
Tool Environments in CORBA-based Medical High Performance Computing

Thomas Ludwig

Markus Lindermeier, Alexandros Stamatakis,
Günther Rackl

Department for parallel and distributed systems
Institute for computer science
Ruprecht-Karls-Universität Heidelberg

thomas.ludwig@informatik.uni-heidelberg.de

Objectives

Applications

- ↳ medical image processing

Tools for parallel systems

- ↳ load balancer, visualizer

How to bring it together?

- ↳ CORBA
 - ↳ integration of legacy code
 - ↳ parallel and concurrent programs
 - ↳ distributed environments

Framework Medical Image Processing

Advanced technology

- ↳ positron emission tomography (PET)
- ↳ functional magnetic resonance imaging (fMRI)

Characteristics

- ↳ image sequences in time during cognitive and motoric exercises
- ↳ computationally intensive

Application

- ↳ neuroscience: Alzheimer disease, epilepsy, ...

Framework Tools Engineering

Objectives

- ➔ design, implementation and deployment of various tool types

Tools and systems

- ➔ debugger, performance analyzer, visualizer, load balancer, ...
- ➔ parallel and distributed target systems

Tool interoperability

- ➔ still under research
- ➔ first examples available

Preceding Work

Medical imaging

- ➔ SPM: statistical parametrical mapping
- ➔ parallelization with PVM and CORBA
- ➔ realignment: compensates patients' movements

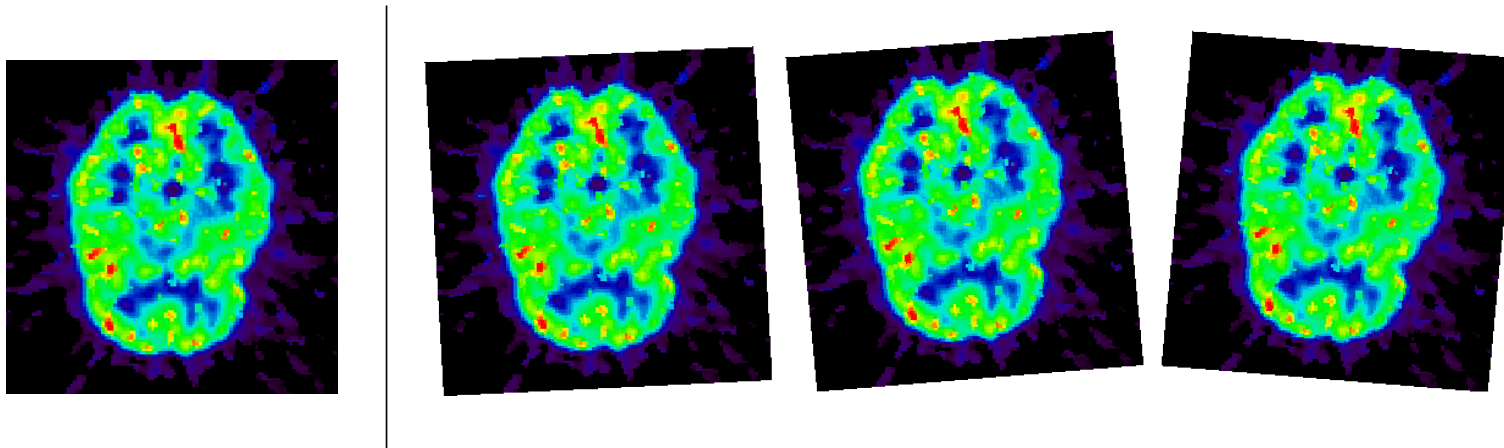
Tools engineering

- ➔ monitoring concepts for CORBA
- ➔ load balancer concept for CORBA

The Realignment Application

Task

- ➔ Given: series of images 1...n
- ➔ Problem: patient's movements have to be compensated



Approach

- ➔ calculate reference data for first picture
- ➔ realign pictures 2...n with respect to first one
- ➔ output 4×4 transformation matrix

Parallelization with CORBA

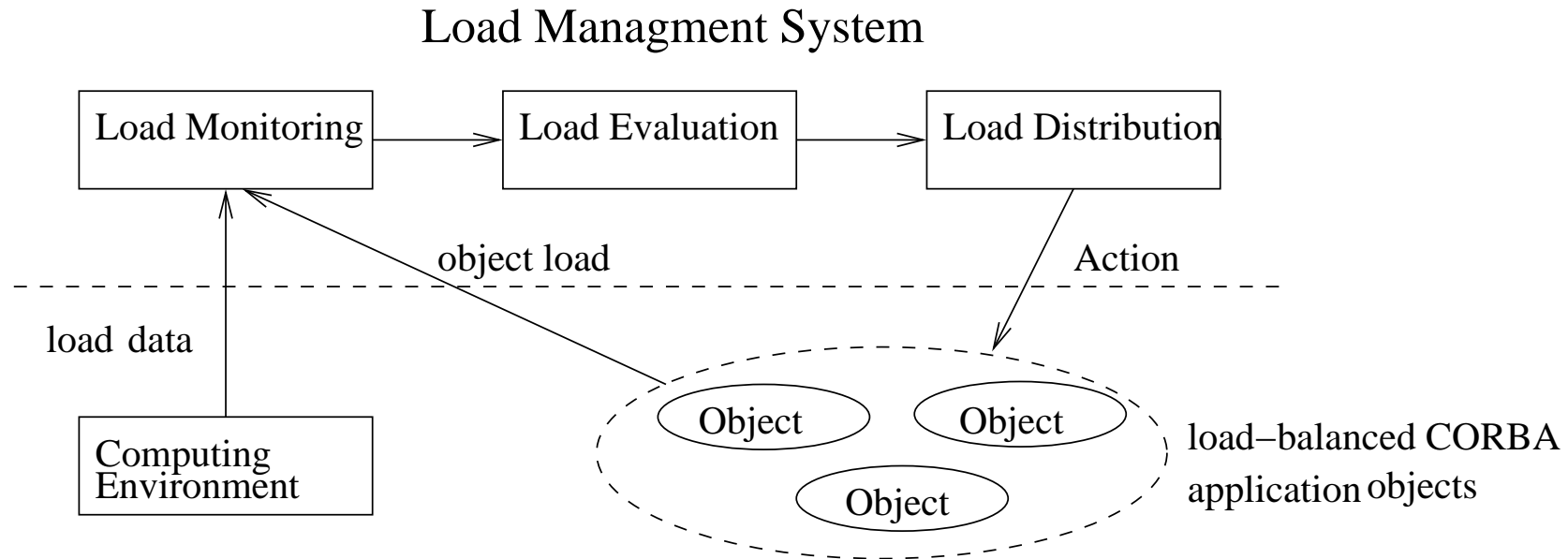
Algorithm

- ➔ client sends pictures and reference data to server
- ➔ server computes transformation matrix and sends it back

Problem

- ➔ load imbalance (different pictures, additional load)
- ➔ reference data has to be cached and re-used

Load Management Concepts



Options

- ➔ distribution: initial placement, migration, replication
- ➔ client/replica-relationship: static or dynamic
- ➔ replica deletion in case of low load

Load Management Implementation

General concept: system integration; transparent to user

Load monitoring

↳ via SNMP on each node

Load evaluation

↳ global comparison of local node values

↳ trigger threshold

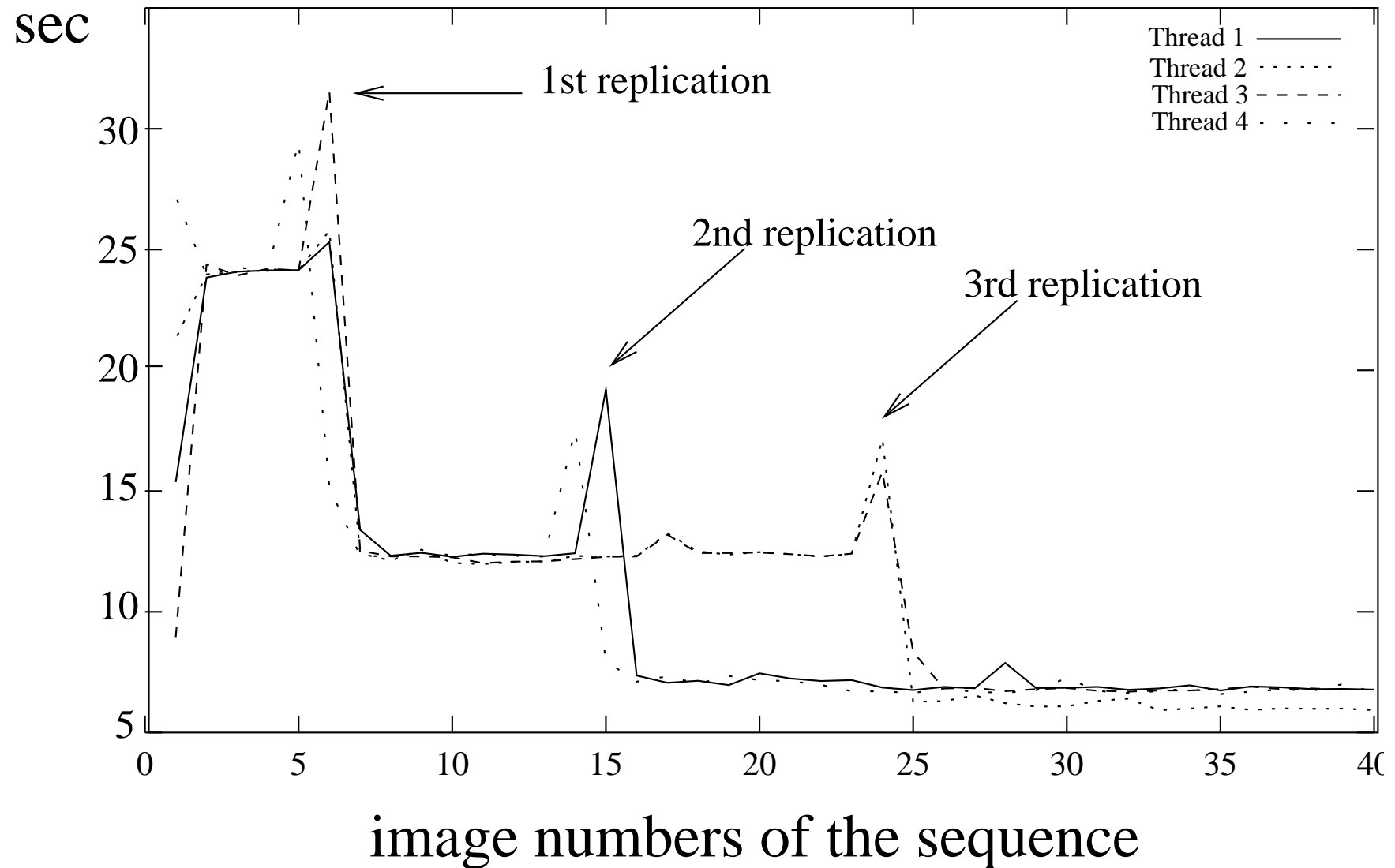
Load distribution

↳ modification of the object request broker (ORB)

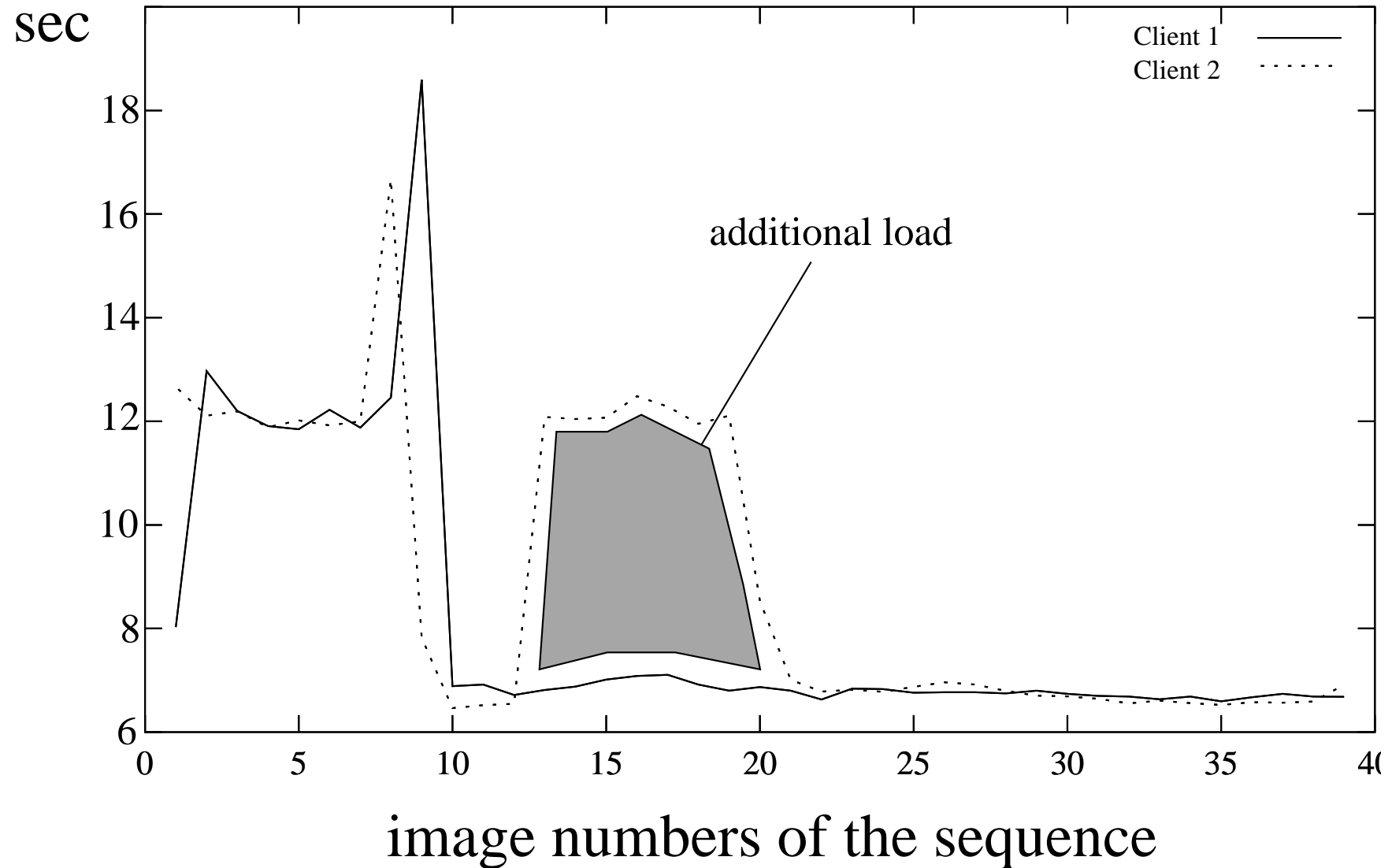
↳ modification of the portable object adapter (POA)

↳ all in Java for JacORB

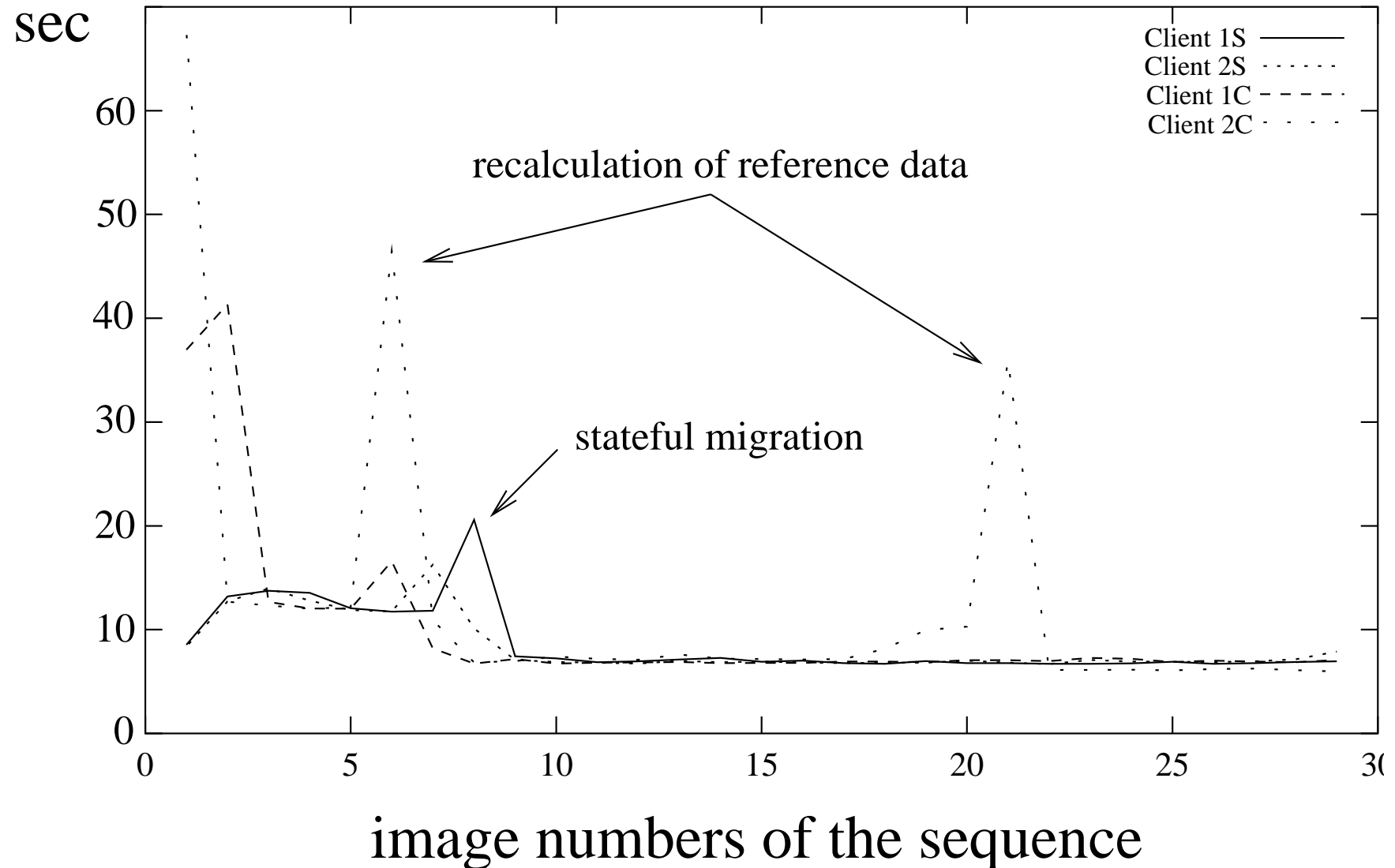
Example: Multi-Threaded Client



Example: Migration and Additional Load



Example: Replication and Status



Tool Interoperability

Interactive tool environment

- ➔ middleware monitoring system MIMO
- ➔ middleware visualization system MiVis

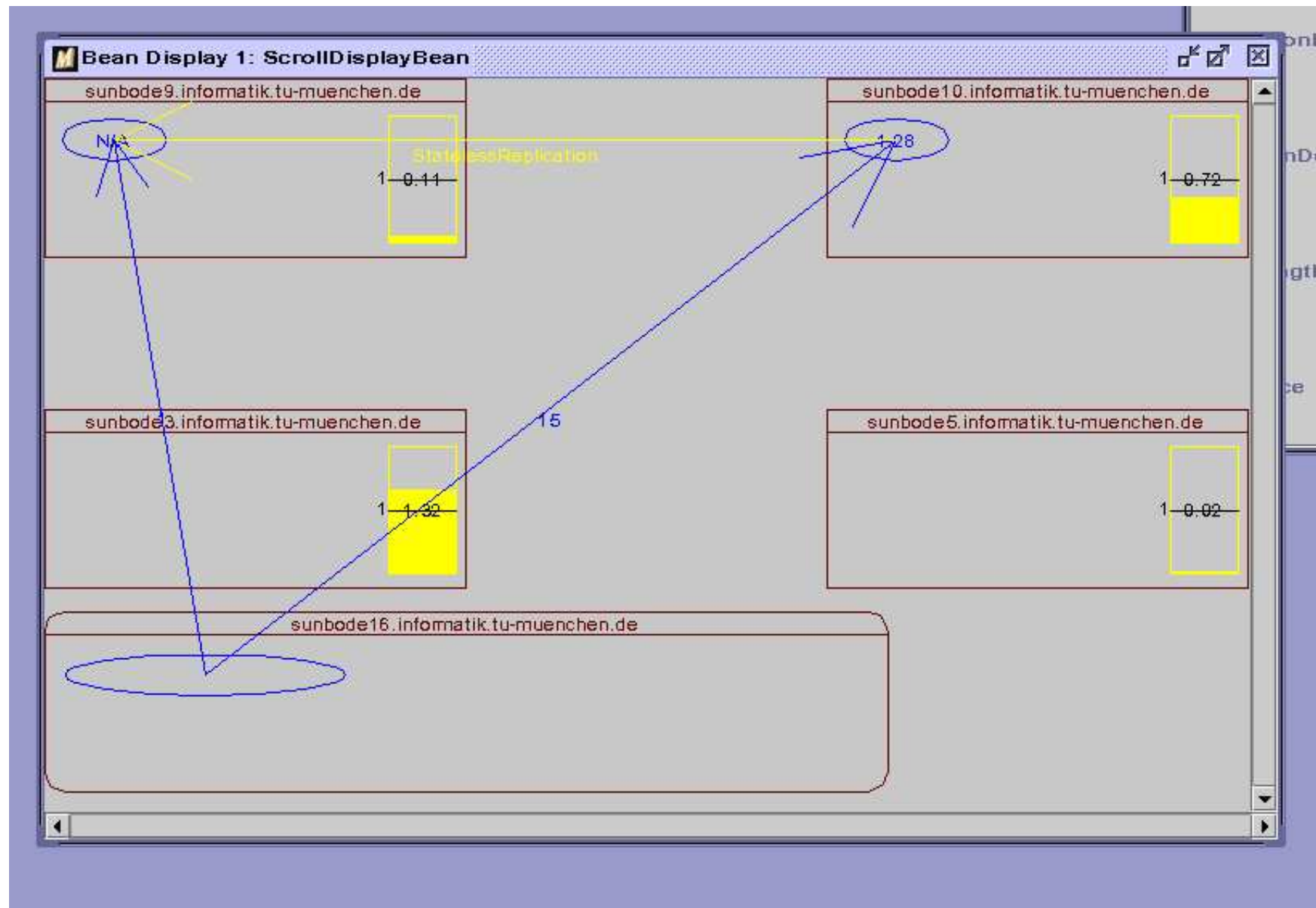
MiVis extension

- ➔ add new display types as Java Beans

Interoperability

- ➔ visualizer shows load management behavior
- ➔ visualizer manipulates load management behavior

Screenshot of Visualizer



Finally...

Summary

- ✎ CORBA is appropriate for parallelization of certain algorithms
- ✎ CORBA-based applications can efficiently be load balanced
- ✎ tool integration provides new functionality

Future Work

- ✎ elaborate load balancer mechanisms and strategies
- ✎ integrate distributed medical imaging into clinical routine