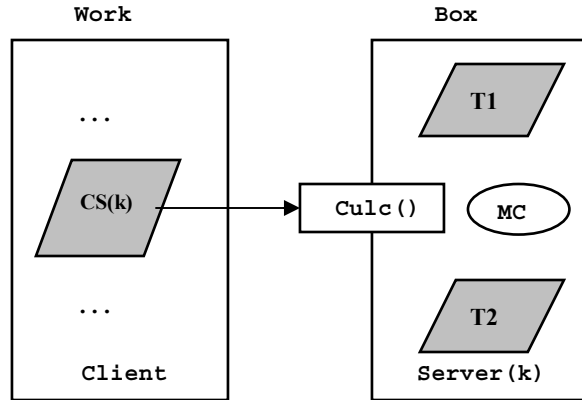


Figure 3. Client-Server realization

Remote procedure $Culc()$ receives the data from client, executes the operation $MA_{2H} = MB_{2H} * MC$ and returns the result MA_{2H} to the client.

The server part of the distributed application for each SMP node is optimized for dual processors. Procedure $Culc()$ creates and runs the two tasks $T1$ and $T2$ that calculate the result $MA_H = MB_H * MC$. Tasks interact via atomic variables, semaphores or protected objects. Tasks communication is related with access to shared variable MC , task synchronization - with a start and finish of tasks.

A complete Ada distributed application requires a configuration file. The partition for the Client includes a procedure $Work$ with seven tasks CS . The partitions for the Servers include package Box with remote procedure $Culc()$.

3.2 Testing

The results of our experimental analysis for Ada95 and Java matrix multiplication programs with sockets and RPC (RMI) for $N=1000$ and $N=1500$ are presented in Tables 1-4 in APPENDIX. A graph of the speed up for the Ada RPC programs is presented in figure 4.

The results of experiments for matrix multiplication:

1. As one would expect, a cluster system with 1 – 7 servers' dual nodes decreases the time for matrix multiplication. The speedup ranges from 1.7 to 8.9 for Ada and Java programs.
2. The best speedup (8.92) was observed for CS-SMP8 with 7 server nodes with an Ada program based on RPC.
3. Both Ada and Java distributed application based on sockets and RPC (RMI) have statistically equal efficiency.
4. Results are better for $N = 1500$ then for $N = 1000$ because the computing time in nodes is more significant than the time for node communication.
5. Speedup begins to slow down for servers number $P > 4$. The reason is that a volume of sent data (MB_{2H} and MC) increases and Fast Ethernet does not cope with such a data flow. It is necessary to use others parallel matrix multiplication algorithms that decrease the size of sent matrix MC with additional

calculations, complete result MA in client node, and to use Gigabit Ethernet.

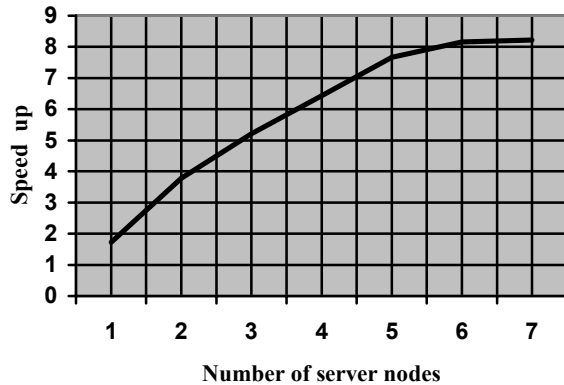


Figure 4. Speed up

4. CONCLUSION

In this work we have examined the feasibility of using Ada95 for programming in cluster systems. We have reviewed the approaches that Ada has taken for supporting concurrency. It provides high-level model based explicit communication and structured approach to mutual exclusion. The extensive collection of Ada tasks communication tools allows us to use them in different combination. We have show that Ada distributed model based on realization of RPC mechanism meets or exceeds the performance of the Java models. Ada surpasses Java in usability. Distributed Ada application can be built via pragmas category easier then Java or MPI distributed application.

Experimental tests in our cluster system have shown that speedups of Ada distributed programs are not worse than speedups for Java, PVM and MPI application. However the actual running time of the Ada application is greater than the equivalent Java programs. We feel this time is a small price to pay for the extra reliability obtained.

In general Ada95 has advanced facilities for parallel and distributed programming and can be used for development an efficient applications for cluster systems with different structure organizations.

5. ACKNOWLEDGMENTS

Our thanks to Professor John W. McCormick from University of Northern Iowa for support Ada in Ukraine a long time.

6. REFERENCES

- [1] Korochkin A. *Ada95: Introduction in Programming*. Swit, Kiev, 1999.
- [2] Korochkin D., Korochkin S. Experimental Performance Analysis of the Ada95 and Java Program on SMP System In *Proceeding of the ACM Annual Conference (SIGADA'02)* (The Houston, Texas, USA, December 8-12, 2002) ACM Press, New York, NY, 2002.
- [3] Korochkin A., Rusanova O. Scheduling Problems for Parallel and Distributed Systems In *Proceeding of the ACM Annual Conference (SIGADA'99)* (The Redondo Beach, CA, USA, October 17-21, 2001) ACM Press, New York, NY, 1999.
- [4] Andres G. *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley, Reading, MA, 2000.
- [5] Taft S., Duff R., Brukardt R., Ploedereder T. *Consolidated Ada Reference Manual. Language and Standard Libraries*, Springer, Berlin, 2001.
- [6] Korochkin A., Rovto V. Experimental Performance Analysis of Ada95 Program in Distributed Systems In *Proceeding of the 1-st Conference (ASCN-2003)* (Lviv, Ukraine, September 10-12, 2003) LPI, Lviv, 2003.

7. APPENDIX

In appendix the results of testing Ada and Java programs for clusters systems are presented.

Table 1. Results for Ada with sockets

Nodes number	Servers number	Processors number	Speed up	
			N=1000	N=1500
2	1	4	1.79	1.84
3	2	6	3.32	3.42
4	3	8	5.23	5.36
5	4	10	5.87	6.14
6	5	12	6.74	7.12
7	6	14	7.43	8.02
8	7	14	7.94	8.24

Table 3. Results for Java with sockets

Nodes number	Servers number	Processors number	Speed up	
			N=1000	N=1500
2	1	4	1.86	1.88
3	2	6	3.48	3.55
4	3	8	5.01	5.16
5	4	10	5.82	5.94
6	5	12	6.27	6.53
7	6	14	7.02	7.68
8	7	16	7.88	8.11

Table 2. Results for Ada with RPC

Nodes number	Servers number	Processors number	Speed up	
			N=1000	N=1500
2	1	4	1.72	1.84
3	2	6	3.78	3.81
4	3	8	5.21	5.34
5	4	10	6.43	6.44
6	5	12	7.67	7.76
7	6	14	8.16	8.85
8	7	16	8.22	8.92

Table 4. Results for Java with RMI

Nodes number	Servers number	Processors number	Speed up	
			N=1000	N=1500
2	1	4	1.76	1.86
3	2	6	3.45	3.85
4	3	8	5.15	5.77
5	4	10	5.92	6.74
6	5	12	7.09	8.14
7	6	14	7.56	8.74
8	7	16	8.03	8.83