OMIS

On-line Monitoring Interface Specification

Roland Wismüller
Thomas Ludwig, Vaidy Sunderam, Arndt Bode

Sept. 15, 1995

Technische Universität München
Lehrstuhl Rechnertechnik und Rechnerorganisation (LRR-TUM)
Institut für Informatik, 80290 München

Tel.: +49-89-2105-8164

email: wismuell@informatik.tu-muenchen.de

Agenda

1. Presentation of OMIS project
   - Motivation and Goals
   - Requirements
   - Monitor structure
   - Monitoring Interface
2. Open Discussion on Monitoring Interface

Motivation

Parallel programming environments suffer from a lack of on-line development tools, especially for distributed memory architectures

E.g. the PVM environment:

- several on-line tools are available, but
- built-in monitoring too restrictive (only event traces)

  $\Rightarrow$ new tools difficult to implement
- no common monitor layer exists

  $\Rightarrow$ tools can not be used in combination

Every new tool and every new target architecture requires implementation of a new monitoring system

$\Rightarrow$ tools can not easily be ported

$\Rightarrow$ $n * m$ different monitoring systems for n tools on m architectures

Project Goals

1. Provide an open interface standard for on-line monitoring which can be used and extended by others
2. Design and implement an OMIS compliant monitoring system for PVM running on networks of workstations
3. Design and implement OMIS based tools

   ⇒ THE TOOL-SET
4. Coordinate extensions being made to OMIS

All products will be released under the GNU license conditions

Requirements

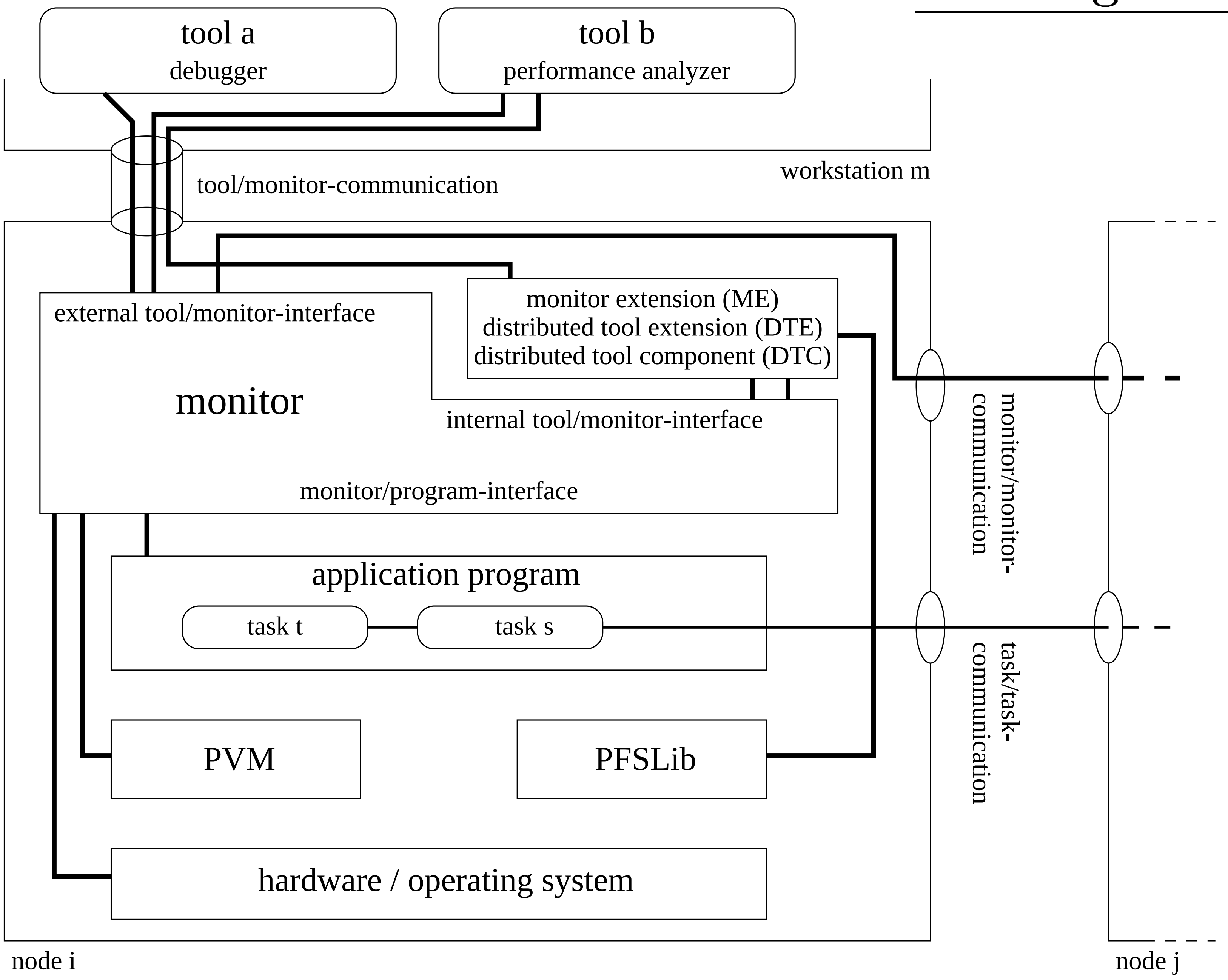
Situations to support:

- different tools: performance analyzers, debuggers, load balancers, ...
- different tool architectures: centralized, completely distributed, ...
- different event processing: traces (no standard format!), on-line evaluation with distributed preprocessing, ...
- special needs of some tools, extensions to programming model

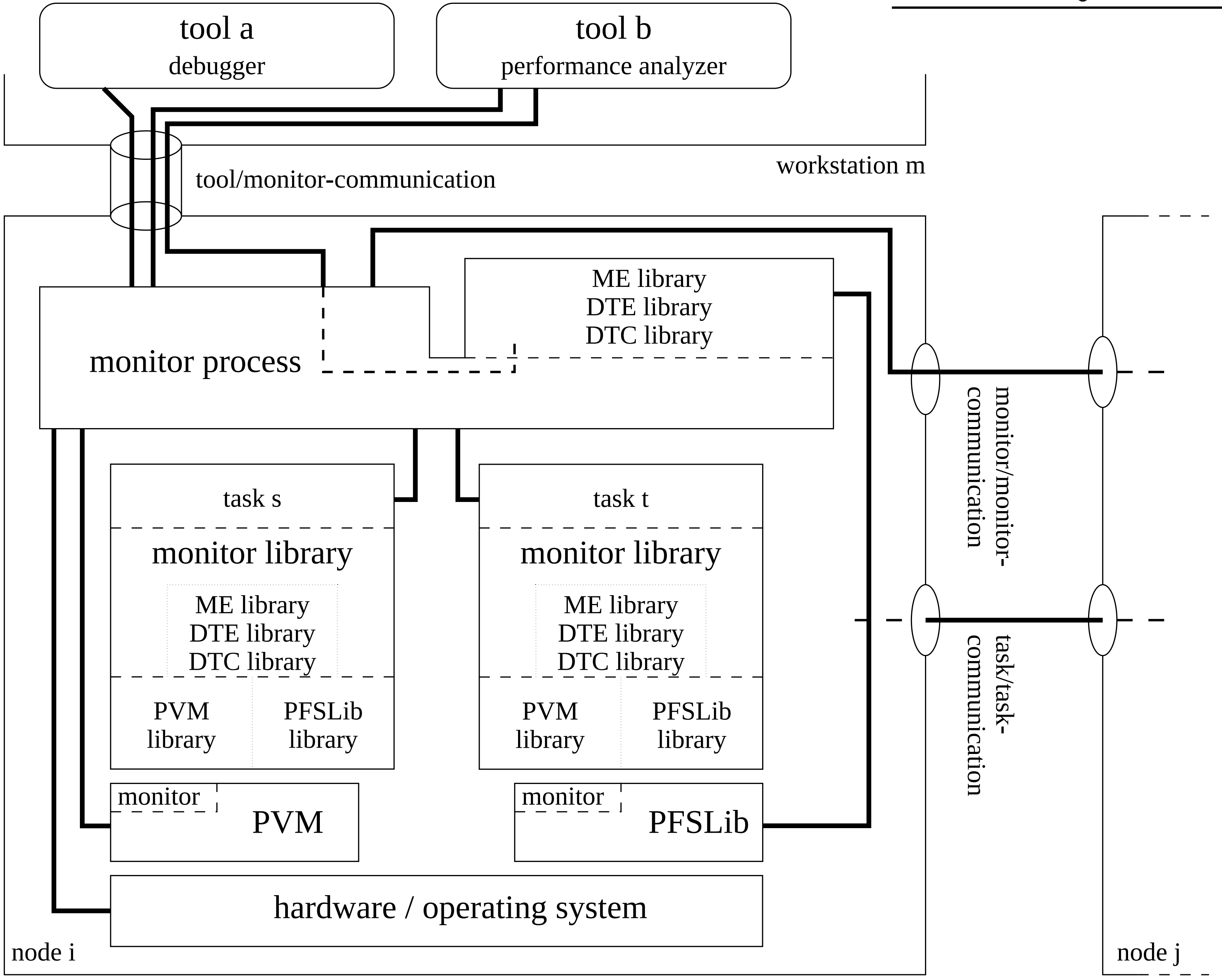
Requirements for monitoring interface:

- Versatility and flexibility
  $\rightsquigarrow$ different tools, observed objects, distribution of tools
- Extensibility
  $\rightsquigarrow$ new tool concepts, new objects (e.g. I/O, language constructs)
- Efficiency
  $\rightsquigarrow$ minimize communication between tool and monitoring system
  $\rightsquigarrow$ support efficient distributed processing of events

The Logical Structure



The Physical Structure



The Monitoring Interface (1)

OMIS offers services that are split into requests and (any number of) replies:

- Manipulation services
- Synchronous services
- Asynchronous services (events)
- Monitor control services

General syntax of requests (based on event-action scheme):

$$
\begin{array}{lll}
\mathit{request} & ::= & [\ \mathit{event} :\ ]\ \mathit{action\_list} \\
\mathit{action\_list} & ::= & \mathit{action1},\ \mathit{action2},\ \ldots \qquad \text{(parallel execution)} \\
 & | & \mathit{action1};\ \mathit{action2};\ \ldots \qquad \text{(sequential execution)} \\
\mathit{event/action} & ::= & \mathit{comm\_id}\quad \mathit{target\_ids}\quad \mathit{service\_no}\ (\ \mathit{params}\ )
\end{array}
$$

Notes:

- Each monitor accepts requests for any set of monitors.
- The special symbols \$1, \$2, etc. can be used to reference result parameters of events.

Monitoring Services

1. **Processes:**
   - start, stop, continue, kill, call function, goto address, write memory, ...
   - list processes, process information, stack backtrace, read registers, ...
   - spawn, terminate, block/unblock, signal, breakpoint, single step, ...
2. **Messages:**
   - insert/delete message, tag message, ...
   - queue information, type information, ...
   - start/end send/receive, receipt of tagged message, ...
3. **Hardware:**
   - number of nodes, list of machines, node information, ...
   - addition/deletion of hosts, ...
4. **Monitor:**
   - initialization, define/raise user events, enable/disable events, ...
   - user defined events, alarm clock, ...

Extensions

Three reasons why extensions should be linked into the monitoring system:

1. Extensions to PVM, e.g. I/O systems like PFSlib or PIOUS
   ⇒ new objects and new services
2. Different handling of events, e.g. XPVM will trace events, a performance analyzer will calculate some metrics.
   ⇒ tool-specific actions for events
3. Fully distributed tools, like load balancers
   ⇒ procedural interface for efficiency reasons

⇒ Allow libraries with new services and initialization routines.

Problem: Name conflicts ⇒ Define fixed ranges of service numbers.

- Each group making extensions gets its own range on request
- Each group provides initialization routine, called during startup
- Dummy library allows monitor to be used without extentions

The Monitoring Interface (2)

Example 1: Performance Analysis

```
start_send(4178): start_integrator(1), add_counter(2,$3)
end_send(4178):   stop_integrator(1)
start_barrier():  trace("barrier0",$1,$2,$3)
end_barrier():    trace("barrier1",$1,$2)
```

Example 2: Debugging

```
define_user_event(5)
start_send(1123):       raise_event(5,$1)
breakpoint(1234,fe08): raise_event(5,$1)
user_event(5):          return($1); stack_backtrace($1);
                        read_regs($1,0,32)
```

The internal interface accepts the same syntax, but uses an additional callback:

`OMIS_request("...", callback)`

$\Rightarrow$ invoke `callback("...")` when reply is available.

Open Problems and Questions

- Is such an interface useful for you?
- Would you use it for new tools? Would you port existing tools?
- Are there additional requirements not met by this interface?
- Are there services of common interest (e.g. timers/traces) that should be added?
- Is there a better way to do extensions?
- Are there tools that need information on more than one application?
- Which kind of communication should we use (PVM, sockets)?
- Should there be a synchronous, blocking form of the internal interface?
- Actions may be called from different processes. How can we handle this?

Please:

- Look at our written document `http://wwwbode.informatik.tu-muenchen.de/parallelrechner/tools/omis.html`
- Send comments to {`ludwig,wismuell`}`@informatik.tu-muenchen.de`