JUSTGrid
A Pure Java HPCC Grid Architecture for Multi-Physics Solvers Using Complex Geometries.

Thorsten Ludewig*, Jochem Hauser*, Torsten Gollnick*, and Hans-Georg Paap†

*Dept. of High Performance Computing
Center of Logistics and Expert Systems (CLE) GmbH
Salzgitter, Germany

†HPC Consultant
Barbing, Germany

presented at

42nd AIAA Aerospace Science Meeting and Exhibit,
Reno, Nevada, USA, 5-8 January 2004
Overview

- Why Java for HPCC?!
- What is JUSTGrid? (Framework)
- JUSTGrid Communication Overview
- Object Oriented Programming (OOP)
- Threads
- Graphical Applications
- Java Performance Results
- Conclusions
- Future Work
Java as the Language for HPCC

- platform independence
- simple and straightforward parallelization
- unique included network capabilities
  - JDBC (Java Database Connectivity)
  - RMI (Remote Method Invocation)
  - Secure Connections over the Inter- and Intranet
- easy generation of object reflecting the engineering design process
- "code reusability" - simplifies code design
Why we like to use Java for writing high-quality portable parallel programs?

- pure object formulation (i.e. an object representation of a wing, fuselage, engine etc. described by a set of classes containing the data structures and methods for a specific item)
- strong typing
- exception model
- elegant threading
- portability
What is JUSTGrid

This is an age of possibility, and IT is the driving force behind this change that occurs on a global range.

High Performance Computing and Communications (HPCC) on a global scale is the key of this new economy.

The need for accurate 3D simulation in numerous areas of computationally intensive industrial applications, including the rapidly evolving field of bioscience, requires the development of ever more powerful HPCC resources for a computational Grid based on the Internet.
What is JUSTGrid

• The Java language has the potential to bring about a revolution in computer simulation. Using Java's unique features, a multi-disciplinary computational Grid, termed JUSTGrid, can be built entirely in Java in a transparent, object-oriented approach.

• JUSTGrid provides the numerical, geometric, parallel, and network infrastructure for a wide range of applications in 3D computer simulation thus substantially alleviating the complex task of software engineering.
Scope of JUSTGrid

JUSTGrid a framework for HPCC in engineering, science, and life sciences

Navier Stokes (fluid dynamics)  Maxwell (electromagnetics)  Schrödinger (quantum mechanics)  Results  Debugging Session Tracking  Surface Conversion

Solver  Visualization

Dynamic Load Balancing  Parallelization

3D Complex Geometries

Interactive Steering  Collaborative Engineering  System Security  Outsourcing

Internet
Scope of JUSTGrid

- A solver only needs to contain the physics and numerics of the simulation task for a single block or a single domain.

- The solver does not need to know anything about the geometry data or the parallelization.

- It has a simple structure

- The solver can be tested independently before its integration

- Replacing the default solver by one's own solver.
Communication Overview

Client

- sends the data AND the *java solver code* to the server and receives a *unique session id* to identify the session on the server.

Server

- provides a framework for multi physics solver

Interactive steering
Communication Overview

- with the *unique session id* everyone can connect to a specific session on the server
  - start / stop session
  - changing the conditions of the computation
  - visualization
  - debugging
  - collaborative work
OOP - Object Oriented Programming

Encapsulation of data in an Object

<table>
<thead>
<tr>
<th>class name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+variable1: integer</td>
</tr>
<tr>
<td>+variable2: double</td>
</tr>
<tr>
<td>+method1</td>
</tr>
<tr>
<td>+method2</td>
</tr>
</tbody>
</table>

Abstract Data Types

- The association between the declaration of a data type and the declaration of the code that is intended to operate upon variables of this type

Data hiding and encapsulation

- Protecting the data of an object from improper modification by forcing the user to access the data through a method.
OOP Example

Programming in 'C'

```c
struct my_date {
    int day, month, year;
} date;

date.day = 32;
```

Programming in 'Java'

```java
public class MyDate {
    private int day, month, year;

    public void setDay( int day ) {
        ... validation code ...
    }
}
```

```java
MyDate myDate = new MyDate();
myDate.setDay( 32 );
```

ERROR!
Threads
an efficient way of parallelizing codes

What are Threads?

- Multithreaded programs extends the idea of multitasking by taking it one level further: individual programs (processes) will appear to do multiple tasks at the same time.

- Programs that can run more than one thread at once are said to be \textit{multithreaded}.

- Each task is usually called \textit{thread} which is the short form for thread of control.
What are threads?

Three Parts of a Thread

- A virtual CPU
- The code the CPU is executing
- The data the code works on
Why Threads are good for CFD

- Threads as a general parallelization strategy for CFD codes
- Sophisticated dynamic load balancing algorithms on shared-memory machines
- Advanced numerical schemes in CFD, i.e. GMRES, do not require the same computational work for each grid cell.
ClientGUI

Why we like to use simple graphical user interface for the JUSTGrid?

Because for a non-Programmer it is too difficult to collect all different parts needed for a JUSTGrid session into a Java source text and compile it for himself.

It is easier to run a quick test case without falling into common programming traps.
### JUSTGrid Simple Client GUI

#### JavaGrid Client GUI

**File**
- Help

**Settings**
- **Server**
- **Classes**
- **Input Data**
- **Output Data**

**Server name**

**Output Log**
- RMI URL = rmi://JpMaster
- Session ID = 4707464153434212577
- establish connection to server and init session 0.953s
- Init multiblock solver handler 0.217s
- sending 3d command file 0.599s
- sending plot3d grid file 24.113s

*** Start ***

*** Ready ***

- computation 35.125s

**Session Information**

**Session Id**

4707464153434212577

**Session Status**

- Start
- Pause
- Stop
- Clear Log

---

Thorsten Ludewig, 42nd AIAA, Aerospace Science Meeting and Exhibit, Reno, Nevada, USA, 5-8 January 2004
GRXTool

- Convert grid data files into XML based GRX Format
- Adding additional Information
- Rapid prototype for Euler 2D

Thorsten Ludewig, 42nd AIAA, Aerospace Science Meeting and Exhibit, Reno, Nevada, USA, 5-8 January 2004
GRX Tool

Online Visualization
Interactive steering

Thorsten Ludewig, 42nd AIAA, Aerospace Science Meeting and Exhibit, Reno, Nevada, USA, 5-8 January 2004
GRX Tool

- Online video generation
- Integrated video player
GRX 3D Tool
Virtual Visualization Toolkit (VVT / ShowMe3D)

- The idea of ShowMe3D is to develop a light weight application, which is easy to use, with a limited (but useful) set of functionality.

  - visualization
    - Geometry (surface)
    - results of a computation
  - debugging
    - online client - server connection
    - e.g. boundary updates
  - surface converting
    - e.g. quads to triangles
ShowMe3D: Motivation

- This program is designed for all programmers of Solver Objects.
- Based on the online "view" into the Server it can be very helpful for debugging.
- It is **NOT** designed to provide complete post processing like *TecPlot* or *Ensight*
ShowMe3D (continued)

Today's implementation of ShowMe3D contains:

- Visualization
  - Geometry data
  - A tree view of the Java 3D scene graph
- Surface conversion
  - For Alias Wave Front Objects only
ShowMe3D: main

- The main window contains all the GUI elements for the file input/output and visualization options

- load geometry
- save geometry
- shaded view
- wire frame view
- system properties
ShowMe3D: file types

ShowMe3D can load 2D and 3D object data in different file formats:

- Triangle
- Plot3D
- Plane 2D
- Plane 3D
- Volume 3D
- Autodesk/AutoCAD DXF
- AliasWavefront Objects
- 3D StudioMAX
- LightWave
ShowMe3D: Application
Java High Performance and Communications Test Suite

- In the following series of slides we will demonstrate Java's amazing numerical performance gains obtained over the last few years.

- Java numerical performance now rivals or exceeds that of C or C++ codes used in engineering.
Simple numeric Benchmark

Simple numeric without communication 10e11 iterations

![Graph showing simple numeric benchmark results on a Sun Microsystems Enterprise 10000 with 64 UltraSPARC II CPUs.]

- Simple numeric benchmark on a Sun Microsystems Enterprise 10000 with 64 UltraSPARC II CPUs

Thorsten Ludewig, 42nd AIAA, Aerospace Science Meeting and Exhibit, Reno, Nevada, USA, 5-8 January 2004
Mandelbrot Set dimension 7200 x 4800, max iterations 5000 running 400 threads on a Sun Microsystems Enterprise 6000 with 28 processors.

This code tests the self-scheduling of threads.
Matrix Multiply

for comparison, we have a Java and a C++-coded version of the sequential block-matrix multiply that does not use threads and multithreaded Java and C++ version.

- to compare floating point performance for scientific applications between C++ and Java on the test machines

- to measure parallel efficiency of a multithreaded application

Exactly the following source was used for both benchmarks. (C++ and Java)

```c
// get start time here
for( n=0; n<maxIterations; n++)
{
    for( i=0; i<dim; i++ )
    {
        for( j=0; j<dim; j++ )
        {
            for( k=0; k<dim; k++ )
            {
                c[i][j] += a[i][k]*b[k][j];
            }
        }
    }
}
// get end time here
```
Sequential Matrix Multiplication

<table>
<thead>
<tr>
<th>Runtime (2GHz Pentium 4, 1GB Memory)</th>
<th>1 run</th>
<th>2 run</th>
<th>3 run</th>
<th>4 run</th>
<th>5 run</th>
<th>6 run</th>
<th>7 run</th>
<th>8 run</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU g++ -O3 -mcpu=pentium4 -march=pentium4 -Wall (Version 3.3.1)</td>
<td>3,15</td>
<td>3,19</td>
<td>3,22</td>
<td>3,16</td>
<td>3,15</td>
<td>3,17</td>
<td>3,16</td>
<td>3,16</td>
</tr>
<tr>
<td>Sun Java HotSpot Client VM (Version 1.4.2_02-b03)</td>
<td>3,86</td>
<td>3,88</td>
<td>3,90</td>
<td>3,90</td>
<td>3,90</td>
<td>3,89</td>
<td>3,90</td>
<td>3,90</td>
</tr>
<tr>
<td>Sun Java HotSpot Server VM (Version 1.4.2_02-b03)</td>
<td>3,55</td>
<td>3,51</td>
<td>2,12</td>
<td>2,12</td>
<td>2,12</td>
<td>2,12</td>
<td>2,13</td>
<td>2,12</td>
</tr>
</tbody>
</table>

- A sequential (1 thread) matrix multiplication using a 30 times 30 matrix doing 10000 iterations on a single processor Pentium 4 PC running Linux.
- After the two warmup phases in the Sun Java HotSpot Server VM. This runtime is about 1.5 times faster then the compiled C++ binary.
- Due to a Linker error we could not use the -fast option with the Intel compiler.

Thorsten Ludewig, 42nd AIAA, Aerospace Science Meeting and Exhibit, Reno, Nevada, USA, 5-8 January 2004
Multithreaded Matrix Multiplication

<table>
<thead>
<tr>
<th>Runtime</th>
<th>time in s</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.8_14</td>
<td>516,94</td>
</tr>
<tr>
<td>1.2.2_08</td>
<td>38,97</td>
</tr>
<tr>
<td>1.3.0_03 Server</td>
<td>37,47</td>
</tr>
<tr>
<td>1.3.1_02 Server</td>
<td>21,69</td>
</tr>
<tr>
<td>1.4.0_01 Server</td>
<td>19,51</td>
</tr>
<tr>
<td>1.4.1_02 Server</td>
<td>17,31</td>
</tr>
<tr>
<td>C++ - GCC</td>
<td>26,65</td>
</tr>
<tr>
<td>C++ - Forte 6u1</td>
<td>17,26</td>
</tr>
</tbody>
</table>

Multithreaded matrix multiplication using a 100 times 100 matrix doing 10000 iterations with 400 threads on a 26 CPU Sun Microsystems Enterprise 6000.
Results a 100 x 100 matrix doing 10,000 iterations with 400 threads on the E6000 (26 CPUs)
Euler 3D Comparison

**JUST Euler 3D**

**CFD++**

- As a reference sample to check the correct communication (boundary update) of the JUSTGrid we computed a 3D cone with the JUST Euler 3D solver and CFD++
Conclusions

- With JUSTGrid a modern, well structured, easy to use and extensible framework for HPCC is provided.
- The code developer is freed from dealing with complex geometries, dynamic load balancing and inter block or domain communication.
- A numerical framework for a system of hyperbolic conservation laws is installed, based on the integral form of the conservation equations.
- The parallel efficiency is obtained if a sufficient number of threads and sufficient computational work within a thread can be provided.
- The execution speed of Java code has increased substantially over the last few years and now rivals the speed of C and C++ codes. More is too be expected.

Further work will be needed, but we following Kernighan's rules *Make it right before you make it faster* as well as *Don't patch bad code, rewrite it*, the latter rule being the reason for a pure Java flow solver code.
Future Work

Extending the JUSTGrid parallel layer to work with distributed memory machines (Beowulf cluster). (e.g., JavaSpaces, JINI, Sockets)
Acknowledgments

• This work is partly funded by the ministry of Sciences and Culture of the State of Lower Saxony, Germany under AGIP 1999.365 EXTV program.

• We are particularly grateful to Sun Microsystems, Benchmark Center, Germany for providing exclusive access to a 28 CPU Sun Enterprise 6000 server.

• We are grateful to Mr. Jean Muylaert for providing information about the European eXperimental Test Vehicle.

• This work is part of the Ph.D. work of Thorsten Ludewig.
References

Ginsberg, M., Häuser, J., Moreira, J.E., Morgan, R., Parsons, J.C., Wielenga, T.J.

Häuser, J., Ludewig, T., Williams, R., Winkelmann, R., Gollnick, T., Brunett, S., Muylaert, J.

Häuser, J., Ludewig, T., Williams, R., Winkelmann, R., Gollnick, T., Brunett, S., Muylaert, J.
*A Test Suite for High Performance Parallel Java* paper presented at 5th National Symposium on Large-Scale Analysis, Design and Intelligent Synthesis Environments, Williamsburg, VA, October 12th to 15th, 1999

Häuser, J., Ludewig, T., Williams, R., Winkelmann, R., Gollnick, T., Brunett, S., Muylaert, J.
*NASA Panel Java Soundbytes* paper presented at 5th National Symposium on Large-Scale Analysis, Design and Intelligent Synthesis Environments, Williamsburg, VA, October 12th to 15th, 1999

Häuser, J., Ludewig, Th., Gollnick, T., Winkelmann, R., Williams, R. D., Muylaert, J., Spel, M.,