Anwendungsorientierte Online-Leistungsanalyse paralleler und verteilter Programme

Roland Wismüller

Institut für Softwarewissenschaft, Universität Wien
LRR-TUM, Technische Universität München

Agenda

1 Anwendungsorientierte Leistungsanalyse

1.1 Hintergrund

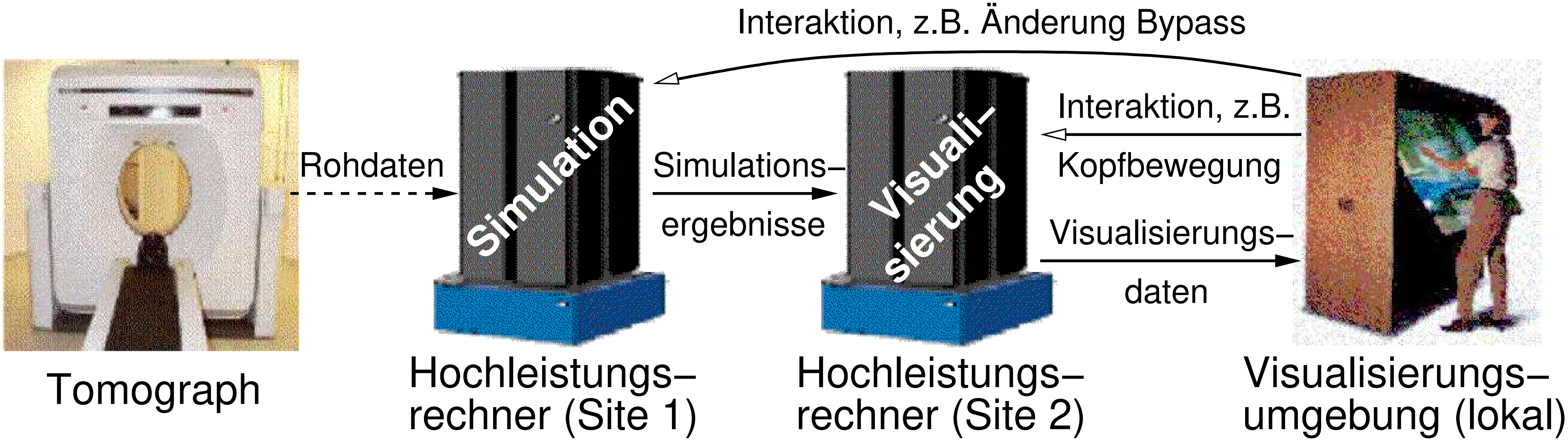
EU-Projekt CrossGrid

- Ziel: Nutzung des Grid für *interaktive* Anwendungen
  - vier parallele / verteilte Anwendungen
  - Werkzeuge (Test, Benchmarks, Leistungsanalyse)
  - Middleware (Portale, Scheduling, Monitoring, Dateizugriff)
  - Testbett
- Teilziel G-PM: nutzbares Leistungsanalyse-Werkzeug für interaktive Anwendungen im Grid
- Laufzeit: März 2002 – Feb. 2005

EU-Arbeitsgruppe APART

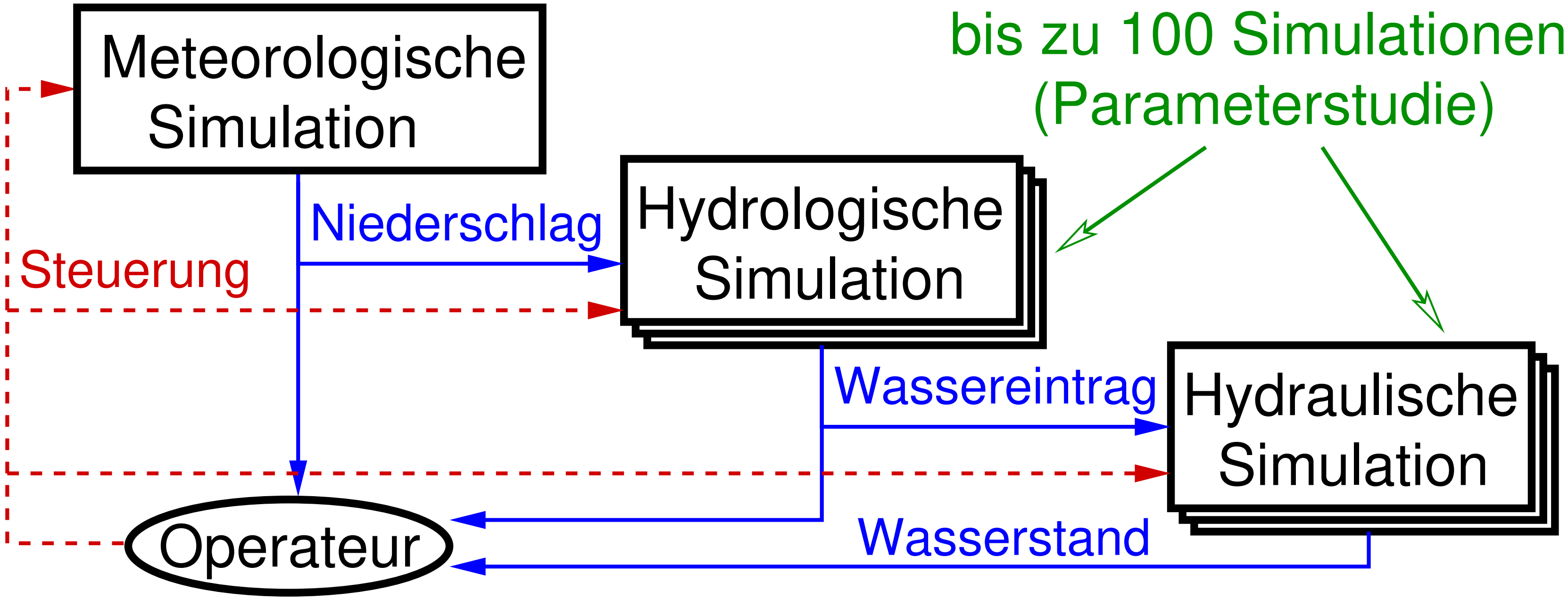
- Automatische Leistungsanalyse

1.1.1 Medizinische Simulation



- Interaktive Simulation und Visualisierung
  - Antwortzeiten sind kritisch
- Spezielle Anforderung an die Leistungsanalyse
  - Messung und Aufschlüsselung der Antwortzeiten
  - Leistungsmessung für einzelne Interaktionen

1.1.2 Hochwasservorhersage



- Interaktive Steuerung (*Steering*)
  - Starten / Beenden von Simulationen (Basis: neue Eingabedaten / Leistungsverhalten / Ressourcen)
- Spezielle Anforderung an die Leistungsanalyse
  - Messung des Konvergenzverhaltens (Rechenfortschritt)
  - Leistungsdaten über die Grid-Infrastruktur

1.1.3 Ziele und Anforderungen

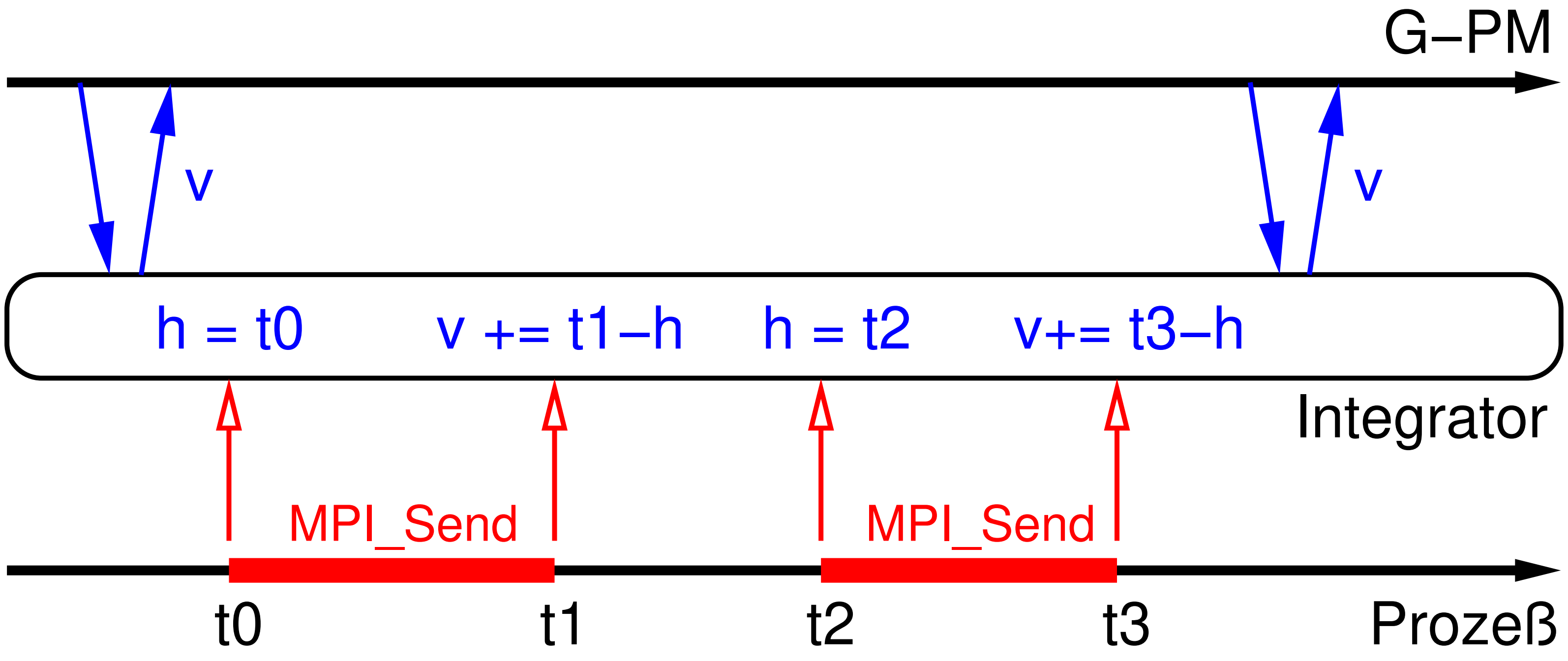
- **Sinnvolle Leistungsmetriken im Grid**
  - Bezug zur Leistung der Grid-Infrastruktur
- **Unterstützung anwendungsspezifischer Metriken**
  - z.B.: Antwortzeit, Konvergenzrate, Daten pro Interaktion, ...
  - mit möglichst minimalem Aufwand für den Anwender
- **Online-Auswertung und Visualisierung**
  - direkter Bezug der Meßergebnisse zu Interaktionen
  - Voraussetzung für *Steering*
- **Effiziente, verteilte Datenauswertung**
  - Skalierbarkeit, Rückwirkung auf Anwendung
  - zentrale Auswertung von Ereignisspuren vermeiden

1.2 Prinzipieller Ansatz

- **Werkzeug G-PM bietet Menge von Standardmetriken an**
  - z.B. Kommunikationszeit, E/A-Volumen; Link-Bandbreite, ...
  - automatische Instrumentierung, transparent für Anwender
- **Anwendung stellt minimale Menge an Information bereit**
  - relevante Ereignisse in der Anwendung
  - Zusammengehörigkeit solcher Ereignisse
  - relevante Daten aus der Anwendung
  - manuelle Quellcode-Instrumentierung (Sonden)
- **G-PM erlaubt Definition neuer Metriken durch Benutzer**
  - Basis: existierende Metriken und Information der Sonden
  - Flexibilität, Wiederverwendung von Metriken / Sonden

1.2.1 Methode der Online-Leistungsmessung

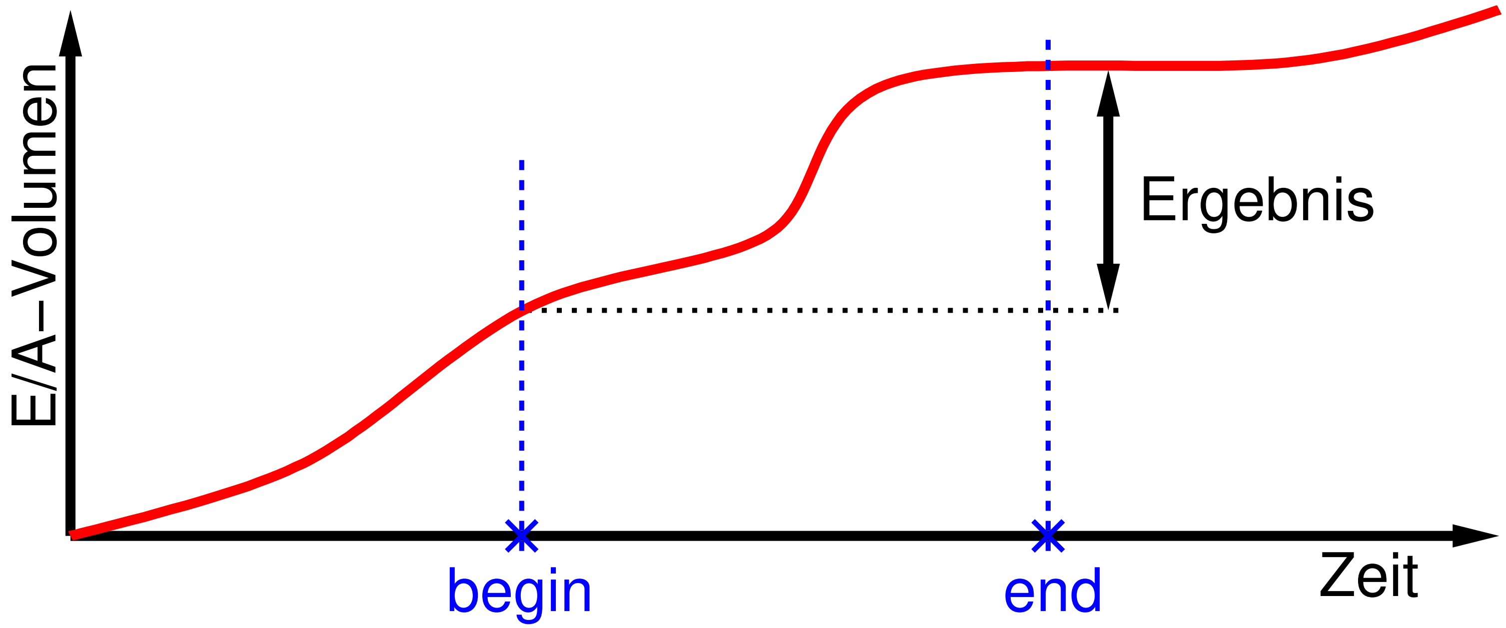
- Überwachung von Anfangs- und Endereignissen (z.B. von Kommunikationsfunktionen)
  - führt zur Aktualisierung von Zählern oder Integratoren
- Periodische Abfrage des Wertes durch G-PM



1.2.2 Prinzip benutzerdefinierte Metriken

Beispiel: E/A-Volumen für eine Interaktion

- Basis: Messung des gesamten E/A-Volumens
- Sonden markieren Anfang und Ende einer Interaktion
- Ergebnis = E/A-Volumen(*end*) – E/A-Volumen(*begin*)



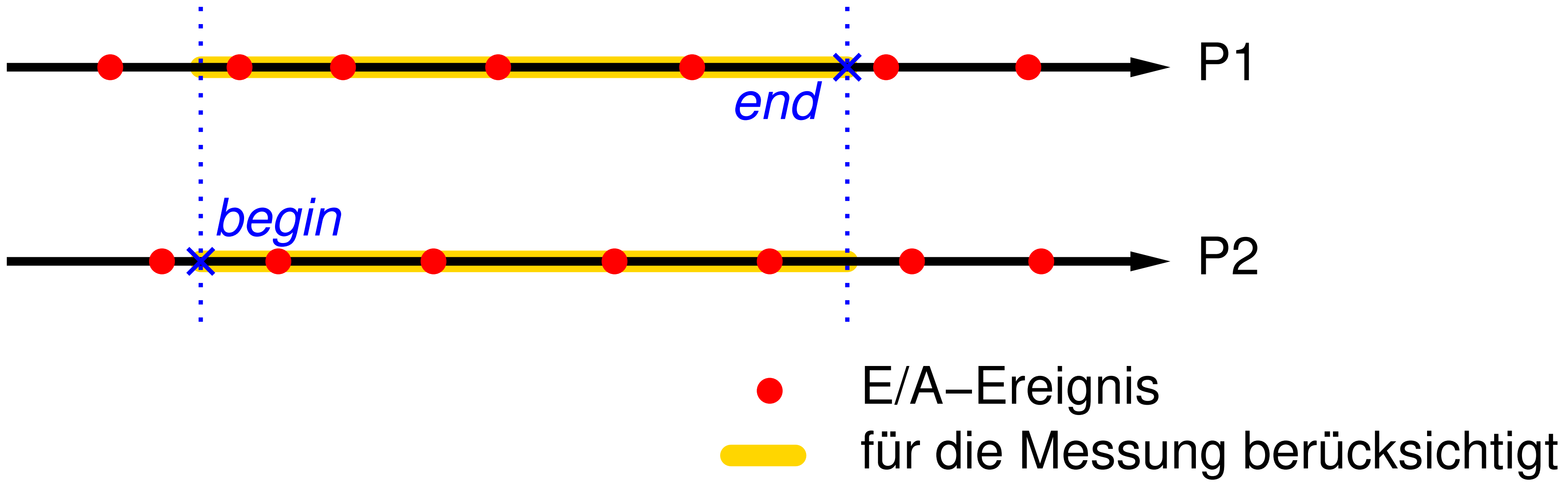
1.3 Probleme / Forschungsgebiete

- (Online-Meßmethode: Semantik, korrekte Realisierung)
- **Semantik benutzerdefinierter Metriken**
  - Ereignisse in verteilten Systemen (zeitliche Abfolge?)
  - Informationsverlust durch Aggregation
- **Einfache, benutzerfreundliche Spezifikation von Metriken**
- **Optimale, verteilte Realisierung von Messungen**

1.3.1 Semantik benutzerdefinierter Metriken

Ereignisse in verteilten Systemen

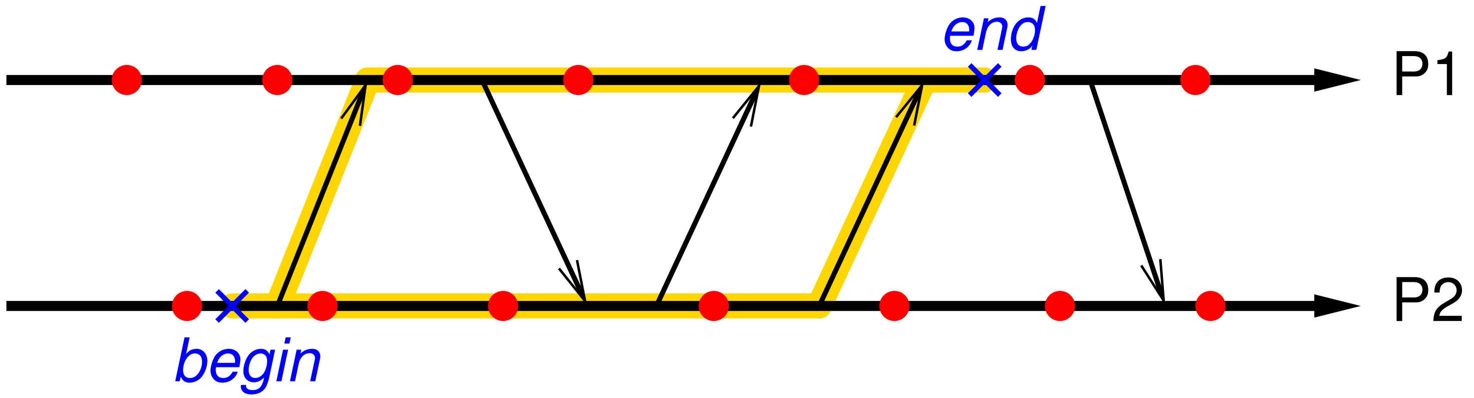
- In verteilten Systemen existiert keine absolute Gleichzeitigkeit
- Die folgende Semantik ist somit nicht realisierbar:



- Meist ist dies aber auch nicht die beabsichtigte Semantik

Ereignisse in verteilten Systemen ...

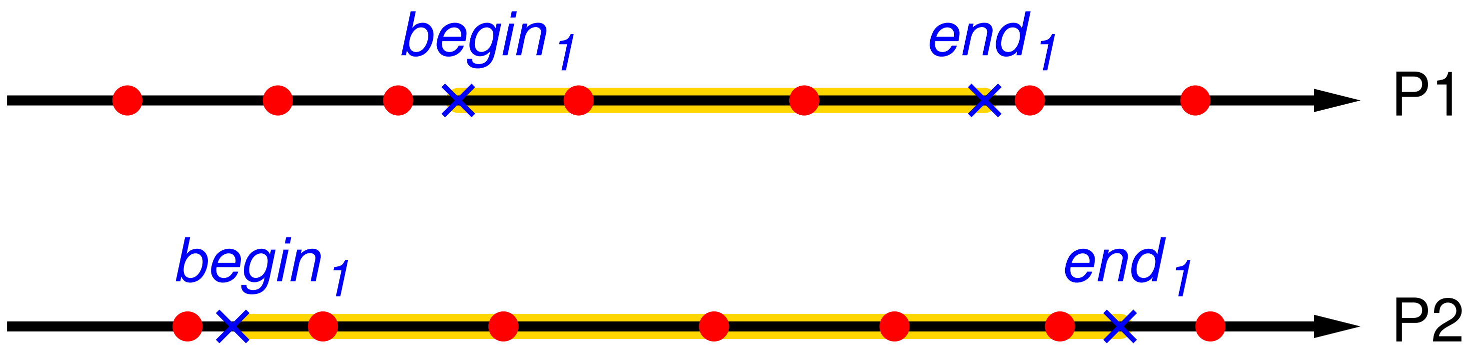
- Alternative: Berücksichtigung kausaler Abhängigkeiten
  - nur E/A-Ereignisse, die durch Interaktion verursacht sind
  - diese Ereignisse liegen kausal zwischen *begin* und *end*



- Probleme:
  - keine Online-Auswertung möglich!
  - nicht immer beabsichtigte Semantik (z.B. Warteschlangen)

Ereignisse in verteilten Systemen ...

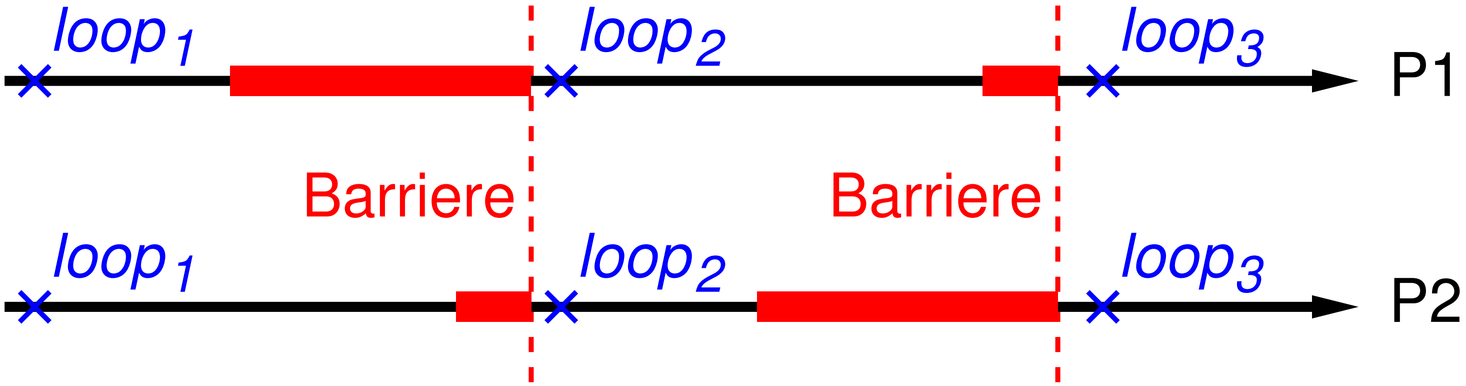
- Konsequenz: jeder Prozeß muß lokale Sonden bereitstellen
- Zusammengehörigkeit durch identische Parameterwerte (*virtuelle Zeit*) ersichtlich



- Vorteil: Semantik vom Benutzer immer eindeutig definierbar
- Nachteil: Mehraufwand (i.a. nicht bei SPMD-Programmen)

Informationsverlust durch Aggregation

- Genereller Verzicht auf Ereignisspuren kann interessante Effekte verbergen
- Z.B. Lastungleichheit in einer Schleife (Wartezeit in Barriere)



- Lösung: Aufzeichnung einer Folge von Meßwerten
  - je ein Wert pro Schleifeniteration (virtueller Zeitpunkt)

1.3.2 Spezifikation von Metriken

- Spezifikation muß (implizit) alle nötigen Aktionen festlegen:
  - bei der Definition einer Messung mit dieser Metrik
  - bei der Ausführung einer Sonde
  - beim Lesen eines Meßwerts
- Existierende Spezifikationssprachen
  - ASL (Apart), JavaPSL (U. Wien): Leistungs-Eigenschaften
  - MDL (U. Madison): tiefste Implementierungsschicht
  - EARL (FZJ): Analyse von Ereignisspuren
- Für G-PM: neue Sprache erforderlich
  - andere Zielsetzung
  - Sprache muß für Anwender beherrschbar sein ($\Rightarrow$ einfach)

1.3.2 Spezifikation von Metriken ...

PMSL: eine deklarative Spezifikationssprache für Metriken

- Basismechanismen: Datenflußprinzip, Unifikation

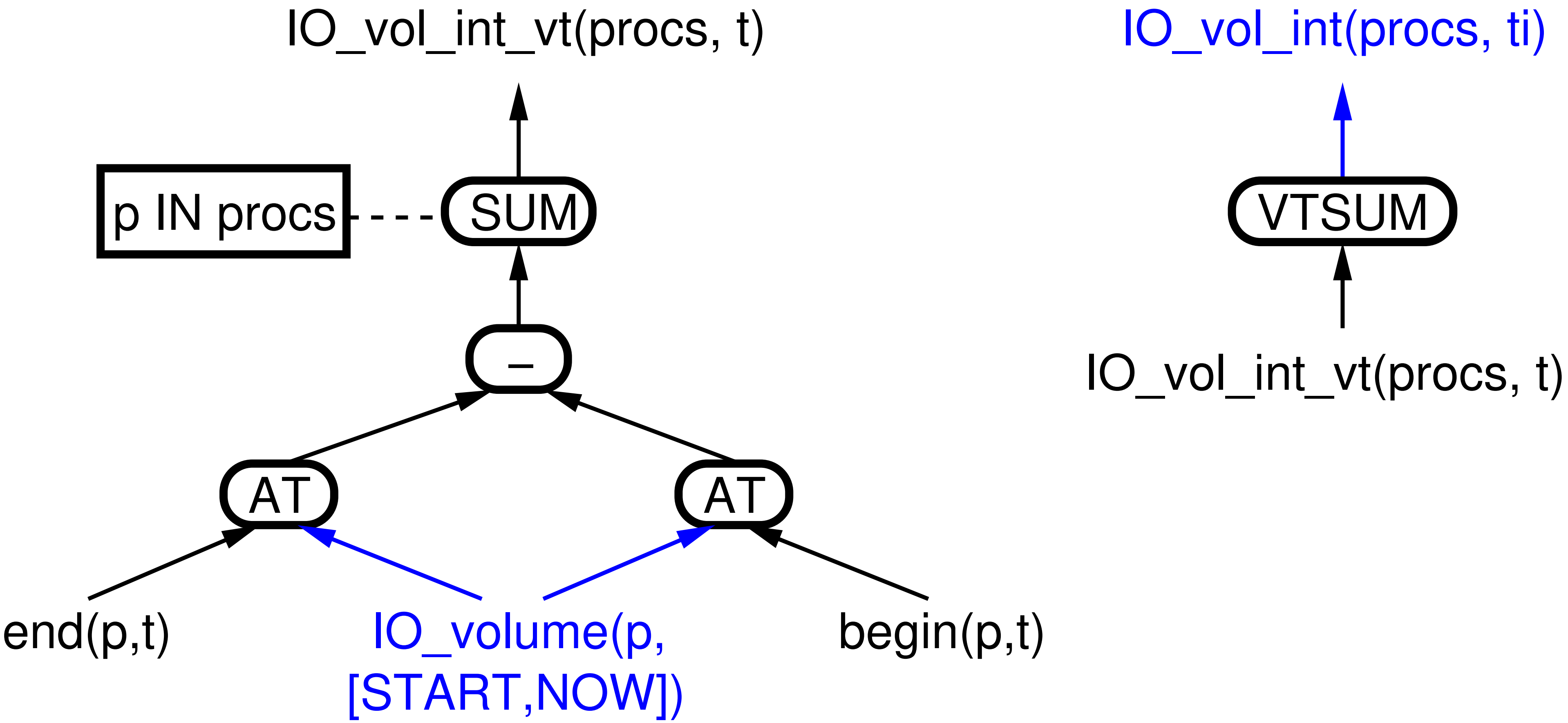
```
IO_vol_int_vt(Process[] procs, VirtualTime t)
{
  Process p; Value vol[];
  vol[p] = IO_volume(p, [START,NOW]) AT end(p, t)
         - IO_volume(p, [START,NOW]) AT begin(p, t);
  return SUM(vol[p] WHERE p IN procs);
}
IO_vol_int(Process[] procs, TimeInterval ti)
{
  VirtualTime t;
  return SUM(IO_vol_int_vt(procs, t) WHERE t.time IN ti);
}
```

1.3.3 Verteilte Realisierung von Messungen

Umsetzung der Spezifikation in verteilten Datenflußgraph

- **Nach Eingabe der Spezifikation der Metrik**
  - Erzeugung einer Graph-Schablone
  - Parameterwerte sind noch unbekannt
- **Bei Definition einer Messung der Metrik**
  - Einsetzen der Parameterwerte und partielle Auswertung
    - dabei: Einsetzen der Graphen für Basismetriken
  - Transformation des Graphen
    - zeitliche Aggregation soweit wie möglich nach unten
  - Verteilen des Graphen
    - lokal auswertbare Teilgraphen an verteiltes Monitor-system senden

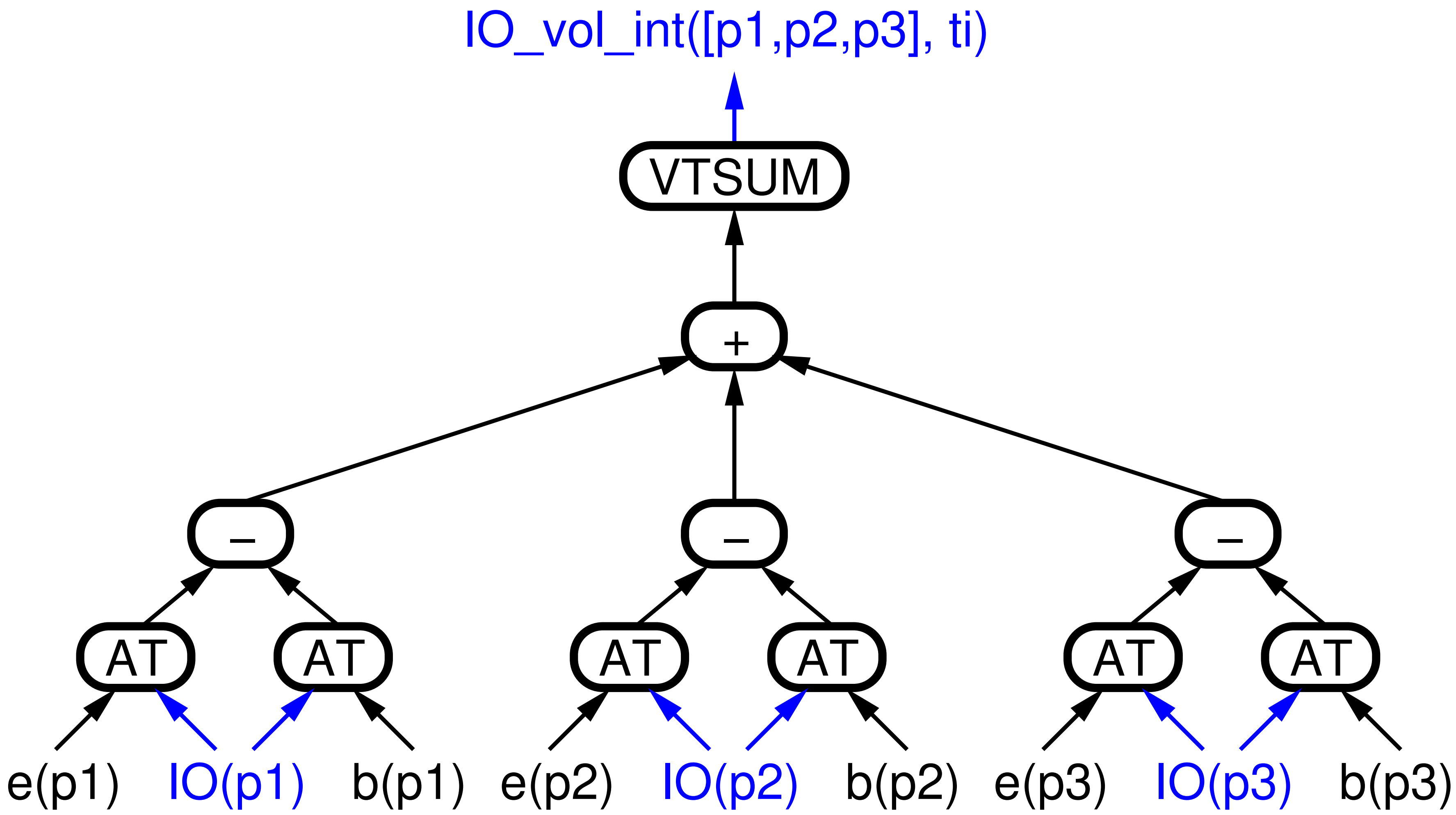
Beispiel: Graph-Schablone



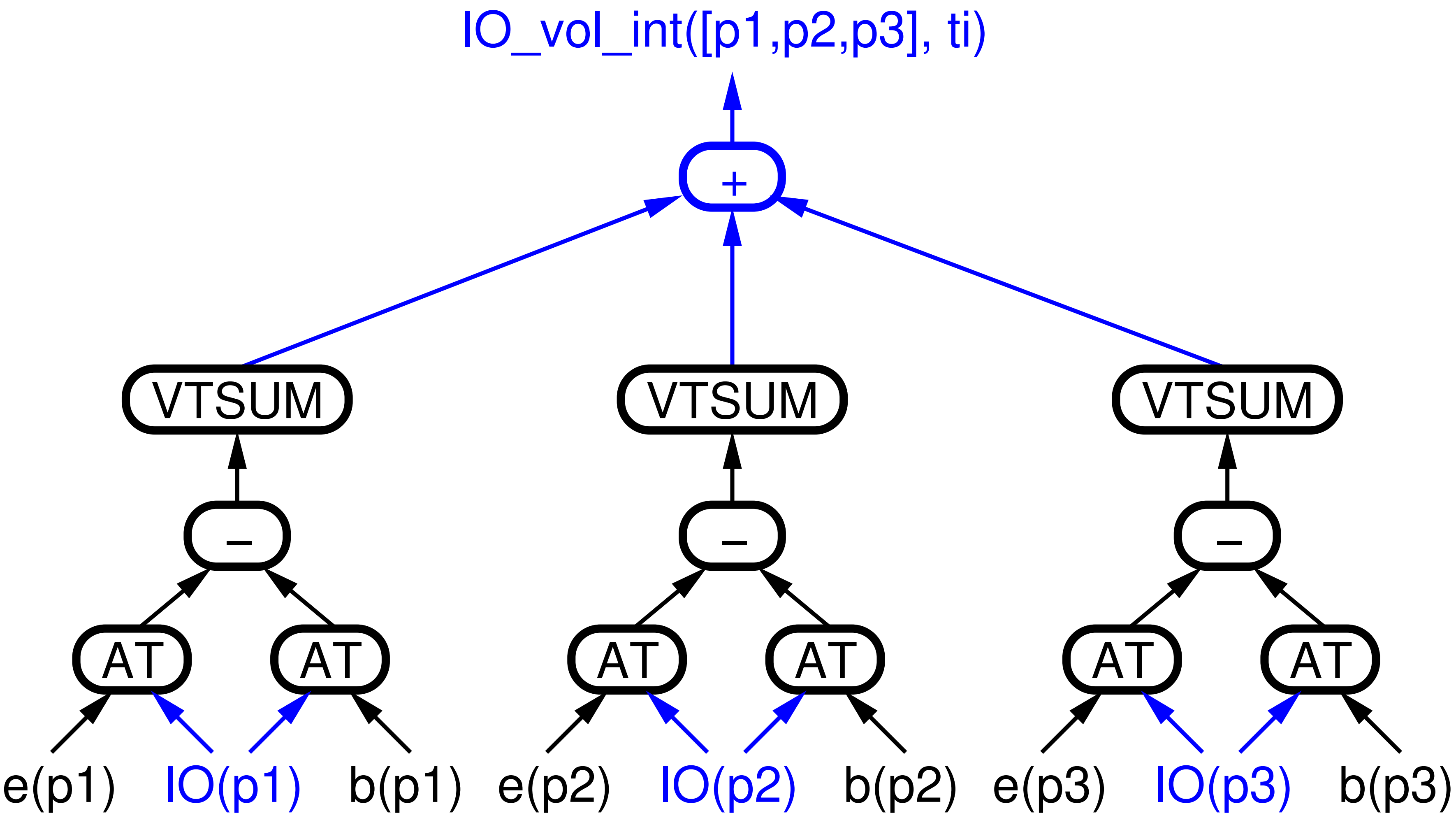
ereignisgetrieben

anfragegetrieben

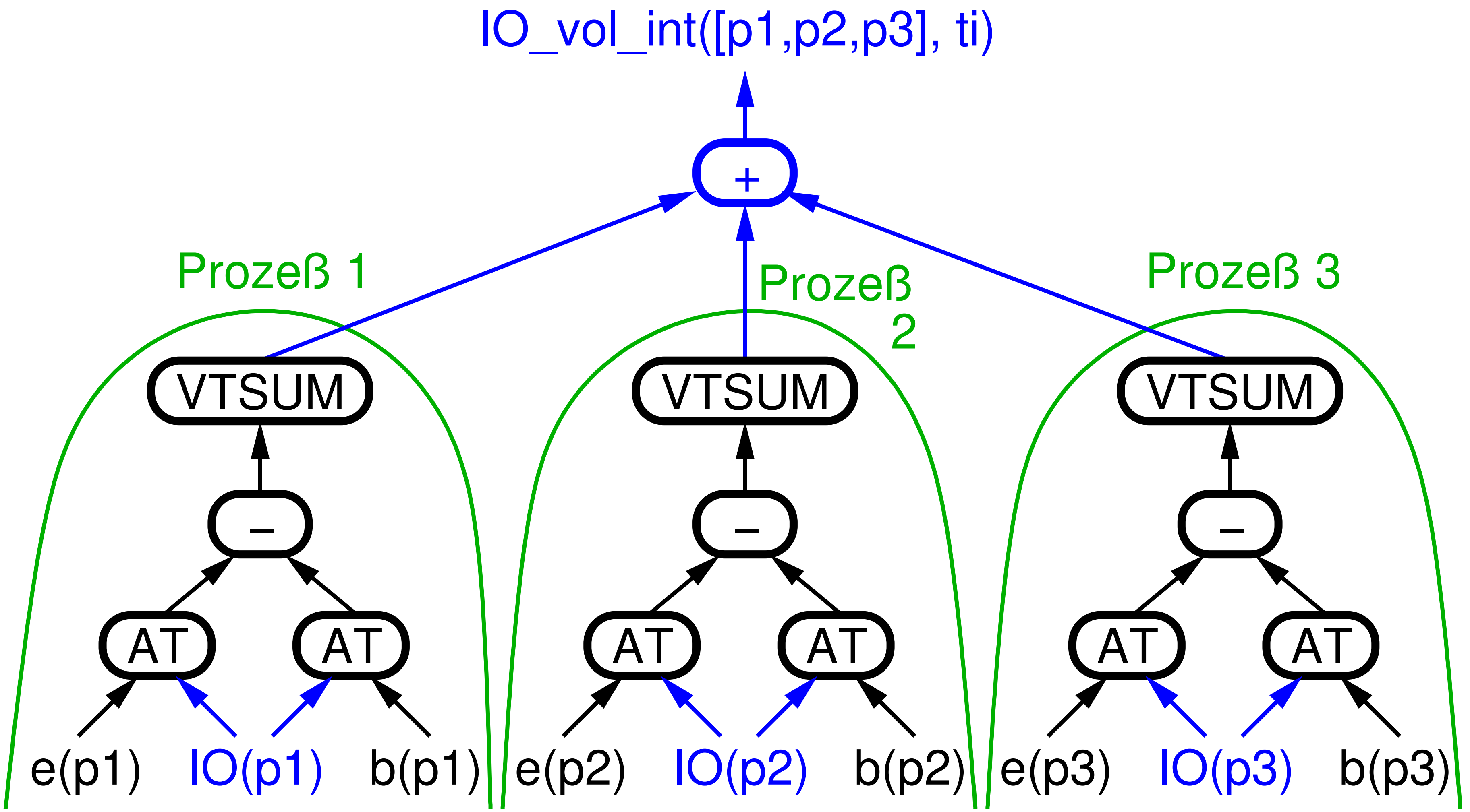
Beispiel: Graph bei Definition der Messung (3 Prozesse)



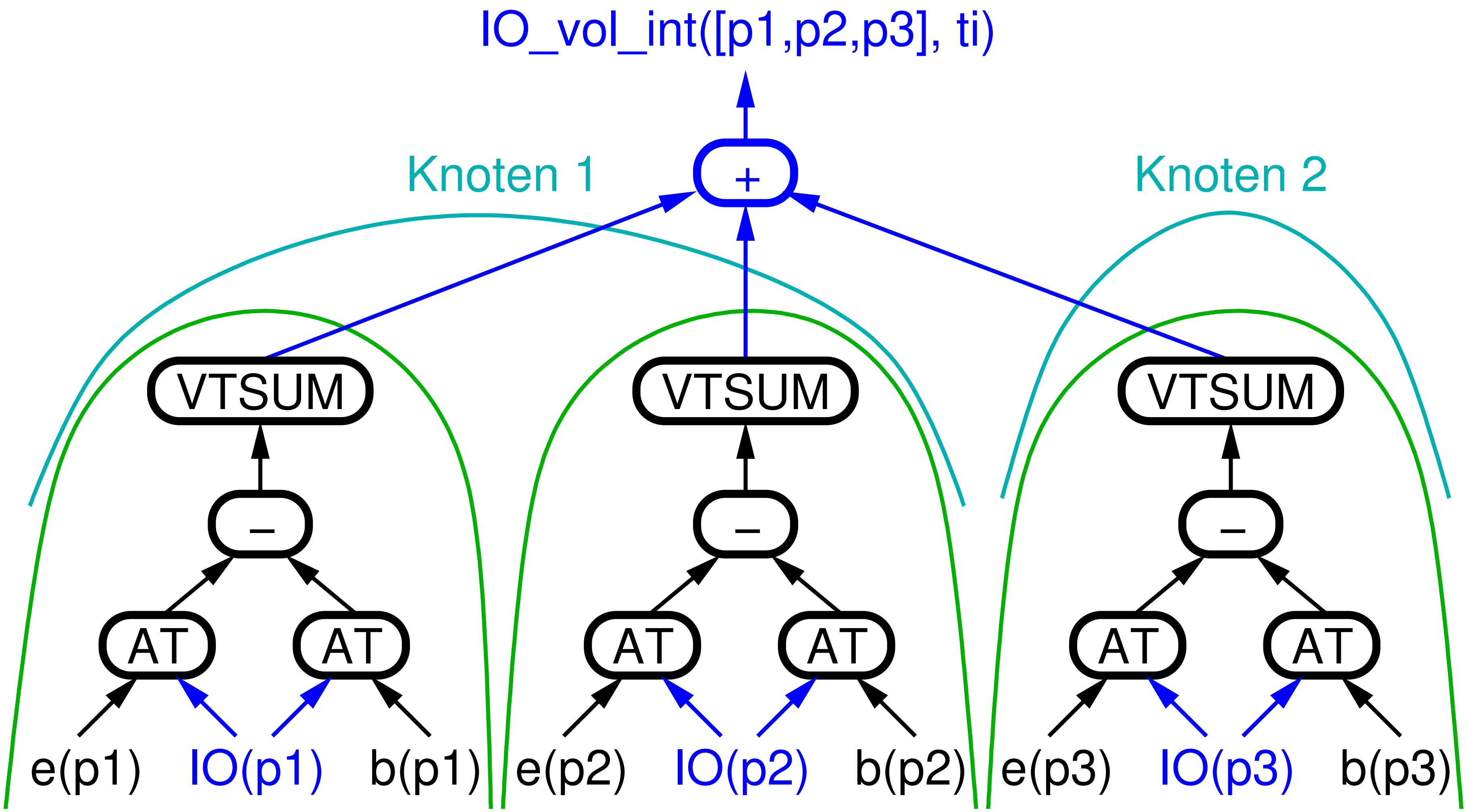
Beispiel: Vertauschung der Summationen



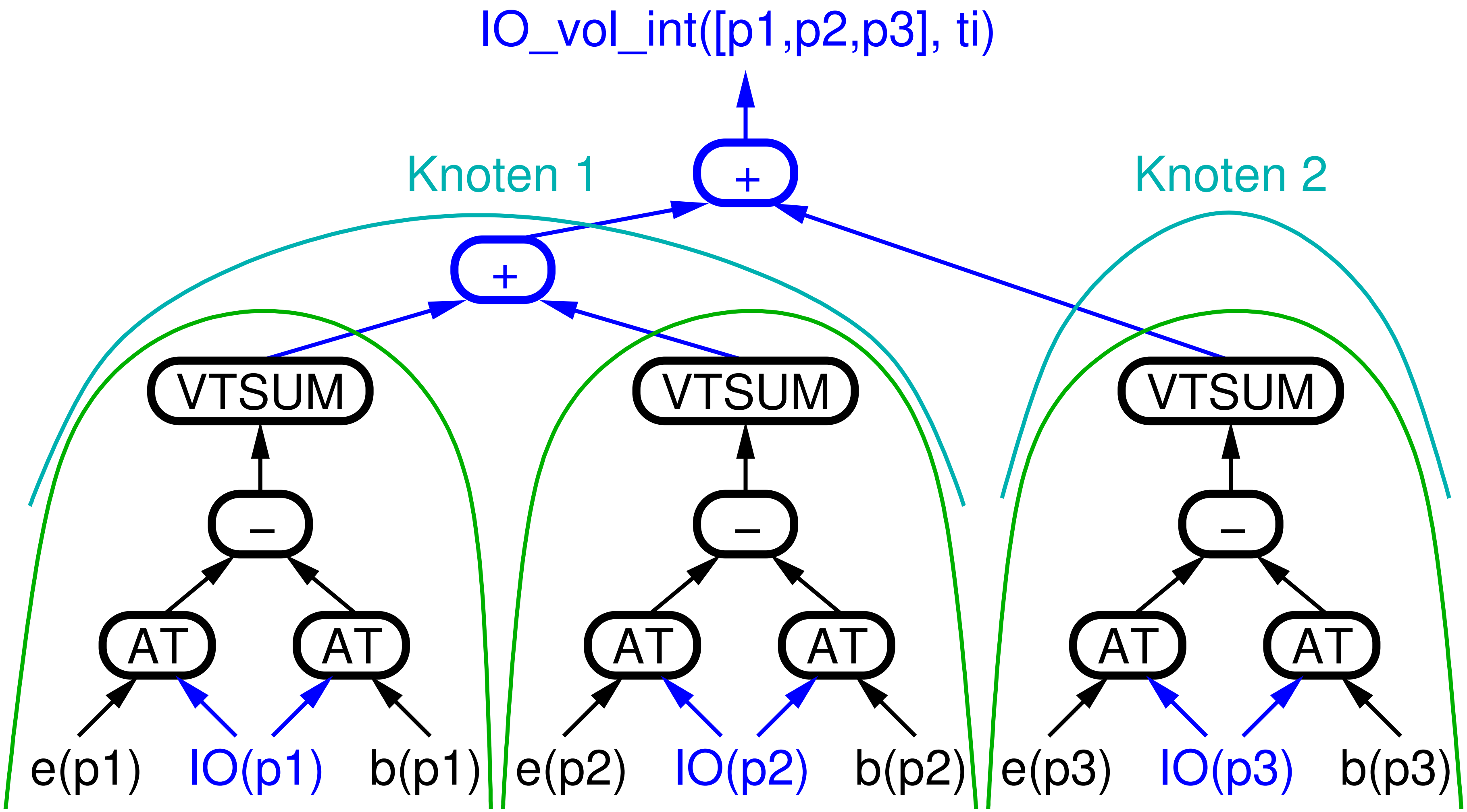
Beispiel: Lokal auswertbare Teilgraphen



Beispiel: Lokal auswertbare Teilgraphen



Beispiel: Hierarchische Aggregation



1.3.3 Verteilte Realisierung von Messungen ...

Infrastruktur zur Auswertung des Graphen

- Monitorsystem OCM, Monitorschnittstelle OMIS

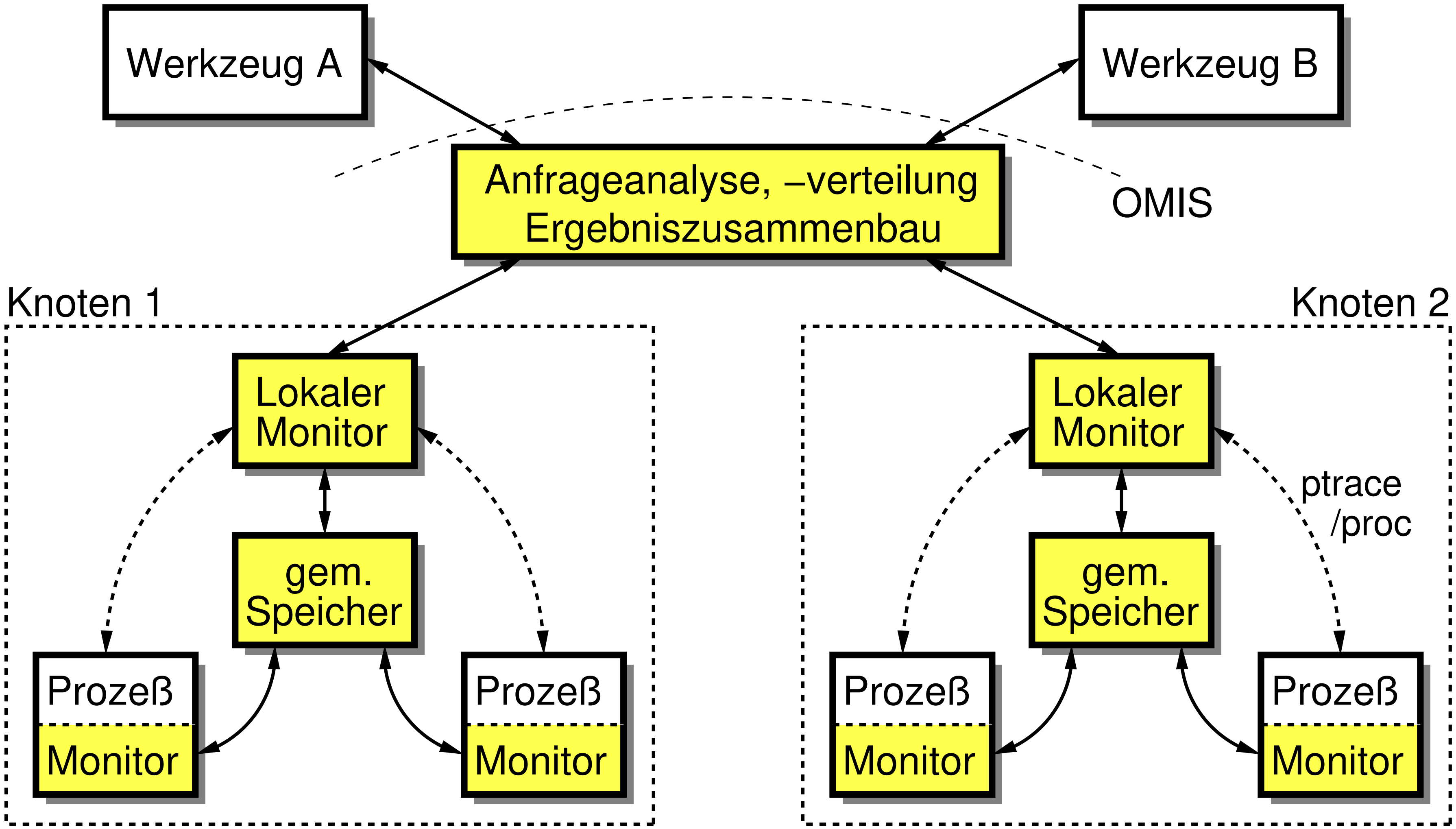
Konzepte von OMIS/OCM

- Objektbasiertes Modell des Zielsystems
- Programmierbare Ereignis/Aktions-Relationen

```
thread_executes_probe([p_1,p_2], "begin") :
    pa_integrator_read(pa_i_1, 0)
    print([$proc])
```

- Objektmengen, Ortstransparenz
- Erweiterbarkeit

Aufbau des OCM



1.4 Projektstatus

- **Derzeitiger Stand:**
  - Erste Version von G-PM einsatzbereit
    - Kommunikationsvolumen/-zeit, CPU-/Speichernutzung
    - Balkendiagramme und Zeitverlauf
    - Demonstrator für benutzerdefinierte Metriken
  - Erste Version von OCM für den Einsatz im Grid
  - Definition der PMSL abgeschlossen
- **Derzeitige Arbeiten:**
  - Realisierung des Parsers für PMSL
  - Erzeugung und Abarbeitung der Datenflußgraphen
- **Folgearbeiten:**
  - Optimierung der Datenflußgraphen

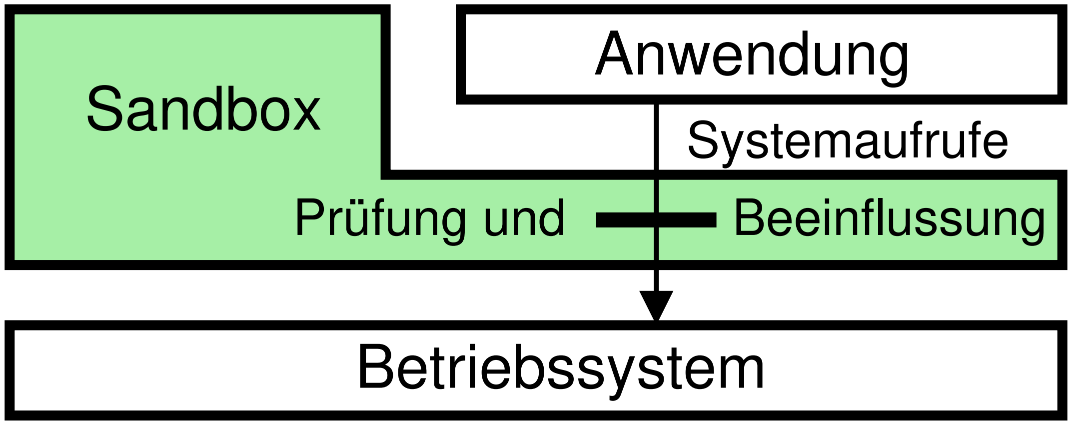
2 Weitere Forschungsarbeiten

2.1 Weiterentwicklung des OCM

- **Überwachung von threadbasierten Programmen**
  - Portabilität zwischen Thread-Implementierungen
  - Nutzung eines Überwachungsthreads in der Anwendung
  - Koop. mit AGH Krakau (Förderung: BMBF WTZ, KBN)
- **Überwachung von JAVA-Programmen**
  - Erweiterung von OMIS für JAVA/RMI-Umgebungen
  - Anbindung des OCM an die JVM
  - Koop. mit AGH Krakau (Förderung: BMBF WTZ, KBN)
- **Kombination von Online-Überwachung und Ereignisspur-aufzeichnung**
  - Adaptive Ereignisspuraufzeichnung
  - Koop. mit Univ. Linz / AGH Krakau (Förderung: ÖAD WTZ)

2.2 Abgesicherte Ausführungsumgebung

- *Community Access Services* im Grid
  - Nutzung derselben Kennung für verschiedene Benutzer
- Problem: Sicherheit
  - Betriebssystem schützt Benutzer nicht mehr voreinander
- Lösung:
  - Ausführung von Prozessen in gesicherter Umgebung („*Sandbox*“), die deren Zugriffsrechte einschränkt
  - Basis: Überwachung aller Systemaufrufe



2.2 Abgesicherte Ausführungsumgebung

Stand der Arbeiten

- Sandbox für Linux einsatzbereit
  - normaler Benutzerprozeß (keine `root`-Rechte)
  - Überwachung beliebiger Prozesse und deren Kinder
- Überwachung von Datei- und Netzwerkzugriffen
  - Rechte für reguläre Mengen von Dateien definierbar
  - Netzzugriffe über Protokollfamilie, Host, Port gefiltert
  - Laufzeit-Overhead: 10-100%
- Weitere Arbeiten:
  - graphische Oberfläche, Portierung auf andere UNIXe
  - Konverter für *Capability*-Sprachen
  - praktischer Einsatz im Grid-Computing

3 Fazit / Ausblick

3 Fazit und Ausblick

- **Forschung im Bereich paralleles / verteiltes Rechnen**
  - Schwerpunkt: Werkzeuge und Middleware
    - Zielsysteme: Cluster und Grid, Multithreading
    - Laufzeitüberwachung und -steuerung
    - Quellcode- und Anwendungsbezug
  - praktische Anwendbarkeit im Vordergrund
- **Zukünftige Pläne:**
  - adaptive Programme, Frameworks, bzw. Bibliotheken
  - Wunsch: Kooperation mit Anwendern und Entwicklern von Compilern und Frameworks
  - weitere Anwendungen der PMSL?