Konzeption portabler Werkzeugumgebungen

Roland Wismüller, Thomas Ludwig

Teilprojektleiter:
Arndt Bode, Hermann Hellwagner

Ergänzungsausstattung:
Stefan Lamberts, Georg Stellner,
Stefan Petri, Jörg Trinitis,
Klaus Tilk

Grundausstattung:
Thomas Ludwig, Roland Wismüller,
Wolfgang Karl, Markus Leberecht,
Peter Luksch, Robert Lindhof

Agenda

- ☐ Parallele Werkzeuge in A1
- ☐ Konventionelle Werkzeugumgebungen
- ☐ OMIS-basierte Werkzeugumgebungen
- ☐ OMIS-basierte Werkzeugentwicklung
- ☐ Anforderungen an die Schnittstelle
- ☐ Struktur der Monitorschnittstelle
- ☐ Implementierung des Monitorsystems
- ☐ OMIS/OCM im SFB 342

Parallele Werkzeuge in A1

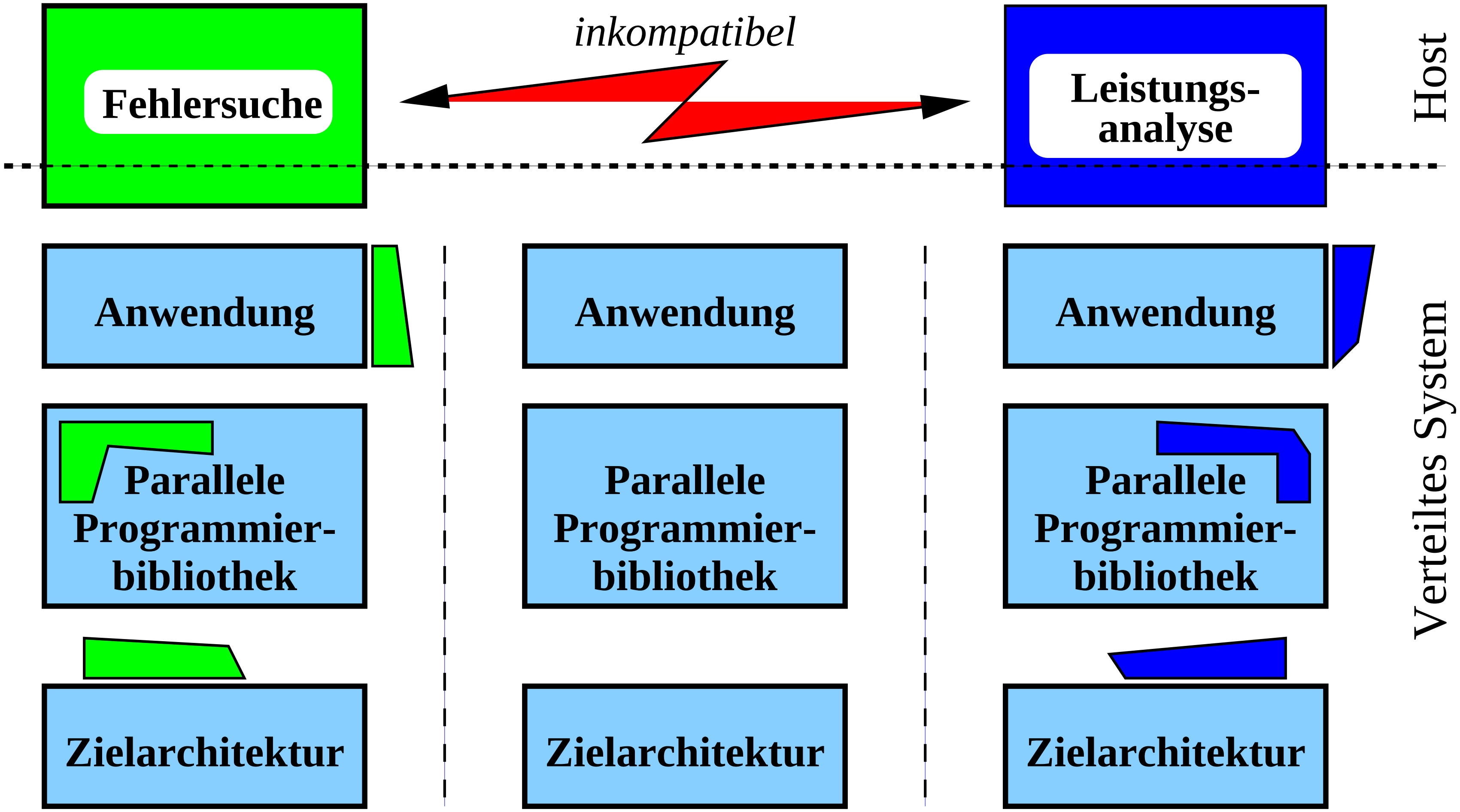
Anforderungen der Anwender

- **Manipulationen am laufenden Programm**
- **Gleichzeitige Verwendung mehrerer Werkzeuge:**
  Fehlersuche auf Sicherungspunkt wiederaufsetzbar
- **A1-Werkzeuge:** Detop, Patop, Vistop, Lastausgleich, Ressourcenverwaltung
- **UPAS** (Universell programmierbare Architektur und Basis-Software)

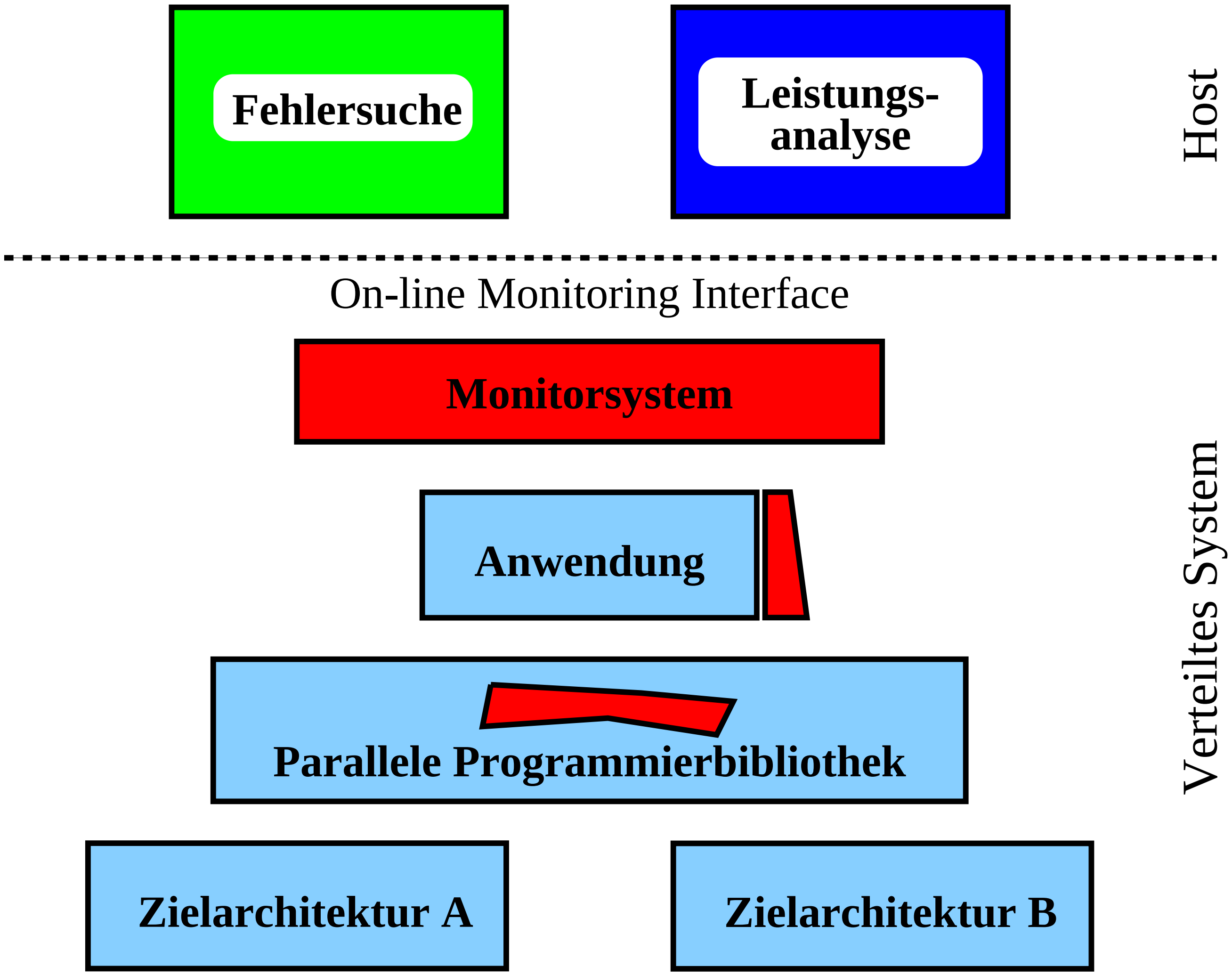
Aufgabenstellung im Berichtszeitraum

- Portierung der Werkzeuge auf andere Rechner und Programmiermodelle (Universalität)
- Methoden der Integration der Werkzeuge, Konstruktion von Werkzeugumgebungen

Konventionelle Werkzeugumgebungen



OMIS-basierte Werkzeugumgebungen



OMIS-basierte Werkzeugentwicklung

Unabhängigkeit

- Entwickler von **Werkzeugen** — Entwickler von **Monitorsystemen**

Interoperable Werkzeuge

- Interaktion über gemeinsame Schnittstelle
- Z. B. Umschalten von Visualisierer auf Fehlersuche
- Z. B. Kopplung interaktiver und automatischer Werkzeuge

Uniforme Werkzeuge

- Portierung des Monitorsystem ⇒ sofortige Verfügbarkeit aller vorhandenen Werkzeuge
- Entwicklung neuer Werkzeuge ⇒ sofortige Verfügbarkeit auf allen OMIS-Plattformen

Anforderungen an die Schnittstelle

Allgemeingültigkeit und Flexibilität

- Verschiedene Werkzeuge zu verschiedenen Zwecken
- Verschiedene Werkzeugarchitekturen
- Verschiedene programmiersprachliche Objekte

Erweiterbarkeit

- Neue, zukünftige Werkzeugkonzepte
- Neue zu überwachende Objekte
- Neue Programmiermodelle

Effizienz

- Geringes Kommunikationsaufkommen zwischen Werkzeugen und Monitorsystem

Struktur der Monitorschnittstelle

Diensteklassen

- **Informationsdienste** z. B. *liefere_knoteninformation*
- **Manipulationsdienste** z. B. *halte_prozeß_an*
- **Benachrichtigungsdienste** z. B. *prozeß_erreicht_adresse*

Dienstanforderungen

- Ereignis-Aktions-Relationen
- Z. B. „*prozeß_erreicht_adresse : halte_alle_prozesse_an*“

Schnittstellenfunktion

- Hauptfunktion: `omis_request(`*dienstanforderung*`,`*rückruffunktion*`,...)`
- *rückruffunktion* überträgt Ergebnisse an das Werkzeug

Das Überwachungssystem

Schnittstellenfunktion (für Werkzeuge und Erweiterungen):

```
Omis_reply
omis_request( char * request,
              void (* callback)(Omis_reply reply, void *param),
              void *param,
              Omis_flags flags )
```

Syntax der Ereignis-Aktionenliste:

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

Eine Aktionenliste kann nebenläufig ausgeführt werden; ein Strichpunkt wirkt als globale Synchronisation

Beispiel: Fehlersuche

```
thread_has_started_lib_call([],"pvm_send") :
    proc_get_info([$proc],12)  thread_get_backtrace($thread,1)
```

Wenn irgendein Prozeß `pvm_send` aufruft: liefere seine `tid`, seinen Argumentenvektor und die Rücksprungadresse des Aufrufs

```
thread_reached_addr([p_12,p_34,p_44],0xfe08) : thread_stop([])
```

Wenn einer der spezifizierten Prozesse den Haltepunkt erreicht: halte die gesamte Anwendung an

Beispiel: Leistungsanalyse

```
thread_has_started_lib_call([p_21],"MPI_Send") :
    pt_integrator_start(pt_i_1)  pt_counter_add(pt_c_1,$par5)
```

Wenn Prozeß p_21 einen Sendeaufruf startet: aktiviere einen integrierenden Zähler und addiere die Nachrichtenlänge ($par5) auf einen Zähler

```
thread_has_ended_lib_call([p_21],"MPI_Send") : pt_integrator_stop(pt_i_1)
```

Wenn der Prozeß den Aufruf beendet: halte den integrierenden Zähler an

Resultat: Verweilzeit im Sendeaufruf und gesendete Nachrichtenmenge

OCM: Die Implementierung von OMIS

OCM = <u>O</u>MIS <u>c</u>ompliant <u>m</u>onitoring system

Implementierung von OMIS für PVM-basierte Anwendungen auf Rechnernetzen

Projektziele:

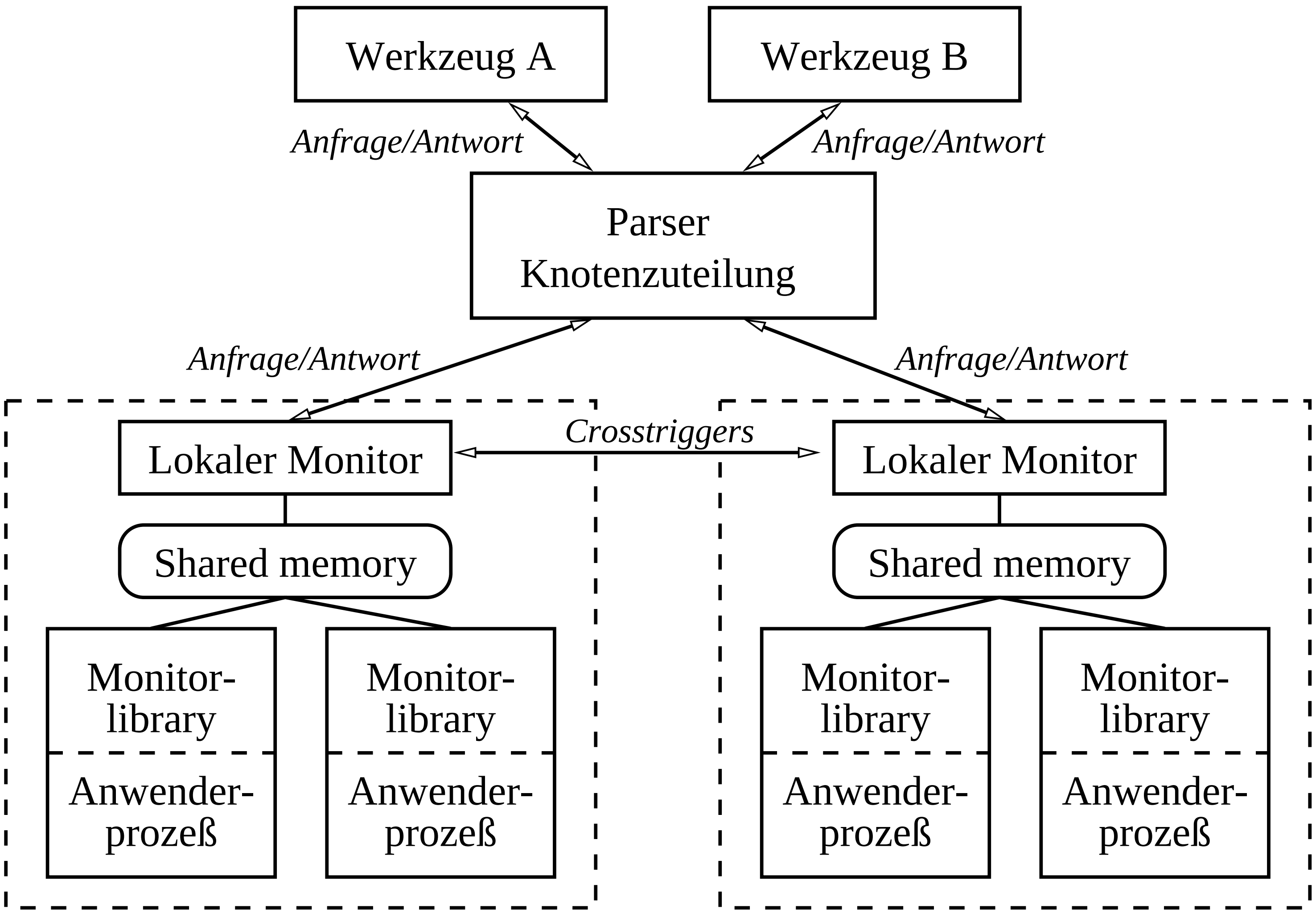
- Verifikation der Anwendbarkeit von OMIS
- Implementierungsplattform für THE TOOL-SET
- Implementierungsplattform für Werkzeuge Dritter
- Referenz für andere OMIS Implementierungen

OCM wird unter GNU Lizenzbedingungen erhältlich sein

Entwurfskonzepte

- Das Überwachungssystem wird aus kommunizierenden Monitoren gebildet, jeweils einem pro Rechnerknoten der virtuellen Maschine
- Globale Aktionsspezifikationen werden zur Definitionszeit an die Knoten verteilt.
- Kommunikation zwischen Monitoren mittels PVM-Mechanismen
- Im ersten Schritt: Zentrales Parsen der Diensteanforderungen
- Ereignisgetriebenes Ausführungsmodell der (Einzelknoten-)Monitore
- Ereigniserkennung und Aktionsausführung sowohl im Monitor als auch in der Anwendung
- Kommunikation zwischen knotenlokalen Monitorkomponenten über gemeinsamen Speicher und Signale

Grobstruktur von OCM



OMIS/OCM im SFB 342

Weiterführung von OCM

- **Implementierungsplattform für die Werkzeuge im SFB**

  Projekt THE TOOL-SET stellt **integrierte interaktive und automatische Werkzeuge** für andere Teilprojekte zur Verfügung

- Validierung der Anwendbarkeit von OMIS
- Implementierungsplattform für Werkzeuge Dritter
- Referenz für andere OMIS-konforme Implementierungen

Weiterführung von OMIS

- **Ausbau für Programmiermodell des gemeinsamen verteilten Speichers**
- Formale Analyse semantischer Aspekte mit Partnern im SFB