Ca’ Foscari University of Venice

Department of Environmental Science, Informatics and Statistics

Master Degree Computer Science

Course of High Performance Computing

(A.A. 2013 - 2014)



dense matrix-matrix multiplication

Versione 1.0 del 6 maggio 2014

Prof. Orlando salvatore

*A cura del gruppo*:

|  |  |  |
| --- | --- | --- |
| Barbon Gianluca | e-mail: | *gbarbon@dsi.unive.it* |
| De Zotti Cristian | e-mail: | *cdezotti@dsi.unive.it* |

Summary

[1. Introduction 3](#_Toc387135910)

[2. Implementation 4](#_Toc387135911)

[2.1 Header files 4](#_Toc387135912)

[2.1.1 Stopwatch.h 4](#_Toc387135913)

[2.1.2 Header.h 4](#_Toc387135914)

[2.1.3 InOut.h 4](#_Toc387135915)

[2.2 Matrix Matrix multiplication 4](#_Toc387135916)

[2.3 Cannon Algorithm 4](#_Toc387135917)

[2.4 Processor Farm 4](#_Toc387135918)

[2.5 Matrix Matrix Fast multiplications 4](#_Toc387135919)

[3. Testing Phase 4](#_Toc387135920)

[3.1 Environment Description 4](#_Toc387135921)

[3.2 Algorithms times 4](#_Toc387135922)

[3.2.1 Matrix Matrix multiplication 4](#_Toc387135923)

[3.2.2 Cannon Algorithm 4](#_Toc387135924)

[3.2.3 Processor Farm 5](#_Toc387135925)

[3.2.4 Matrix Matrix Fast multiplications 5](#_Toc387135926)

[4. Conclusion 5](#_Toc387135927)

[5. Appendix I: MPI primitives 5](#_Toc387135928)

[5.1 MPI\_Send 5](#_Toc387135929)

[5.2 MPI\_Recv 6](#_Toc387135930)

[5.3 MPI\_Sendrecv\_replace 6](#_Toc387135931)

[6. Appendix II: User Manual 7](#_Toc387135932)

[7. Bibliography 8](#_Toc387135933)

# Introduction

dsfsdfsdfsdfsdfsd

# Implementation

## Header files

### Stopwatch.h

### Header.h

### InOut.h

## Matrix Matrix multiplication

## Cannon Algorithm

## Processor Farm

## Matrix Matrix Fast multiplications

# Testing Phase

## Environment Description

## Algorithms times

### Matrix Matrix multiplication

### Cannon Algorithm

### Processor Farm

### Matrix Matrix Fast multiplications

# Conclusion

# Appendix I: MPI primitives

The following is a brief overview of the MPI functions used in the implementation phase, to allow communication between the various nodes involved in the parallel computation. The overview consists of the signature of the function and description of the parameters in input and output, to make it easier to understand the implementation of the algorithms.

## MPI\_Send

int MPI\_Send(const void \*buf, int count, MPI\_Datatype datatype, int dest,

   int tag, MPI\_Comm comm)

MPI\_Send performs a standard-mode, blocking send. This routine will block until the message is sent to the destination.

Input Parameters

**buf:** Initial address of send buffer (choice).

**count:** Number of elements send (nonnegative integer).

**datatype:** Datatype of each send buffer element (handle).

**dest:** Rank of destination (integer).

**tag:** Message tag (integer).

**comm:** Communicator (handle).

## MPI\_Recv

int MPI\_Recv(void \*buf, int count, MPI\_Datatype datatype,

   int source, int tag, MPI\_Comm comm, MPI\_Status \*status)

MPI\_Recv performs a standard-mode blocking receive. This basic receive operation, MPI\_Recv, is blocking: it returns only after the receive buffer contains the newly received message. A receive can complete before the matching send has completed (of course, it can complete only after the matching send has started).

The receive buffer contains a number (defined by the value of count) of consecutive elements. The first element in the set of elements is located at address\_buf. The type of each of these elements is specified by datatype. The length of the received message must be less than or equal to the length of the receive buffer.

Input Parameters

**count:** Maximum number of elements to receive (integer).

**datatype:** Datatype of each receive buffer entry (handle).

**source:** Rank of source (integer).

**tag:** Message tag (integer).

**comm:** Communicator (handle).

Output Parameters

**buf:** Initial address of receive buffer (choice).

**status:** Status object (status).

## MPI\_Sendrecv\_replace

int MPI\_Sendrecv\_replace(void \*buf, int count, MPI\_Datatype datatype,

   int dest, int sendtag, int source, int recvtag, MPI\_Comm comm,

   MPI\_Status \*status)

MPI\_Sendrecv\_replace sends and receives a message using a single buffer. The send-receive operations combine in one call the sending of a message to one destination and the receiving of another message, from another process. The two (source and destination) are possibly the same. A send-receive operation is useful for executing a shift operation across a chain of processes. If blocking sends and receives are used for such a shift, then one needs to order the sends and receives correctly (for example, even processes send, then receive; odd processes receive first, then send) in order to prevent cyclic dependencies that may lead to deadlock. When a send-receive operation is used, the communication subsystem takes care of these issues.

A message sent by a send-receive operation can be received by a regular receive operation or probed by a probe operation; a send-receive operation can receive a message sent by a regular send operation.

MPI\_Sendrecv\_replace executes a blocking send and receive. The same buffer is used both for the send and for the receive, so that the message sent is replaced by the message received.

The semantics of a send-receive operation is what would be obtained if the caller forked two concurrent threads, one to execute the send, and one to execute the receive, followed by a join of these two threads.

Input/Output Parameter

**buf:** Initial address of send and receive buffer (choice).

Input Parameters

**count:** Number of elements in send and receive buffer (integer).

**datatype:** Type of elements to send and receive (handle).

**dest:** Rank of destination (integer).

**sendtag:** Send message tag (integer).

**source:** Rank of source (integer).

**recvtag:** Receive message tag (integer).

**comm:** Communicator (handle).

Output Parameters

**status:** Status object (status).

# Appendix II: User Manual

# Bibliography

Orlando, S. (2013). HPC Course slides

Open MPI Project (2004-2014). Open MPI v1.4.5 documentation