

Thomas Krüger

April 26th, 2003

Parallel and Distributed Computing for an Adaptive Visual Object Retrieval System

**Thomas Krüger, Jochen Wickel,
and Karl-Friedrich Kraiss**

Outline

- Introduction
- Related Work
- System Description
- Dataflow Concept
- Distributed Processing
- Experimental Results
- Conclusion

Requirements

- adaptive to meet different requirements
 - flexibility
 - modularity
- sophisticated algorithms
 - image processing
 - classification / AI
- support for
 - parallelism
 - distribution
 - heterogeneous environments
 - load balancing



Related Work - Image Processing

Visual programming environments for computer vision and image processing

- Application Visualization System
- Khoros / Cantata
- IrisExplorer
- WiT, Mavis, VISSION ...

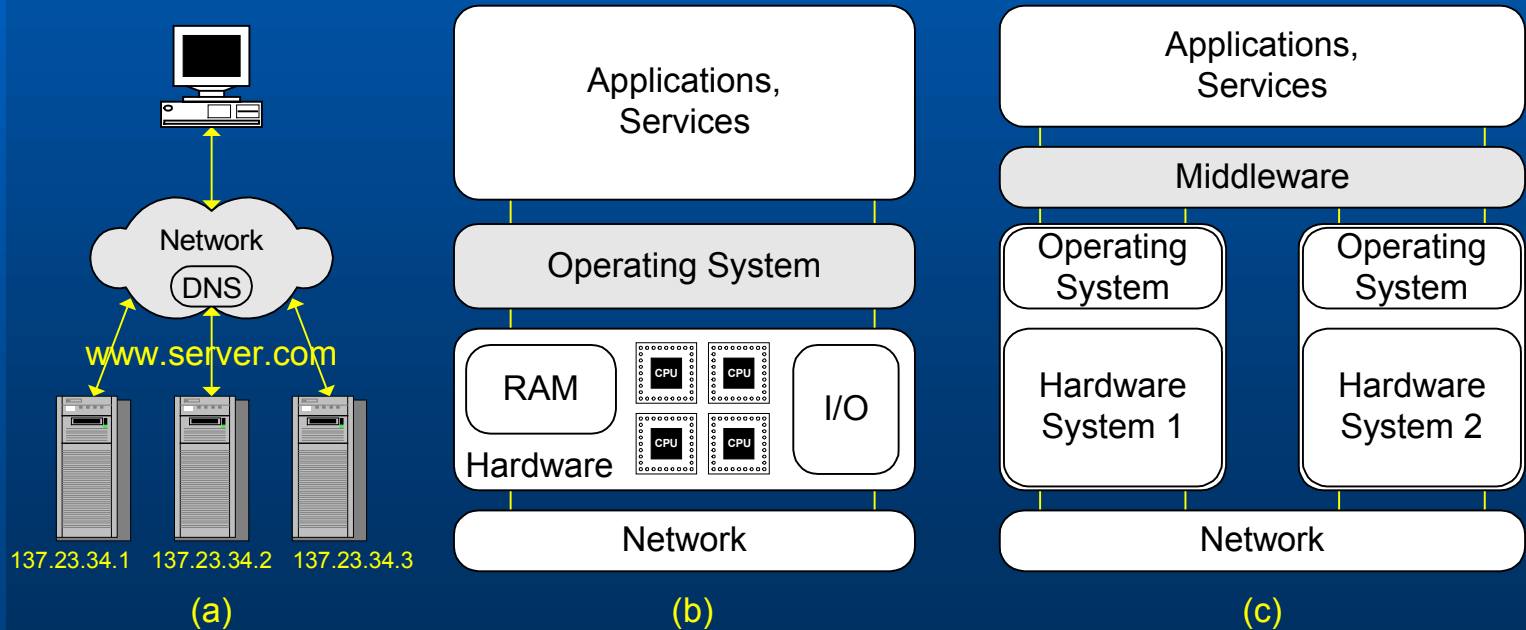
- VuSystem
- Matlab / Simulink, Labview

Parallel & Distributed Computing

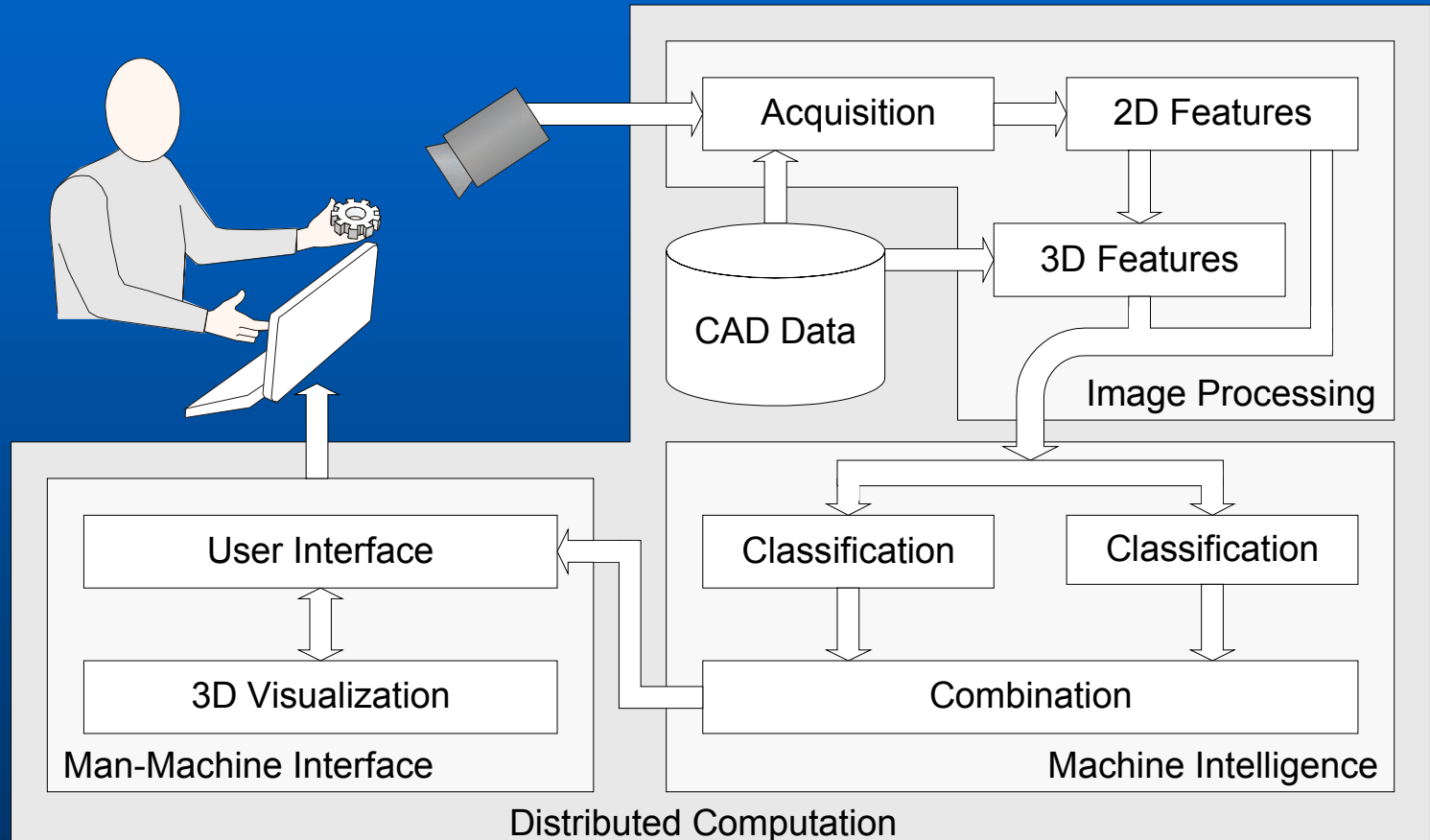
- Multiprocessor systems
 - Uniform Memory Access (UMA)
 - Non-Uniform Memory Access (NUMA)
 - Symmetric Multiprocessors (SMP)
- Coupled Computing System
 - Cluster of Workstations (COW)
 - Message Passing Interface (MPI)
 - Parallel Virtual Machines (PVM)
 - **Network of Workstations (NOW)**

Related Work - Load Balancing

- Network
- Operating System
- Middleware



The Axiom System

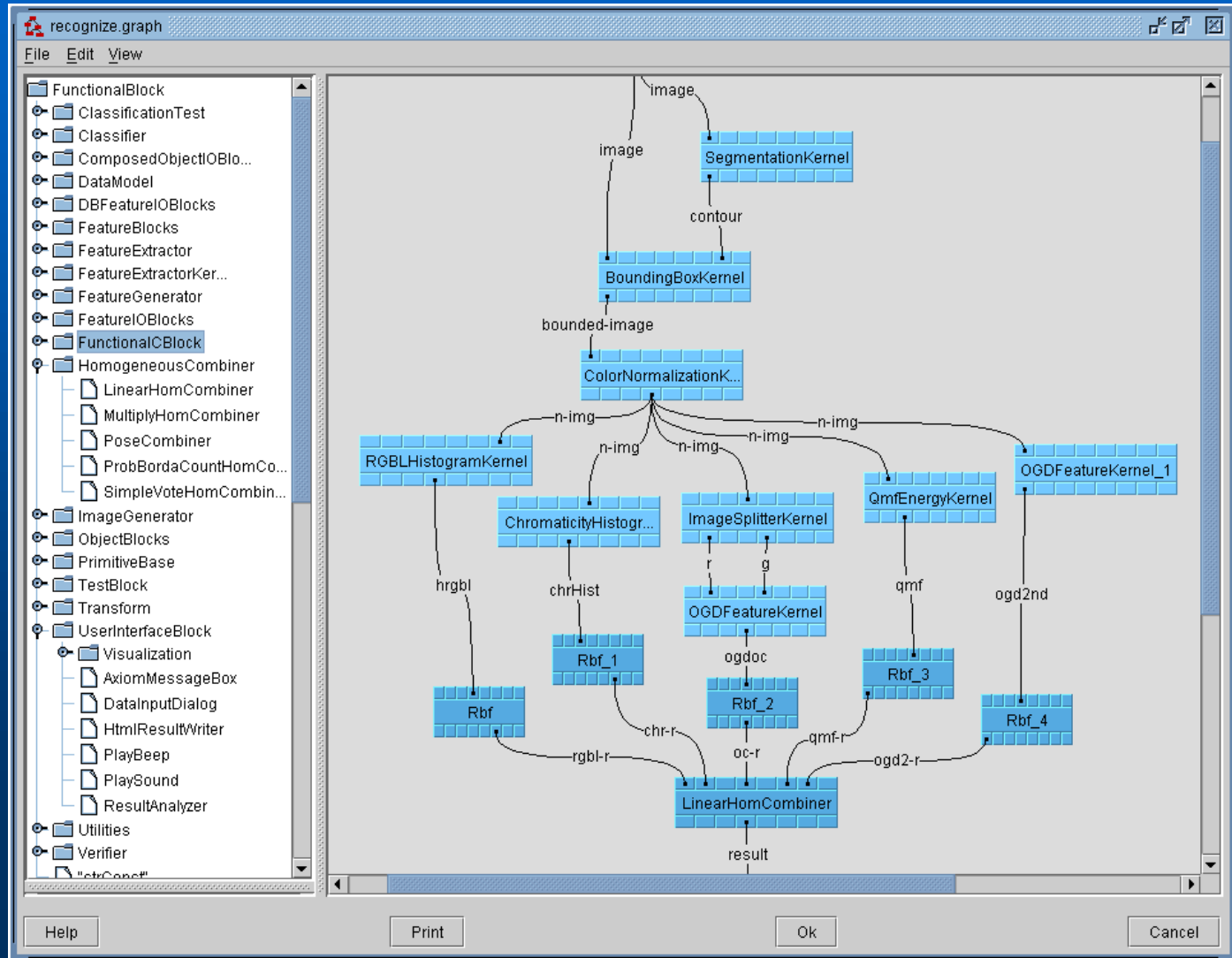


Axiom - Adaptive expert system for intelligent object mining

Dataflow Concept

- natural specification:
dataflow graph, functional program
- C++ library for computer vision
object oriented, encapsulates operations
`http://ltilib.sourceforge.net`
- **F**unctional **O**bject **R**etrieval **C**ontrol **S**yntax
 - similar to Scheme language
 - processing blocks written in Java or C++
 - sequential functional and parallel dataflow programs

Programming Panel

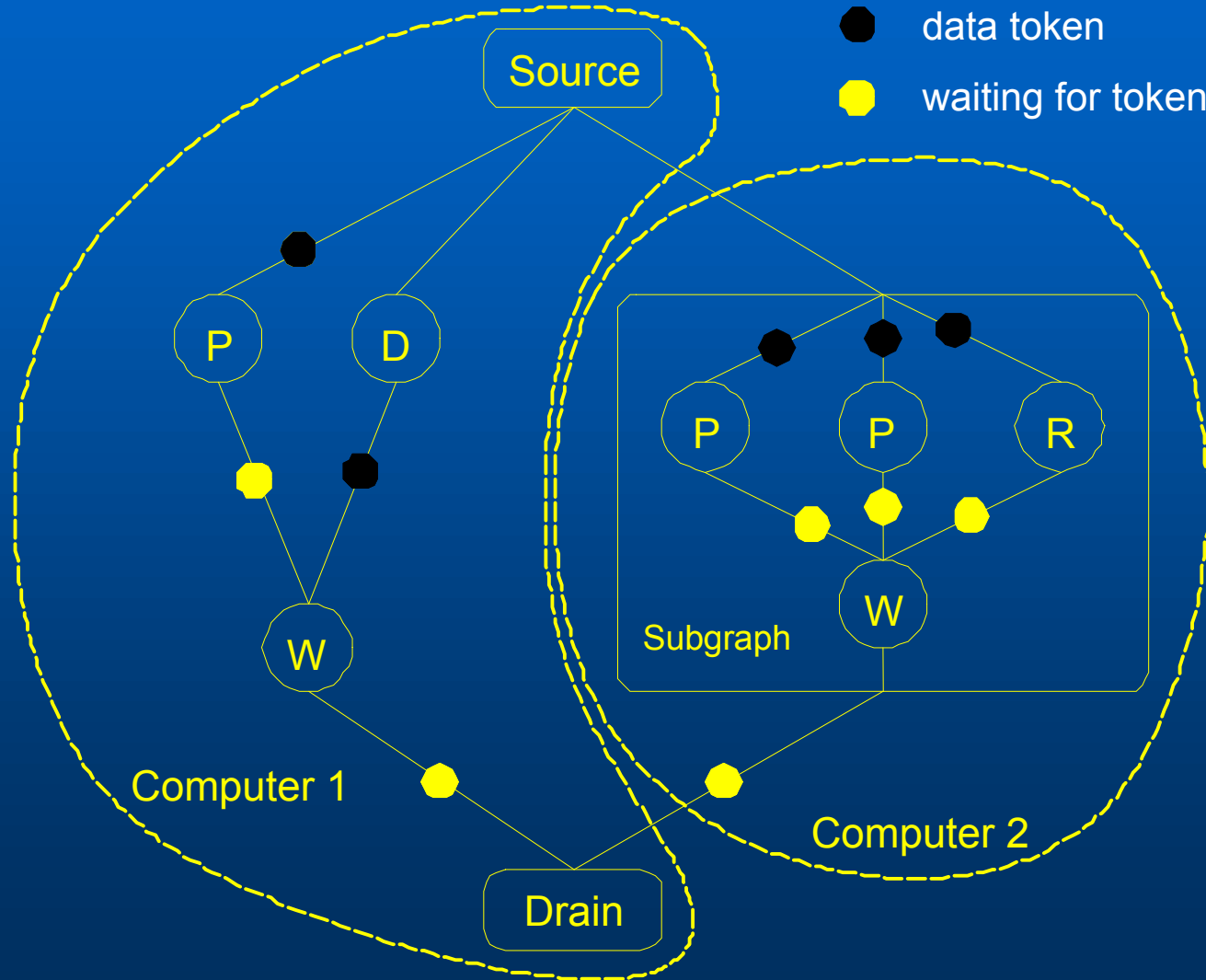


FORCS Code

```
(let contour (features.SegmentationKernel image))
(let bounded-image
  (features.BoundingBoxKernel image contour))
(let n-img (images.ColorNormalizationKernel
  bounded-image))
(let hrgb1 (features.RGBLHistogramKernel n-img))
(let chrHist
  (features.ChromaticityHistogramKernel n-img))
(let lnk.1 (images.ImageSplitterKernel n-img))
(let r (nth 1 lnk.1))
(let g (nth 2 lnk.1))
(let ogdoc (features.OGDFeatureKernel r g))
(let qmf (features.QmfEnergyKernel n-img))
(let ogd2nd (features.OGDFeatureKernel_1 n-img))
(let rgb1-r (neural.rbf.Rbf hrgb1))
(let chr-r (neural.rbf.Rbf_1 chrHist))
```

Distributed Processing

Waiting
 Ready
 Processing
 Done



Distribution with Middleware

- Distributed Computing Environment (DCE)
- Distributed Component Object Model (DCOM)
- Java Remote Method Invocation (RMI)
- Microsoft .NET
- Message Oriented Middleware

- **Common Object Request Broker Architecture (CORBA)**

Load balancing

CORBA load balancing

- request for proposal
- proprietary solutions
- simple techniques

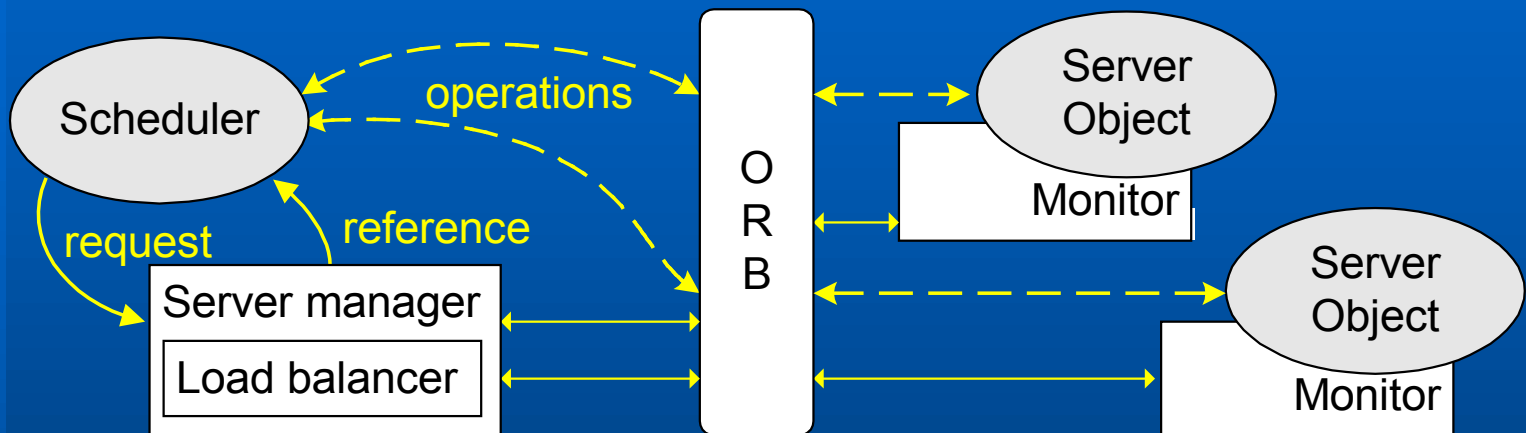
TAO (Othman, Schmidt, 2001)

VisiBroker (Inprise, 2001), Orbix (Iona, 2000)

(Post, 2002), (Lindermeier, 2002),

(Schnekenburger, 1997)

Communication Architecture



Monitor

- one monitor per host
- software monitoring
- monitoring of entire host

Load balancer

CPU-load and clock rate

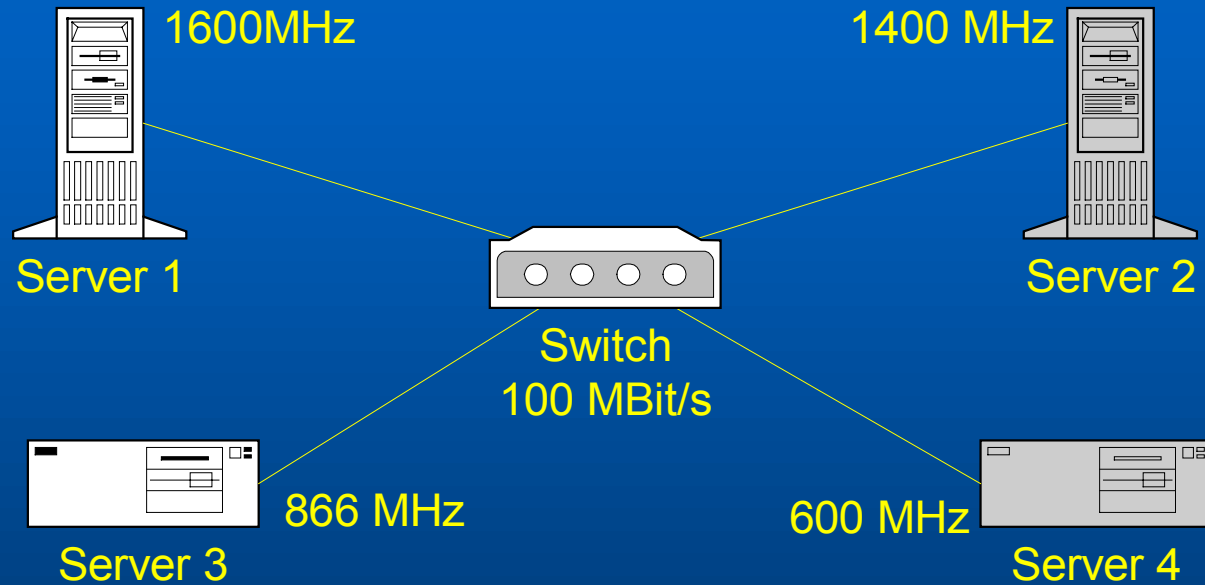
$$CPU_n = (CPU_{n-1} * f_{decay}) + (1 - f_{decay}) \left(\frac{T_{USER} + T_{SYS}}{\sum_{USER, SYS, NICE, IDLE} T} \right)$$

$$CPU_{free} = [\alpha * (l_{max\ Avg} - l_{Avg}) + (1 - \alpha) * l_{Act}] * f_{CPU}$$

server memory and round trip time

$$W_{ges} = w_{CPU} * CPU_{free} + w_{mem} * MEM_{free} - w_{RTT} * t_{RTT}$$

Testbed



Server 1	100%	Server 2	69%
Server 3	42%	Server 4	23%
Scimark2 Benchmark			234%

Scenario

Axiom training sequence:

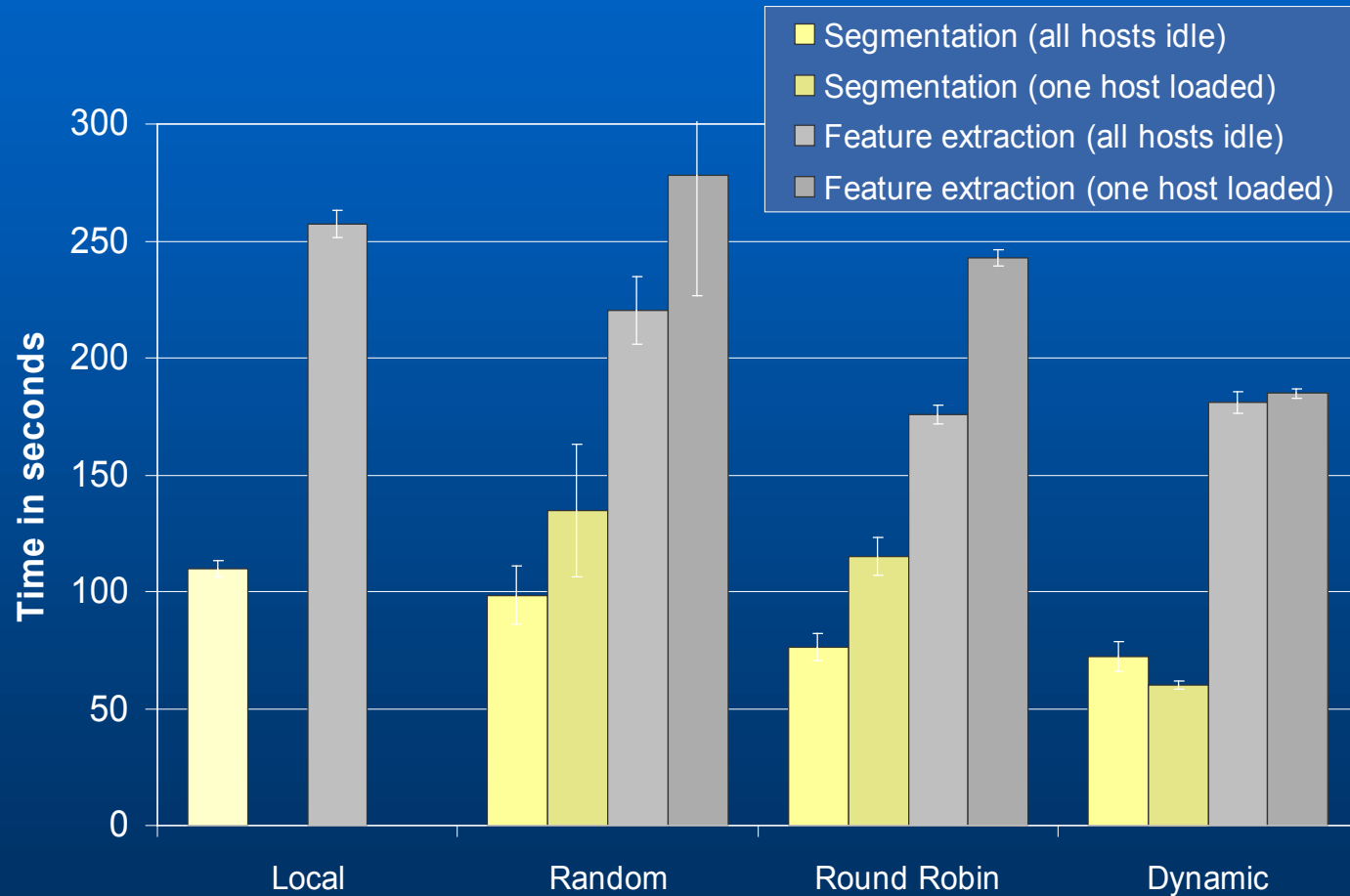
- segmentation of 50 single images
- extraction of local features from 16 single images

SciMark 2 Benchmark

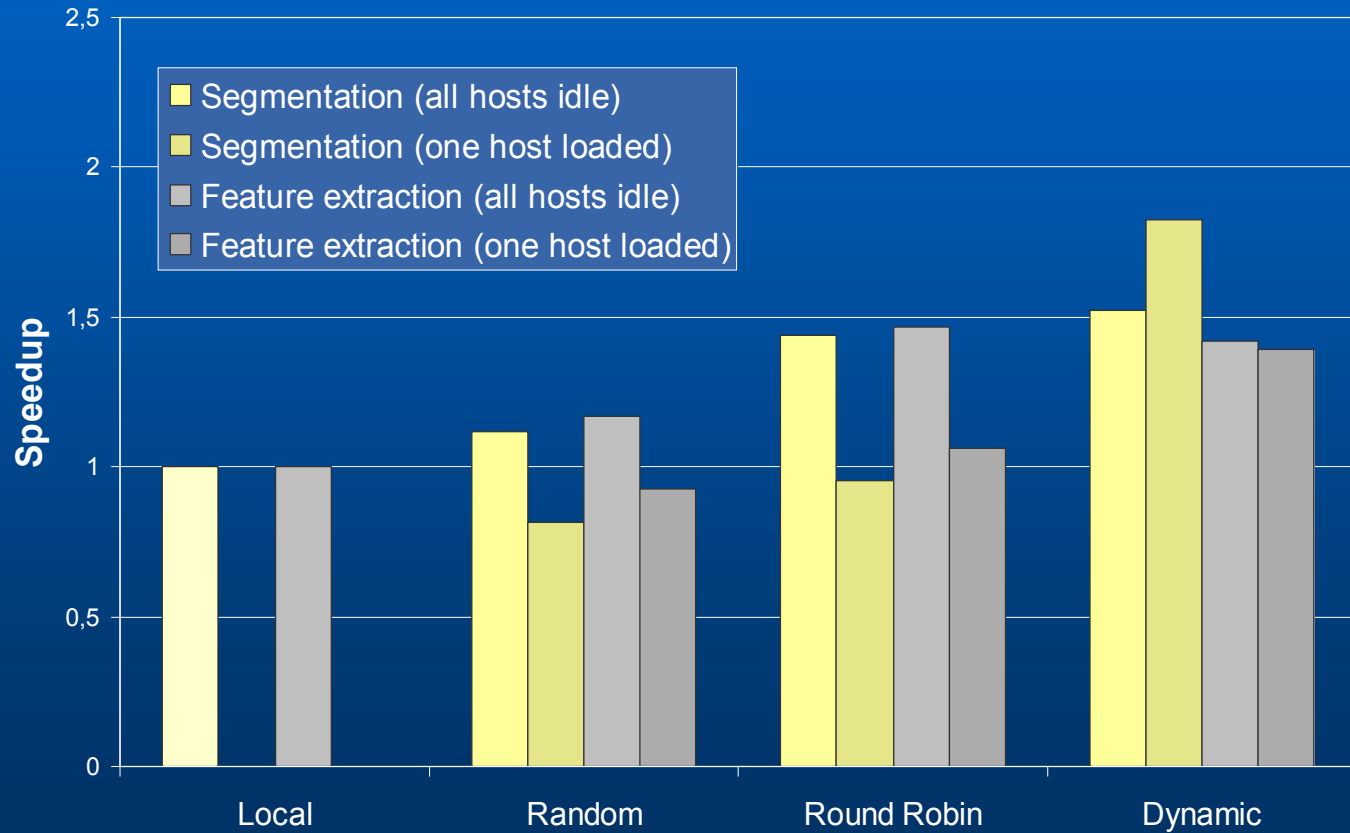
- all hosts idle
- one hosts with additional task

$$S(p) = \frac{T(1)}{T(p)} = \frac{T(1)_{\min}}{T(4)}$$

Experimental Results (I)

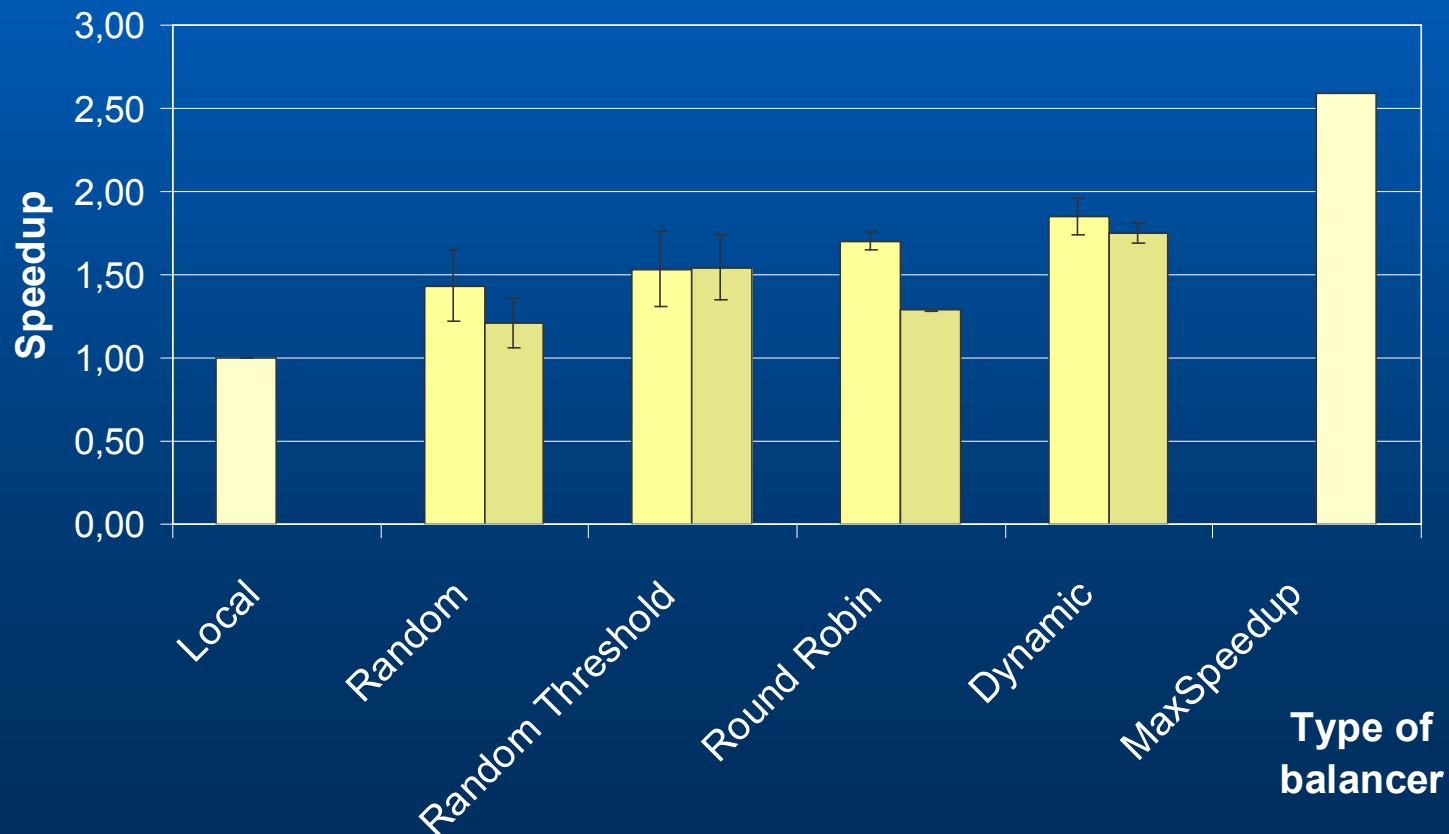


Experimental Results (II)



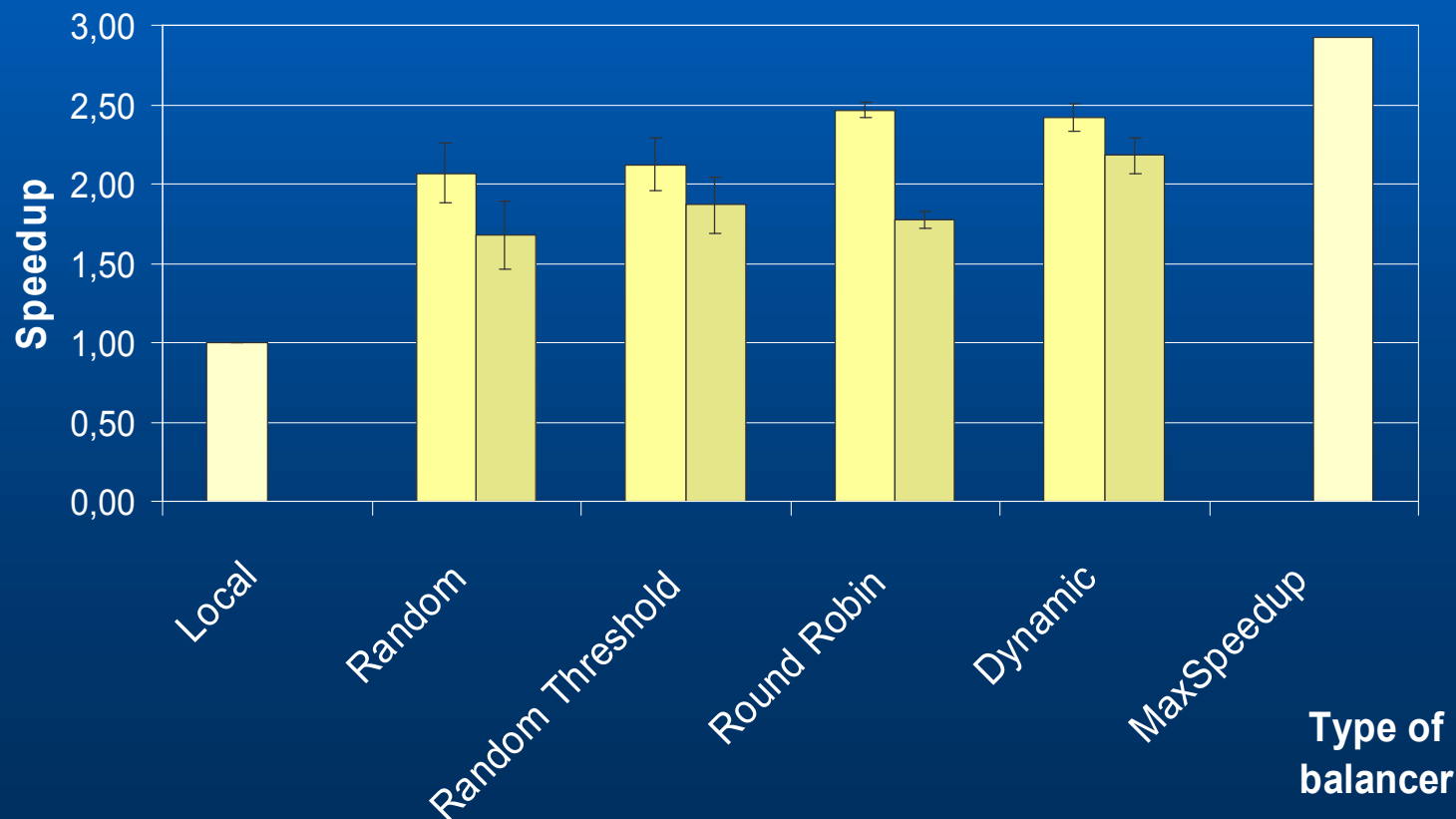
Experimental Results (III)

SciMark 2 FFT Benchmark (small)



Experimental Results (IV)

SciMark 2 FFT Benchmark (large)



Conclusion

- computation architecture for a visual object retrieval system
- visual combination of modular blocks
- control syntax
- distributed computing with CORBA-middleware
- dynamic load balancing in heterogeneous environments

Future Work

- automatic data flow graph analysis
- improve load balancing mechanism
- evaluation of system scalability

LTILib URL

LTILIB <http://ltilib.sourceforge.net/>

Introduction

The LTILib is an object oriented library with algorithms and data structures frequently used in image processing and computer vision. It has been developed by the Chair of Distributed Computer Systems (Distributed User Interfaces) at the University of Technology, as part of many research projects in computer vision dealing with robotics, object recognition and sign language and gesture recognition.

The main goal of the LTILib is to provide an **object oriented** library in C++, which simplifies the code sharing and maintenance, but still providing fast algorithms that can be used in real applications.

It has been developed using [GCC](#) under [Linux](#), and [Visual C++](#) under [Windows NT](#). We have not tested it under other platforms.

Many classes are supplied for the developer to use to simplify the development of his own applications (for example classes for multi threading and synchronization, time measurement and serial port access).

The main purpose of the 300 classes is summarized by the following table:

Linear algebra

Matrices, Vectors, tensors, and functions to extract eigenvalues, eigenvectors, linear equations solutions, statistics, etc. are provided.

Classification and Clustering

Radial Basis Function classifiers, Support vector Machines, K-Means, Fuzzy C-Means, classification statistics are just some examples of what you can do with the LTILib.

Image Processing

The most classes deal with image processing problems. Different segmentation approaches (neural networks, wavelets, steerable filters, and much more) are already available.

Visualization and Drawing Tools

The most difficult part when developing image processing algorithms in C++ is showing temporary images while debugging. Due to the object oriented architecture of the LTILib, you just need to create a viewer object and give it the image you need to show.

On Line Manual
Mailing Lists
FAQ
Sourceforge Summary Page
Tutorial Manual

C++
Download Package
Developer's Guide

Document: C:\msd\ltilib