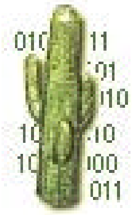
Distributed Supercomputing in Java

Henri Bal

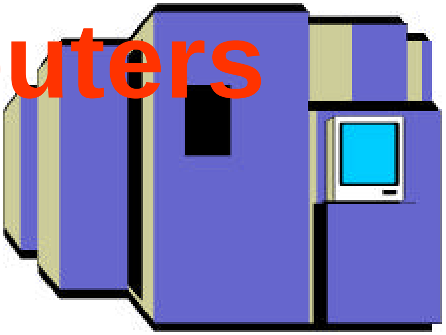
Vrije Universiteit Amsterdam

Keynote talk 11-th Euromicro Conference on Parallel, Distributed and Network based Processing, Genoa, Italy, (February, 5-7, 2003)

Distributed supercomputing

- Parallel processing on geographically distributed computing systems (grids)
- Examples:
  - SETI@home ( ), RSA-155, Entropia, Cactus 
- Currently limited to trivially parallel applications
- Our goals:
  - Generalize this to more HPC applications
  - Provide high-level programming support

Grids versus supercomputers



- Performance/scalability
  - Speedups on geographically distributed systems?
- Heterogeneity
  - Different types of processors, operating systems, etc.
  - Different networks (Ethernet, Myrinet, WANs)
- General grid issues
  - Resource management, co-allocation, firewalls, security, monitoring, authorization, accounting, ...

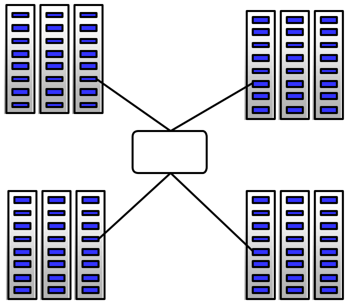
Our approach

- Performance/scalability
  - Exploit hierarchical structure of grids (Albatross project)

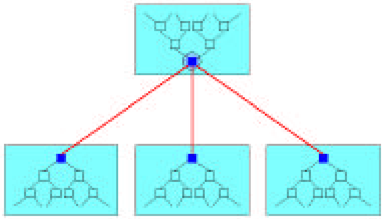
- Heterogeneity
  - Use Java + JVM (Java Virtual Machine) technology

- General grid issues
  - Import knowledge from elsewhere (GGF, GridLab)

Speedups on a grid?



- Grids usually are hierarchical
  - Collections of clusters, supercomputers
  - Fast local links, slow wide-area links
- Can optimize algorithms to exploit this hierarchy
  - Message combining + latency hiding on wide-area links
  - Collective operations for wide-area systems
  - Load balancing



- Successful for many applications
  - Did many experiments on a homogeneous wide-area test bed (DAS) [HPCA 1999, IEEE TPDS 2002]

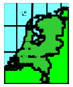
The Ibis system

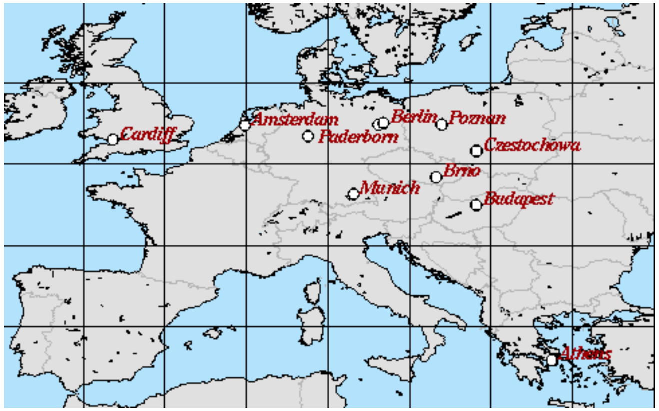
- High-level & efficient programming support for distributed supercomputing on heterogeneous grids
- Use Java-centric approach + JVM technology
  - Inherently more portable than native compilation

    “*Write once, run everywhere* ”
  - Requires entire system to be written in Java
- Use special-case (native) optimizations on demand

Outline

- Programming support
- Highly portable & efficient implementation
- Experiences on DAS-2 and EC GridLab testbeds

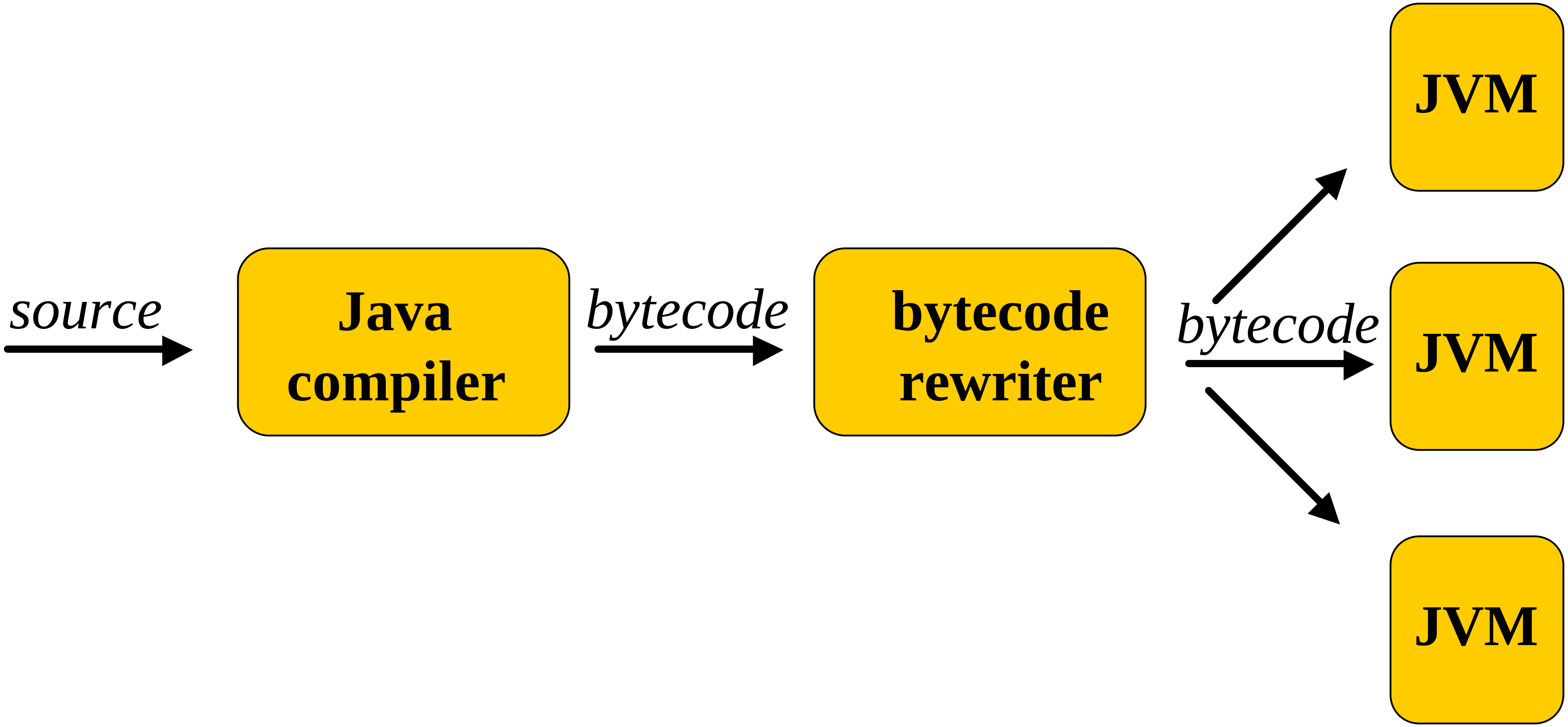




Ibis programming support

- Ibis provides
  - Remote Method Invocation (RMI)
  - Replicated objects (RepMI) - as in Orca
  - Group/collective communication (GMI) - as in MPI
  - Divide & conquer (Satin) - as in Cilk
- All integrated in a clean, object-oriented way into Java, using special “marker” interfaces
  - Invoking native library (e.g. MPI) would give up Java’s “run everywhere” portability

Compiling Ibis programs



source

Java compiler

bytecode

bytecode rewriter

bytecode

JVM

JVM

JVM

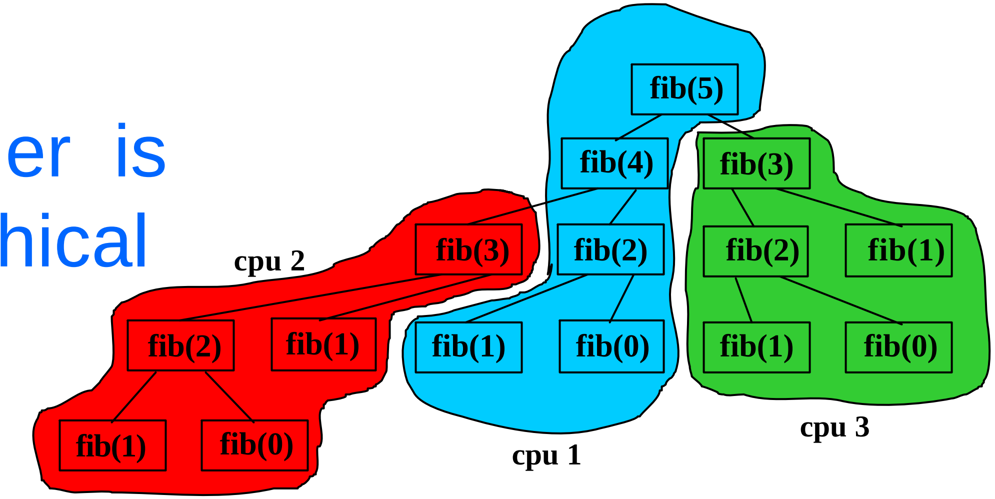
GMI (group communication)

- Generalizes Remote Method Invocation
  - Modify how invocations & results are handled
  - Invoke multiple objects, combine/gather results, etc.
  - Expresses many forms of group communication

```
interface Example extends GroupInterface {
       public int get() throws ...;   ….
}
Example e =  ;                  // get group stub
m = findMethod(”int get()", e); // method to configure
m.useGroupInvocation();  // get() will be multicast
m.useCombineResult(…);   //results are combined
int result = e.get();    // example invocation
```

Divide-and-conquer parallelism

- Divide-and-conquer is inherently hierarchical



- Satin
  - Cilk-like primitives (spawn/sync)
- New load balancing algorithm
  - Cluster-aware random work stealing [PPoPP'01]

Example

```
interface FibInter {
      public int fib(long n);
}

class Fib implements FibInter {
   int fib (int n) {
      if (n < 2) return n;
      return fib(n-1) + fib(n-2);
   }
}
```

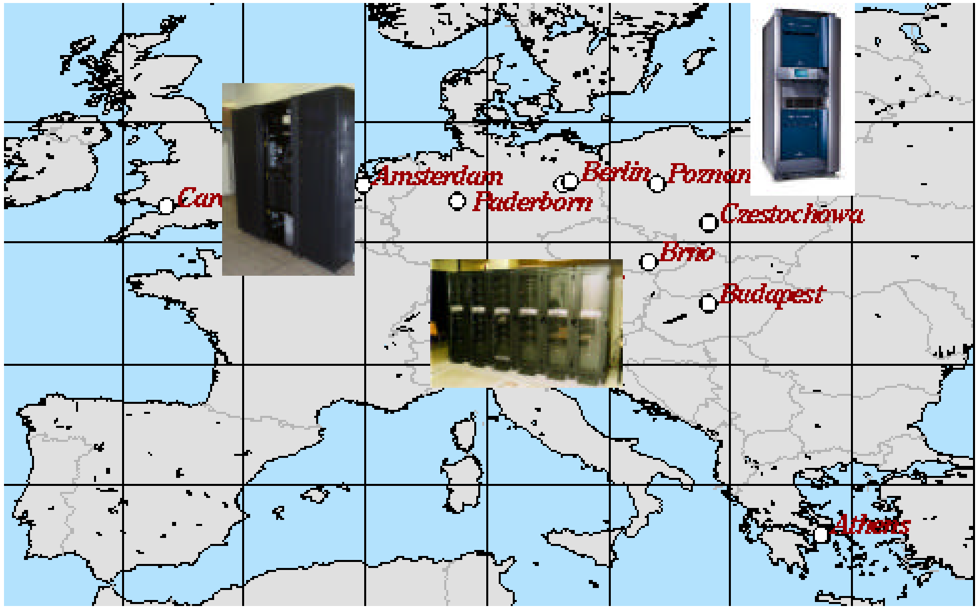
Single-threaded Java

Example

```
interface FibInter
   extends ibis.satin.Spawnable {
      public int fib(long n);
}

class Fib
  extends ibis.satin.SatinObject
  implements FibInter {
   public int fib (int n) {
      if (n < 2) return n;
      int x = fib (n - 1);
      int y = fib (n - 2);
      sync();
      return x + y;
   }
}
```

Java + divide&conquer

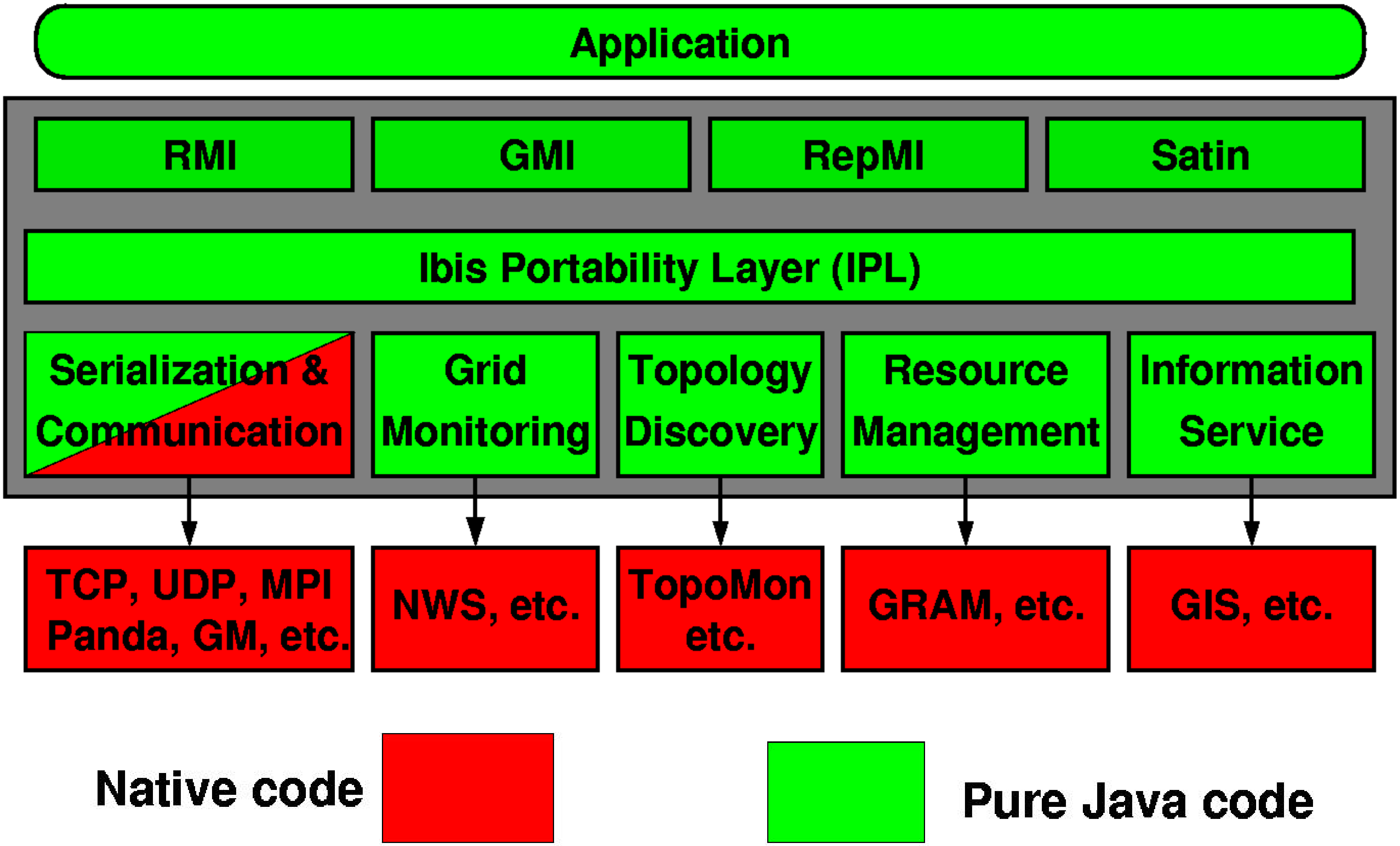


GridLab testbed

Ibis implementation

- Want to exploit Java’s “run everywhere” property, **but**
  - That requires 100% pure Java implementation, **no** single line of native code
  - Hard to use native communication (e.g. Myrinet) or native compiler/runtime system
- Ibis approach:
  - Reasonably efficient pure Java solution (for any JVM)
  - Optimized solutions with native code for special cases

Ibis design



Application

RMI

GMI

RepMI

Satin

Ibis Portability Layer (IPL)

Serialization & Communication

Grid Monitoring

Topology Discovery

Resource Management

Information Service

TCP, UDP, MPI Panda, GM, etc.

NWS, etc.

TopoMon etc.

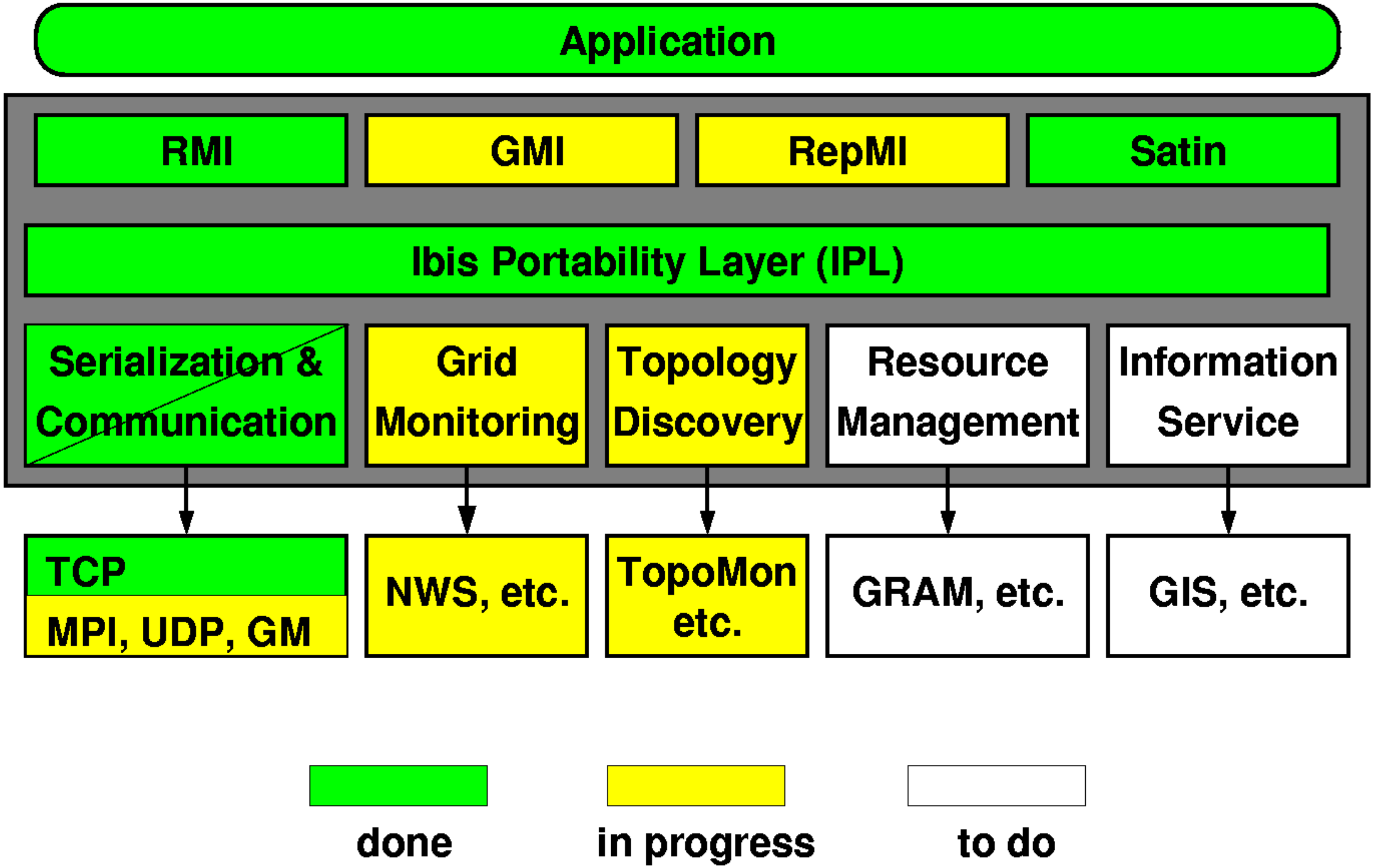
GRAM, etc.

GIS, etc.

Native code 🟥

🟩 Pure Java code

Current status



Application

RMI | GMI | RepMI | Satin

Ibis Portability Layer (IPL)

Serialization & Communication | Grid Monitoring | Topology Discovery | Resource Management | Information Service

TCP / MPI, UDP, GM | NWS, etc. | TopoMon etc. | GRAM, etc. | GIS, etc.

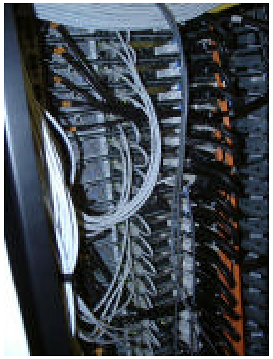
🟩 done

🟨 in progress

□ to do

Challenges

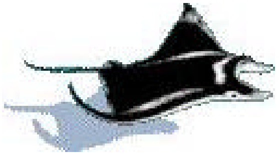
- How to make the system flexible enough
  - Run seamlessly on different hardware / protocols
- Make the pure-Java solution efficient enough
  - Need fast local communication even for grid applications
- Special-case optimizations



Flexibility

- Support different communication substrates
- IPL just defines an *interface* between high-level programming systems and underlying platforms
- Higher levels can ask IPL to load different *implementations* at runtime, using class loading
  - Eg FIFO ordering, reliable communication

Fast communication in pure Java



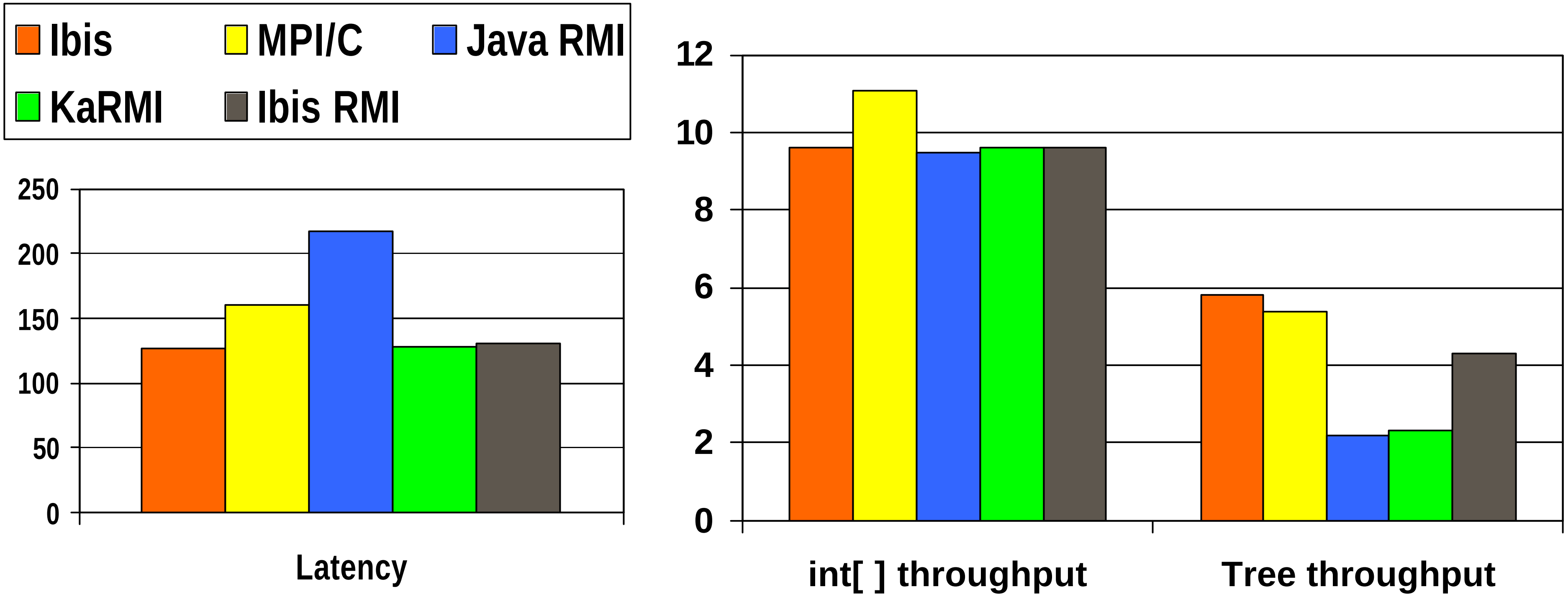
- Manta system [ACM TOPLAS Nov. 2001]
  - RMI at RPC speed, but using native compiler & RTS
- Ibis does similar optimizations, but in pure Java
  - Compiler-generated serialization at bytecode level

    5-9x faster than using runtime type inspection
  - Reduce copying overhead

    Zero-copy native implementation for primitive arrays

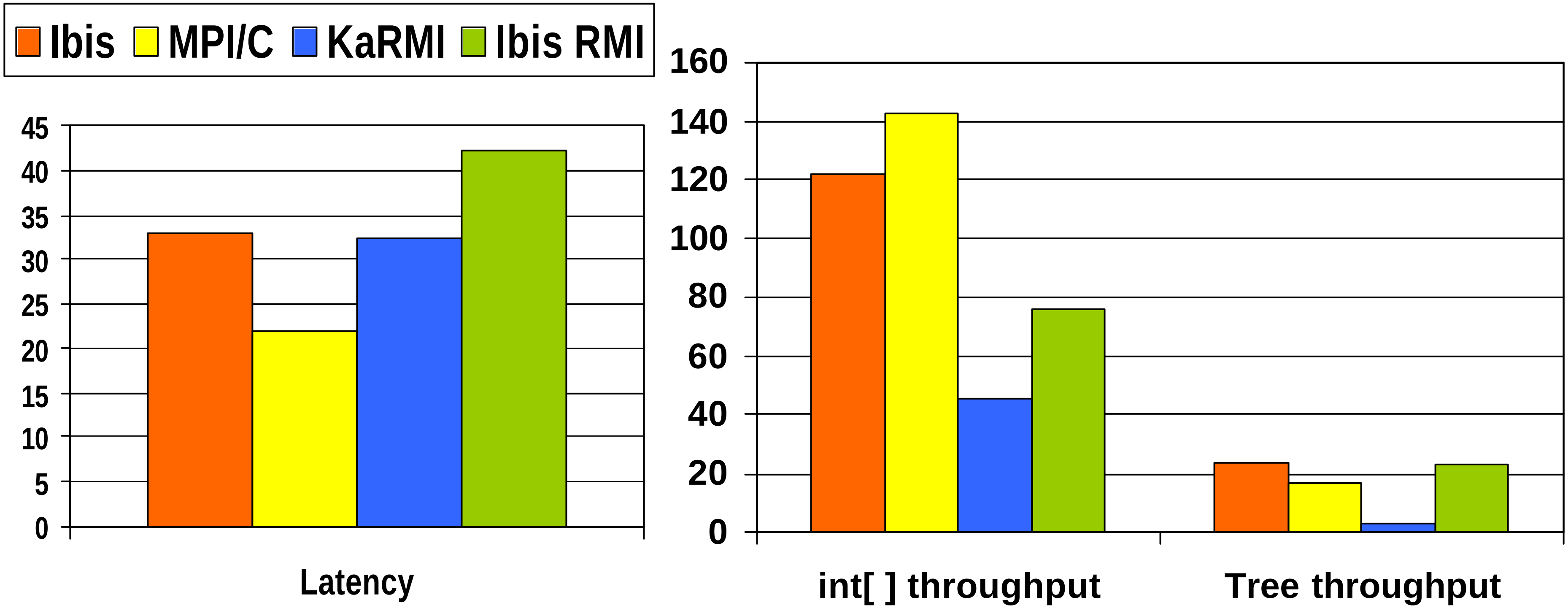
    Pure-Java requires type-conversion (=copy) to bytes

Communication performance on Fast Ethernet



Latency (µs) & throughput (MB/s), measured on
1 GHz Pentium-IIIs (KaRMI = Karlsruhe RMI)

Communication performance on Myrinet

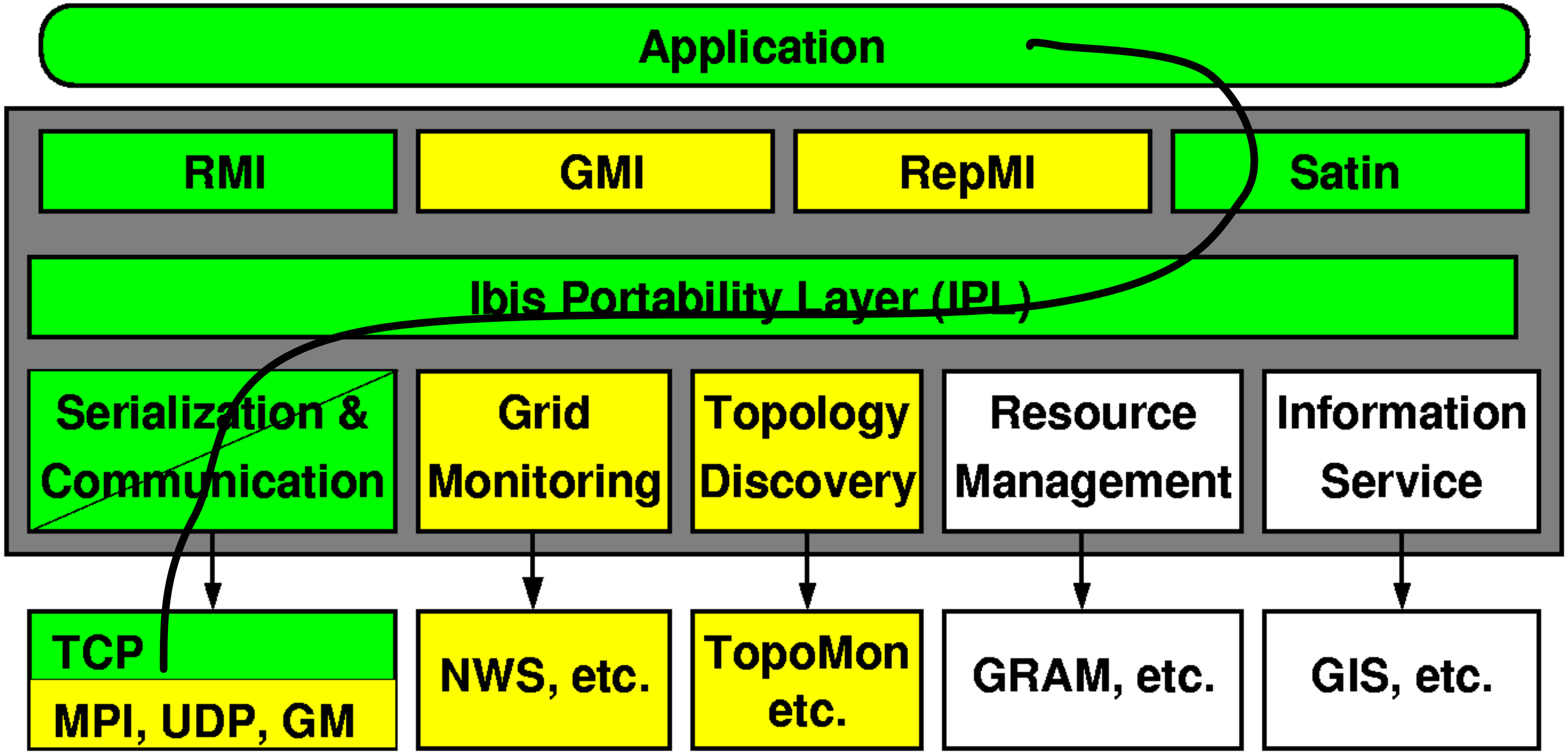


Early Grid experiences with Ibis

- Using Satin divide-and-conquer system
  - Implemented with Ibis in pure Java, using TCP/IP

- Application measurements on
  - DAS-2 (homogeneous)
  - Testbed from EC GridLab project (heterogeneous)

Layers involved



Application

RMI

GMI

RepMI

Satin

Ibis Portability Layer (IPL)

Serialization & Communication

Grid Monitoring

Topology Discovery

Resource Management

Information Service

TCP

MPI, UDP, GM

NWS, etc.

TopoMon etc.

GRAM, etc.

GIS, etc.

🟩 done

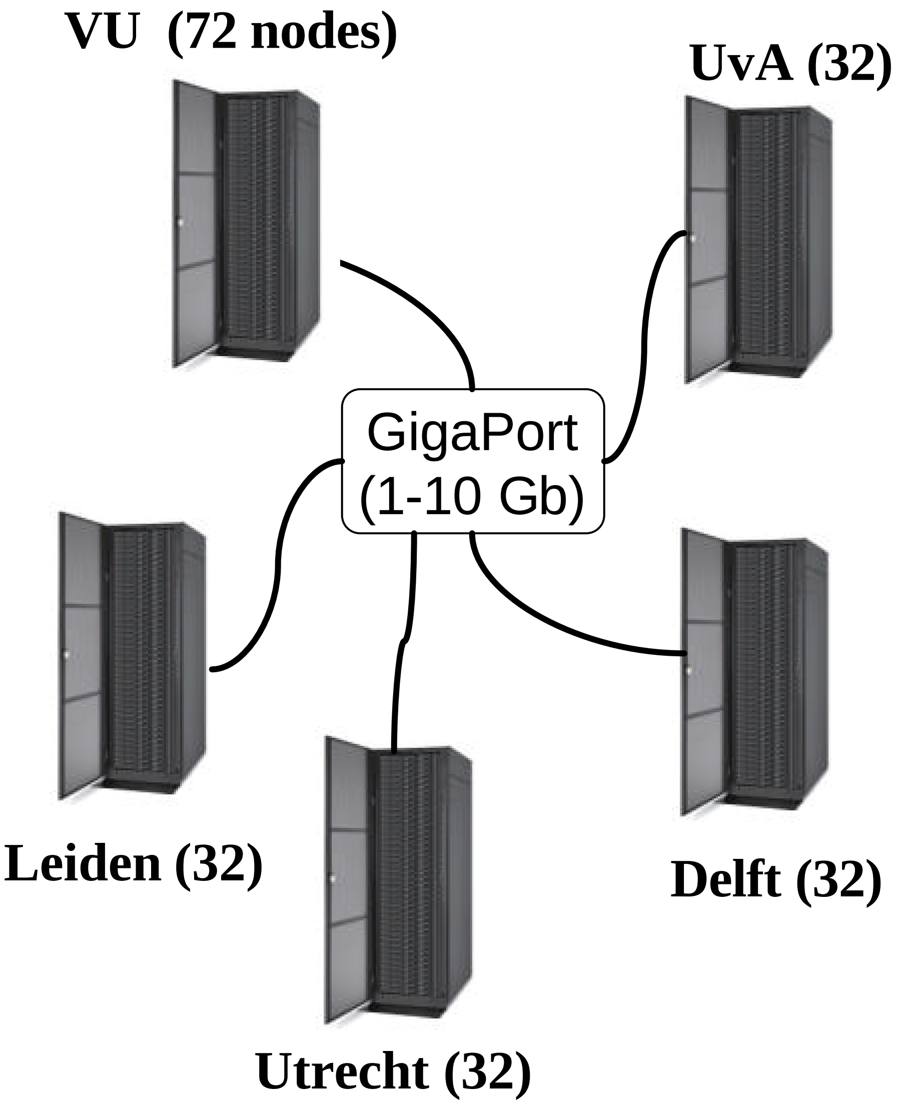
🟨 in progress

⬜ to do

Distributed ASCI Supercomputer (DAS) 2

Node configuration

Dual 1 GHz Pentium-III
>= 1 GB memory
Myrinet
Linux



VU (72 nodes)

UvA (32)

GigaPort
(1-10 Gb)

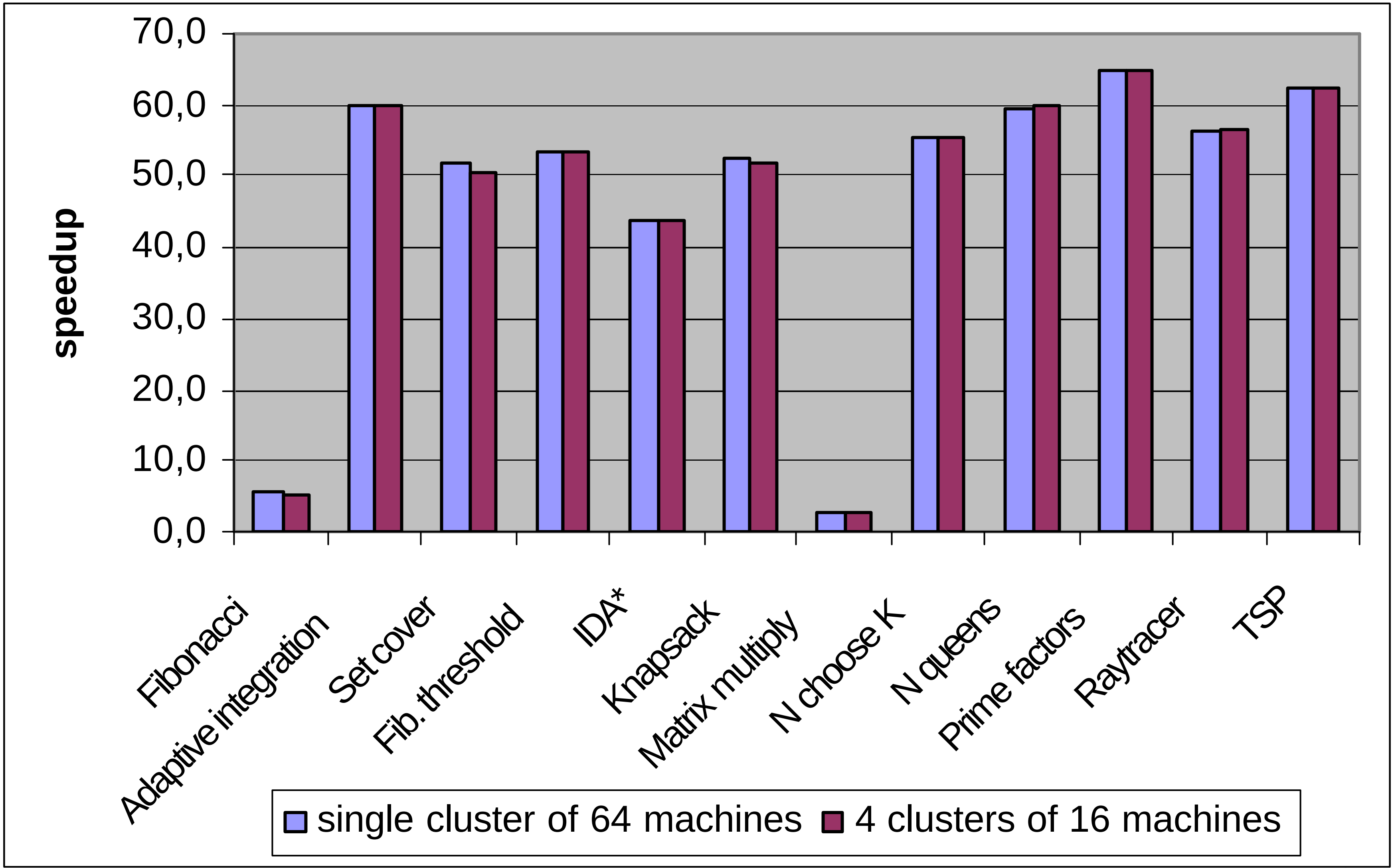
Leiden (32)

Delft (32)

Utrecht (32)

Advanced School for Computing and Imaging

Satin on wide-area DAS-2



Satin on GridLab: Theory versus practice

- No support for co-allocation yet (done manually)
- Firewall problems everywhere
  - Currently: use a range of open ports
  - Future: use multiplexing & ssh
- Java indeed runs everywhere

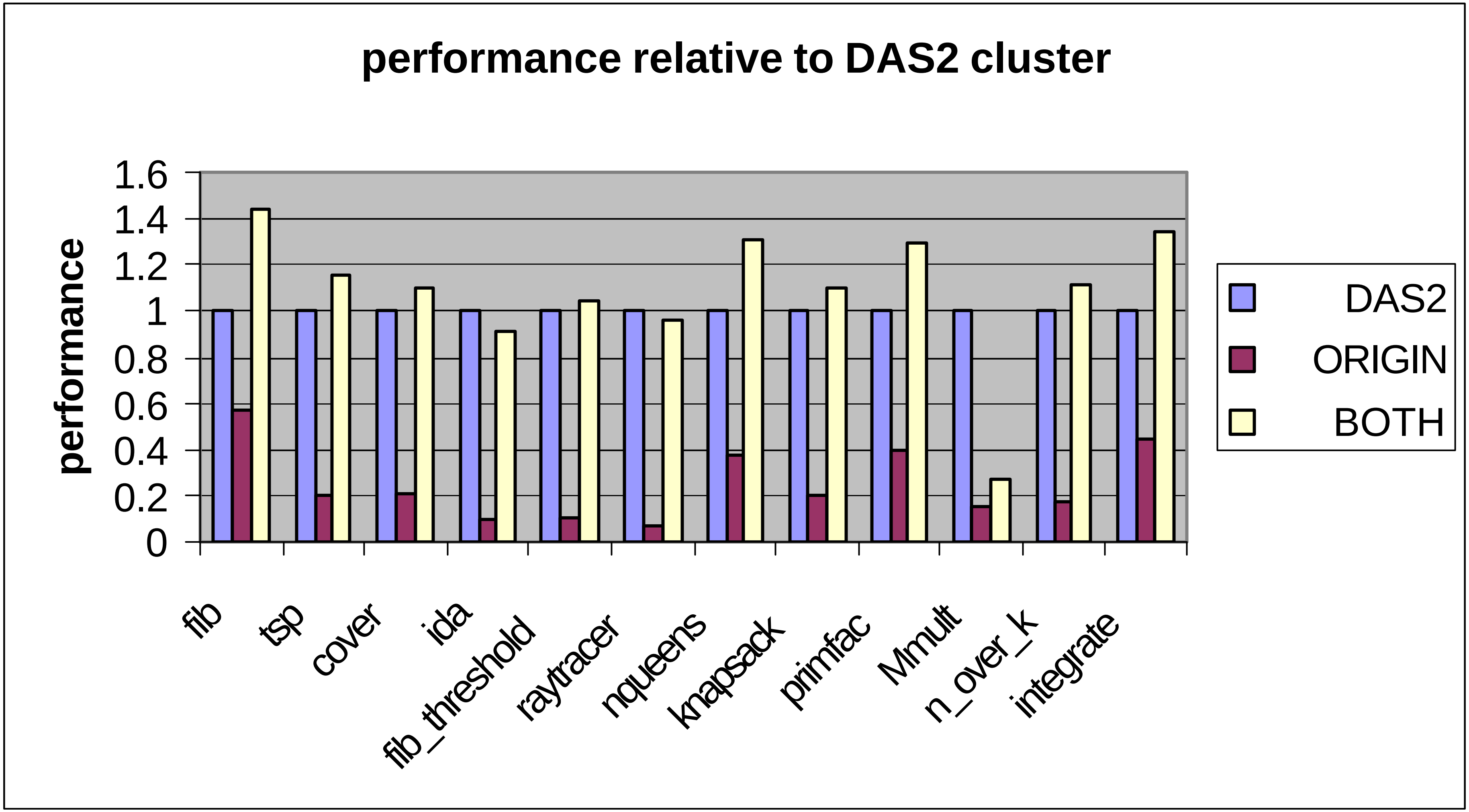
  modulo bugs in (old) JVMs
  - IBM 1.3.1 JIT: bug in versioning mechanism
  - Origin2000 JDK: bug in thread synchronization

But it works ...

- Satin/Ibis program was run simultaneously on

Type	OS	CPU	Location
Cluster	Linux	Pentium-3	VU Amsterdam
Server	Solaris	Sparc	VU Amsterdam
Origin 2000	Irix64	MIPS	AEI Potsdam
Cluster	Linux	Pentium-3	PSNC Poznan

Performance on GridLab



16-node DAS-2 cluster + 16-node Origin

Summary

- Ibis: a programming environment for grids
  - RMI, group communication, divide&conquer
- Portable
  - Using Java’s “write once, run everywhere” property
- Efficient
  - Reasonably efficient “run everywhere” solution
  - Optimized solutions for special cases
- Experience with prototype system on GridLab

Acknowledgements

Rob van Nieuwpoort

Jason Maassen

Thilo Kielmann

Rutger Hofman

Ceriel Jacobs

Gosia Wrzesinska

Olivier Aumage

Kees Verstoep

web site: www.cs.vu.nl/ibis

vrije Universiteit