

Monitoring Technologies for Parallel On-Line Tools

Arndt Bode, Wolfgang Karl, *Thomas Ludwig*, Roland Wismüller

{bode|karlw|ludwig|wismuell}@in.tum.de

Overview

- On-line tools and parallel systems
- The TOPSYS project — early steps
- The PARIX monitoring system — first adaptations
- OMIS — the universal monitoring interface
- OCM — an OMIS compliant monitoring system
- Interoperable tools and new architectures
- MIMO — adaptations to middleware based environments

History

- 1988 On-line tools and parallel systems
- 1991 The TOPSYS project — early steps
- 1993 The PARIX monitoring system — first adaptations
- 1995 OMIS — the universal monitoring interface
- 1998 OCM — an OMIS compliant monitoring system
- 2000 Interoperable tools and new architectures
- 2001 MIMO — adaptations to middleware based environments

On-Line Tools and Parallel Systems

Parallel systems

- Message passing programming paradigm
- Massively parallel computers — workstation clusters

On-line tools

- Interactive tools: Debugger, performance analyzer, visualizer
- Automatic tools: Resource management, load balancing

Necessary functional unit: monitoring system

- Can observe and manipulate the hardware-software environment
- Integrated into the tools

TOPSYS Concepts

TOPSYS — Tools for parallel systems

- DETOP Debugging tool
- PATOP Performance Analysis tool
- VISTOP Visualization tool

TOPSYS Programming Environment

- MMK — Multiprocessor multitasking kernel

Hardware Environment

- Intel iPSC/2 and iPSC/860 parallel computers with distributed memory architecture

TOPSYS Monitoring Techniques

Monitor implemented in software and **hardware**

Designed to observe and manipulate objects of the **MMK programming model**

Monolithic approach: Monitoring system is module of the tools' software

Simple interface: requests refer to one object only

Node-local implementation: tools needed to contact each node separately

Event/action paradigm and basic **programmability**

Monitoring system implemented in the **address space of the run-time environment**

Weak **adaptability** and **portability**

PARIX Adaptations

Successor of the MMK monitoring system adapted to Parsytec's parallel computers with distributed memory running the PARIX environment

No more hardware implementation therefore more powerful concepts

Multi-service requests with result passing

Root monitor for tool-monitor communication

Inter-monitor communication via broadcast and multicast of the target system

No location transparency for object access but already global actions were supported

Well defined interfaces to the operating system and the run-time system

OMIS — On-Line Monitoring Interface Specification

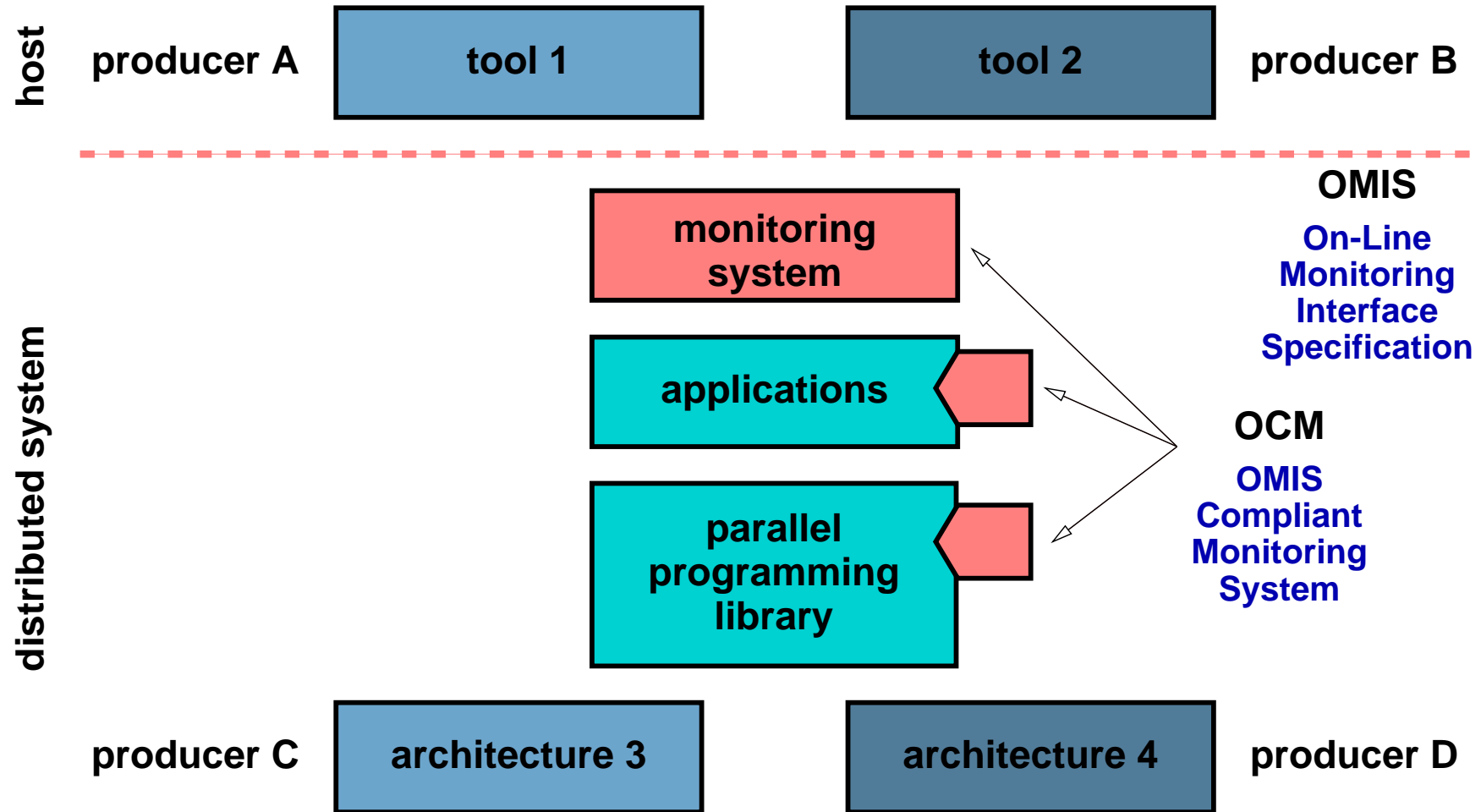
Deficiencies of monitor system construction

- Monolithic approach leads to low portability and adaptability
- Extensibility critical

Poor tool concepts

- No cooperation between tools possible
Reason: Tools do not know of each other
- No concurrent usage supported
Reason: Tools use different infra-structure components that are incompatible to each other

OMIS: The Basic Concept





Object-based Tool Interface

Object classes

- System, node, process, thread, message buffer, message, monitor object

Service classes

- Services for information, manipulation and events

Working principle: event/action model

- **Event** = change of state in the program system
- **Action** = information or manipulation service requests

Requests

- Behaviour of the monitoring system is controlled by event/action-relations
- Relations are programmed via the interface



Interface Syntax and Semantics

```
Omisi_reply omisi_request( char * request ,  
                           void (* callback)(Omisi_reply reply, void *param),  
                           void *param, Omisi_flags flags)
```

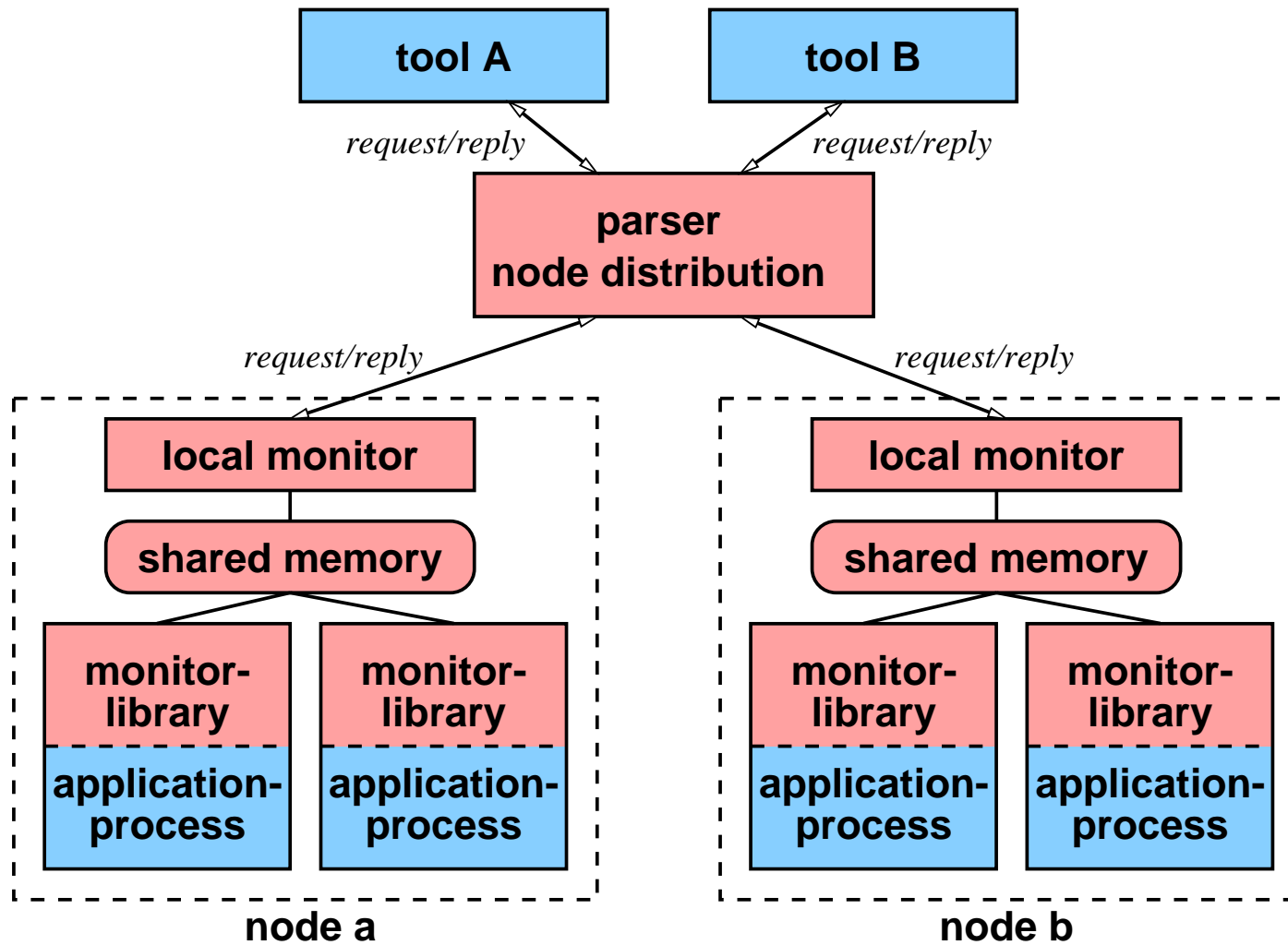
Syntax of event/action-relations

```
request ::= [ event_definition ] : action_list  
event_definition ::= service_name ( parameters )  
action_list ::= action | action [ ; ] action_list  
action ::= service_name ( parameters )
```

Semantic details

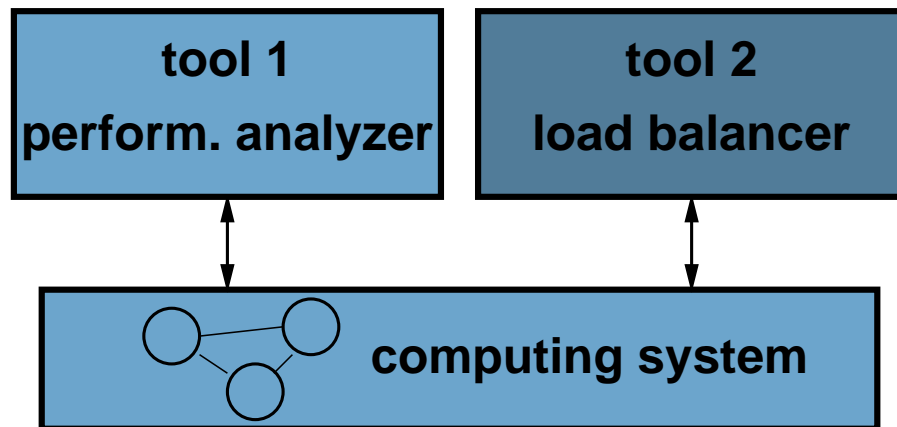
- Services work on single objects and object sets
- Services accept object references on any hierarchy level
- Actions can be performed concurrently

OCM — An OMIS Compliant Monitoring System





Interoperable Tools



Problem

- More than one tool knows the state of object O
- At least one tool modifies the state of O
- Question: what about the other tools?

Prerequisites

- Uniform interface for all tools
- Shared monitoring system



The Model of Interoperability

Starting point: object view of the system

Question: who accesses objects when in which way?

Access modes: reading, writing

Access conflict

- Incompatible overlapping access of two tools to a shared object where at least one access modifies the object

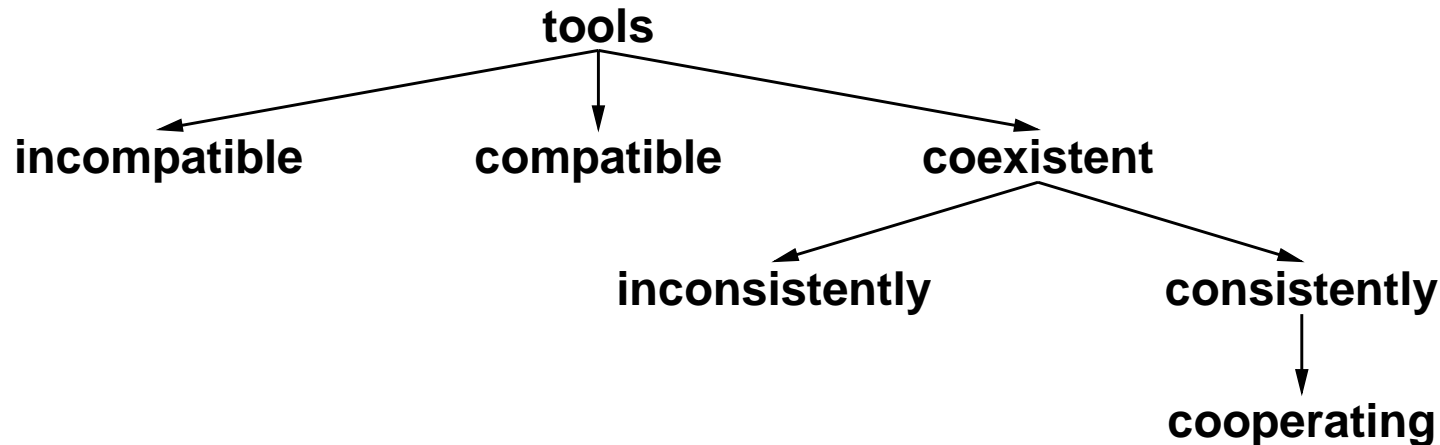
Variants: RW, WR, WW

Possible consequences of access conflicts

- Program crash
- Wrong information in the tool
- Crash of tool session



New Tool Classification



Example: Performance Analysis and load balancing

- Inconsistently coexisting: performance analyzer gets confused by the appearing and disappearing processes
- Consistently coexisting: performance analyzer recognizes migrations and visualizes them if the user wants that. Otherwise it hides them.
- Cooperating: performance analyzer requests load management; load management requests performance analyzer to visualize certain load indices

Prototype Demonstration

Interoperable tool set

- Debugger
- Checkpointing system
- Deterministic program execution
- Program flow visualizer

To be demonstrated by [Roland Wismüller](#) in the afternoon



New Architectural Paradigms

OCM initially oriented towards message passing programming models

OMIS extensible

Adaptation to distributed shared memory programming

- SCI target architecture designed in the SMiLE project
- Memory accesses are recorded with a special hardware monitor device
- Information processing in an OMIS SCI-DSM extension
- Main problem: abstraction levels between hardware memory access and data access at program level.

Adaptation to middleware-based architectures

- MIMO project — monitoring and managing distributed middleware environments

Scientific and Technical Progress in the last Decade

Progress in monitor system construction techniques

- Versatile interface with high abstraction programming language
- Support for efficient tool design and implementation
- Generic interface with high degree of extensibility
- Concepts for interoperable tools

Technical aspects

- Object based approach
- Location transparency
- Object token conversion
- Monitor programming language