The Ibis e-Science Software Framework

Frank J. Seinstra, Jason Maassen, Niels Drost

Jungle Computing Research & Applications Group
Department of Computer Science
VU University, Amsterdam, The Netherlands

vrije Universiteit amsterdam

Mini-course: General Overview

- 10:00 – 10:20: General Summary
- 10:20 – 12:00: Ibis as ‘Master Key’
- 12:00 – 13:00: Lunch (free)
- 13:00 – 14:00: Ibis as ‘Advanced Master Key’
- 14:00 – 14:30: Ibis as ‘Glue’
- 14:30 – 15:00: Questions + Discussion + Future

Enlighten Your Research 3

- Jan Bot (et al.):
  - First Prize Winner
    - Next Generation Networking for Next Generation Sequencing

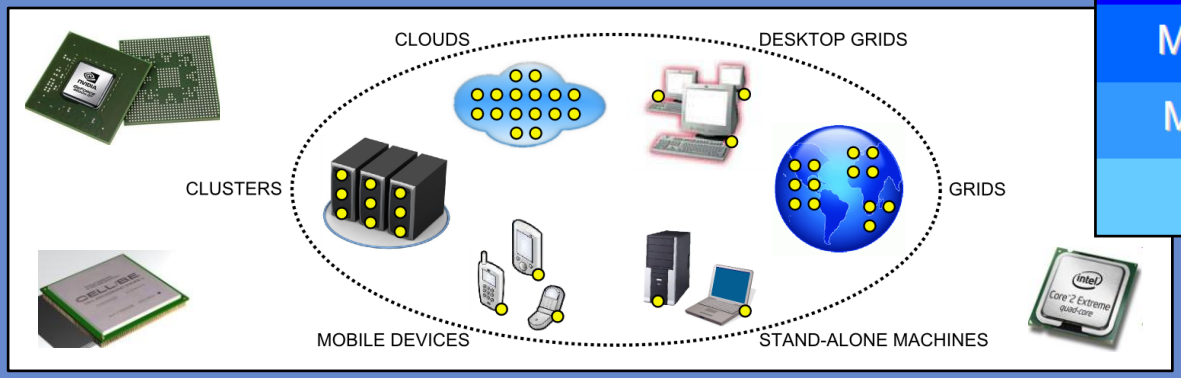
- Ibis Team (+ LU, UU):
  - Sustainability Prize Winner
    - High-Performance Distributed Multi-Model / Multi-Kernel Simulations
- e-Infrastructures
  - By: SURFnet, SARA, BigGrid

General Summary

- Jungle Computing + Domain Examples
- ‘Problem Solving’ vs. ‘System Fighting’
- The Ibis Software Framework
  - Requirements and Tools
- Disclaimers
- The 3 Common Uses of Ibis
  - Ibis as ‘Master Key’, Ibis as ‘Glue’, Ibis as ‘HPC solution’
- General Outline of Today
  - See also: http://www.cs.vu.nl/ibis/tutorial.html
  - See also: email Kees Verstoep ivm DAS-3/DAS-4

Jungle Computing

- ‘Worst case’ computing as required by end-users
  - Distributed
  - Heterogeneous
  - Hierarchical (incl. multi-/many-cores)

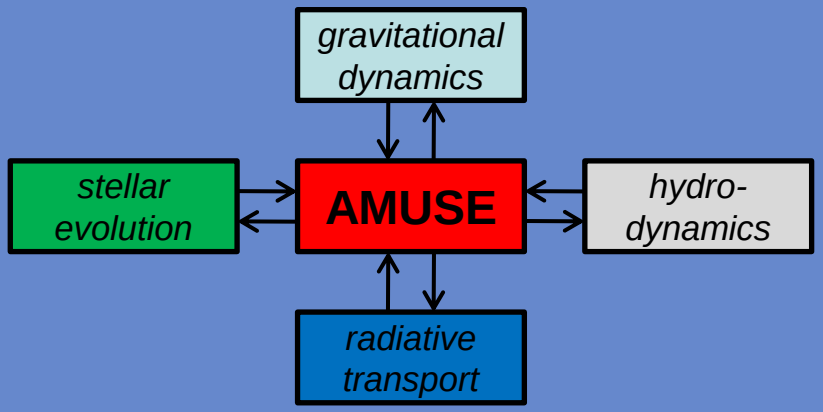


Multi-wide-area (jungle)
Multi-local-area (wide-area)
Multi-node (local-area)
Multi-processor (node)
Multi-core (processor)
Core

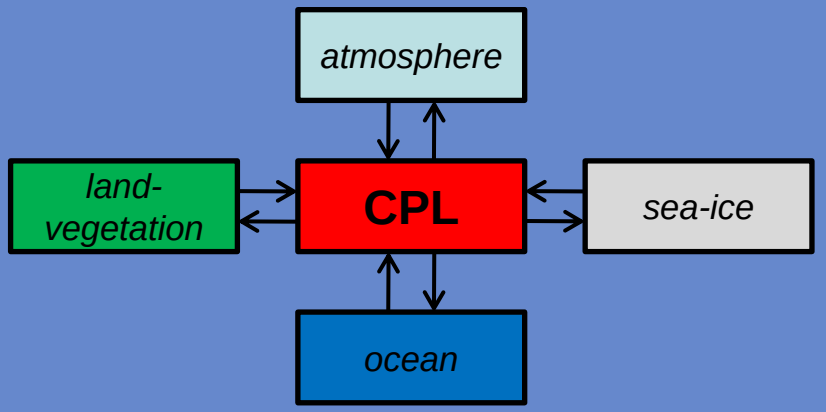
- Note: most users do not need ‘worst case’
  - Ibis aims to apply to any subset

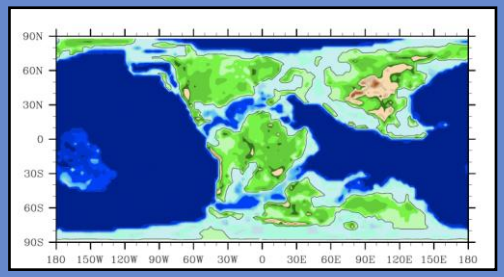
Example Application Domains

- Computational Astrophysics
  (Leiden: AMUSE)



- Climate Modeling
  (Utrecht: CPL/CESM)

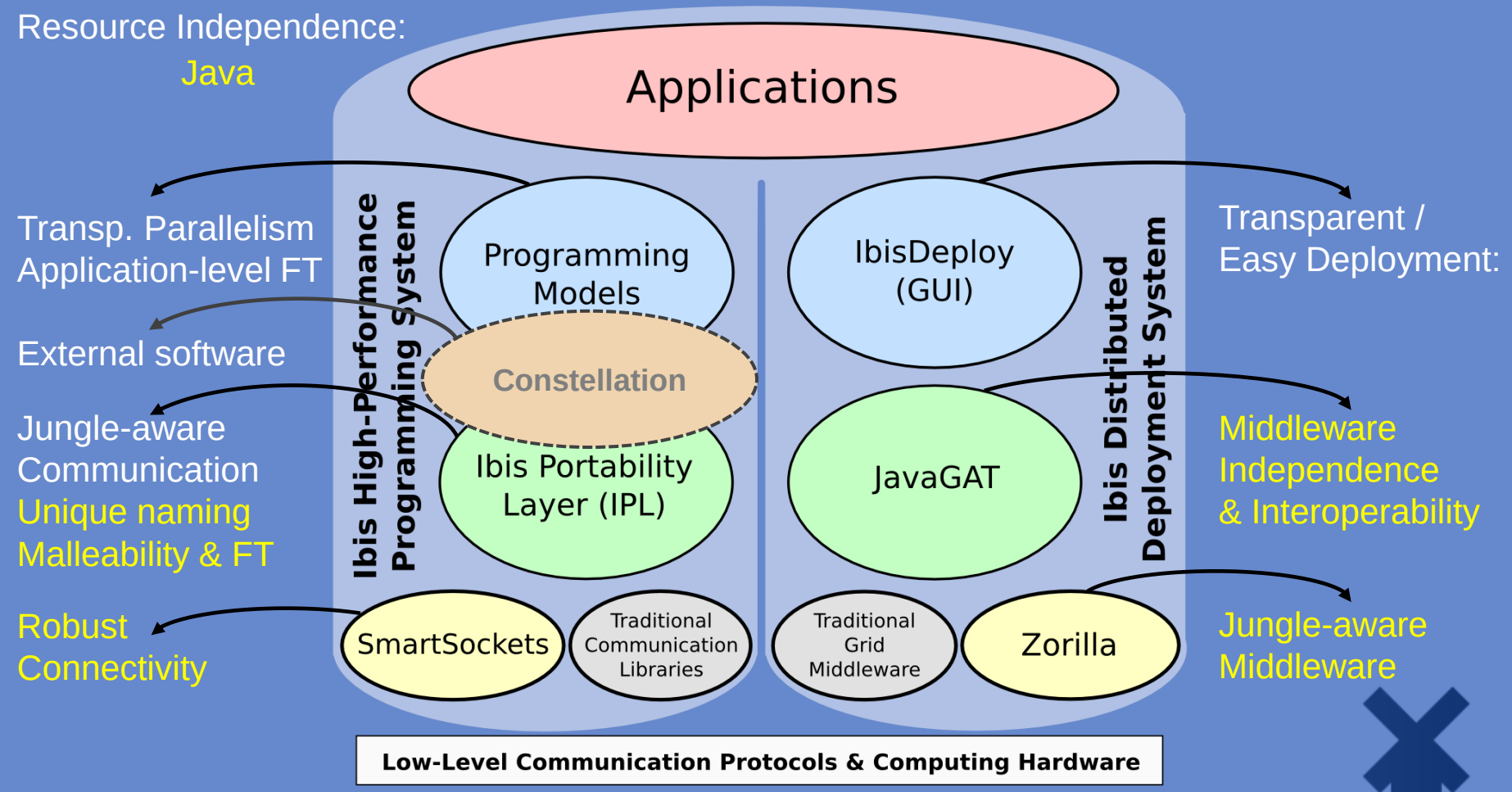




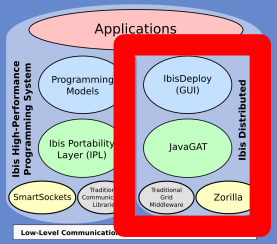
Problem Solving vs. System Fighting

- Jungle Computing for domain scientists?
  - Hardware heterogeneity
  - Middleware heterogeneity
  - Software heterogeneity
    - Kernels in C, MPI, Fortran, Java, CUDA, scripts, …
  - Connectivity problems
    - e.g. firewalls, NAT, …
  - Infrastructure often dynamic, faulty
  - ….
- Need for integrated, user-friendly solution/toolbox
  - Focus on ‘problem solving’, not ‘system fighting’

Ibis Software Stack

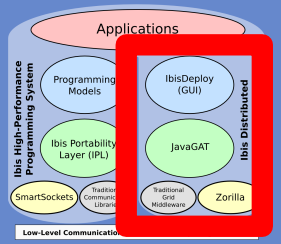


JavaGAT

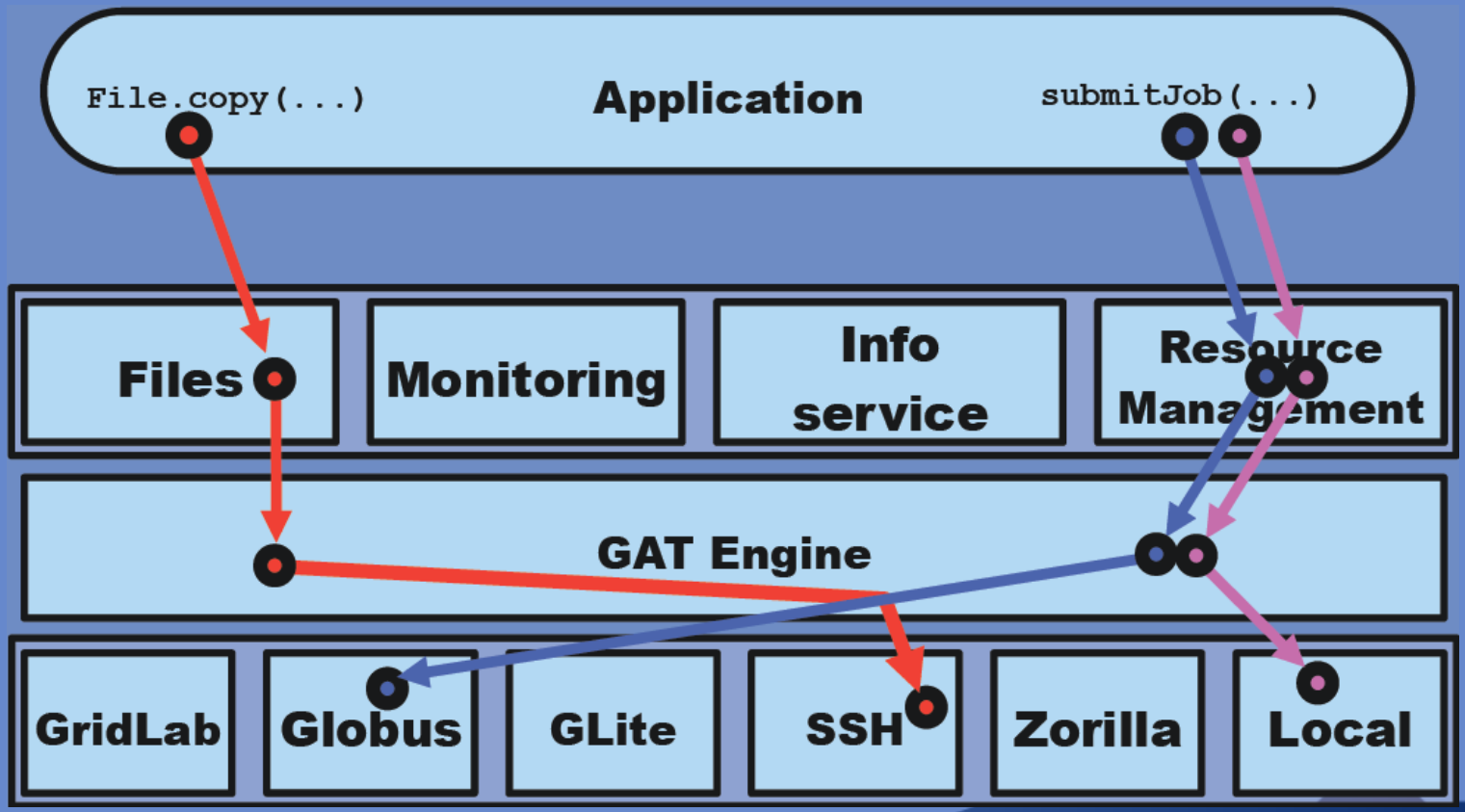


- Java *Grid* Application Toolkit
  - High-level API for developing (*Grid*) applications *independently* of the underlying (*Grid*) middleware
  - Use (*Grid*) services; file cp, resource discovery, job submission, …
  - Overcomes problems, incl:
    - Functionality may not work on all sites, or for all users, …
    - Middleware version differences & complex codes…
- Note: SAGA API standardized by OGF
  - Simple API for *Grid* Applications (a.o. with LSU)
  - SAGA on top of JavaGAT (and v.v.)

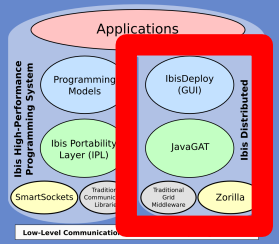
JavaGAT



- 'API' vs. 'Engine' vs. 'Adaptors'



JavaGAT Examples



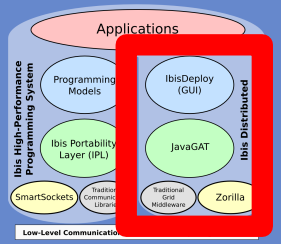

```
import org.gridlab.gat.GAT;
import org.gridlab.gat.URI;

public class Copy {

    public static void main(String[] args) throws Exception {
        GAT.createFile(args[0]).copy(new URI(args[1]));
        GAT.end();
    }
}
```

- Do file copy

- Run Job incl. File Staging

```
public class RunJobWithStaging {
    // USAGE: machine executable input [arguments…] output
    public static void main(String[] args) throws Exception {
        ResourceBroker broker = GAT.createResourceBroker(new URI(args[0]));

        SoftwareDescription sd = new SoftwareDescription();
        sd.setExecutable(args[1]);
        sd.setStdout(GAT.createFile("stdout.txt"));
        sd.setStderr(GAT.createFile("stderr.txt"));

        sd.addPreStagedFile(GAT.createFile(args[2]));
        sd.addPostStagedFile(GAT.createFile(args[args.length - 1]));

        sd.setArguments(getArguments(args));
        Job job = broker.submitJob(new JobDescription(sd));

        do {
            System.out.println("Current state: " + job.getState());
            Thread.sleep(1000);
        } while ((job.getState() != JobState.STOPPED)
                && (job.getState() != JobState.SUBMISSION_ERROR));

        GAT.end();
    }
```

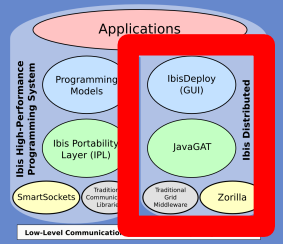
JavaGAT Examples

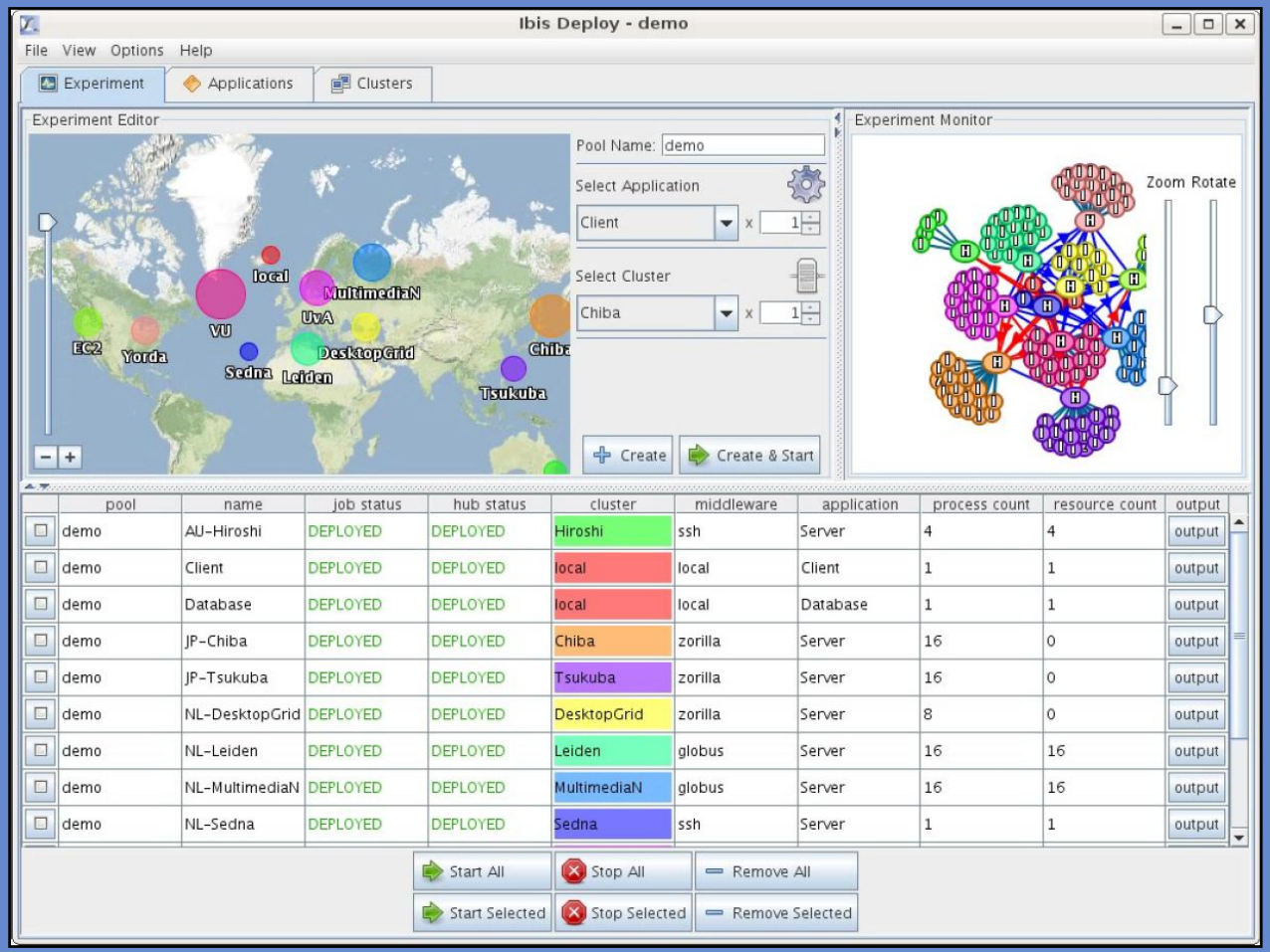


- Simple Task Farming
- Go multi-site:
  - Multiple brokers

```
public class SimpleTaskFarming {
    //USAGE: MACHINE EXECUTABLE INPUT_DIR [ARGUMENTS ...] OUTPUT_DIR
    public static void main(String[] args) throws Exception {
        ResourceBroker broker = GAT.createResourceBroker(new URI(args[0]));

        String executable = args[1];
        String inputdir = args[2];
        String outputdir = args[args.length - 1];

        String[] inputs = listInputs(inputdir, ".jpg");

        Job[] jobs = new Job[inputs.length];

        for (int i = 0; i < inputs.length; i++) {
            SoftwareDescription sd = new SoftwareDescription();
            sd.setExecutable(executable);

            File input = GAT.createFile(inputdir + File.separator + inputs[i]);
            sd.addPreStagedFile(input);
            File output = GAT.createFile(outputdir + File.separator + "out-" + input.getName());
            sd.addPostStagedFile(GAT.createFile(output.getName()), output);

            sd.setStdout(GAT.createFile("stdout-" + i + ".txt"));
            sd.setStderr(GAT.createFile("stderr-" + i + ".txt"));

            // Set the arguments and submit the job.
            sd.setArguments(prepareArguments(input.getName(), getArguments(args),
                    output.getName()));

            jobs[i] = broker.submitJob(new JobDescription(sd));
        }

        waitUntilFinished(jobs);
        GAT.end();
    }
```

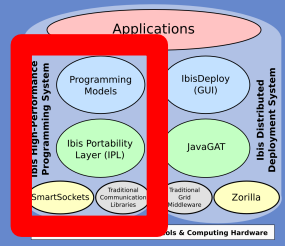
IbisDeploy





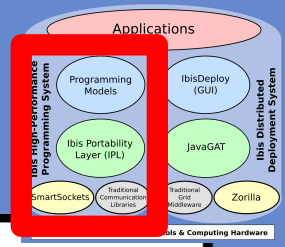
pool	name	job status	hub status	cluster	middleware	application	process count	resource count	output
demo	AU-Hiroshi	DEPLOYED	DEPLOYED	Hiroshi	ssh	Server	4	4	output
demo	Client	DEPLOYED	DEPLOYED	local	local	Client	1	1	output
demo	Database	DEPLOYED	DEPLOYED	local	local	Database	1	1	output
demo	JP-Chiba	DEPLOYED	DEPLOYED	Chiba	zorilla	Server	16	0	output
demo	JP-Tsukuba	DEPLOYED	DEPLOYED	Tsukuba	zorilla	Server	16	0	output
demo	NL-DesktopGrid	DEPLOYED	DEPLOYED	DesktopGrid	zorilla	Server	8	0	output
demo	NL-Leiden	DEPLOYED	DEPLOYED	Leiden	globus	Server	16	16	output
demo	NL-MultimediaN	DEPLOYED	DEPLOYED	MultimediaN	globus	Server	16	16	output
demo	NL-Sedna	DEPLOYED	DEPLOYED	Sedna	ssh	Server	1	1	output

Ibis Portability Layer (IPL)



- Java-centric ‘run-anywhere’ communication library
  - Unidirectional pipes: send ports + receive ports (types must match)
  - Ports communicate using ‘messages’ (allows streaming)
  - Unique naming: IbisIdentifier
- Supports fault-tolerance and malleability
  - Resource tracking (JEL model)
  - Separate registry component (server; various implementations)
- Efficient
  - Highly optimized object serialization
  - Can use optimized native libraries (e.g. MPI, MX)

IPL Examples



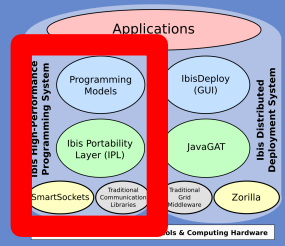

```
PilotJob() throws Exception {
    ibis = IbisFactory.createIbis(Shared.ibisCapabilities, null,
            Shared.portTypeServer, Shared.portTypeSlave);

    IbisIdentifier server = ibis.registry().getElectionResult("JobServer");

    rp = ibis.createReceivePort(Shared.portTypeSlave, "receiver");
    rp.enableConnections();

    sp = ibis.createSendPort(Shared.portTypeServer);
    sp.connect(server, "receiver");
}
```

pilot1.png

```
package

import

public
    Ibi
    Rec
    Sen

    PilotJob() throws Exception {

    Job get

    void ru

    public
}
```

```
static PortType portTypeSlave = new PortType(
        PortType.COMMUNICATION_RELIABLE, PortType.SERIALIZATION_OBJECT,
        PortType.RECEIVE_EXPLICIT, PortType.CONNECTION_ONE_TO_ONE);

static PortType portTypeServer = new PortType(
        PortType.COMMUNICATION_RELIABLE, PortType.SERIALIZATION_OBJECT,
        PortType.RECEIVE_EXPLICIT, PortType.CONNECTION_MANY_TO_ONE);

static IbisCapabilities ibisCapabilities = new IbisCapabilities(
        IbisCapabilities.ELECTIONS_STRICT);
```

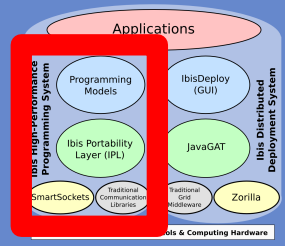
IPL Examples



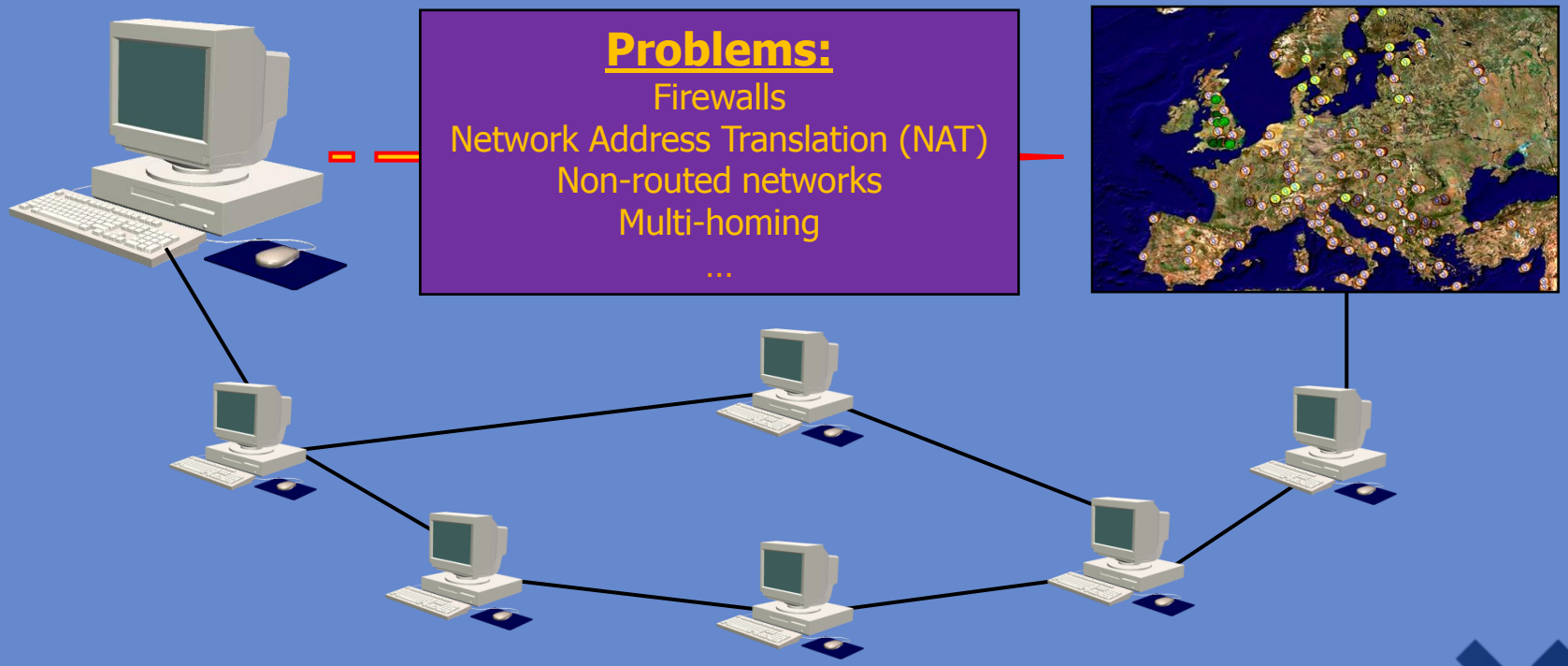

```
package tutorial20.glu

import ibis.ipl.Ibis;

public class PilotJob
    Ibis ibis;
    ReceivePort rp;
    SendPort sp;

    PilotJob() throws Exception {

    Job getWork(Result previousResult) throws Exception {

    void run() throws Exception {

    public static void main(String[] args) {
}
```

```
Job getWork(Result previousResult) throws Exception {
    WriteMessage wm = sp.newMessage();
    wm.writeObject(previousResult);
    wm.finish();

    ReadMessage rm = rp.receive();
    Job job = (Job) rm.readObject();
    rm.finish();
    return job;
}
```

SmartSockets

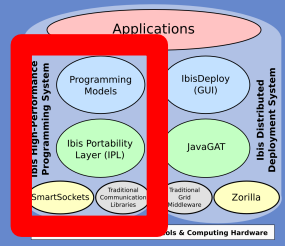


- Robust connection setup

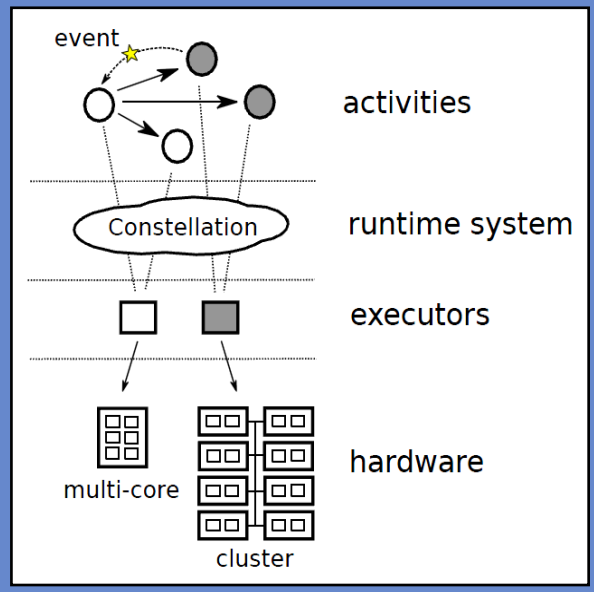


- Always connection in 30 different scenarios

“The Future of Ibis”



- Ibis/Constellation:
  - Generalized programming framework for ‘all’ Jungle Computing applications
  - Automatically maps *any* application activity (task) onto *any* appropriate executor (HW)
  - By way of ‘contexts’, for example:
    - Activity's context: “I need a GPU”
    - Executor’s context: “I represent a GPU”



- Note:
  - Activities may represent any type of task:
    - Also legacy codes, scripts, 3rd party software, …

Disclaimers

- This tutorial only ‘scratches the surface’
  - Not incl. a.o. Ibis Programming Models, Ibis tomorrow & beyond, …
- Ibis not the-be-all-and-end-all solution
  - Modular toolbox; sometimes low-level; development continues
- Ibis scope constantly re-defined
  - We continue to learn with new users / use cases
- Outreach to Application Domains
  - We’re happy to support users, but must do research primarily
  - Future: through NLeSC, or…?
  - Roadmap to ‘Ibis Coding Community’

3 Common Uses of Ibis

- Ibis as ‘Master Key’
  - Use JavaGAT / IbisDeploy to access ‘any’ system
  - Middleware independence, simple, portable
- Ibis as ‘Glue’
  - Use IPL + SmartSockets – mostly for wide-area communication
  - Link up ‘any’ task in ‘any’ language/tool; robust connection setup
- Ibis as ‘HPC Solution’
  - Use Ibis as replacement for e.g. C++/MPI (req. re-implementation)
  - High-level programming models; many properties
  - *Not* discussed in this course
- With each use separately or combined:
  - Fully functional solution possible, so: not Java only!

General Outline of Today

- Hands-on: Ibis as ‘Master Key’ (Niels)
  - Basic use of JavaGAT + task farming
  - Integrated requests: specific use of existing systems / middleware
- Hands-on: Ibis as ‘Advanced Master Key’ (Jason)
  - Workflow using JavaGAT (requested by NBIC, low-granularity)
  - Bag-of-tasks (pilot job alternative, requested by SARA)
- Hands-on: Ibis as ‘Glue’ (Jason / Niels)
  - Pilot Job framework with IbisDeploy
  - Integrated requests: use of scripts for added flexibility
- Discussion (all), a.o.:
  - Use of Pilot Jobs vs submitting large nr. of jobs
  - Future support / collaboration Ibis team

End of Summary

www.cs.vu.nl/ibis/