Message Passing Interface

In Java for AgentTeamwork (MPJ)
By Zhiji Huang
Advisor:
Professor Munehiro Fukuda
2005
AgentTeamwork

- User requests AgentTeamwork for some computing nodes.
- AgentTeamworking manages the resources for performance and fault tolerance automatically.
# AgentTeamwork Layers

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Operating Systems</th>
<th>Java Virtual Machine</th>
<th>User Program Wrapper</th>
<th>mpiJava Socket</th>
<th>mpiJava API</th>
<th>User applications in Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>AgentTeamwork</td>
<td>GridTcp</td>
<td>Java Socket</td>
<td>mpiJavaAteam</td>
<td>mpiJavaSocket</td>
<td>Java Virtual Machine</td>
<td>User applications in Java</td>
</tr>
</tbody>
</table>

- User applications in Java
- mpiJava API
- User Program Wrapper
- mpiJavaSocket
- Java Socket
- mpiJavaAteam
- GridTcp
- AgentTeamwork
- Java Virtual Machine
- Operating Systems
- Hardware
Agent Teamwork
GridTCP

- Extends TCP by adding message saving and check-pointing features.
- Automatically saves messages.
- Provides check-pointing, or snapshots of program execution.
- Ultimately allows programs to recover from errors.
  - Node crashes, etc.
public class MyApplication {
    public GridIpEntry ipEntry[];
    public int funcId;
    public GridTcp tcp;
    public int nprocess;
    public int myRank;

    public int func_0( String args[] ) {
        MPJ.Init( args, ipEntry, tcp );
        ....;
        return 1;
    }

    public int func_1() {
        if ( MPJ.COMM_WORLD.Rank() == 0 )
            MPJ.COMM_WORLD.Send( ... );
        else
            MPJ.COMM_WORLD.Recv( ... );
        ....;
        return 2;
    }

    public int func_2() {
        ....;
        MPJ.finalize( );
        return -2;
    }
}
Message Passing Interface

- API that facilitates communications (or message passing) for distributed programs.
- Usually exists for FORTRAN, C/C++, Java.
- Current implementations in Java are actually Java wrappers around native C code.
  - Disadvantages with portability and is not suitable to concept of AgentTeamwork.

![Diagram of Message Passing Interface](image)
Message Passing Interface – Basic Functions

- Init()
- Send()/Recv()
- Bcast()
- Gather()
MPJ – mpiJavaS & mpiJavaA
Contains main MPI operations.

Call to traditional Init(string[]) initializes Java socket-based connections.

Call to Init(string[], IpTable, GridTcp) initializes connections with GridTCP

Also provides Rank(), Size(), Finalize(), etc.
Communicator

- Provides all communications functions.
- Point to point
  - Blocking – Send(), IRecv()
  - NonBlocking – Isend(), Recv()
- Collective – Gather(), Scatter(), Reduce(), and variants.
JavaComm & GridComm

- **JavaComm:**
  - Java Sockets, SocketServers

- **GridComm:**
  - GridTcp Sockets, GridTcp object, IpTable
  - And others needed by GridTCP.

- **Both:**
  - InputStreamForRank[]
  - OutputStreamForRank[]
  - Allows for socket communications using bytes.
  - Can use same communications algorithms for both GridComm and JavaComm.
  - Clean interface between the two layers.
Creation of Java byte arrays/buffers very expensive. Greatly reduces performance.

One solution: use permanent buffers for serialization
- byte buffer[64k]
- Serialize into buffer until full, write buffer, serialize remaining data.

Not effective with collective communication algorithms.
- Either requires extra byte storage to handle/save serialized data.
- Or requires serialization/deserialization at every read/write.
Raw Bandwidth – no serialization (just bytes).
Serialization – Doubles and other primitives

- Doubles - only 20% of performance.
- Other primitives see 25-80% performance.
- Necessity to “serialize” or turn items into bytes very costly
- In C/C++
  - Cast into byte pointer – 1 instruction.
- In Java
  - ```java
    int x;
    byte[] arr[4];
    arr[3] = (byte) ( x );
    arr[2] = (byte) ( x >>> 8);  //shift, cast, copy
    arr[1] = (byte) (x >>> 16);  //repeat
    arr[0] = (byte) (x >>> 24);
    ```
  - Lots of instructions, extra memory for byte buffer.
  - Cost x2 due to deserialization on other side.
PingPong (send and recv) – Doubles

PingPong of Doubles

Number of Doubles

MBytes/s

mpiJavaS (mpj)
mpiJava
PingPong Objects

PingPong Objects

Number of Objects

Objects/s

mpiJavaS (mpj)
mpiJava
Bcast – 8 processes Doubles

Bcast (np = 8) Doubles

Number of Doubles

MB/s

mpiJavaS (mpj)

mpiJava
Bcast – 8 processes

Objects

Bcast (np = 8) Objects

Number of Objects

Objects/s

mpiJavaS (mpj)

mpiJava
Performance Analysis

- **Raw bandwidth**
  - mpiJavaS comes to about 95-100% of maximum Java performance.
  - mpiJavaA (with checkpointing and error recovery) incurs 20-60% overhead, but still overtakes mpiJava with bigger data segments.

- **Doubles & Objects**
  - When dealing with primitives or objects that need serialization, a 25-50% overhead is incurred.

- **Memory issues related to mpiJavaA** – runs out of memory.
Conclusion

The next step is to develop a tool to automatically parses a user program into GridTcp functions for best performance.

Ultimately, automate user job distribution, management, and error recovery.
A few helpful classes…

- CSS432 Networking
- CSS430 Operating Systems
- CSS360 Software Engineering
- CSS422 Hardware
- CSS343 Data Structures & Algorithms
Questions?