From:
CCGrid 2008 European Projects Showcase

The Ibis Project:

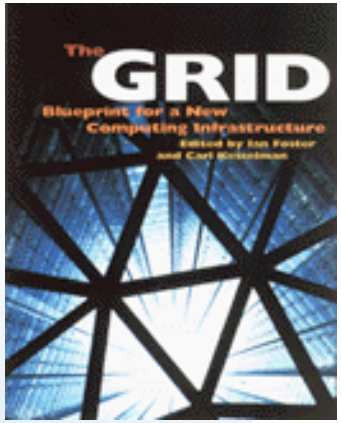
Simplifying Grid Programming & Deployment

Henri Bal, Jason Maassen, Rob van Nieuwpoort,
Thilo Kielmann, Niels Drost, Ceriel Jacobs,
Kees van Reeuwijk, Frank Seinstra,
Roelof Kemp, Kees Verstoep

Vrije Universiteit Amsterdam

vrije Universiteit

vl·e



The 'Promise of the Grid'

Efficient and transparent (i.e. easy-to-use) wall-socket computing over a distributed set of resources [Sunderam ICCS'2004, based on Foster/Kesselman]

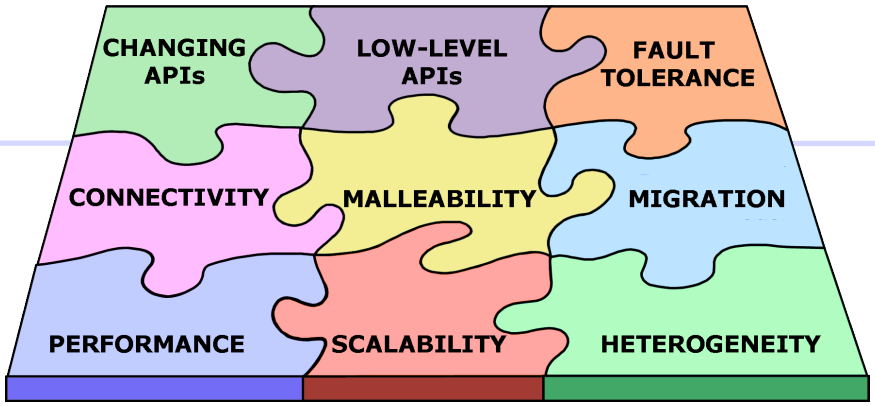
ibis

vrije Universiteit

Reality: 'Problems of the Grid'

- Performance & scalability
- Heterogeneous
- Low-level & changing programming interfaces
- Connectivity issues
- Fault tolerance
- Malleability

User



Wide-Area Grid Systems

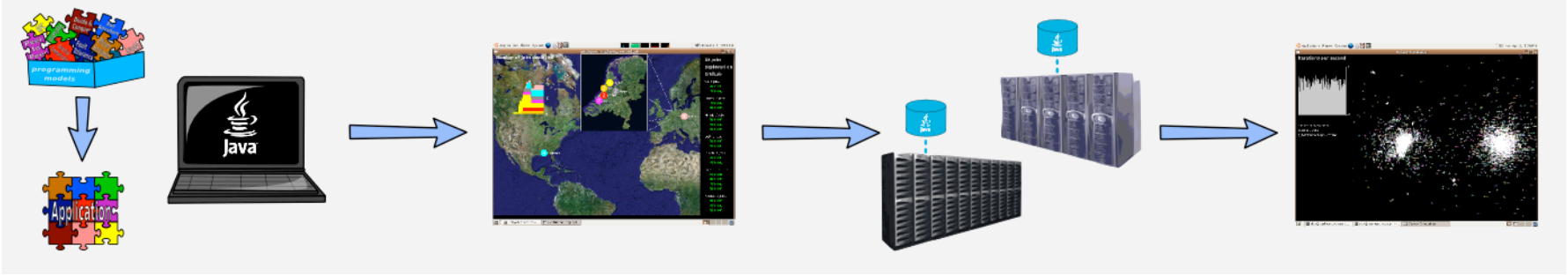
- writing & deploying grid applications is hard

ibis

vrije Universiteit

The Ibis Project

- Goal:
  - drastically simplify grid programming/ deployment
  - ``Grids as promised''

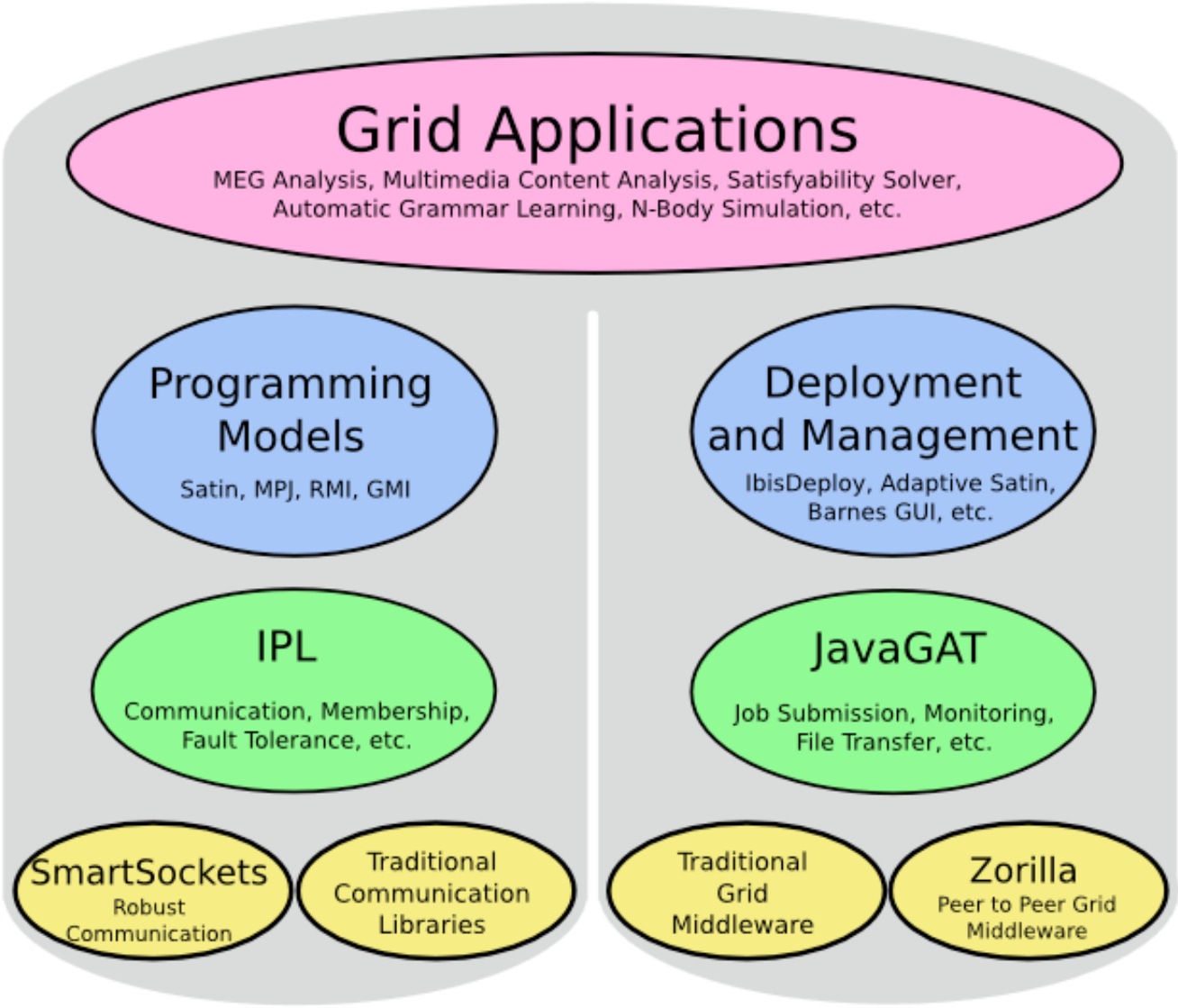


Approach

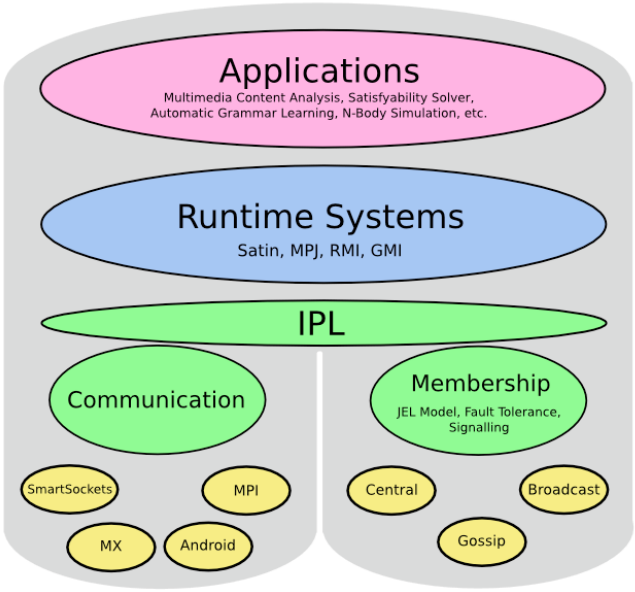
- Minimal assumptions about execution environment
  - Virtual Machines (Java) deal with heterogeneity
- Use middleware-independent APIs
- Different programming abstractions
- Designed for dynamic/unhelpful grid environments
- Modular and flexible: can replace Ibis components by different implementations

vrije Universiteit

Global picture



Grid programming



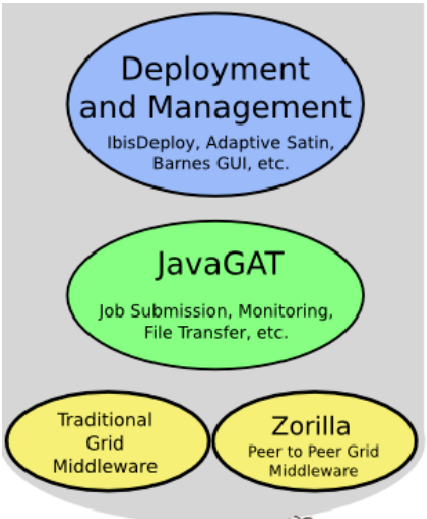
- Programming models:
  - Message passing (RMI, MPJ)
  - Divide-and-conquer (Satin)
  - Stream processing (Maestro)
- IPL (Ibis Portability Layer)
  - Java-centric “run-anywhere” library
  - Can handle fault tolerance, malleability
- SmartSockets
  - Solve connectivity problems automatically (firewalls, NAT, addressing problems)

ibis

vrije Universiteit

Grid deployment

- JavaGAT: Java Grid Application Toolkit
  - Make applications independent of underlying grid
  - Used for file copying, resource discovery, job submission & monitoring, user authentication
  - API is currently standardized (SAGA)
- Zorilla P2P system
  - Job management, gossiping, clustering, flood scheduling



Ibis applications

- e-Science (VL-e)
  - Brain MEG-imaging
  - Mass spectroscopy

- Multimedia content analysis
- Video processing
- Many HPC applications
  - SAT-solver, N-body, grammar learning, ….

ibis

vrije Universiteit

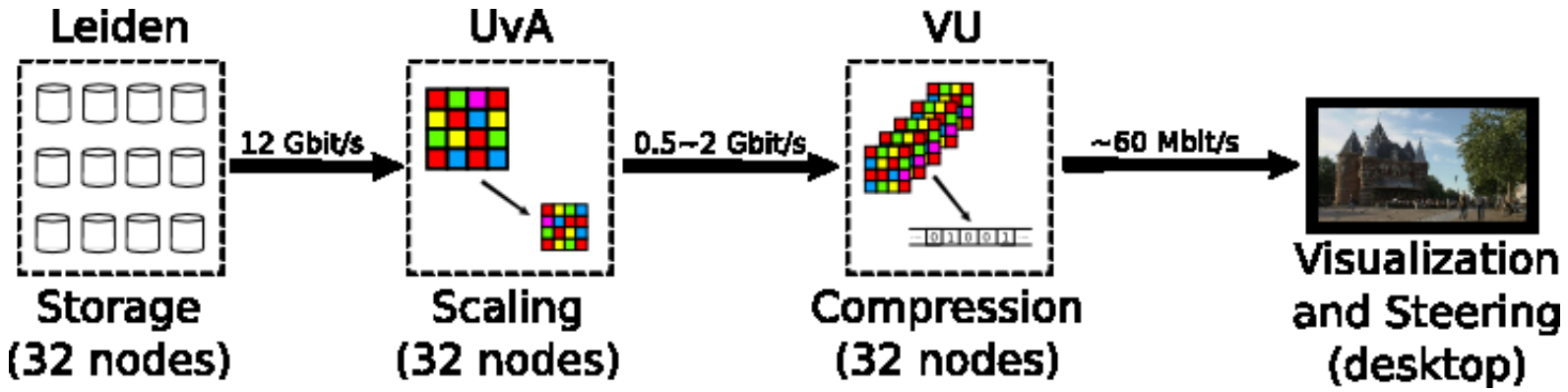
vl·e virtual laboratory for e·science

European users

- D-Grid: Workflow engine for astronomy
- U. Erlangen: grid file system
- INRIA: ProActive on Ibis RMI
- U. Patras: Jylab scientific computing
- UPC Barcelona: Grid Superscalar
- HITACHI: Peta-scale data management

High Resolution Video Processing

- Realtime processing of CineGrid movie data
  - 3840x2160 (4xHD) @ 30 fps = 1424 MB/sec
- Multi-cluster processing pipeline
  - Using DAS-3, StarPlane and Ibis



ibis

vrije Universiteit

Summary

- Goal: Simplify grid programming/deployment
- Key ideas in Ibis
  - Virtual machines (JVM) deal with heterogeneity
  - High-level programming abstractions (Satin)
  - Handle fault-tolerance, malleability, connectivity problems automatically (Satin, SmartSockets)
  - Middleware-independent APIs (JavaGAT)
  - Modular

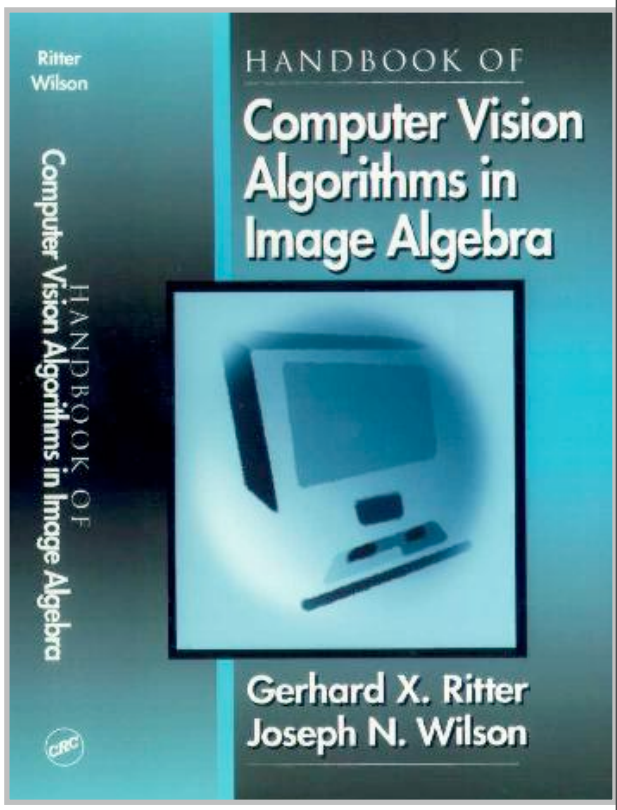
Image processing

- Frank Seinstra's research
- Previous: Horus: parallel implementation of low-level image processing operations (UvA, Arnold Smeulder's group)
- New: Jorus: Horus in Java and on Ibis

vrije Universiteit

Low Level Image Processing `Patterns' (1)

- Unary pixel Operations
  - absolute value
- Binary pixel Operations
  - addition
- N-ary Pixel Operations:
  Template, Kernel, Filter,
  Neighborhood operations
  - Gauss filter

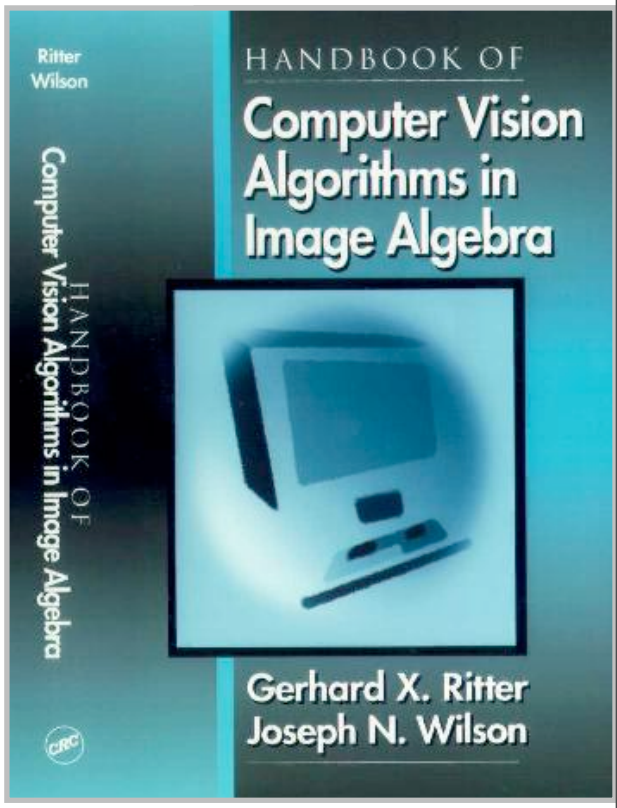


ibis

vrije Universiteit

Low Level Image Processing `Patterns' (2)

- Reduction Operation
  - sum of pixel values
- N-Reduction Operation
  - histogram
- Geometric Transformation
  - rotation



ibis

vrije Universiteit

Example Application: Template Matching

```
for all images {
    inputIm = readFile ( … );
    unaryPixOpI ( sqrdInIm,  inputIm,  “set” );
    binaryPixOpI ( sqrdInIm,  inputIm,  “mul” );
    for all symbol images {
        symbol = readFile ( … );
        weight = readFile ( … );
        unaryPixOpI (filtIm1, sqrdInIm, “set”);
        unaryPixOpI (filtIm2, inputIm, “set”);
        genNeighborhoodOp (filtIm1, borderMirror, weight, “mul”, “sum”);
        binaryPixOpI (symbol,  weight,  “mul” );
        genNeighborhoodOp (filtIm2, borderMirror, symbol, ”mul”, “sum”);
        binaryPixOpI (filtIm1, filtIm2, “sub”);
        binaryPixOpI (maxIm, filtIm1, “max”);
    }
    writeFile ( …, maxIm, … );
}
```

ibis

vrije Universiteit

Implementation

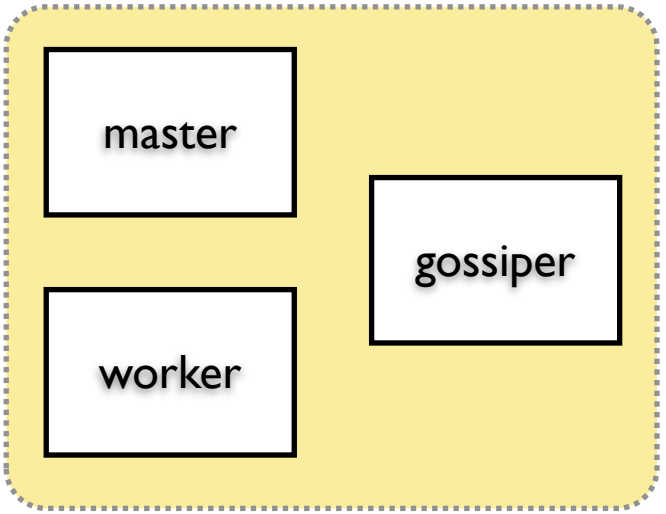
- Load distribution by Horus
- Optimizer to generate the best sequence of operations
- All parallelism is hidden from the user: still the same Horus code
- But different Horus code can have better or worse parallelism

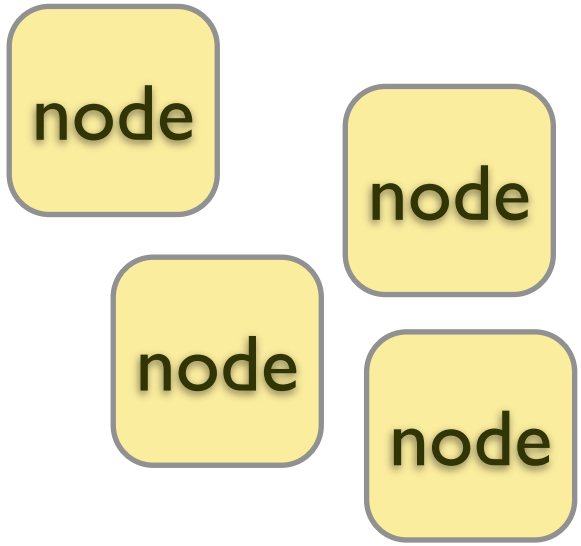
Maestro

- Programming model for stream processing
- Self-organizing: no central coordination
- Learning (but requires repeatable performance in the processing steps)
- Inherently very robust

Maestro nodes

- Master selects fastest worker for each task, based on past performance and total completion time for job type
- Worker reports back to master, and submits next job in the flow to local master
- Gossiper helps ensure recent performance info for all nodes





Example application

Convert video frames:

```
convertJob = jobs.createJob(
  "benchmark",
  new GenerateFrameTask(),
  new ScaleUpFrameTask( 2, slowScale, allowScale ),
  new SharpenFrameTask( slowSharpen, allowSharpen ),
  new CompressFrameTask(),
  new SaveFrameTask( dir )
);
```

Results

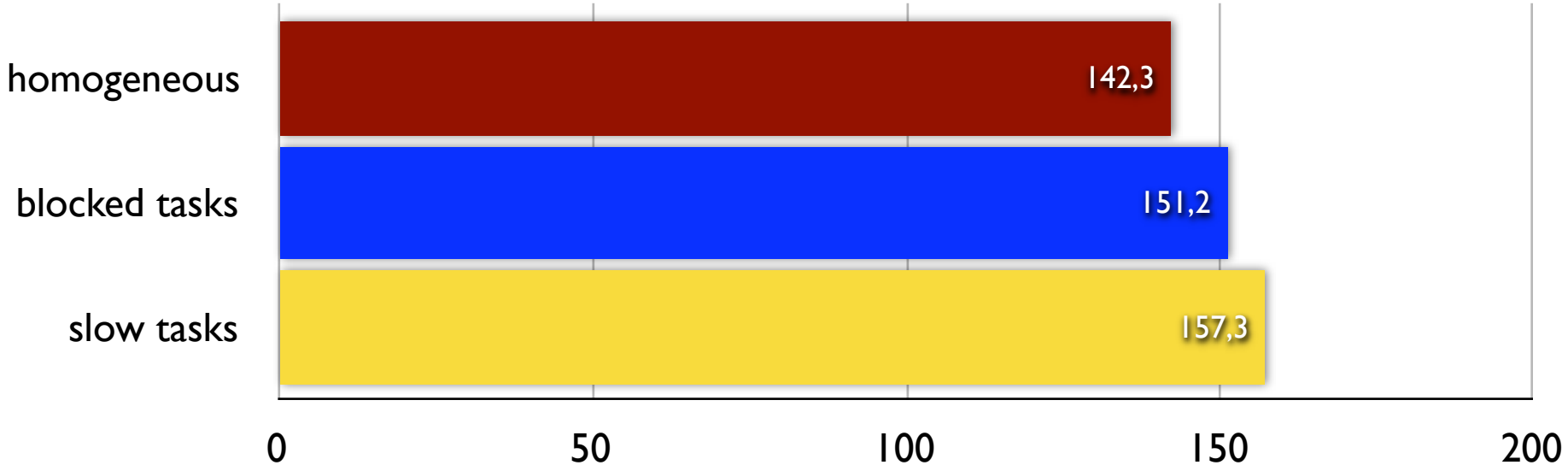
Run image processing on 40 nodes: 20000 frames

- All nodes the same (homogeneous)
- 20 nodes no sharpening, 20 no scaling
- 20 nodes 4 times slower sharpen, 20 slow scale

Results

Run image processing on 40 nodes: 20000 frames

- All nodes the same (homogeneous)
- 20 nodes no sharpening, 20 no scaling
- 20 nodes 4 times slower sharpen, 20 slow scale



Summary

- Our group is very interested in astronomy applications
- We can offer:
  - Robust grid computing
  - Efficient parallel image processing
  - Stream computing
- Typical research question: can XXX be done on grid systems?