TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Monitoring Globus Components with MIMO

Günther Rackl

TUM-I0006
SFB-Bericht Nr. 342/06/00 A
März 00

TUM–INFO–03-I0006-50/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Monitoring Globus Components with MIMO*

— Technical Report —

Günther Rackl

Distributed Systems Laboratory
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
USA
rackl@mcs.anl.gov

LRR-TUM
Lehrstuhl für Rechnertechnik und Rechnerorganisation
Institut für Informatik
Technische Universität München
80290 München
Germany
rackl@in.tum.de
http://wwwbode.in.tum.de/~rackl

November 1999

*Research supported by DOE/ANL and German Science Foundation SFB 342 (TP A1)

Foreword

The distributed systems laboratory in the Mathematics and Computer Science Division of the Argonne National Laboratory and Lehrstuhl für Rechnertechnik und Rechnerorganisation (LRR-TUM) of Technische Universität München share common interests in the development of software for parallel and distributed computing. Argonne has developed the GLOBUS metacomputing infrastructure, LRR-TUM has developed many instances of online tools for distributed program and development.

Günther Rackl, a researcher at LRR-TUM's grant "Methods and Tools for parallel and distributed Systems" (German Science Foundation grant: DFG SFB 342, TP A1) had developed the MIMO MIddleware monitor with CORBA-based interfaces. It was a very good oportunity for him to use MIMO to observe the behaviour of GLOBUS components. For this purpose, Günther Rackl had a research stay at Argonne National Laboratory from September to November 1999. This paper reports on the work done at Argonne.

LRR-TUM is especially grateful to the Argonne National Laboratory, especially to Ian Foster for hosting Günther Rackl and thereby strengthening the research links between both institutions.

Munich, March 10th 2000

Arndt Bode

Abstract

This report describes work done during a research stay at Argonne National Laboratory from September to November 1999. It is about connecting components of the Globus metacomputing infrastructure developed at Argonne National Laboratory to the MIMO monitoring system developed at LRR-TUM. —

MIMO MIddleware monitor is built to enable easy connection of diverse middleware platforms. This is enabled by introducing a well defined interface for data collection components ("intruders"). This report shows how to realize an adapter for the Globus metacomputing infrastructure to MIMO, allowing to observe the behavior of several Globus components. As MIMO's interfaces are CORBA-based and Globus is completely implemented in C, an ORB providing a C language mapping was used for this purpose. With the resulting C-adapter for MIMO, exemplary instrumentations of the Globus gatekeeper and jobmanager components have been carried out in order to show the appropriateness of the approach. The C-adapter itself is kept generic and allows for usage with other C-based middleware platforms.

Contents

1 Introduction

Developing and maintaining large distributed software environments is one of the major challenges in computer science at the time. The usage of middleware platforms abstracting from diverse and heterogeneous computing platforms is a common approach to handle the complexity of such systems. Middleware platforms include general purpose distributed object-computing environments [RZB99] like CORBA or DCOM, message-oriented middleware (MOM), transaction processing monitors (TPMs), or meta-computing infrastructures like Globus.

This report first introduces the MIMO MIddleware MOnitor infrastructure, a monitoring and management system that addresses the following issues:

- *Support for the whole software lifecycle:* In order to be able to build tools for the complete software lifecycle, information on all abstraction levels of the system has to be gathered. This includes low-level information, e.g. needed for debugging purposes, as well as high-level information for application management issues. MIMO solves this problem by introducing a multi-layer-monitoring model that is used to classify data collected from the observed system.

- *Integration of monitoring and management functionality:* Supporting the complete software lifecycle allows us to use a single system for monitoring and management tasks; as the term "monitoring" is mostly used for low-level aspects, and "management" rather for high-level administrative tasks, the multi-layer-model makes it possible to build both kinds of tools only using MIMO.

- *Integration of heterogeneous middleware platforms:* MIMO is designed to enable monitoring of different middleware platforms simultaneously. This is done by introducing a generic interface for middleware platforms to MIMO, such that heterogeneous systems can be easily integrated. As interoperable applications are getting more and more popular (see e.g. CORBA-COM bridges), the ability to observe heterogeneous systems simultaneously is one of the key features for future monitoring systems.

Subsequently, we describe the connection of the Globus metacomputing infrastructure [FK98] to the MIMO system. This possibility to easily add very diverse middleware platforms to the monitoring system is one of the key features of MIMO's design. The integration of Globus is enabled by means of an adapter providing functionality to instrument Globus components implemented in C to MIMO. All communication is handled via CORBA, which abstracts from difficulties arising when different programming languages and hardware/operating system platforms might be used.

2 MIMO MIddleware MOnitor

This section gives a brief overview of the MIMO system. First, we introduce the applied multi-layer-monitoring model (MLM), and subsequently we give an overview of MIMO's features and usage policies.

2.1 The Multi-Layer-Monitoring Model

Here we describe the system model and multi-layer-monitoring approach, on which the MIMO system is based. Figure 1 shows an illustration of a typical distributed middleware environment that we consider.

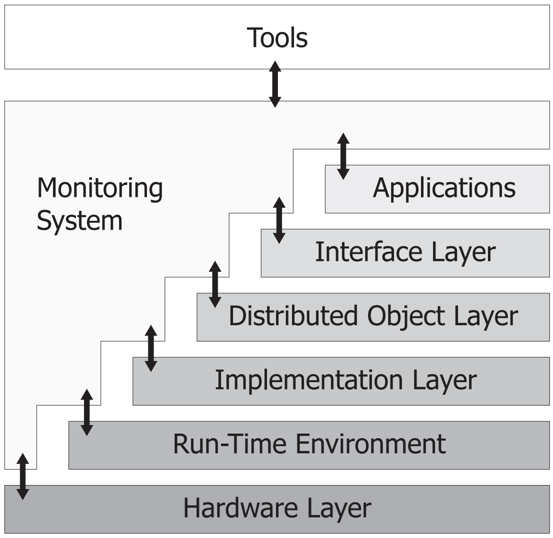

Figure 1: Multi-Layer Monitoring Model

The system to be monitored consists of six abstraction layers, from which the monitor collects information and provides it to tools only by means of the tool-monitor interface. All information gathered by the monitoring system is classified according to this layer model. For various middleware platforms, this abstract model can be mapped to concrete entity types related to the respective middleware environments. Consequently, a *multi-layer monitoring* (MLM) approach [Rac99] which closely reflects the structure of distributed object-environment is well suited for a large class of online tools. For obtaining information from all abstraction layers, specialized modules adapted to requirements of the layer to be observed can be inserted into the monitoring system. Thus, the monitor is kept very modular and flexible and can easily be adjusted to changes of the distributed environment.

2.2 MIMO Monitoring Scenario

MIMO's architecture is based on the monitoring scenario shown in Figure 2. The

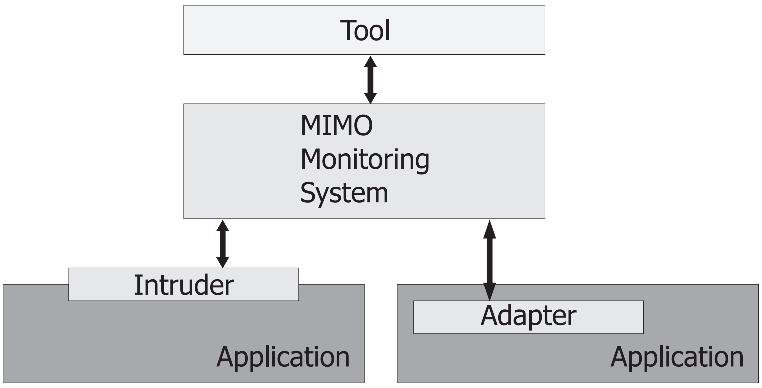

Figure 2: Monitoring Scenario

fundamental architecture relies on the separation of the tools from the monitoring system and the observed applications [LWSB97]. The resulting 3-tier model shows tools making use of MIMO by means of a tool-monitor-interface, while MIMO collects information from the monitored applications by means of intruders or adapters which communicate with MIMO through a intruder-monitor-interface. The difference between intruders and adapters is that intruders are transparently integrated into the application (without rebuilding the application), while adapters might be built by inserting code into the application (and rebuilding it).

2.3 MIMO Architecture

An illustration of the basic MIMO architecture is shown in Figure 3. Every instance of MIMO keeps information about the current system state, i.e. data about the applications currently attached to this instance via the intruders/adapters. Furthermore, information about currently attached tools and their active requests is stored. Information can also be exchanged between various MIMO instances, but it is only stored once at the MIMO instance who's intruder/adapter provided the data.

2.3.1 Tool-Monitor-Interaction

The only entrance point for tools to MIMO is the tool-monitor-interface, which is a CORBA IDL interface that basically provides methods for attaching to and detaching from MIMO, and for starting and stopping requests.

When requests are started, the result can either be returned synchronously if this is possible (e.g. for system state queries), or in an event-based manner, which is

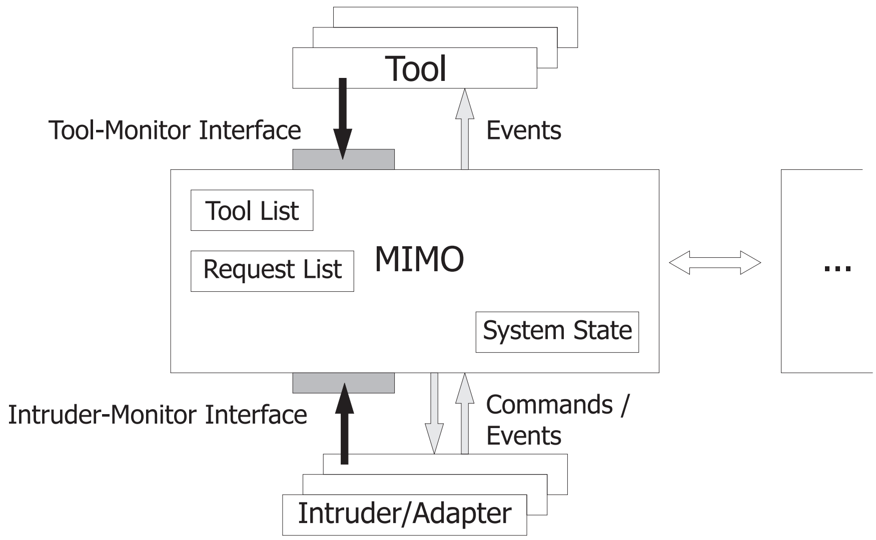

Figure 3: MIMO Architecture

necessary for passing results of asynchronously occurring events (e.g. interactions between entities). Events are passed from MIMO to the tool through a CORBA event channel which is set up during the attachment of the tool.

2.3.2 Intruder/Adapter-Monitor-Interaction

The entrance point for adapters and intruders is the intruder-monitor-interface, which provides methods for attaching and detaching intruders/adapters. After initialization, communication between MIMO and the clients is only handled via two CORBA event channels which are set up at startup. Event channels are mandatory because an asynchronous way of interaction is needed in order to influence the observed system as little as possible. Whenever an “interesting” event occurs within the monitored application, the intruder builds a CORBA event and passes it to MIMO via the monitor-intruder event channel. Similarly, whenever MIMO needs to pass information to the intruder, e.g. for configuring the intruder, it passes a CORBA event to the intruder via its intruder-monitor event channel. This way of communication results in a decoupled intruder/adapter-MIMO interaction scheme. Moreover, it is very flexible due to the standardized protocol which allows for easy integration of different types of middleware platforms which need to be attached by very different intruder/adapter implementations.

For details see [RLRS00].

3 Globus Adapter for MIMO

This section describes the design and implementation of a Globus adapter for MIMO, i.e. a component for collecting information from Globus applications and delivering it to MIMO in an appropriate way.

3.1 MLM Mapping for Globus

First, the MLM model needs to be mapped to Globus entities, yielding the following assignment of MIMO's abstract entities to concrete Globus entities shown in Table 1.

Abstract MIMO layer	*Concrete Globus Entities*
Application	*Globus applications*
Interface	*Globus gatekeeper*
Distributed Objects	*Globus job identification*
Implementation	*Local executable*
Run-time	*Local PID*
Hardware	*Node*

Table 1: Mapping of MIMO layers to Globus job execution

This mapping mainly concentrates on running Globus jobs, which are started on different hosts by using the gatekeeper's interface to authenticate and authorize a user. Other Globus information sources like the heartbeat monitor or MDS could also be integrated into the model, but have not been taken into account up to now.

3.2 Globus Adapter Design Issues

The Globus adapter delivering gathered data to MIMO has to use MIMO's CORBA interfaces for attaching and detaching, and CORBA event channels for asynchronous communication. Therefore, an ORB handling the communication with MIMO needs to be integrated into the adapter. As the Globus adapter needs to be integrated and linked with Globus components, it is conveniently implemented in C like the Globus components used for the prototype implementation. Under given circumstances, it would also be possible to develop a C++ library which could be linked with wrapping C-Code, but the proper approach is to use C for implementing the adapter itself.

3.2.1 ORBs Providing a C Language Mapping

The following criteria were important for selecting a C ORB to implement the Globus adapter:

- Light-weight:
  The ORB should be light in size and performance in order to influence the observed system as little as possible
- Free Availability:
  The chosen ORB should be freely available without any constraints.
- Functionality:
  The ORB has to provide a C-mapping (clearly) and basic naming and event services.
- CORBA compliance:
  Many ORBs still show a lack of CORBA compliance. This results in a lacking interoperability with other ORBs provided by different vendors. As MIMO is implemented using the ORBacus ORB [Obj99], CORBA compliance and interoperability are crucial criteria for the selected ORB.

Unfortunately, the number of ORBs fulfilling the above requirements is rather small, because C-mappings are not provided at all with most ORB implementations. For an overview of ORBs and their properties, also see [Eng99].

After comparing existing ORBs fulfilling our requirements the ORBit ORB [Red99] was chosen. ORBit is developed under the GNOME projects (and therefore freely available with a GNU license), provides a C-mapping, and is a very light-weight and high-performing implementation. Also CORBA compliance and interoperability with ORBacus is given. ORBit's implementation is still ongoing and therefore some bugs can be expected in the code, but the developers are very responsive and provide good support.

The only alternative to ORBit is ILU by Xerox, which also provides a C-mapping, but has the disadvantage of being a very heavy-weight product (supporting even more than "just" CORBA). Thus, ORBit was favored because of its compactness.

3.2.2 Globus Adapter Design

The Globus adapter is implemented as a C library providing basic functions for attaching to and detaching from MIMO, and for providing event-based asynchronous data in case of state changes. At present, only event flow from the adapter to MIMO has been implemented, whereas manipulation events for e.g. steering applications by MIMO have not yet been considered.

4 Globus Adapter Realization

As said above, the Globus adapter is implemented as a set of C functions to be called from Globus components. The instrumentation of the C code itself can be done in various ways, e.g. source code instrumentation or transparent instrumentation of other libraries.

4.1 Globus Adapter Interface

Figure 4 shows the functions that are available for communicating with MIMO.

```
/* attach to and detach from MIMO: */

int MIMO_attach(int argc, char *argv[]);
int MIMO_detach(void);

/* push events: */

/* generic event: */
int MIMO_event(char *evtype, CORBA_any descr);

/* object lifecycle and interaction: */
int MIMO_newEvent(char *entities[6] );
int MIMO_delEvent(int layer,char *entname);
int MIMO_interactionEvent(int layer1, char *entname1,
                          int layer2, char* entname2,
                          char *method);
```

Figure 4: Globus Adapter Functions

The attach and detach operations are used for setting up and terminating connections to MIMO. Whenever no MIMO is running within the environment, any function call to the adapter will fail without further influence to the observed program.

4.2 Attach and Detach Operations

The parameters to be passed with the above attach and detach operations are defined in MIMO's intruder-monitor interface defined using the CORBA IDL interface description language shown in Figure 5.

```
module MIMO {
  typedef long T_IntruderID;

  interface IntruderMonitor {

    // Attach:
    // result is an intruder id, and two event
    // channels, one for
    // communication from the intruder to the monitor,
    // the other one for a command channel from the
    // monitor to the intruder

    T_IntruderID attach(in string iname,
         out CosEventChannelAdmin::EventChannel ievch,
         out CosEventChannelAdmin::EventChannel mevch)
      raises(MIMOException);

    // Detach:
    // intruder id needed as parameter
    void detach(in T_IntruderID iid)
        raises(MIMOException);
  };
};
```

Figure 5: MIMO Intruder-Monitor Interface

4.3 Event Passing

The generic MIMO_event operation can be used to generate any kind of event for MIMO. Its format contains a string giving the type of the event and a CORBA any keeping the description. The time of the event is automatically added by the Globus adapter.

For often occurring events like lifecycle events (construction and destruction of objects) and object interaction, additional operations have been added to the Globus adapter. The operations are MIMO_newEvent, MIMO_delEvent, and MIMO_interaction event.

The parameters passed to them comply with the IDL description of possible MIMO intruder events shown in Figure 6. All MIMO types are based on the general notion of an entity, who's IDL description is given in Figure 7.

```
module MIMO {
  // general event definition:
  struct IntruderEvent {
    // time of event:
    string time; // long long (= java long)
                    not yet supported by ORBacus
    // type of event:
    string etype; // new/del, interaction, mw_call
    // description of the event:
    any description;
  };
  // new event:
  struct newIntruderEvent {
    EntityList e6list;   // list of entitiy-ids
                            on all layers
  };
  // delete event:
  struct delIntruderEvent {
    Entity delEnt;       // the entity to be deleted
  };
  // interaction event:
  struct interactionIntruderEvent {
    Entity e1,e2;  // the two interacting entities
    string method; // the method name
  };
};
```

Figure 6: MIMO Intruder Events

```
module MIMO {
  typedef long ToolID;
  typedef long RequestID;
  typedef string RequestName;
  typedef long Result;

  // Possible Entity Types:
  enum EntityType { E_Appl, E_Interface, E_Object,
                    E_Impl, E_Runtime, E_Node };

  // an identifier for an entity:
  typedef string EntityID;

  // a single entity:
  struct Entity {
    EntityType etype;
    EntityID   eid;
  };

  // a list of arbitrary entities:
  typedef sequence<Entity> EntityList;

  // a list of entities with the same type:
  typedef sequence<EntityID> EntityIDList;
  struct EEntityList {
    EntityType   etype;
    EntityIDList elist;
  };
};
```

Figure 7: MIMO Entity Definition

5 Instrumentation of Globus Components

An important concept of MIMO is not to depend on any special kind of instrumentation. Therefore various instrumentation techniques can be applied, as long as communication with MIMO is handled through its IDL interfaces and event channels.

5.1 Source Code Instrumentation

For simplicity, the gatekeeper and jobmanager Globus components were instrumented by inserting code into the Globus sources. The following things need to be done to apply this kind of instrumentation:

- Include the "globus-adap.h" header file
- Add MIMO_attach and MIMO_detach operations at the beginning and at the end
- Add MIMO event operations wherever events need to be generated
- Link the Globus adapter library with the executable

Clearly, the drawback of this approach is that sources have to be modified and rebuilt, but the advantage is that events can be generated with a very fine granularity, determined by the programmer in order to concentrate on specific aspects to be monitored. Moreover, the basic changes to the sources are kept very small as only three lines of code have to be added.

5.2 Gatekeeper and Jobmanager Instrumentation

In order to test the techniques described above, the Globus gatekeeper and jobmanager have been instrumented. By observing the gatekeeper, authentication and authorization behavior within Globus can be monitored, because the gatekeeper serves as the "entrance point" for users to a machine; this is the reason why is is classified in the interface layer of the MLM. Instrumenting of the jobmanager then allows for monitoring of running Globus jobs and their queuing system behavior.

Thus, instrumentation of both components with the approach described above enables generation of events for MIMO whenever new job submissions arrive at the observed nodes.

Of course, the instrumentation technique used here could be improved in order not to require source code manipulation, e.g. by means of instrumenting libraries used by the gatekeeper and jobmanager. In this case, no recompilation and no relinking would be necessary, but event evaluation gets more complicated in this case.

6 Conclusion and Future Work

This report showed how to connect job submission components of the Globus metacomputing infrastructure to the MIMO monitoring system. For this purpose, an adapter for the CORBA C-language mapping has been developed and successfully deployed with the Globus gatekeeper and jobmanager components. This proof of concept demonstrates the applicability and flexibility of MIMO's multi-layer-monitoring approach. Moreover, the developed C-adapter can be used for other C-based middleware platforms.

What remains to be done is a more sophisticated instrumentation of Globus components, including performance measurements and evaluation of the gained results using appropriate tools.

Also, other Globus information sources could be included to provide information for MIMO, for example the Globus heartbeat monitor and Globus MDS. Hence, MIMO could then be seen as an "information-source broker", providing a central point for collecting different information coming from different sources. This could be a step to a more general approach for discovering information or event sources within a distributed computing environment.

A Software

All the developments were made using the following software environment:

Hardware/OS	Sun with Solaris 2.6
MIMO:	
Java	JDK 1.2
ORB	ORBacus Java 3.2
Globus Adapter:	
C compiler	gcc (egcs-1.1.2 release)
ORB	ORBit 0.5.1

The interoperability between ORBacus 3.2 and ORBit 0.5.1 caused no problems.

References

[Eng99] Ben Eng. The Free CORBA page – ORB Core Feature Matrix, Nov 1999. http://www.vex.net/ ben/corba/orbmatrix.html.

[FK98] I. Foster and C. Kesselman. The Globus project: A status report. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–18. IEEE Computer Society Press, 1998.

[LWSB97] Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, and Arndt Bode. *OMIS — On-Line Monitoring Interface Specification (Version 2.0)*, volume 9 of *Research Report Series, Lehrstuhl für Rechnertechnik und Rechnerorganisation (LRR-TUM), Technische Universität München*. Shaker, Aachen, 1997.

[Obj99] Object Oriented Concepts Inc. ORBacus, Nov 1999. http://www.ooc.com/ob/.

[Rac99] Günther Rackl. Multi-Layer Monitoring in Distributed Object-Environments. In Lea Kutvonen, Hartmut König, and Martti Tienari, editors, *Distributed Applications and Interoperable Systems II — IFIP TC 6 WG 6.1 Second International Working Conference on Distributed Applications and Interoperable Systems (DAIS'99)*, pages 265–270, Helsinki, June 1999. Kluwer Academic Publishers.

[Red99] Red Hat Inc. ORBit, Nov 1999. http://www.labs.redhat.com/orbit/.

[RLRS00] Günther Rackl, Markus Lindermeier, Michael Rudorfer, and Bernd Süss. MIMO — An Infrastructure for Monitoring and Managing Distributed Middleware Environments. April 2000. Accepted for Middleware'2000.

[RZB99] Günther Rackl, Ivan Zoraja, and Arndt Bode. Distributed Object Computing: Principles and Trends. In *Proceedings of SoftCOM '99*, pages 121–132, Oct 1999.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

Liste aller erschienenen Berichte von 1990-1994
auf besondere Anforderung

342/01/95 A	Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids
342/02/95 A	Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law
342/03/95 A	Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space
342/04/95 A	Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes
342/05/95 A	Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems
342/06/95 A	Maximilian Fuchs: Formal Design of a Model-N Counter
342/07/95 A	Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology
342/08/95 A	Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies
342/09/95 A	Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness
342/10/95 A	Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design
342/11/95 A	Thomas Schnekenburger: The ALDY Load Distribution System
342/12/95 A	Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm
342/13/95 A	Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming
342/14/95 A	Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks
342/15/95 A	Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs
342/16/95 A	Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik
342/17/95 A	Georg Stellner: Using CoCheck on a Network of Workstations
342/18/95 A	Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications
342/19/95 A	Thomas Schnekenburger: Integration of Load Distribution into ParMod-C
342/20/95 A	Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

Reihe A

342/21/95 A	Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control
342/22/95 A	Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes
342/23/95 A	Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation
342/24/95 A	M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems
342/01/96 A	Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries
342/02/96 A	Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs
342/03/96 A	Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes
342/04/96 A	Thomas Huckle: Efficient Computation of Sparse Approximate Inverses
342/05/96 A	Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-line Monitoring Interface Specification
342/06/96 A	Ekkart Kindler: A Compositional Partial Order Semantics for Petri Net Components
342/07/96 A	Richard Mayr: Some Results on Basic Parallel Processes
342/08/96 A	Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht
342/09/96 A	P.P. Spies, C. Eckert, M. Lange, D. Marek, R. Radermacher, F. Weimer, H.-M. Windisch: Sprachkonzepte zur Konstruktion verteilter Systeme
342/10/96 A	Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFSLib – A File System for Parallel Programming Environments
342/11/96 A	Manfred Broy, Gheorghe Ştefănescu: The Algebra of Stream Processing Functions
342/12/96 A	Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete
342/13/96 A	Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks
342/14/96 A	Giannis Bozas, Michael Jaedicke, Andreas Listl, Bernhard Mitschang, Angelika Reiser, Stephan Zimmermann: On Transforming a Sequential SQL-DBMS into a Parallel One: First Results and Experiences of the MIDAS Project
342/15/96 A	Richard Mayr: A Tableau System for Model Checking Petri Nets with a Fragment of the Linear Time μ -Calculus
342/16/96 A	Ursula Hinkel, Katharina Spies: Anleitung zur Spezifikation von mobilen, dynamischen Focus-Netzen
342/17/96 A	Richard Mayr: Model Checking PA-Processes

Reihe A

342/18/96 A	Michaela Huhn, Peter Niebert, Frank Wallner: Put your Model Checker on Diet: Verification on Local States
342/01/97 A	Tobias Müller, Stefan Lamberts, Ursula Maier, Georg Stellner: Evaluierung der Leistungsfähigkeit eines ATM-Netzes mit parallelen Programmierbibliotheken
342/02/97 A	Hans-Joachim Bungartz and Thomas Dornseifer: Sparse Grids: Recent Developments for Elliptic Partial Differential Equations
342/03/97 A	Bernhard Mitschang: Technologie für Parallele Datenbanken - Bericht zum Workshop
342/04/97 A	nicht erschienen
342/05/97 A	Hans-Joachim Bungartz, Ralf Ebner, Stefan Schulte: Hierarchische Basen zur effizienten Kopplung substrukturierter Probleme der Strukturmechanik
342/06/97 A	Hans-Joachim Bungartz, Anton Frank, Florian Meier, Tilman Neunhoeffer, Stefan Schulte: Fluid Structure Interaction: 3D Numerical Simulation and Visualization of a Micropump
342/07/97 A	Javier Esparza, Stephan Melzer: Model Checking LTL using Constraint Programming
342/08/97 A	Niels Reimer: Untersuchung von Strategien für verteiltes Last- und Ressourcenmanagement
342/09/97 A	Markus Pizka: Design and Implementation of the GNU INSEL-Compiler gic
342/10/97 A	Manfred Broy, Franz Regensburger, Bernhard Schätz, Katharina Spies: The Steamboiler Specification - A Case Study in Focus
342/11/97 A	Christine Röckl: How to Make Substitution Preserve Strong Bisimilarity
342/12/97 A	Christian B. Czech: Architektur und Konzept des Dycos-Kerns
342/13/97 A	Jan Philipps, Alexander Schmidt: Traffic Flow by Data Flow
342/14/97 A	Norbert Fröhlich, Rolf Schlagenhaft, Josef Fleischmann: Partitioning VLSI-Circuits for Parallel Simulation on Transistor Level
342/15/97 A	Frank Weimer: DaViT: Ein System zur interaktiven Ausführung und zur Visualisierung von INSEL-Programmen
342/16/97 A	Niels Reimer, Jürgen Rudolph, Katharina Spies: Von FOCUS nach INSEL - Eine Aufzugssteuerung
342/17/97 A	Radu Grosu, Ketil Stølen, Manfred Broy: A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing
342/18/97 A	Christian Röder, Georg Stellner: Design of Load Management for Parallel Applications in Networks of Heterogenous Workstations
342/19/97 A	Frank Wallner: Model Checking LTL Using Net Unfoldings
342/20/97 A	Andreas Wolf, Andreas Kmoch: Einsatz eines automatischen Theorembeweisers in einer taktikgesteuerten Beweisumgebung zur Lösung eines Beispiels aus der Hardware-Verifikation – Fallstudie –
342/21/97 A	Andreas Wolf, Marc Fuchs: Cooperative Parallel Automated Theorem Proving

Reihe A

342/22/97 A T. Ludwig, R. Wismüller, V. Sunderam, A. Bode: OMIS - On-line Monitoring Interface Specification (Version 2.0)
342/23/97 A Stephan Merkel: Verification of Fault Tolerant Algorithms Using PEP
342/24/97 A Manfred Broy, Max Breitling, Bernhard Schätz, Katharina Spies: Summary of Case Studies in Focus - Part II
342/25/97 A Michael Jaedicke, Bernhard Mitschang: A Framework for Parallel Processing of Aggregat and Scalar Functions in Object-Relational DBMS
342/26/97 A Marc Fuchs: Similarity-Based Lemma Generation with Lemma-Delaying Tableau Enumeration
342/27/97 A Max Breitling: Formalizing and Verifying TimeWarp with FOCUS
342/28/97 A Peter Jakobi, Andreas Wolf: DBFW: A Simple DataBase FrameWork for the Evaluation and Maintenance of Automated Theorem Prover Data (incl. Documentation)
342/29/97 A Radu Grosu, Ketil Stølen: Compositional Specification of Mobile Systems
342/01/98 A A. Bode, A. Ganz, C. Gold, S. Petri, N. Reimer, B. Schiemann, T. Schnekenburger (Herausgeber): "Anwendungsbezogene Lastverteilung", ALV'98
342/02/98 A Ursula Hinkel: Home Shopping - Die Spezifikation einer Kommunikationsanwendung in FOCUS
342/03/98 A Katharina Spies: Eine Methode zur formalen Modellierung von Betriebssystemkonzepten
342/04/98 A Stefan Bischof, Ernst W. Mayr: On-Line Scheduling of Parallel Jobs with Runtime Restrictions
342/05/98 A St. Bischof, R. Ebner, Th. Erlebach: Load Balancing for Problems with Good Bisectors and Applications in Finite Element Simulations: Worst-case Analysis and Practical Results
342/06/98 A Giannis Bozas, Susanne Kober: Logging and Crash Recovery in Shared-Disk Database Systems
342/07/98 A Markus Pizka: Distributed Virtual Address Space Management in the MoDiS-OS
342/08/98 A Niels Reimer: Strategien für ein verteiltes Last- und Ressourcenmanagement
342/09/98 A Javier Esparza, Editor: Proceedings of INFINITY'98
342/10/98 A Richard Mayr: Lossy Counter Machines
342/11/98 A Thomas Huckle: Matrix Multilevel Methods and Preconditioning
342/12/98 A Thomas Huckle: Approximate Sparsity Patterns for the Inverse of a Matrix and Preconditioning
342/13/98 A Antonin Kucera, Richard Mayr: Weak Bisimilarity with Infinite-State Systems can be Decided in Polynomial Time
342/01/99 A Antonin Kucera, Richard Mayr: Simulation Preorder on Simple Process Algebras
342/02/99 A Johann Schumann, Max Breitling: Formalisierung und Beweis einer Verfeinerung aus FOCUS mit automatischen Theorembeweisern – Fallstudie –

Reihe A

342/03/99 A	M. Bader, M. Schimper, Chr. Zenger: Hierarchical Bases for the Indefinite Helmholtz Equation
342/04/99 A	Frank Strobl, Alexander Wisspeintner: Specification of an Elevator Control System
342/05/99 A	Ralf Ebner, Thomas Erlebach, Andreas Ganz, Claudia Gold, Clemens Harlfinger, Roland Wismüller: A Framework for Recording and Visualizing Event Traces in Parallel Systems with Load Balancing
342/06/99 A	Michael Jaedicke, Bernhard Mitschang: The Multi-Operator Method: Integrating Algorithms for the Efficient and Parallel Evaluation of User-Defined Predicates into ORDBMS
342/07/99 A	Max Breitling, Jan Philipps: Black Box Views of State Machines
342/08/99 A	Clara Nippl, Stephan Zimmermann, Bernhard Mitschang: Design, Implementation and Evaluation of Data Rivers for Efficient Intra-Query Parallelism
342/09/99 A	Robert Sandner, Michael Mauderer: Integrierte Beschreibung automatisierter Produktionsanlagen - eine Evaluierung praxisnaher Beschreibungstechniken
342/10/99 A	Alexander Sabbah, Robert Sandner: Evaluation of Petri Net and Automata Based Description Techniques: An Industrial Case Study
342/01/00 A	Javier Esparza, David Hansel, Peter Rossmanith, Stefan Schwoon: Efficient Algorithm for Model Checking Pushdown Systems
342/02/00 A	Barbara König: Hypergraph Construction and Its Application to the Compositional Modelling of Concurrency
342/03/00 A	Max Breitling and Jan Philipps: Verification Diagrams for Dataflow Properties
342/04/00 A	Günther Rackl: Monitoring Globus Components with MIMO

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B	Wolfgang Reisig: Petri Nets and Algebraic Specifications
342/2/90 B	Jörg Desel: On Abstraction of Nets
342/3/90 B	Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems
342/4/90 B	Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme
342/1/91 B	Barbara Paech: Concurrency as a Modality
342/2/91 B	Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox - Anwenderbeschreibung
342/3/91 B	Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen
342/4/91 B	Werner Pohlmann: A Limitation of Distributed Simulation Methods
342/5/91 B	Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis
342/6/91 B	Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme
342/7/91 B	W. Reisig: Concurrent Temporal Logic
342/1/92 B	Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support
	Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support
342/2/92 B	Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen
342/1/93 B	Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung
342/2/93 B	Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick
342/1/94 B	Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS