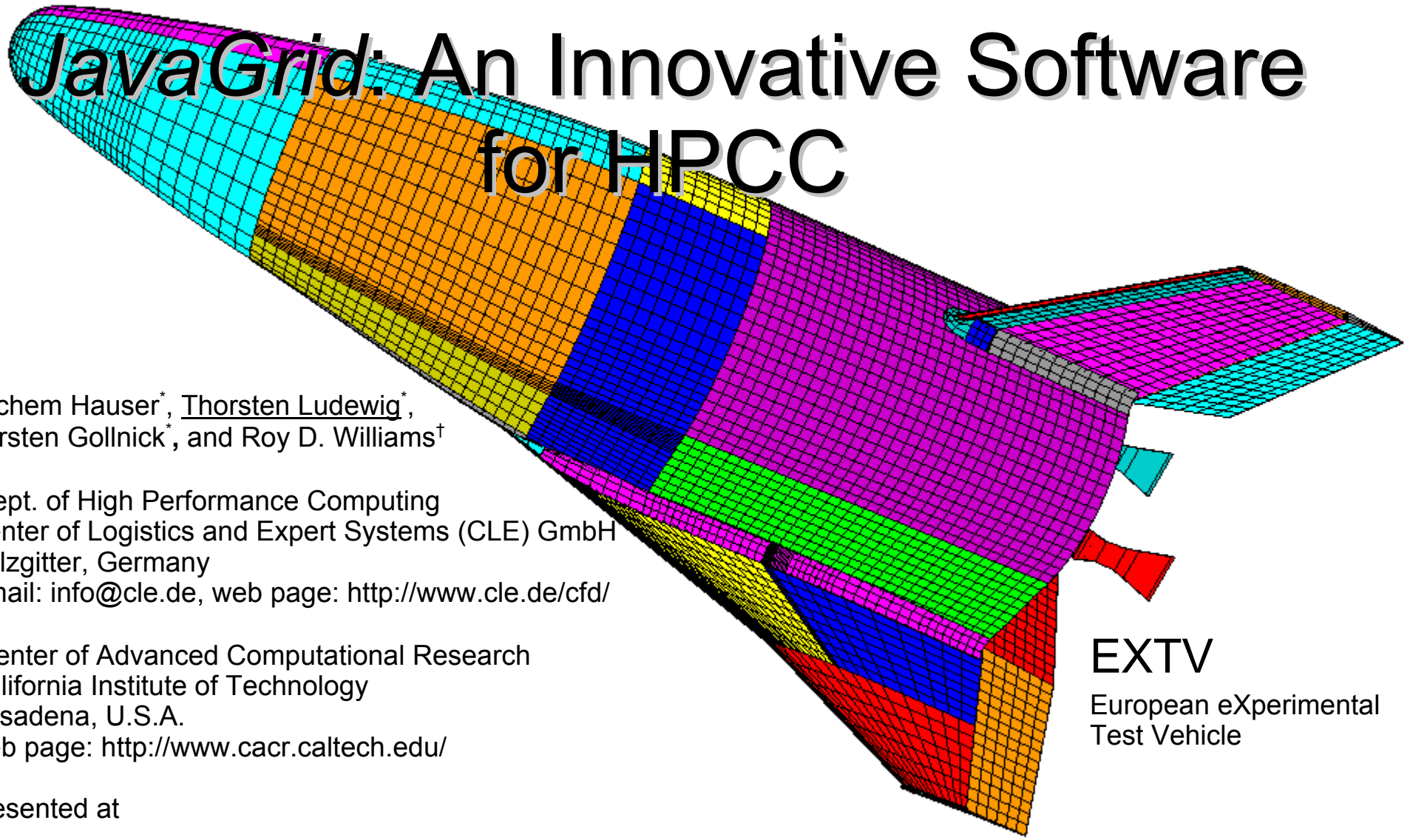
JavaGrid: An Innovative Software for HPCC



EXTV

European eXperimental Test Vehicle

Jochem Hauser*, Thorsten Ludewig*,
Torsten Gollnick*, and Roy D. Williams†

*Dept. of High Performance Computing
Center of Logistics and Expert Systems (CLE) GmbH
Salzgitter, Germany
Email: info@cle.de, web page: http://www.cle.de/cfd/

†Center of Advanced Computational Research
California Institute of Technology
Pasadena, U.S.A.
web page: http://www.cacr.caltech.edu/

presented at

ECCOMAS COMPUTATIONAL FLUID DYNAMICS CONFERENCE, SWANSEA 2001

Overview

- Scope of JavaGrid (Framework)
- Architecture Overview
- Object Oriented Programming (OOP)
- Threads
- Graphical Applications (ClientGUI & ShowMe3D)
- Java Performance Results
- Conclusions
- Future Work

Java as the Language for HPCC

- platform independence
- simple and straight forward 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 JavaGrid

- 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 JavaGrid

- 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 **JavaGrid**, can be built entirely in Java in a transparent, object-oriented approach.

- **JavaGrid 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 *JavaGrid*

JavaGrid a framework for HPCC in engineering, science, and life sciences providing:

- full portability to any computing platform
- an infrastructure to couple ,,legacy codes"
- all layers for
  - dealing with arbitrary complex *3D geometries*
  - parallelization without libraries
  - scheduling (automatic dynamic load balancing)

Scope of *JavaGrid*

- all layers for (continued)
  - internet connectivity
    - collaborative engineering
    - outsourcing
    - interactive steering
  - system security (secure communication over the internet)
  - post processing with the Virtual Visualization Toolkit (VVT / *ShowMe3D* with Java3D)

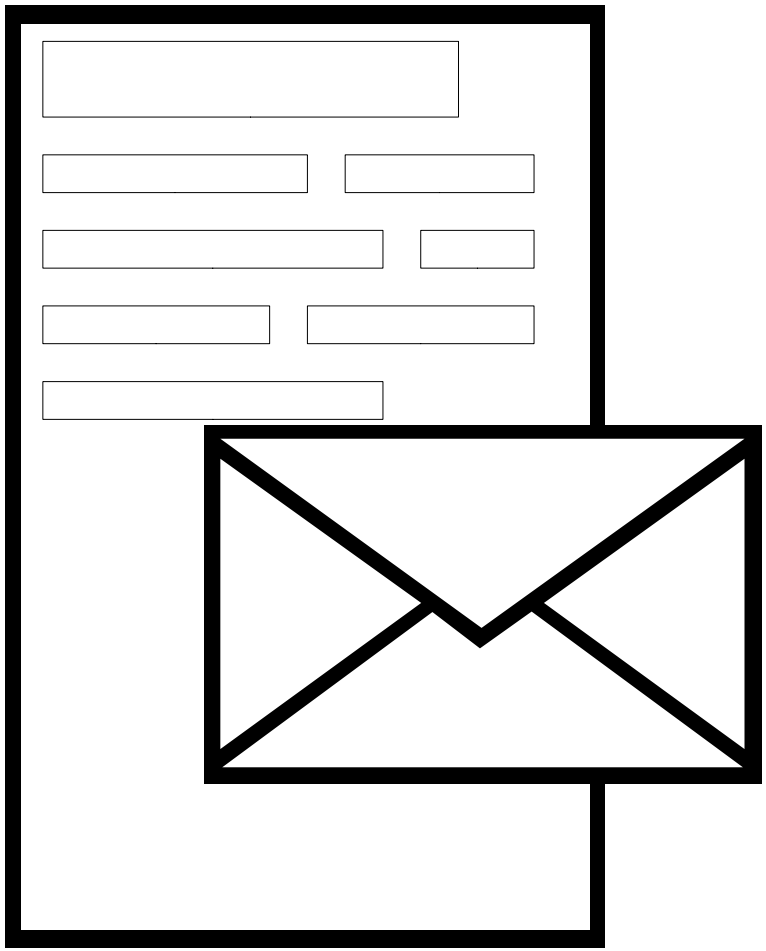
Scope of *JavaGrid*

- all layers for (continued)
  - solver package layer
    - contains all methods to construct a solver whose physics is governed by a set of conservation laws.
    - provides the infrastructure for a large class of computational problems
      - Navier-Stokes equations (fluid dynamics)
      - Maxwell's equations (electromagnetics, including semiconductor simulation)
      - Schrödinger's equation (quantum mechanics)

Scope of *JavaGrid*

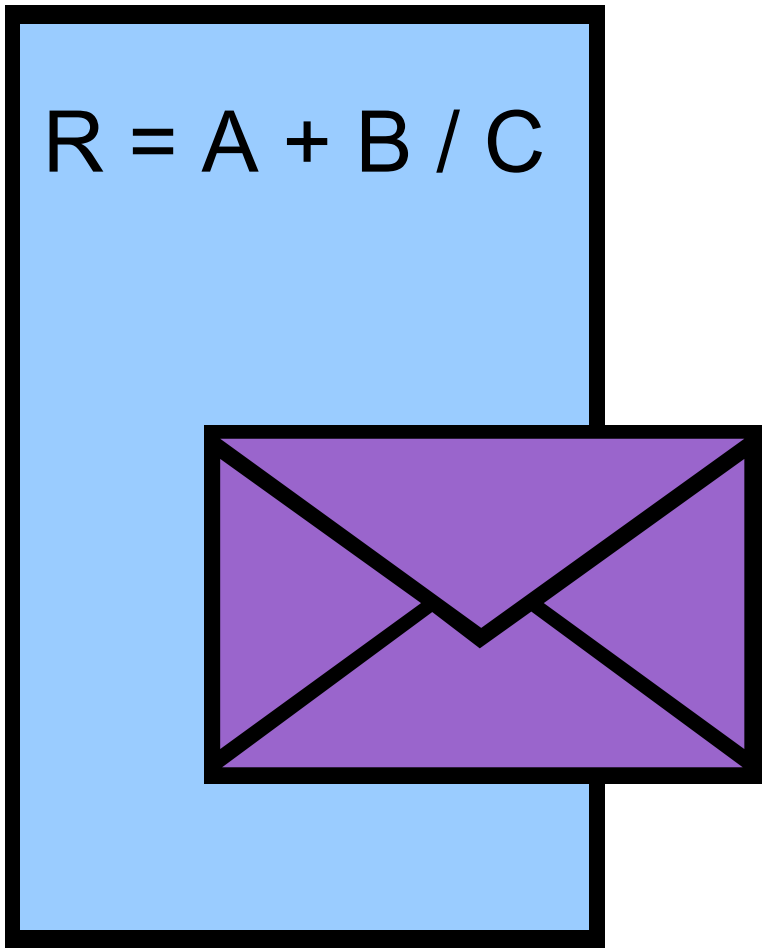
- A solver only needs to contain the physics and numerics of the simulation task for a single block or a single domain.
- The solver does not need to know anything about the geometry data or the parallelization.
- It has a simple structure
- The solver can be tested independently before its integration
- **Replacing the default solver by one's own solver.**

Current "static numeric" Solvers



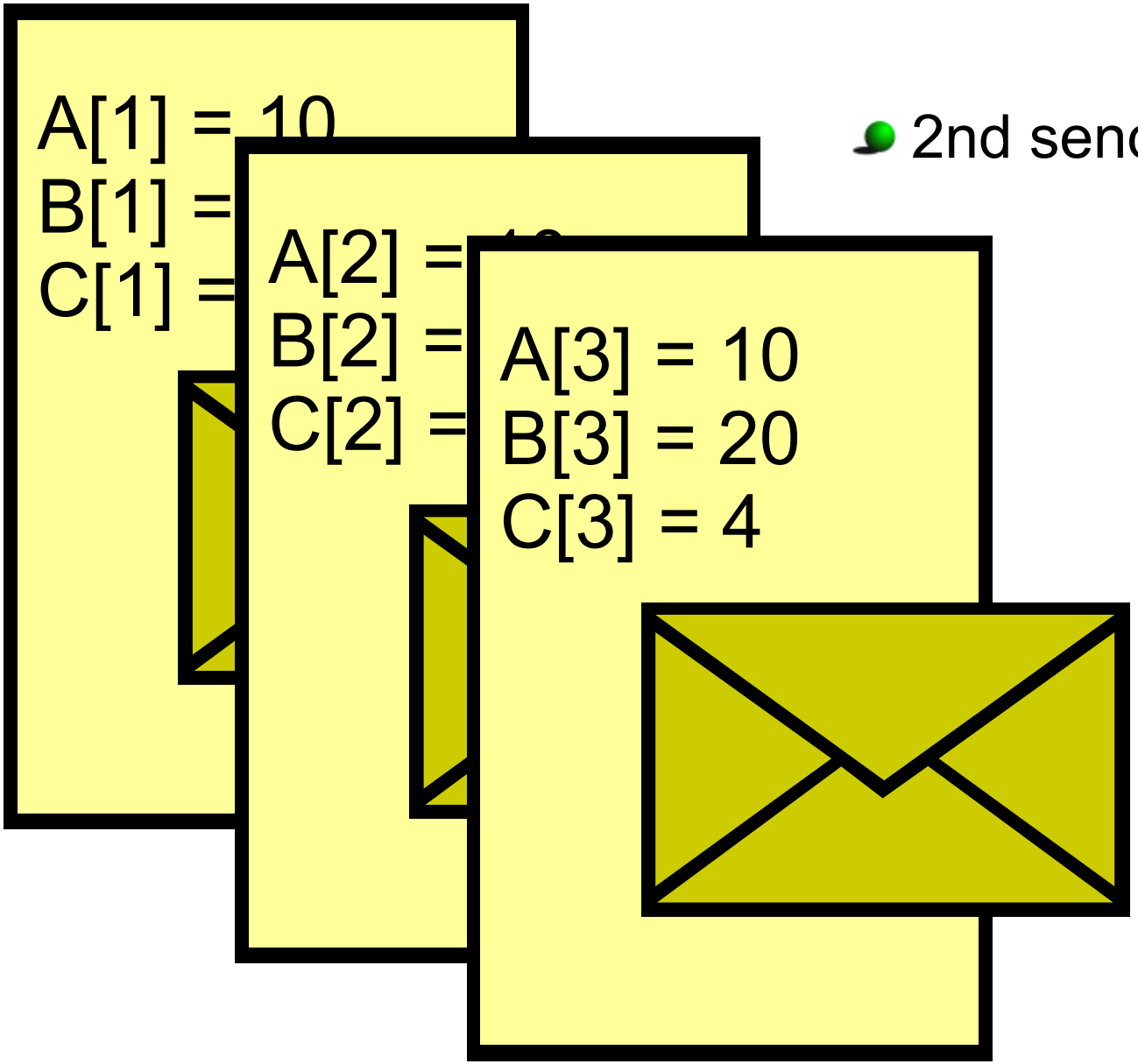
- the solver numerics is "static"
- all provided numerical strategies have do be declared at build time of the solver executable
- the user data will be handled like filling out predefined form

"dynamic numerics" Framework



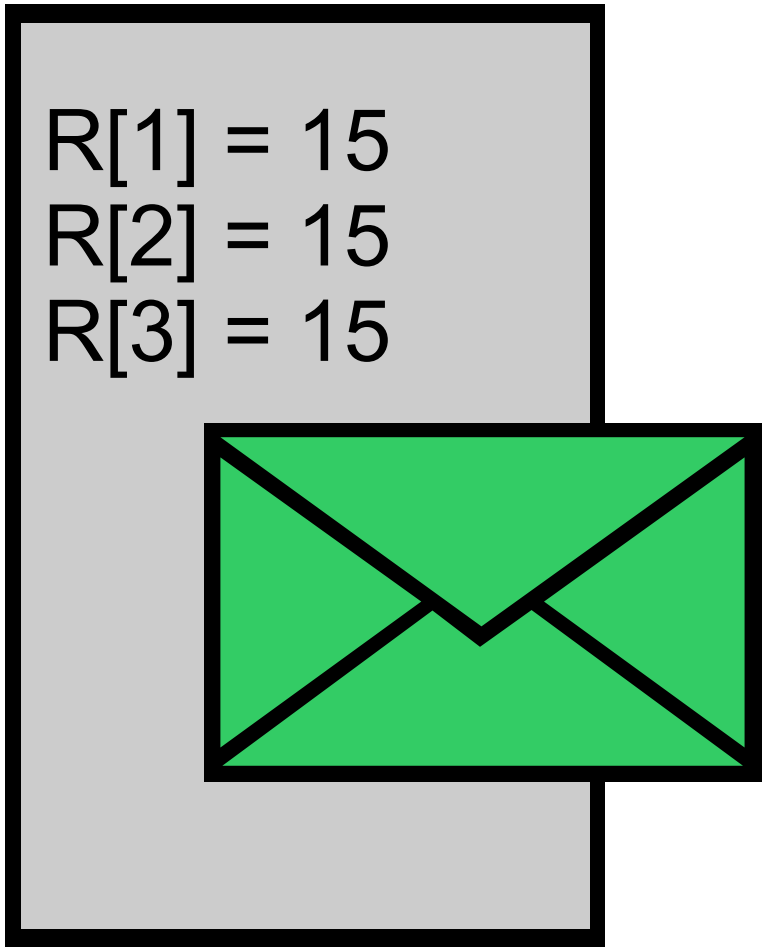
- the numerical strategies can be exchanged during runtime
- 1st send the numerics

Framework



- 2nd send the data

Framework



- 3rd receive the **self-defined** result.

Architecture Overview

Solver/Code Development

Session-ID

RMI based Network

Session-ID

Server

Visualization

Session-ID

Session-ID

Collaborative Engineering

Architecture Overview

Client

Server

- 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

Architecture Overview

Client

Server

- 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

class name
+variable1: integer +variable2: double
+method1 +method2

- 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 trough a method.

OOP Example

- Programming in 'C'

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

date.day = 32;
```

- Programming in 'Java'

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

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

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

ERROR!

LOGISTIKZENTRUM CLE

Threads

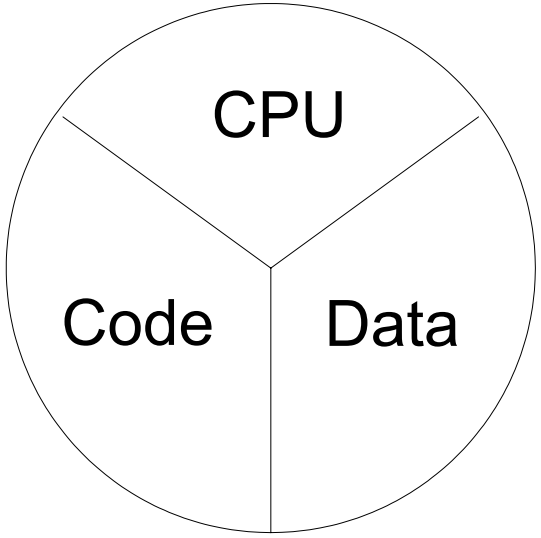
an efficient way of parallelizing codes

- 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.
  - Each task is usually called *thread* which is the short form for thread of control.
  - Programs that can run more than one thread at once are said to be *multithreaded*.

What are threads?

- Three Parts of a Thread
  - A virtual CPU
  - The code the CPU is executing
  - The data the code works on

A thread or
execution context

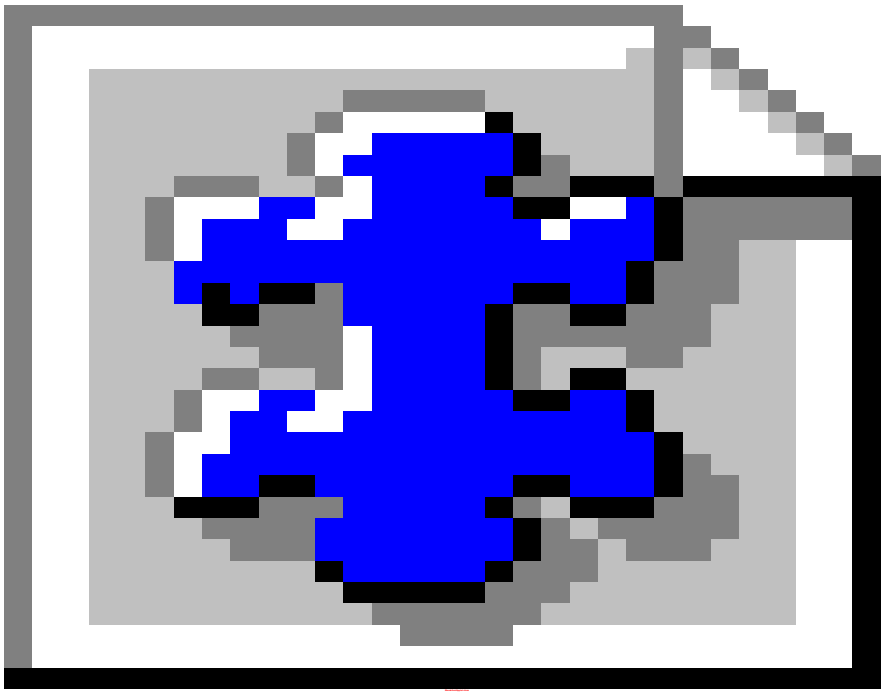


Why Threads are good for CFD

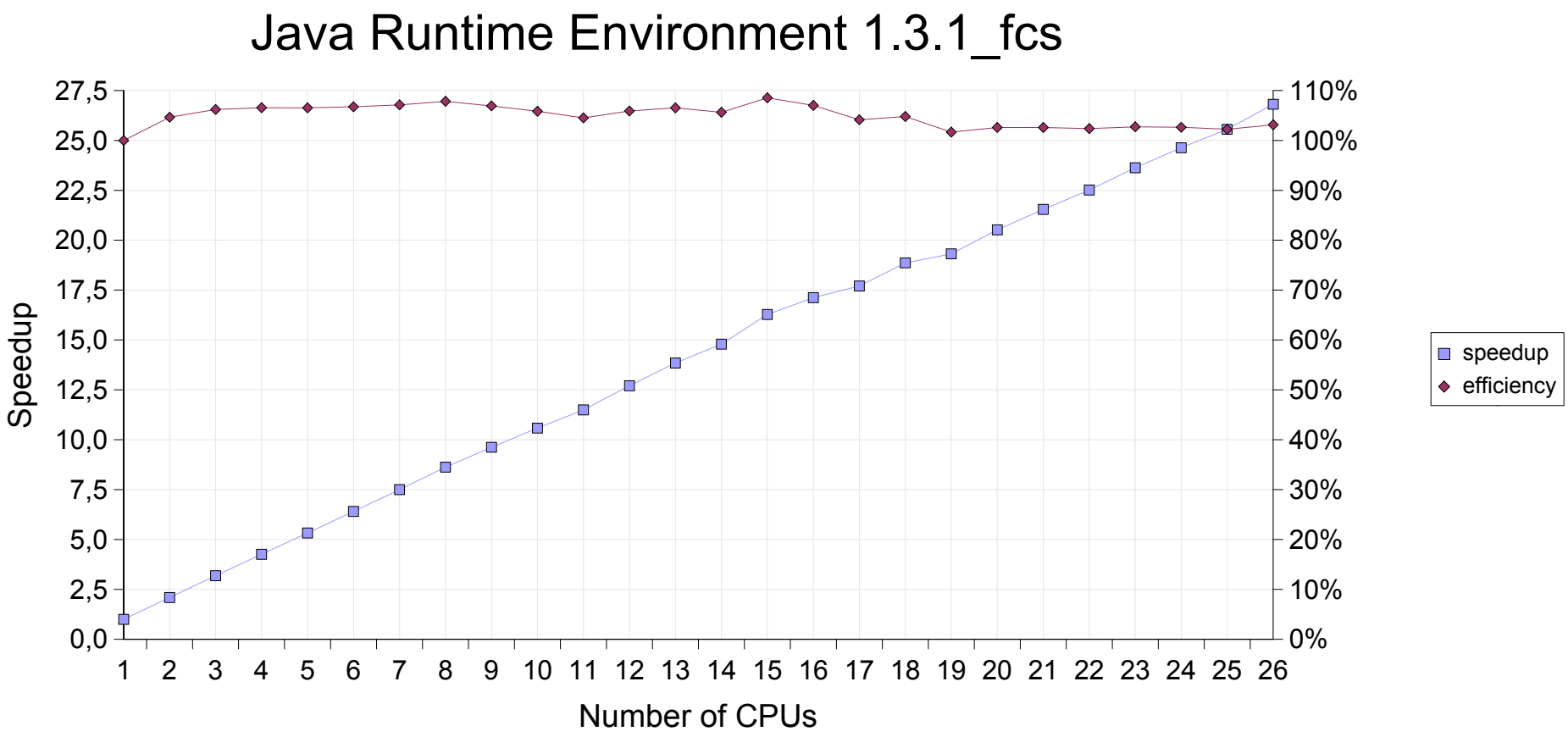
- Threads as a general parallelization strategy for CFD codes
- Advanced numerical schemes in CFD, i.e. GMRES, do not require the same computational work for each grid cell.
- Sophisticated dynamic load balancing algorithms on distributed-memory machines

Mandelbrot

- This code tests the self-scheduling of threads
- Computation of the Mandelbrot set performed on a 2D grid in the complex plane

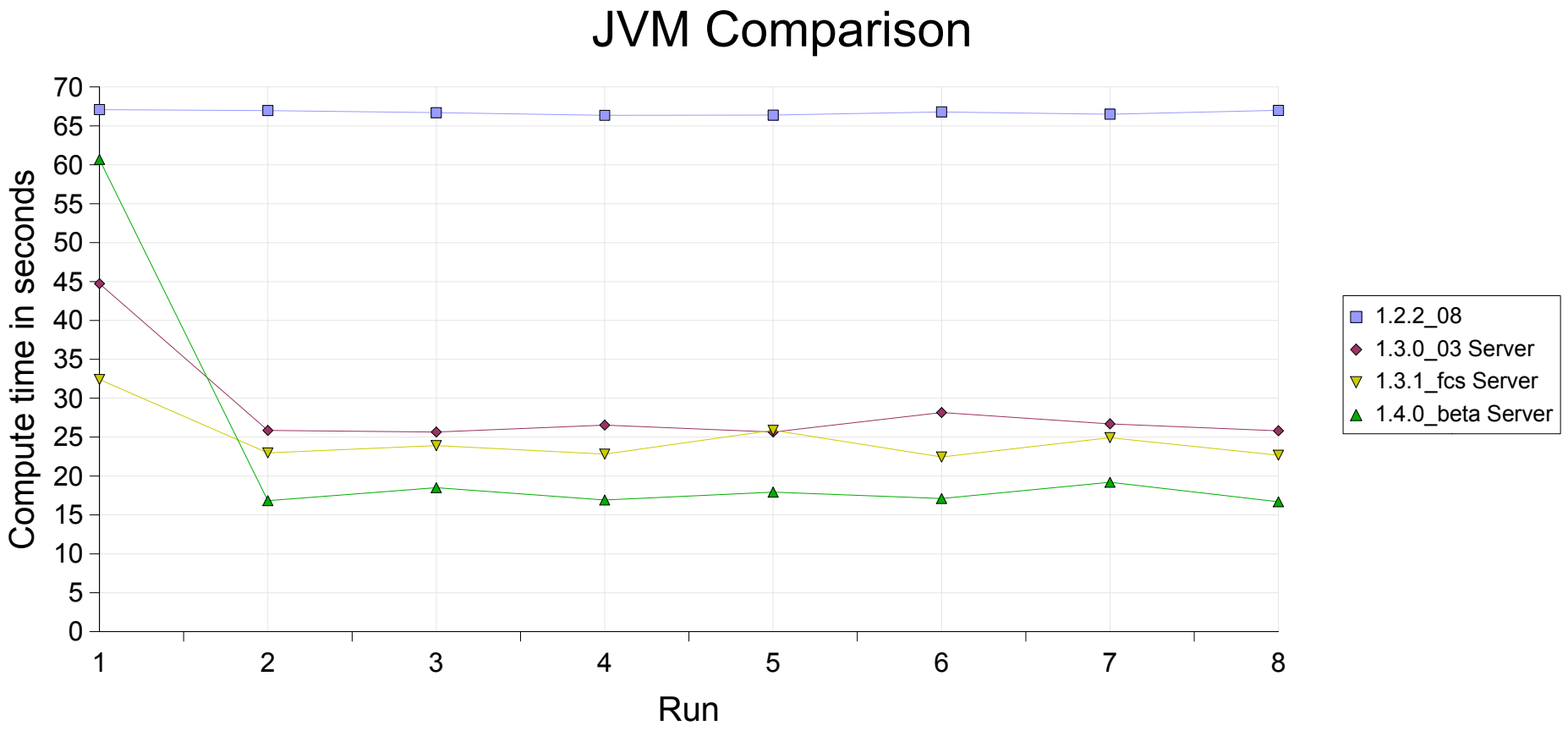


Mandelbrot Set - dimension 7,200 x 4,800, max iterations 5,000 running 400 threads



Sun Enterprise 6000
26 CPUs, 400 MHz, 8 MB Cache, 32 GB Memory

Mandelbrot Set - dimension 7,200 x 4,800, max iterations 5,000 running 400 threads

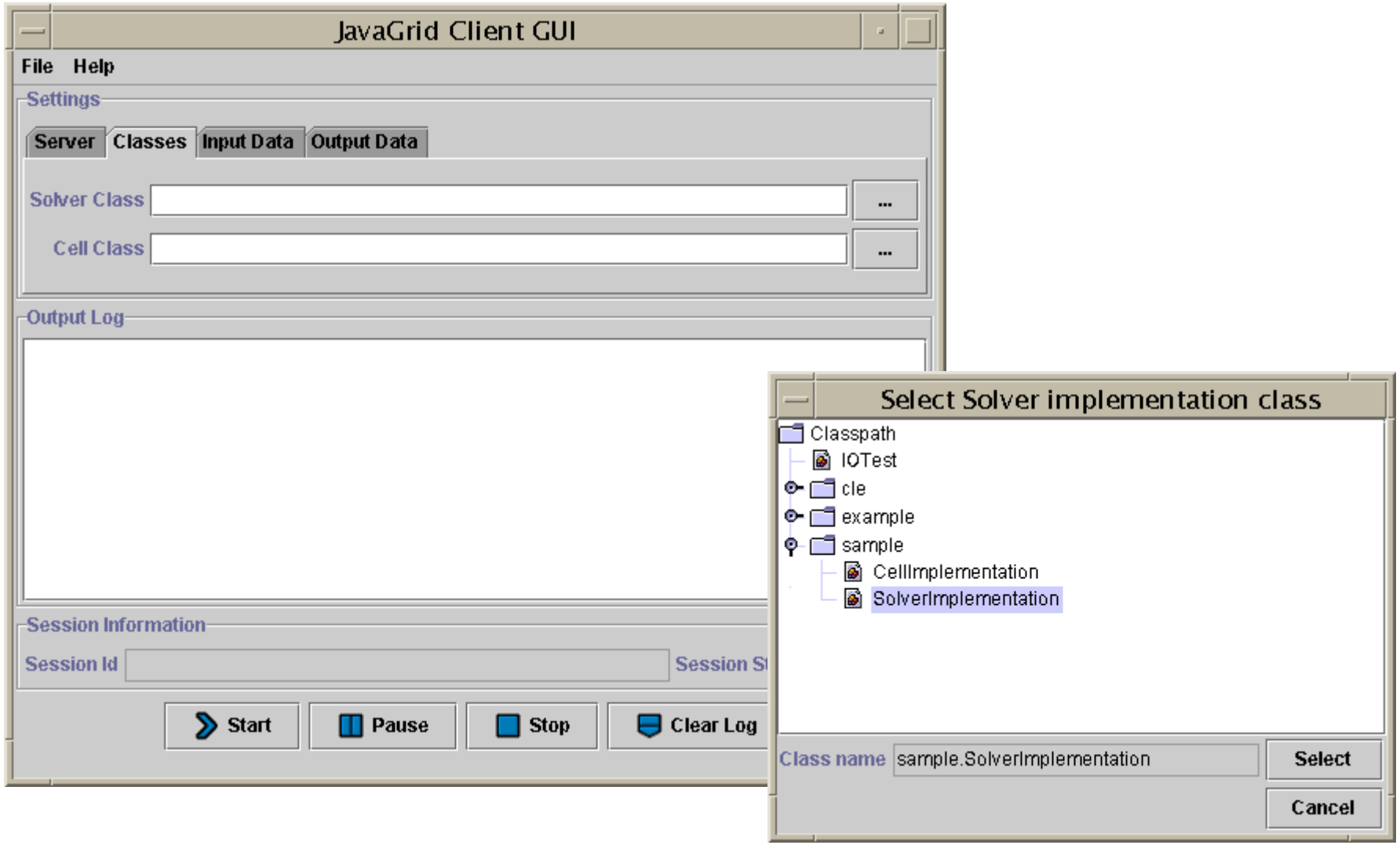


Sun Enterprise 6000
26 CPUs, 400 MHz, 8 MB Cache, 32 GB Memory

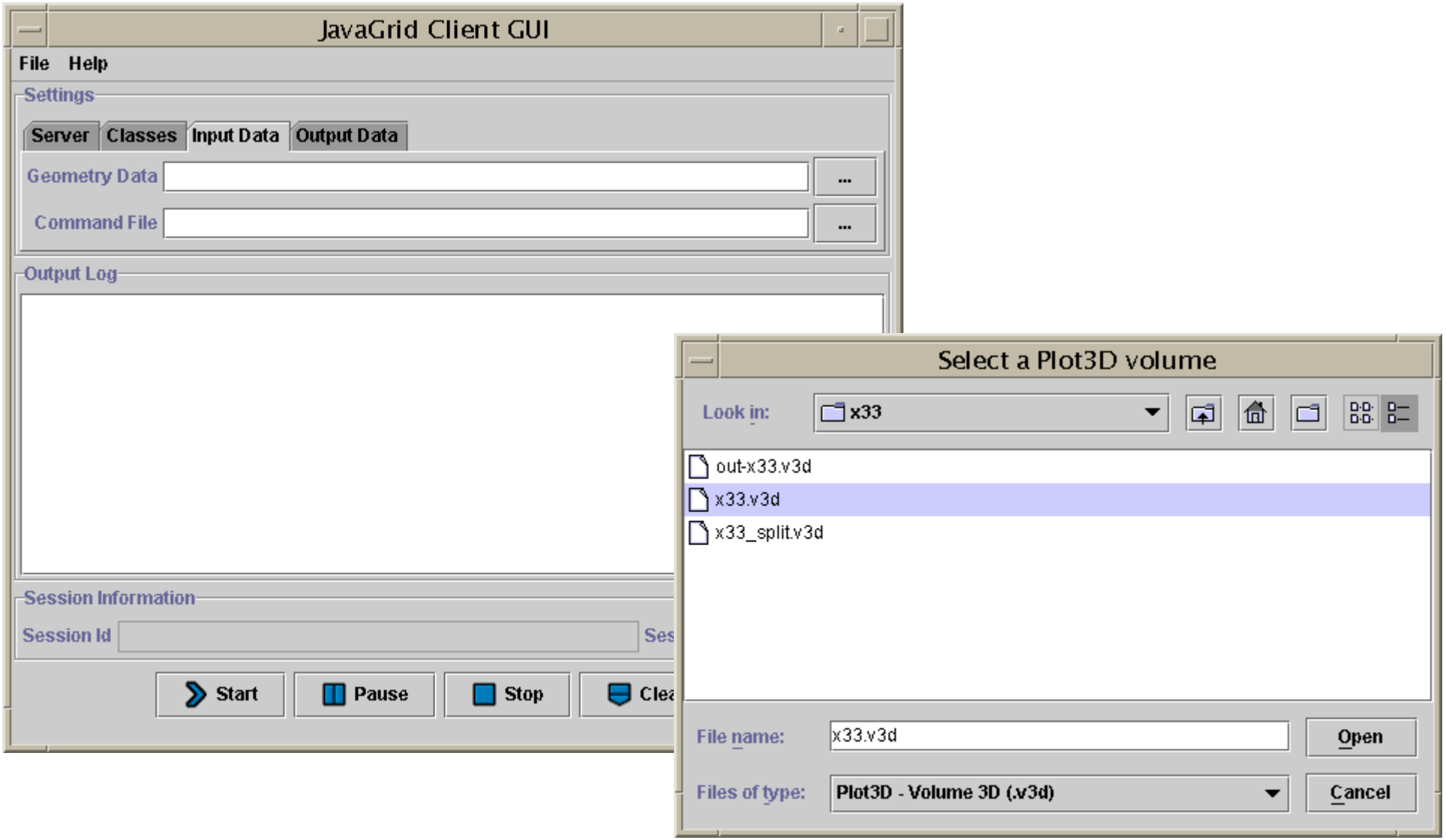
ClientGUI (2D & 3D)

- Why we like to use simple graphical user interface for the *JavaGrid*?
  - Because for a non-Programmer it is to difficult to collect all different parts needed for a *JavaGrid* session into a Java source text and compile it for himself.
  - It's easier to run a quick test case without falling into common programming traps.

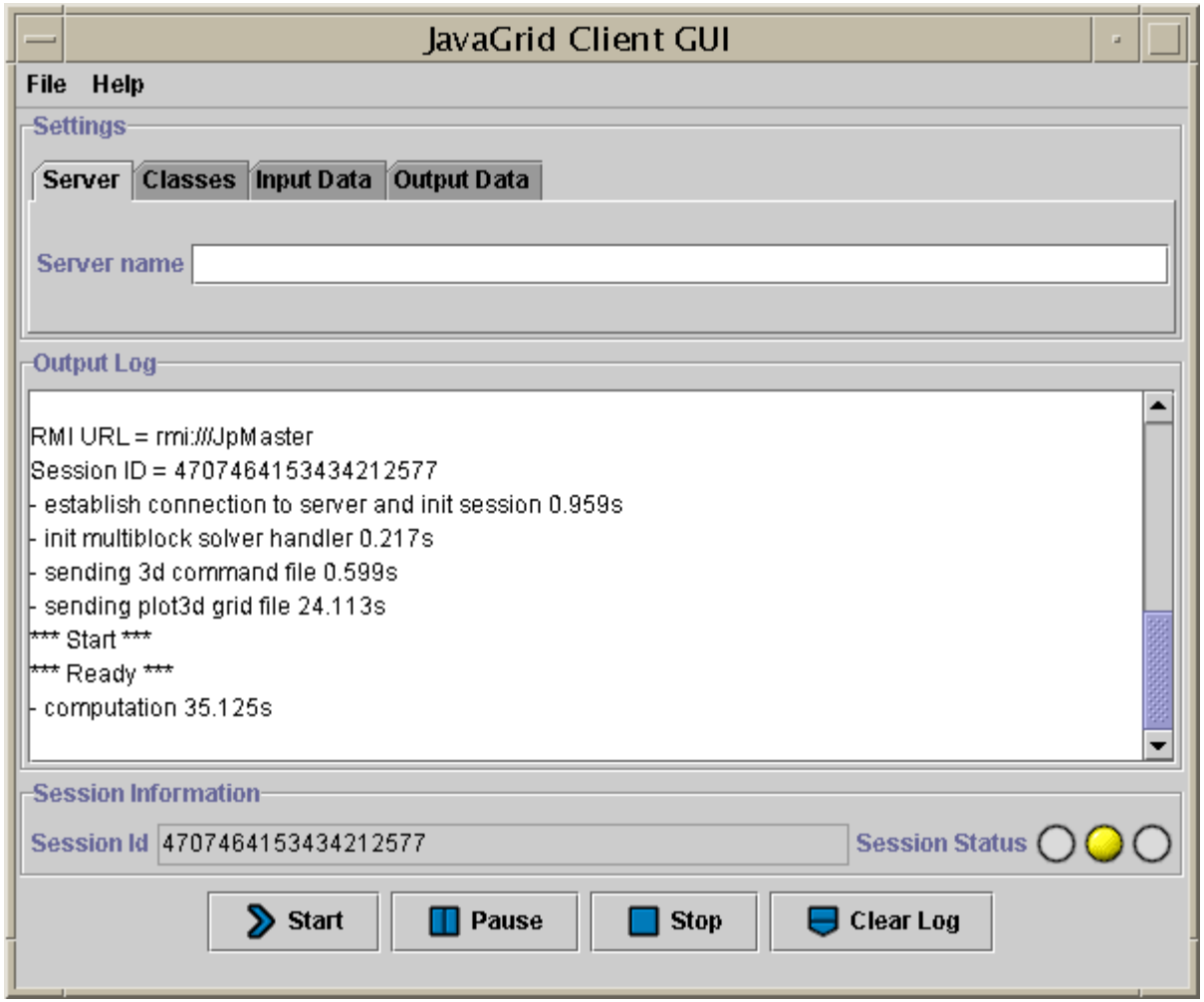
Setting Solver & Cell Class



Setting Geometry & Command files



Receiving Result



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
    - f.e. boundary updates
  - surface converting
    - f.e. quads to triangles

ShowMe3D: Motivation

- This program is at first 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)

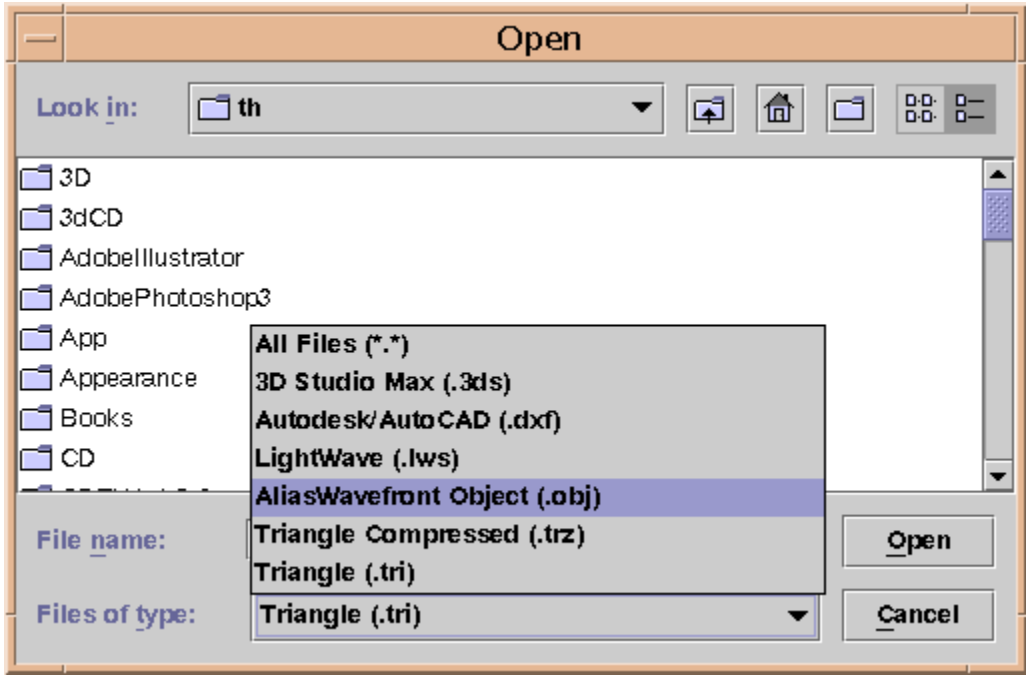
- Today's implementation of ShowMe3D contains
  - Visualization
    - Geometry data
    - a tree view of the Java 3D scene graph
  - Surface converting
    - for Alias Wave Front Objects only

ShowMe3D: main

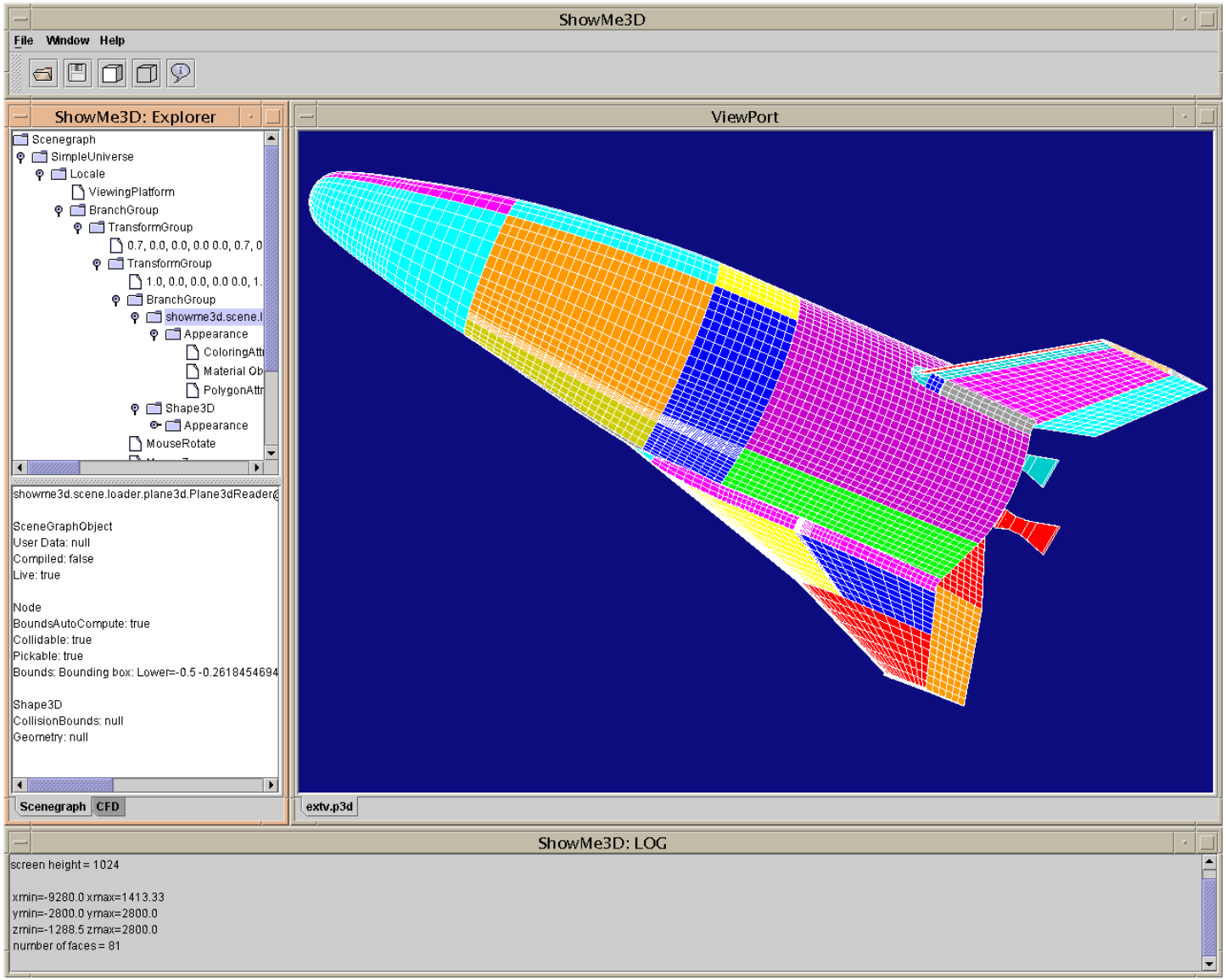
- The main window contains all the GUI elements for the file input / output and visualization options
  - load geometry
  - save geometry
  - shaded view
  - wire frame view
  - system properties

ShowMe3D: file types

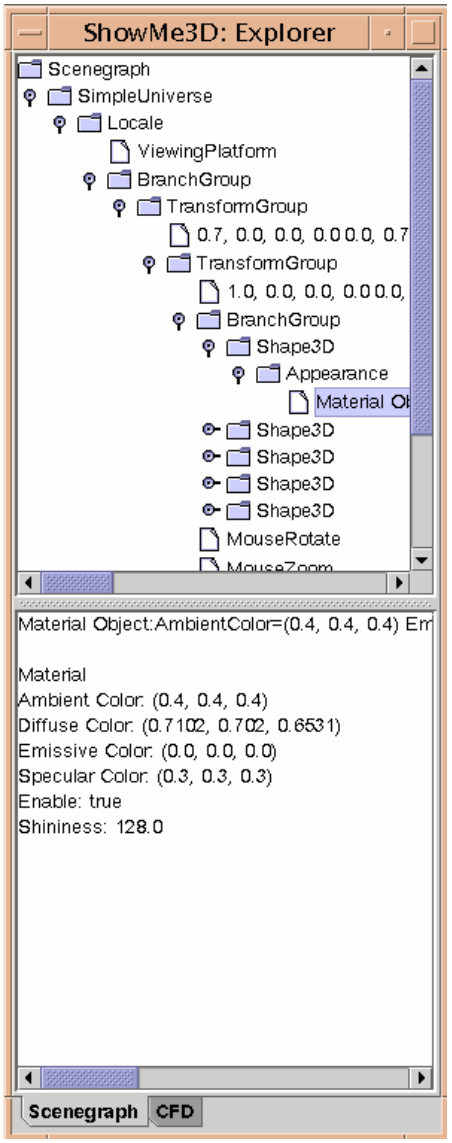
- ShowMe3D can load 2D and 3D object data in different file formats
  - Triangle
  - Plot3D
    - Plane 2D
    - Plane 3D
    - Volume 3D
  - Autodesk/AutoCAD DXF
  - AliasWavefront Objects
  - 3D StudioMAX
  - LightWave



ShowMe3D: Application

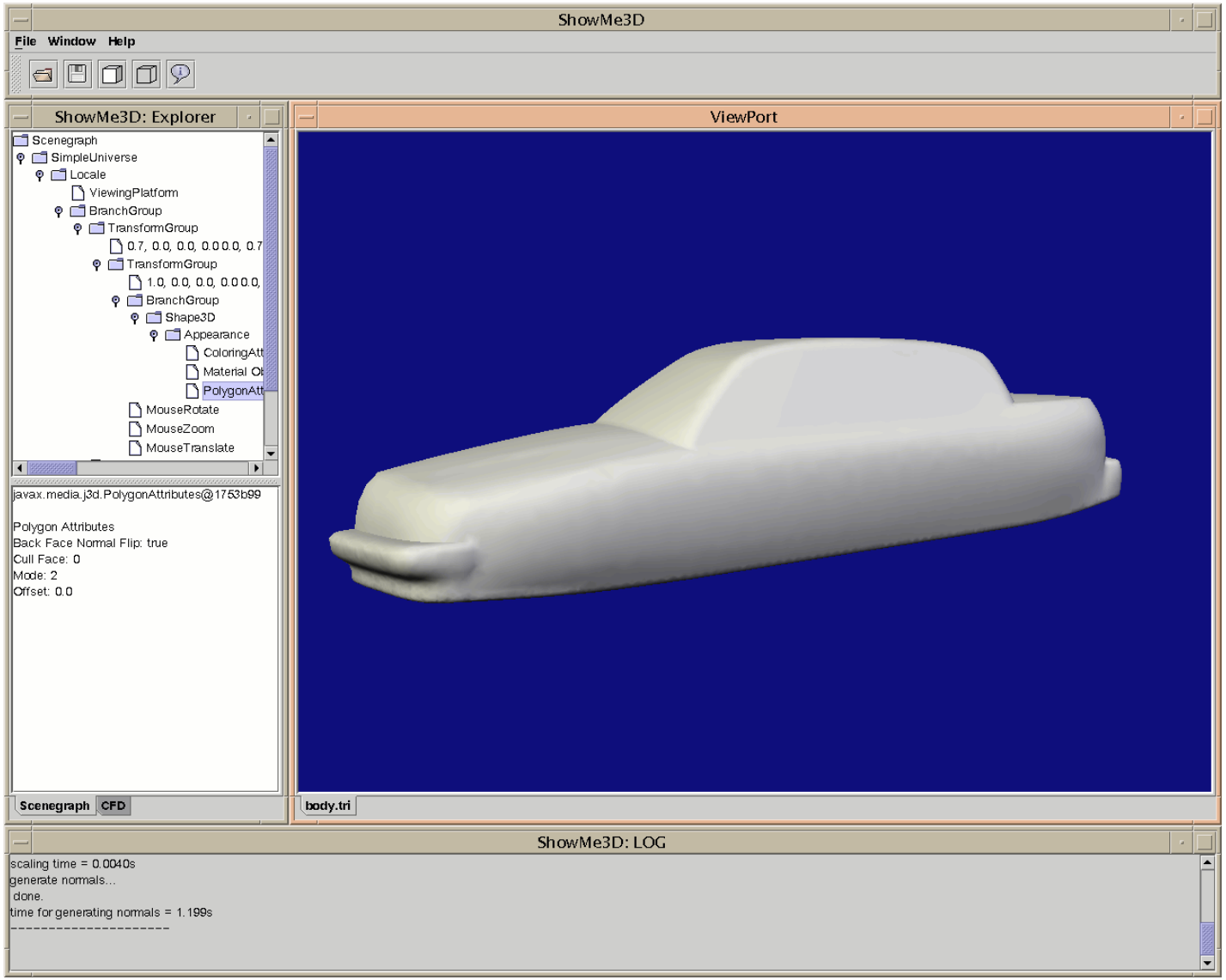


ShowMe3D: Explorer

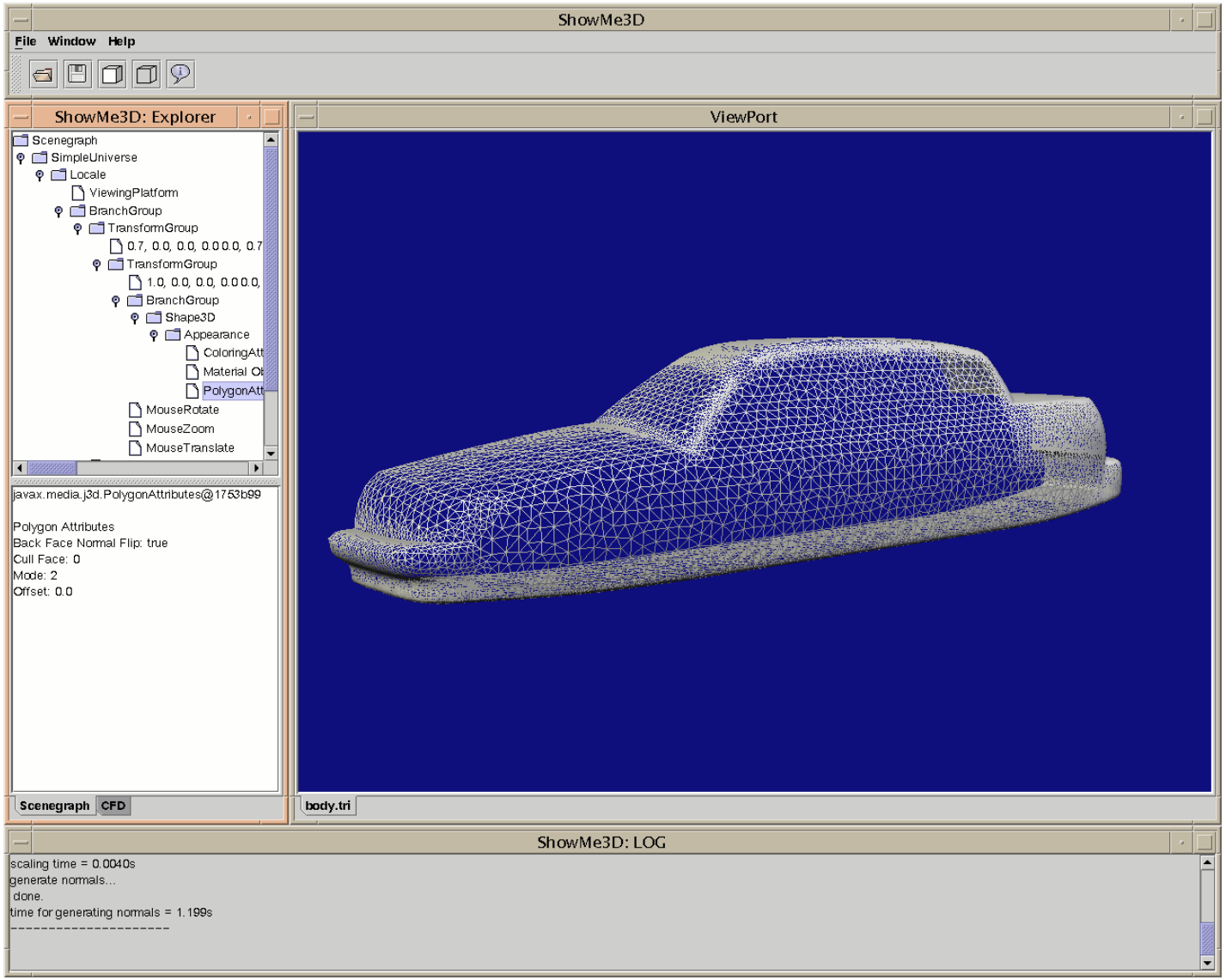


- The explorer shows the Java 3D scene graph in a tree view
- In the bottom text area the properties of the selected component are printed.

ShowMe3D: shaded view



ShowMe3D: wire frame view

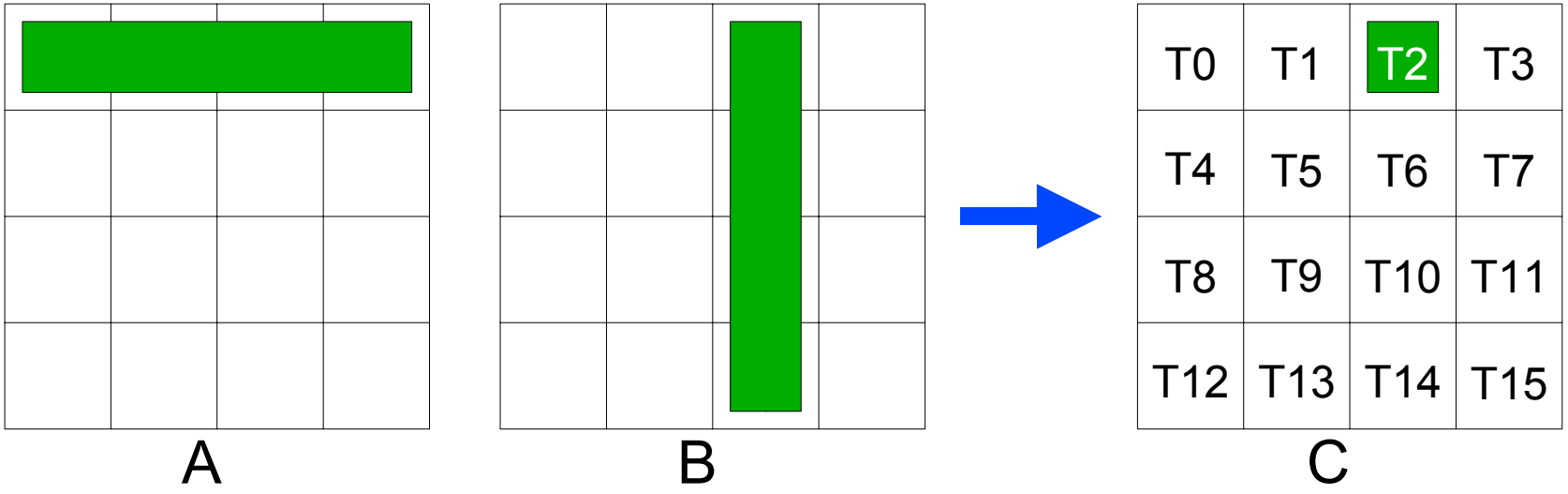


Java High Performance and Communications Test Suite

- **In the following series of slides we will demonstrate Java's amazing numerical performance gains obtained over the last two years.**
- **Java numerical performance now rivals or exceeds that of C or C++ codes used in engineering.**

Matrix Multiply

- Parallel matrix multiplication is implemented by block matrices, as shown in the Figure Matrices **A** and **B** are multiplied to produce **C**



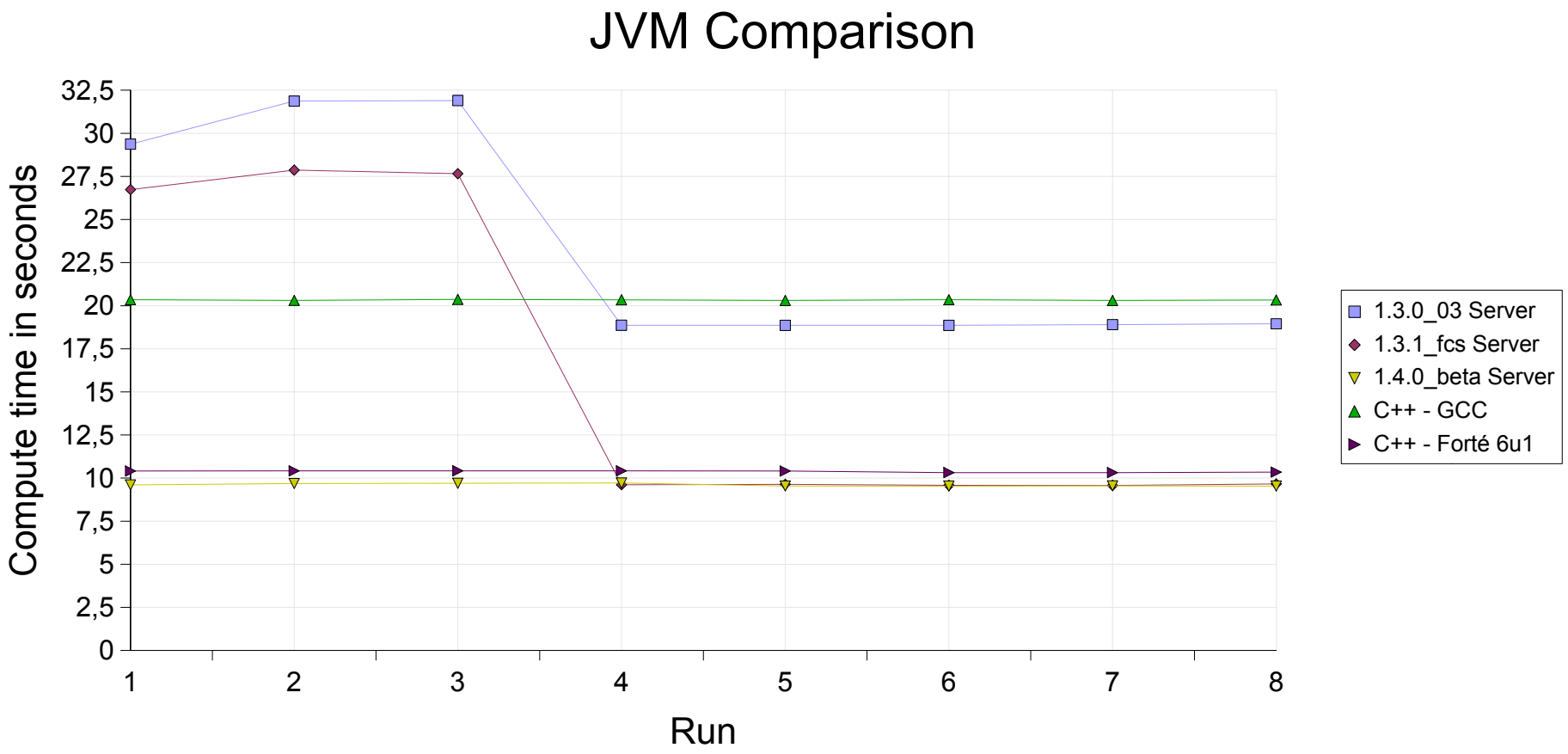
The multi-threaded matrix multiplication is performed by splitting matrix C into partitions. Each partition is then calculated by one thread, with the thread numbering as shown for matrix C. Concurrent access to the memory containing A and B is necessary: here we see the memory that thread 2 accesses.

LOGISTIKZENTRUM CLE

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

Sequential Matrix Multiplication



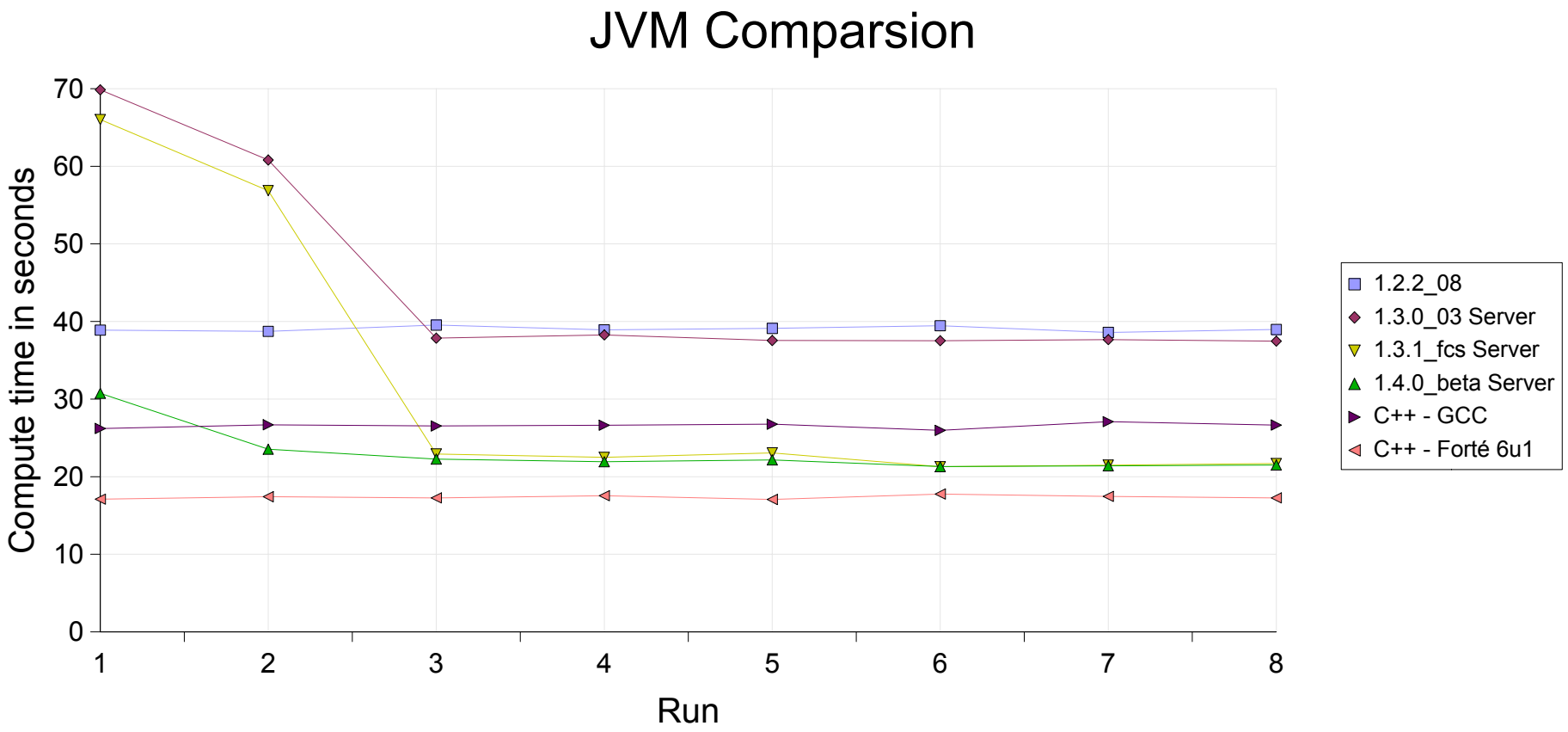
Results a 30 x 30 matrix doing 10,000 iterations on the E6000

Sequential Matrix Multiplication

run	1	2	3	4	5	6	7	8
1.1.8_14	56,15	56,14	56,16	56,16	56,16	56,17	56,16	56,16
1.2.2_08	21,59	21,57	21,64	21,71	21,63	21,73	21,56	21,61
1.3.0_03 Client	36,25	36,23	36,27	36,42	36,42	36,43	36,23	36,27
1.3.0_03 Server	29,37	31,87	31,9	18,87	18,86	18,86	18,9	18,95
1.3.1_fcs Client	36,3	36,26	36,31	36,26	36,31	36,46	36,26	36,44
1.3.1_fcs Server	26,74	27,86	27,66	9,62	9,64	9,58	9,58	9,66
1.4.0_beta Client	47,61	47,51	47,55	47,51	47,51	47,55	47,56	47,51
1.4.0_beta Server	9,6	9,69	9,7	9,71	9,54	9,54	9,55	9,54
C++ - GCC	20,35	20,31	20,36	20,34	20,31	20,35	20,31	20,33
C++ - Forté 6u1	10,41	10,42	10,42	10,42	10,41	10,32	10,32	10,35

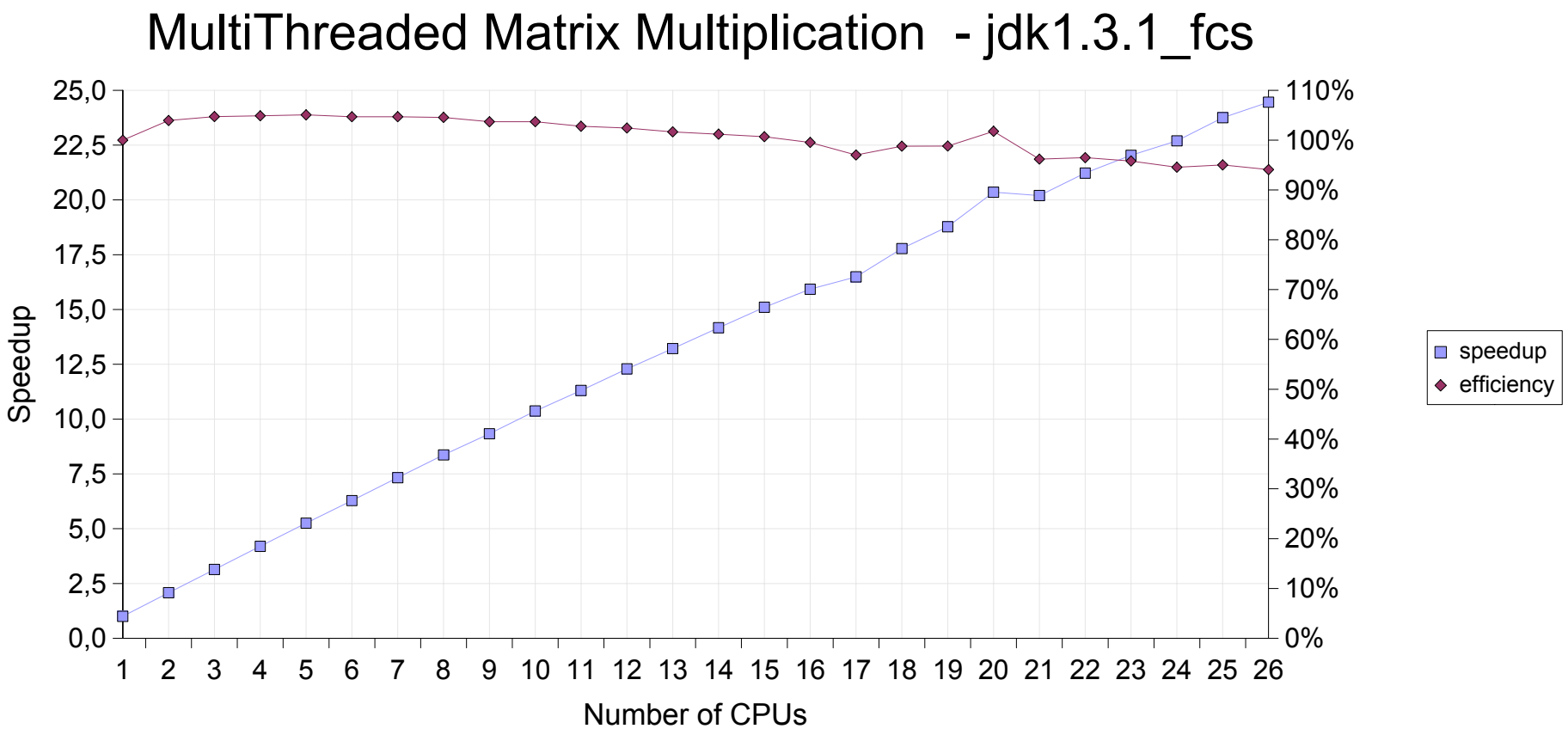
Results a 30 x 30 matrix doing 10,000 iterations on the E6000

Multi-threaded Matrix Multiplication



Results a 100 x 100 matrix doing 10,000 iterations with 400 threads on the E6000 (26 CPUs)

Multi-threaded Matrix Multiplication

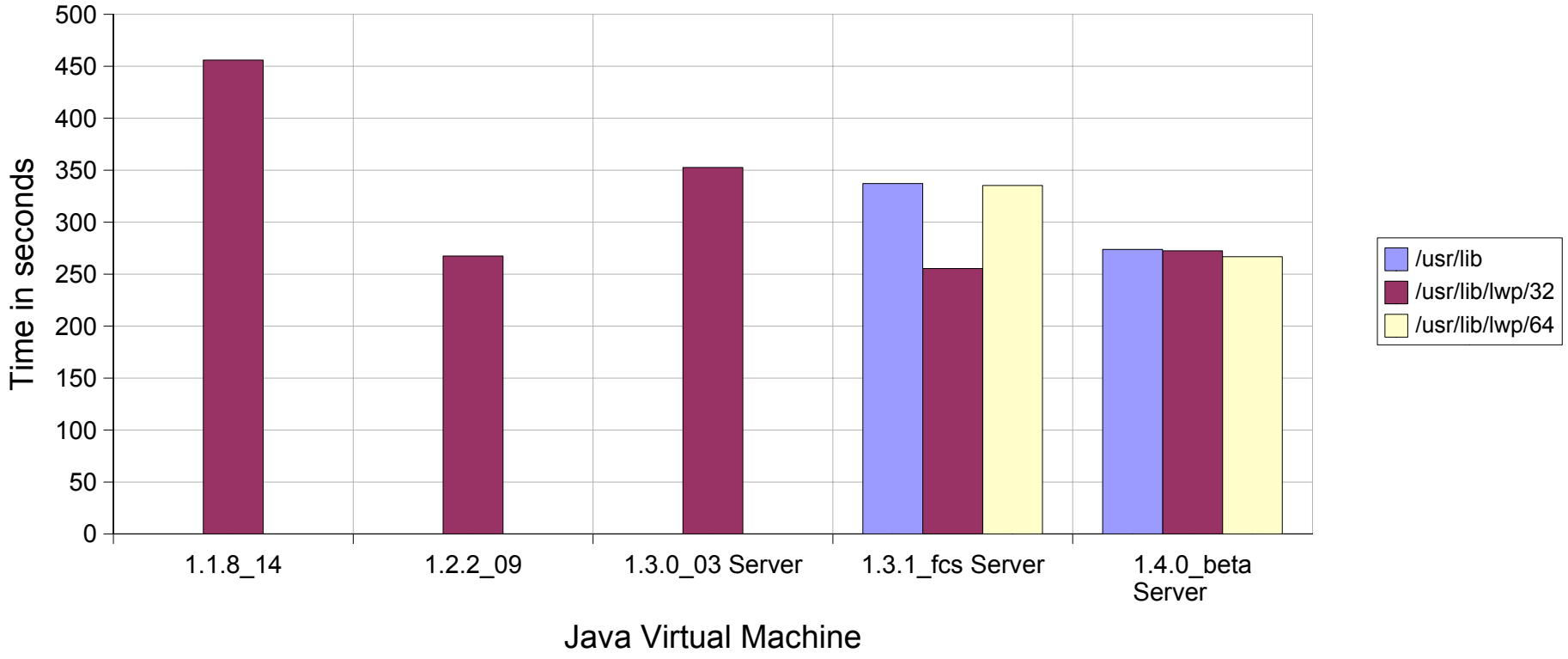


Results a 100 x 100 matrix doing 10,000 iterations with 400 threads on the E6000 (26 CPUs)

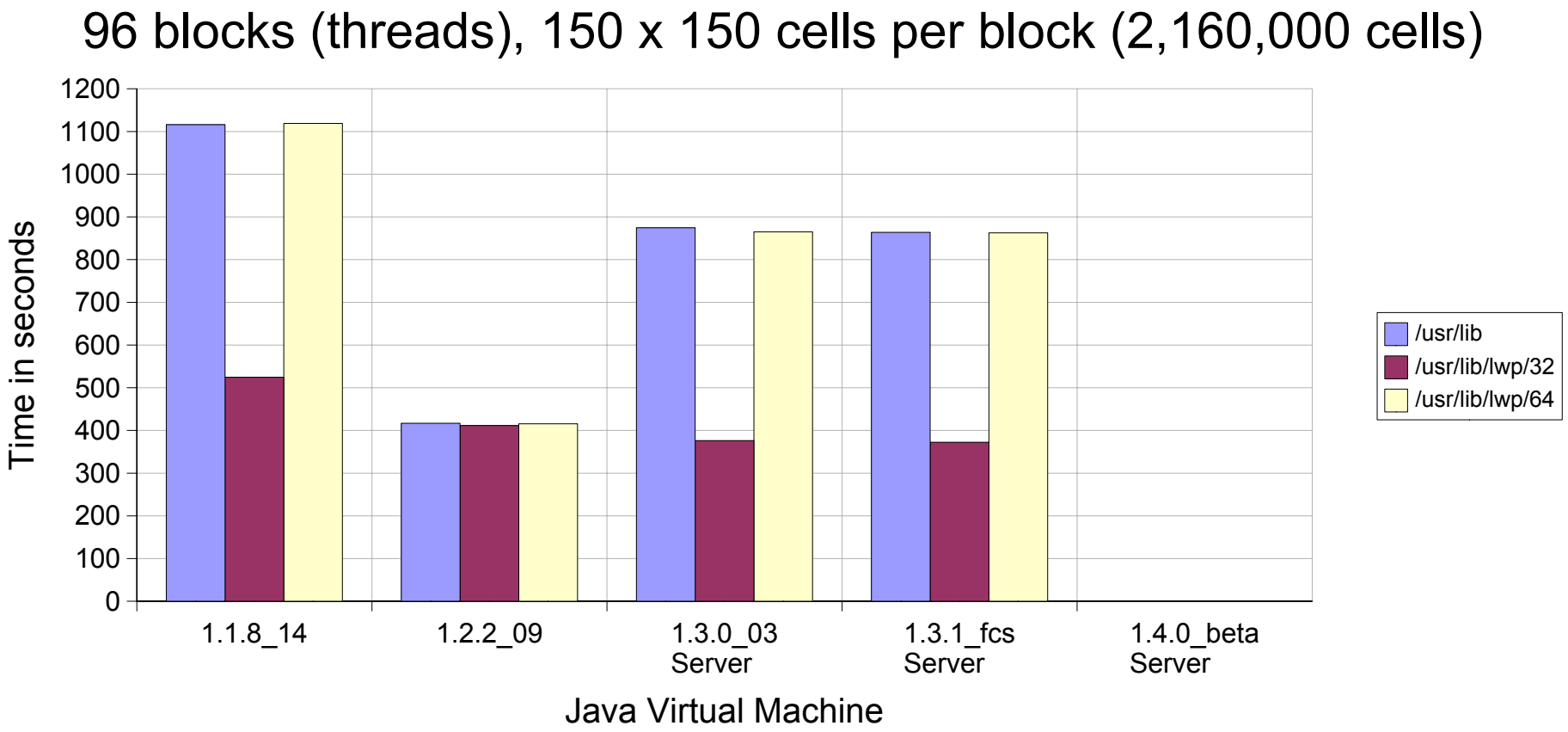
Laplace Solver

- In this test program we are using a global thread synchronization to test the systems behavior if it have to synchronize different numbers of threads.

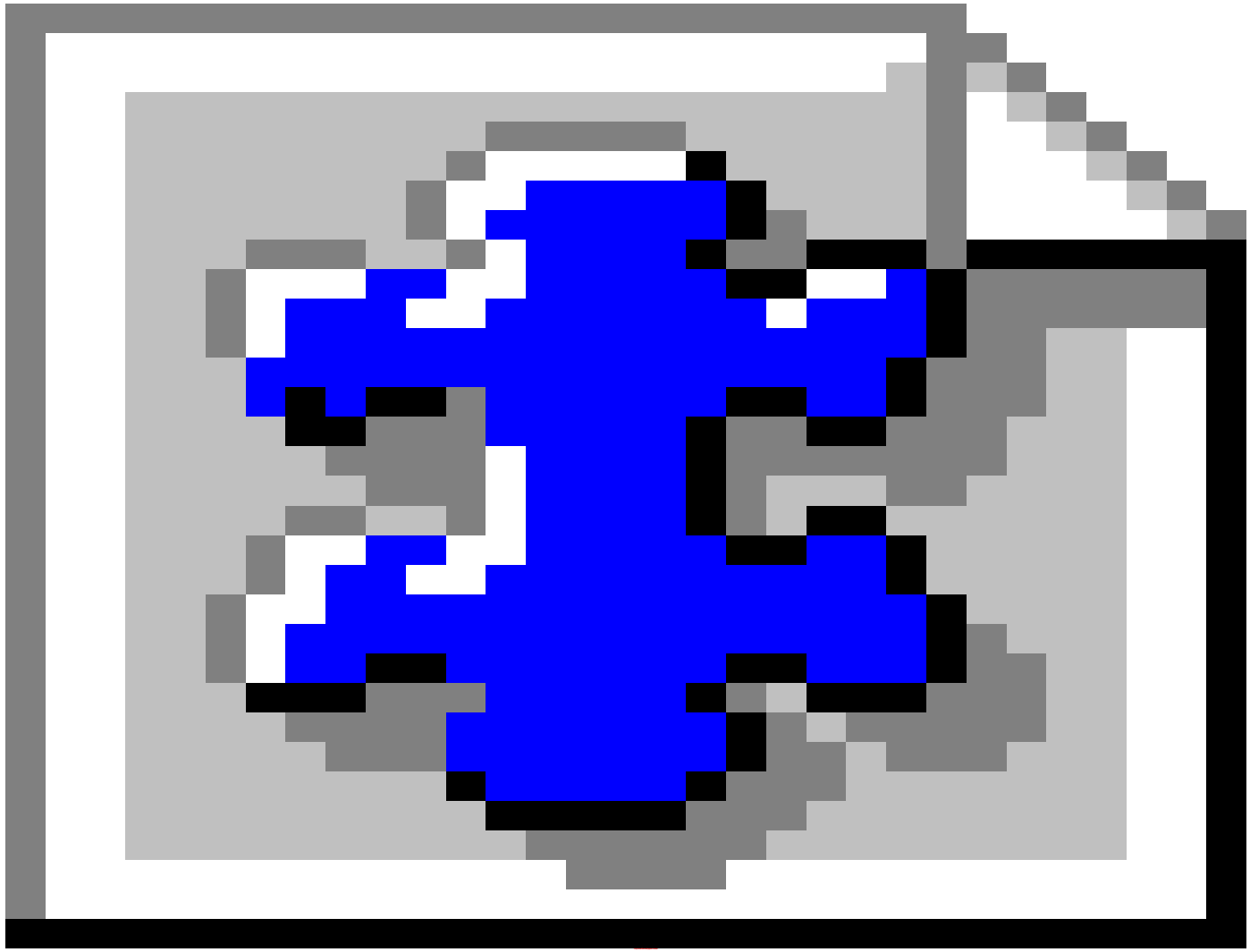
24 blocks (threads), 300 x 300 cells per block (2,160,000 cells)



Laplace Solver

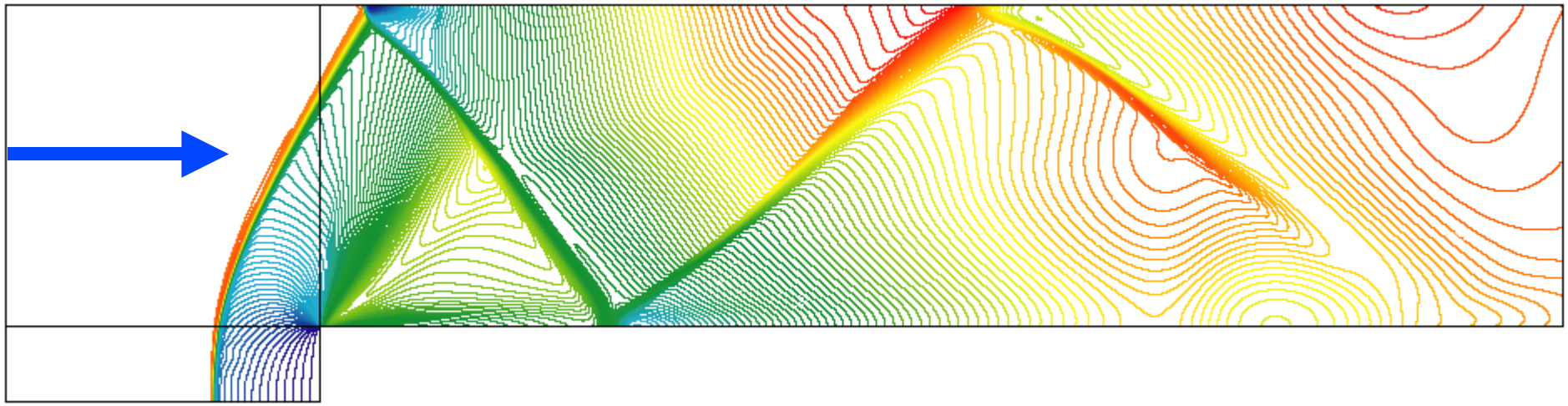


Laplace Solver



JParEuler

- Is an Euler solver plugin for the *JavaGrid*
- Testcase is a Mach 3 Euler flow past a forward facing-step
- The simulations are based on structured multi-block grids



The computation is explicit and first order accurate. Shown is the Mach-number distribution.

JParEuler

number of blocks	number of cells	time in seconds		parallel speedup
		single processor	multi processor (16)	
16	121104	3246,73	541,13	6,00
16	200704	6908,88	1077,20	6,41
16	484416	12905,88	2720,48	4,74
48	118803	2980,93	225,76	13,20
48	202800	5190,54	436,09	11,90
48	480000	12663,30	1162,54	10,89

HP V-Class times for 320 iterations

JParEuler

number of blocks	number of cells	time in seconds		parallel speedup
		single processor	multi processor (4)	
16	121104	852,79	258,26	3,30
16	200704	1571,76	532,74	2,95
16	484416	4277,96	1593,45	2,68
48	118803	756,24	195,95	3,86
48	202800	1409,96	378,61	3,72
48	480000	3892,67	1077,06	3,61

Sun E450 times for 320 iterations

Conclusions

- The parallel efficiency is obtained if a sufficient number of threads and sufficient computational work within a thread can be provided.
- **With the scientific and engineering problems that we have in mind, these requirements are easily satisfied.**
- The execution speed of Java code has increased substantially over the last 2 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 *Don't patch bad code, rewrite it*, the latter rule being the reason for a pure Java flow solver code.

Future Work

- **JavaGrid** will be extended to accommodate both structured and unstructured as well as hybrid general 3D grids to allow complete geometric flexibility.
- Extend the **JavaGrid** parallel layer to work with distributed memory machines (Beowulf class). (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.
  Panel Session: future directions and challenges for Java implementations of numeric-intensive industrial applications published in: Advances in Engineering Software, Elsevier, 31,2000, p.743-751

- Häuser,J., Ludewig, T., Williams, Roy D., Winkelmann, R., Gollnick, T., Brunett, S., Muylaert, J.
  A Test Suite for High Performance Parallel Java published in: Advances in Engineering Software, Elsevier,31, 2000, p.687-696

- Häuser,J., Ludewig, T., Williams, Roy D., 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, Roy D., 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.,
  A Pure Java Parallel Flow Solver, published in: Proceedings of the 37th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 99-0549 Reno, NV, USA, January 11.-14., 1999.

- Häuser J., Williams R.D, Spel M., Muylaert J., ParNSS: ***An Efficient Parallel Navier-Stokes Solver for Complex Geometries***, AIAA 94-2263, AIAA 25th Fluid Dynamics Conference, Colorado Springs, June 1994.