Online-Monitoring paralleler und verteilter Systeme

Roland Wismüller
Univ. Siegen, Fachbereich 12
roland.wismueller@uni-siegen.de

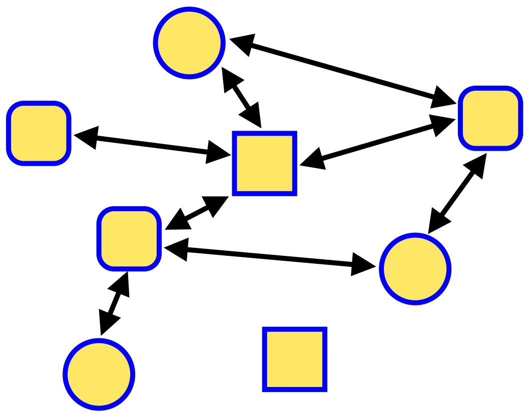
Agenda

- Begriffsklärung: verteiltes System, Monitoring
- Monitorsysteme: Architektur, Techniken, Forschungsziele
- Monitoring paralleler Programme
- Verteilte Auswertung von Leistungsmetriken
- Laufende und künftige Forschungsarbeiten

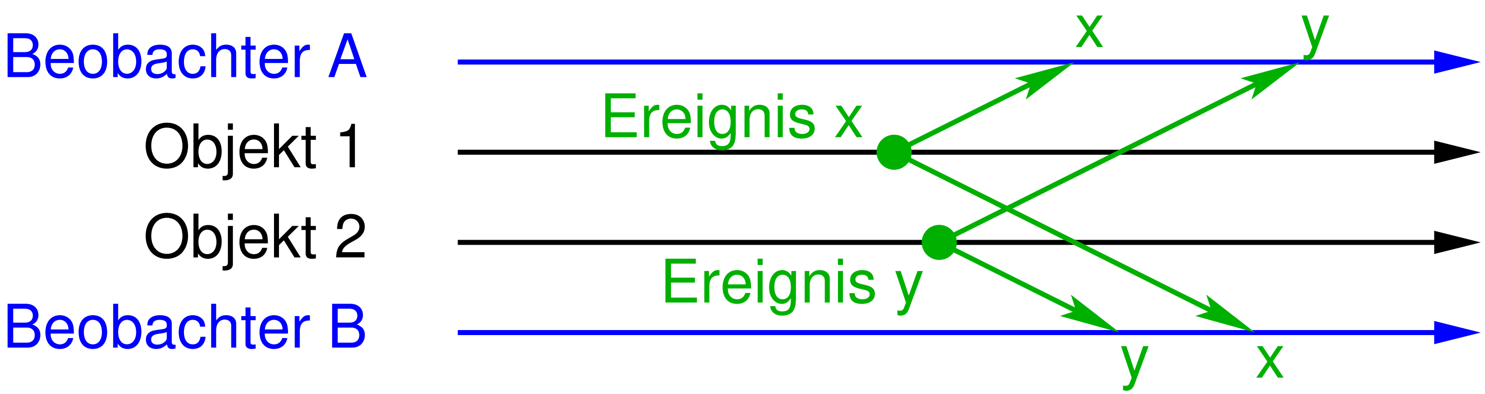
1. Begriffsklärung

Paralleles / verteiltes System

- Menge von interagierenden, aktiven Objekten
- Nebenläufige Aktivitäten in den Objekten
- Keine instantane Interaktionen möglich



- Es gibt weder eine globale Zeit noch einen globalen Zustand:



  - Kausalität wird zentrales Konzept

1. Begriffsklärung ...

Monitoring

- Beobachtung eines Systems (Ereignisse, lokale Zustände)
- Zielgerichtete (Vor-)Auswertung der Beobachtungsdaten
  - globale Aussagen über das Gesamtsystem
- Auch: steuernder Eingriff in das System

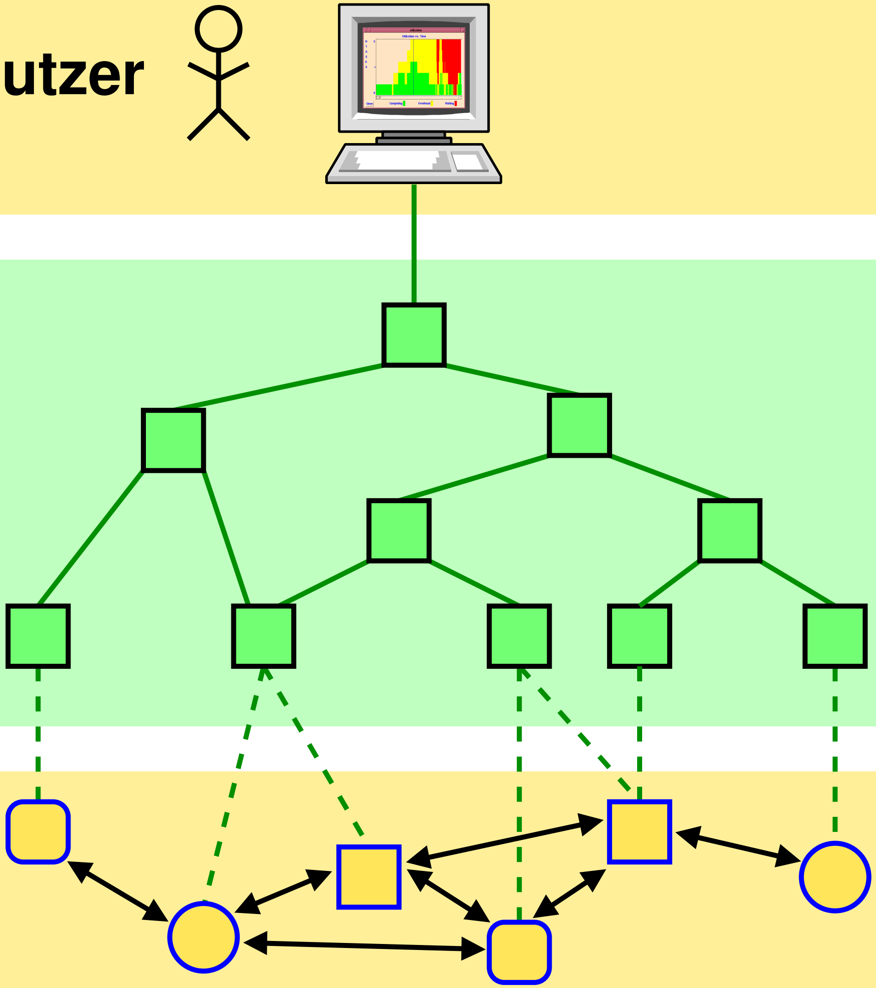
Beispiele

- Verteilte Programme: Debugging, Leistungsanalyse
- PC-Cluster: Zustand / Auslastung von Rechnern und Netz
- Produktionsanlage: Zustand der Maschinen (z.B. Störungen)
- Gebäudetechnik: Temperatur, Türen, Fenster, Klimaanlage

2. Monitorsysteme

Prinzipielle Architektur eines Monitorsystems

Benutzer



Software–Werkzeug (zentral)
- Auswertung, Darstellung
- Benutzerschnittstelle

Monitorsystem (verteilt)
- Erfassung, Transport, Vorverarbeitung
- Abstraktion des Zielsystems

Zielsystem (verteilt)

2. Monitorsysteme ...

Monitoring-Techniken

- **Datenanalyse**
  - offline
  - online
    - Voraussetzung für steuernde Eingriffe
- **Datenerfassung**
  - Ereignisaufzeichnung (*Logging, Tracing*)
    - Rohdaten (Ereignisse) werden unverarbeitet an das Werkzeug geleitet
  - verteilte (Vor-)Auswertung
    - Monitorsystem nimmt Datenauswertung und -reduktion vor (gemäß Vorgabe des Werkzeugs)

2. Monitorsysteme ...

Verteilte Auswertung

- Monitorsystem wird von Werkzeug „programmiert“
  - welche Rohdaten sollen erfaßt werden?
  - wie sollen Daten verarbeitet (zusammengefaßt) werden?

- Motivation
  - Kosten der Datenerfassung (Rückwirkung auf System)
  - Skalierbarkeit (Datenmenge beim zentralen Werkzeug)
  - Einsparung von Kommunikation
    - Belastung des Kommunikationsnetzes (Rückwirkung)

2. Monitorsysteme ...

Forschungsziel: universelles, generisches Monitorsystem

- Unabhängig von Zielsystem und konkreter Anwendung
- Einfache, flexible Schnittstelle zu Werkzeugen
  - globale, konsistente Sicht auf das verteilte System
  - Erweiterbarkeit (Datenerfassung und -auswertung)
- Skalierbarkeit und Effizienz
  - automatisch verteilte Auswertung

2. Monitorsysteme ...

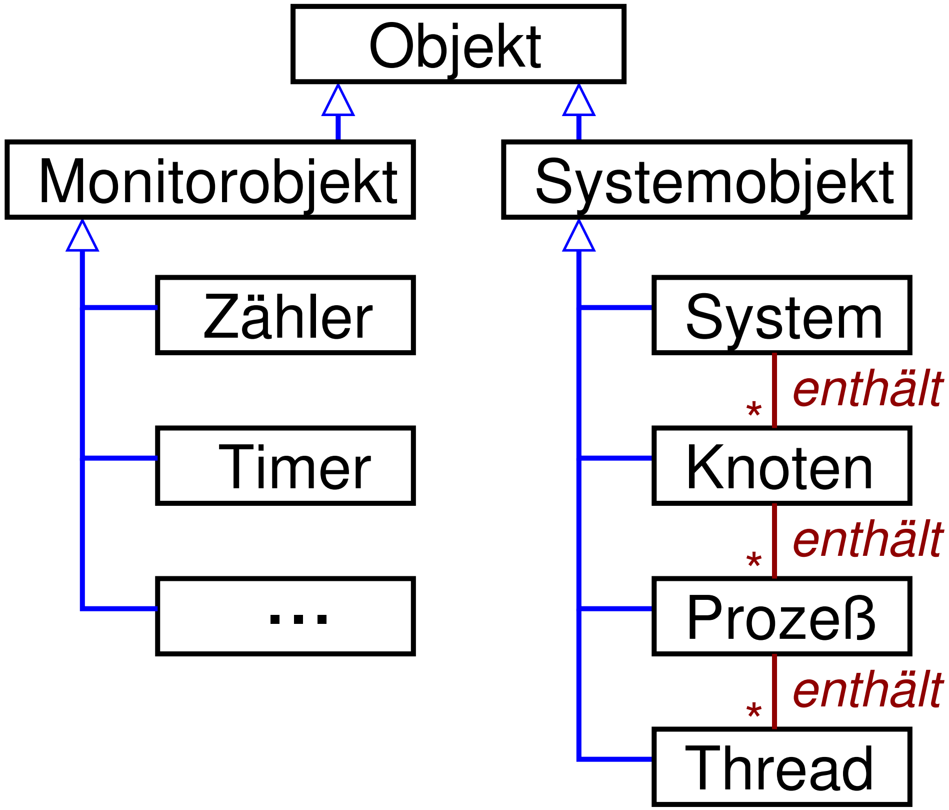
Typischer Einsatz

- Kombination von Routine- und *Ad-hoc*-Überwachung
  - Benutzer (oder Werkzeug) verfeinert die Überwachung bei Verdacht auf Probleme
    - interaktive / automatische Problemeingrenzung
  - ggf. Eingriff in das System zur Problembehebung
- Nicht (primär):
  - reine Protokollierung von Überwachungsdaten
- (Noch) nicht:
  - automatische Steuerung im Sinne der Steuer- und Regeltechnik (harte Realzeitanforderungen)

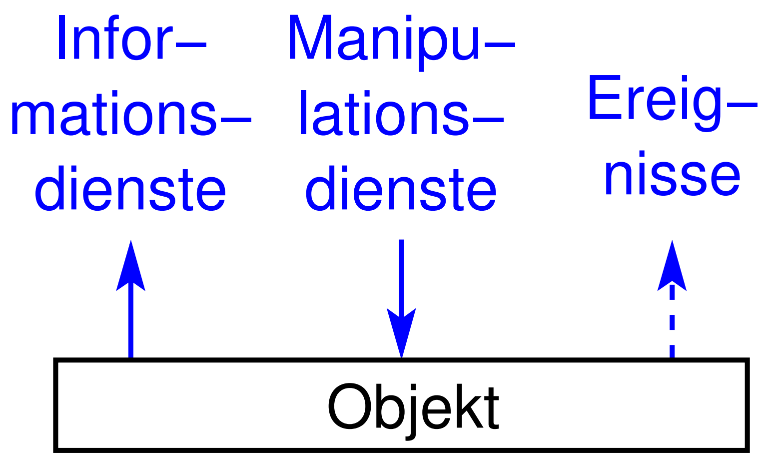
©2005, Roland Wismüller

3. Monitoring paralleler Programme

- OMIS / OCM: Spezifikation / Implementierung einer Schnittstelle zum Online-Monitoring paralleler Programme
- Basis: objektbasiertes Modell des Zielsystems



Schnittstelle der Objekte:



Monitor-Schnittstelle

- Definition von Ereignis/Aktions-Relationen

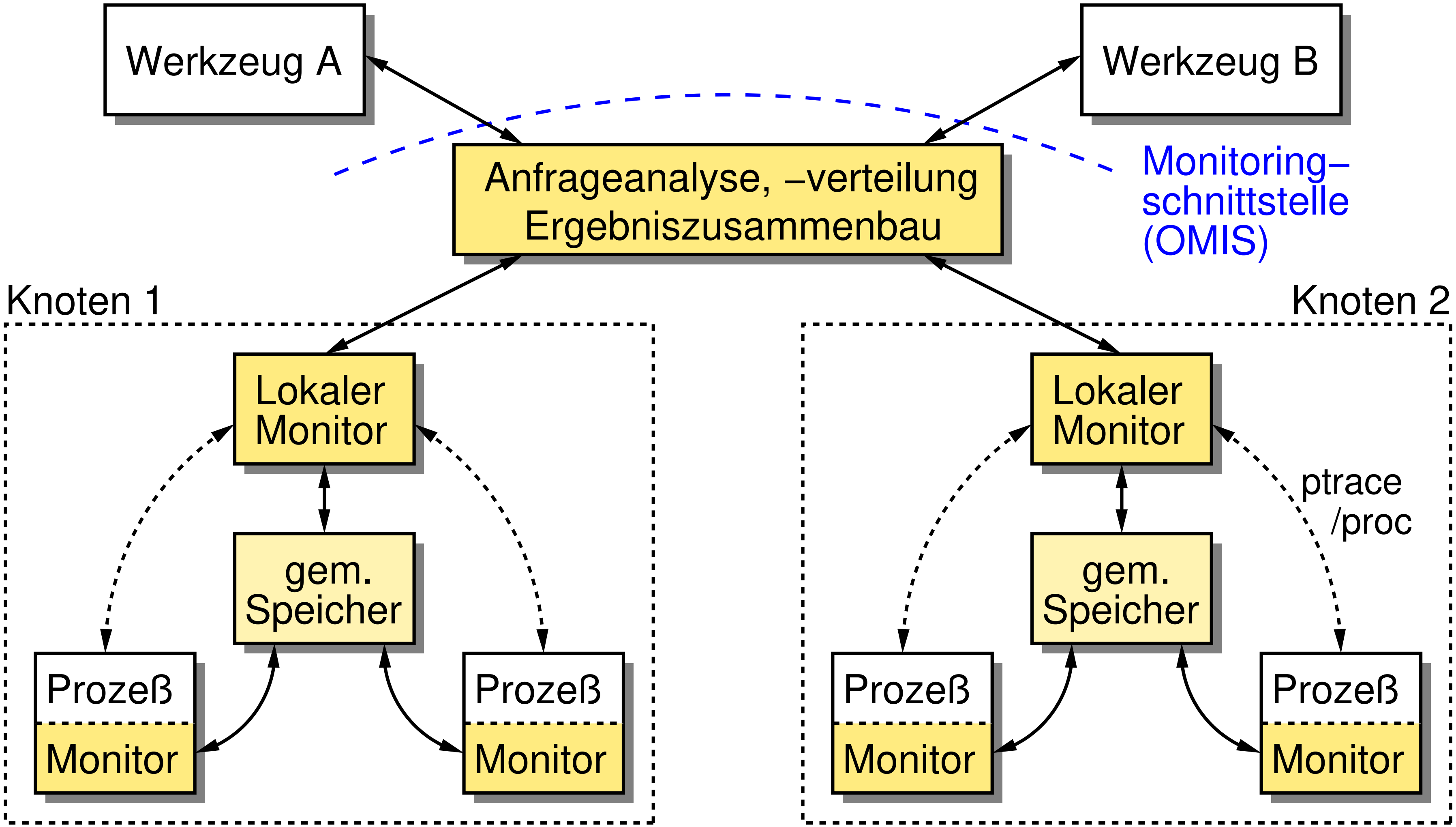
```
thread_started_lib_call([p_1,p_2], "MPI_Send") :
    pa_counter_increment(pa_c_1, 1)
```

```
thread_started_lib_call([p_1,p_2], "MPI_Send") :
    thread_stop([a_]) thread_get_backtrace([$thread])
```

- Dienst-Anfragen für Objektmengen
- Ortstransparenz: automatische Verteilung der Anfragen
- Konsistenzsicherung (Sperren, Uhrensynchronisation)
- Dynamische Erweiterbarkeit durch *Plug-In*-Schnittstelle
    - für neue Objekte, Ereignisse und Aktionen

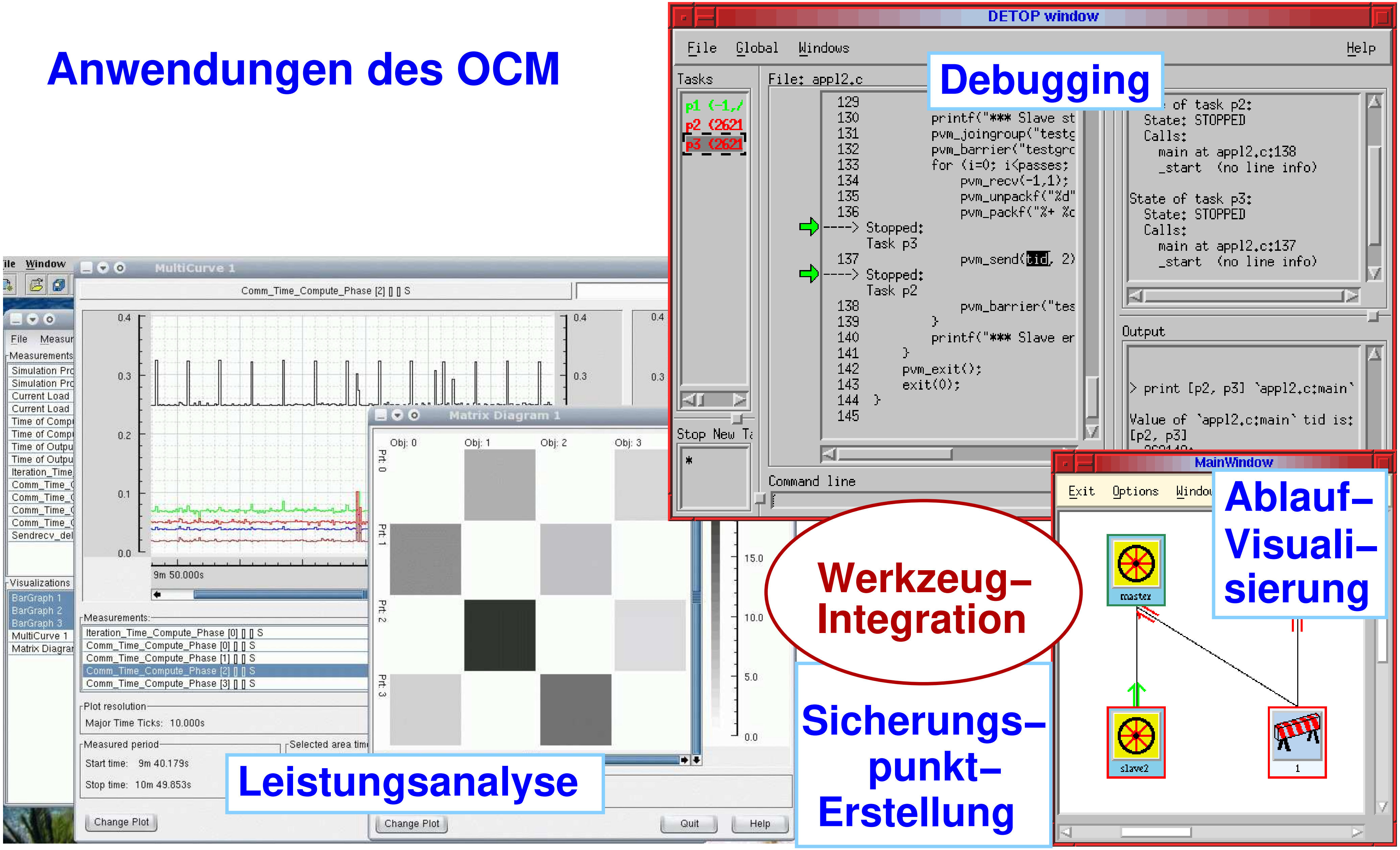
©2005, Roland Wismüller

Aufbau des OCM



3. Monitoring paralleler Programme ...

Anwendungen des OCM



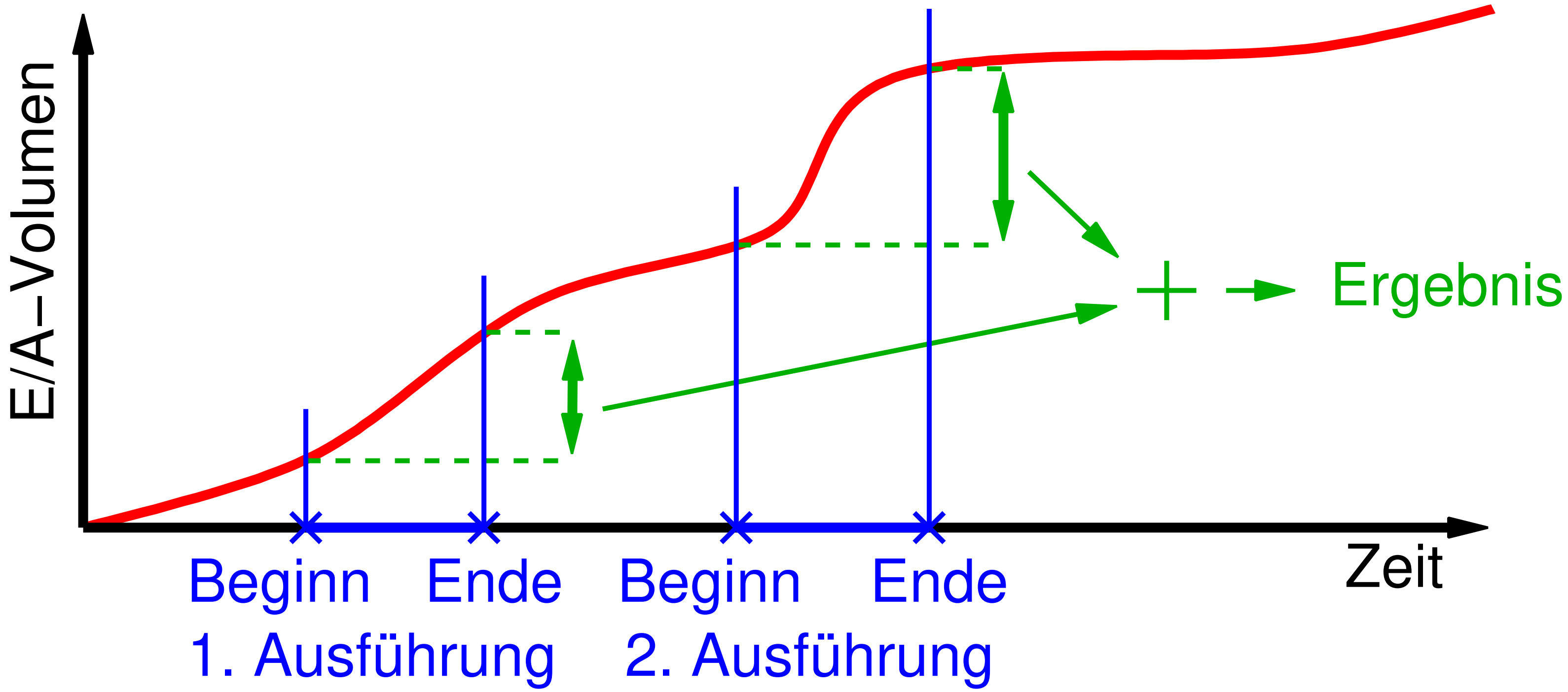
4. Verteilte Auswertung von Leistungsmetriken

- Hintergrund: EU-Projekt *CrossGrid* (2002-2005)
  - interaktive Anwendungen im *Grid*
- Teilprojekt G-PM: anwendungsorientierte Leistungsanalyse
  - Anwender kann Leistungsmetriken selbst definieren
  - Basis: vorhandene Metriken, Ereignisse im Programm
- Ziele:
  - einfache, deklarative Spezifikation der Metriken
    - ohne Angabe der Implementierung
    - Verteiltheit muß nicht berücksichtigt werden
  - effiziente, skalierbare Auswertung
    - automatische Ableitung eines verteilten Auswertungs-schemas

4. Verteilte Auswertung von Leistungsmetriken ...

Beispiel: E/A-Aufkommen in einer Programmphase

- Basis:
  - Metrik für das gesamte bisherige E/A-Volumen
  - Ereignisse zu Beginn und Ende der Programmphase



4. Verteilte Auswertung von Leistungsmetriken ...

Spezifikation der Metrik

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

4. Verteilte Auswertung von Leistungsmetriken ...

Spezifikation der Metrik

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

4. Verteilte Auswertung von Leistungsmetriken ...

Spezifikation der Metrik

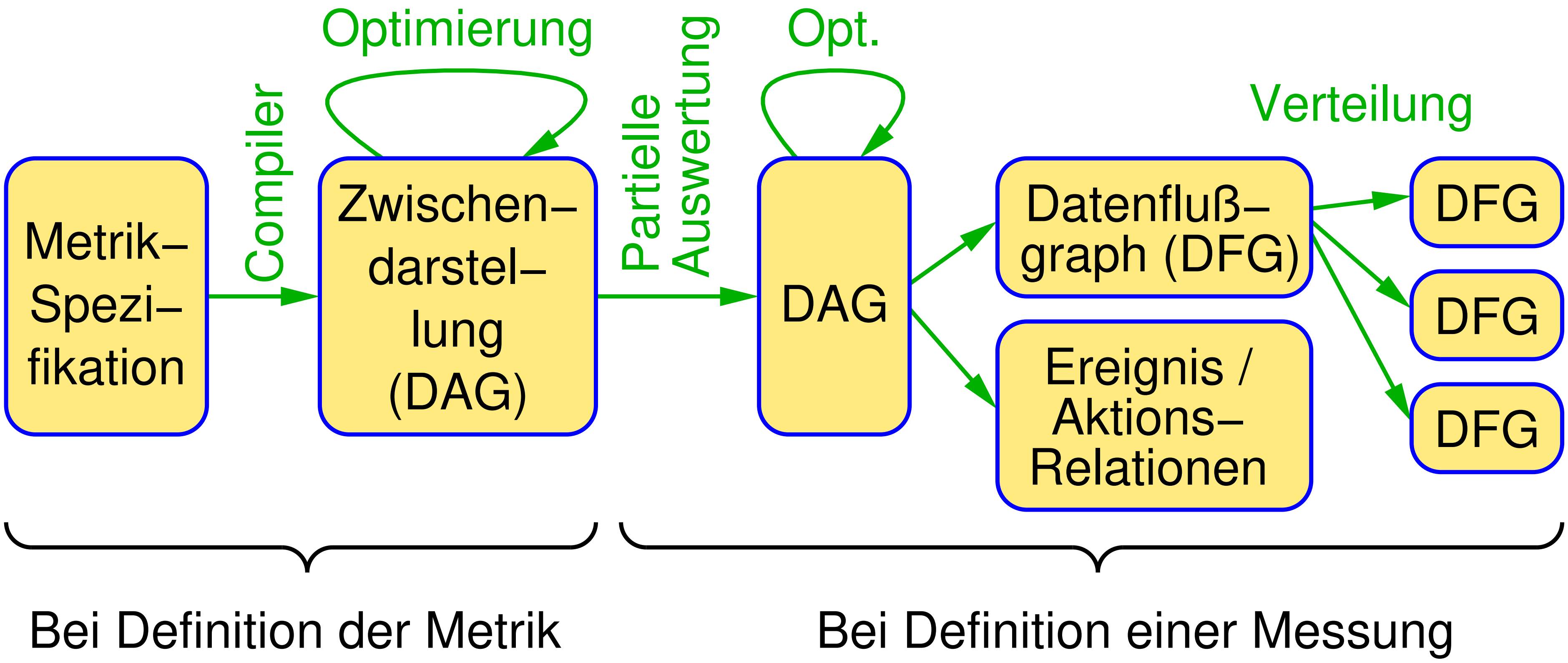
```
IO_in_Phase(Process[] procs, TimeInterval ti)
{
   Process p; VirtualTime vt;
   Value lvol[][], vol[];
   lvol[p][vt] =  IO_volume(p, [START, NOW])
                  AT end(p, vt)
                - IO_volume(p, [START, NOW])
                  AT begin(p, vt);
   vol[vt] = SUM(lvol[p][vt] WHERE p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

4. Verteilte Auswertung von Leistungsmetriken ...

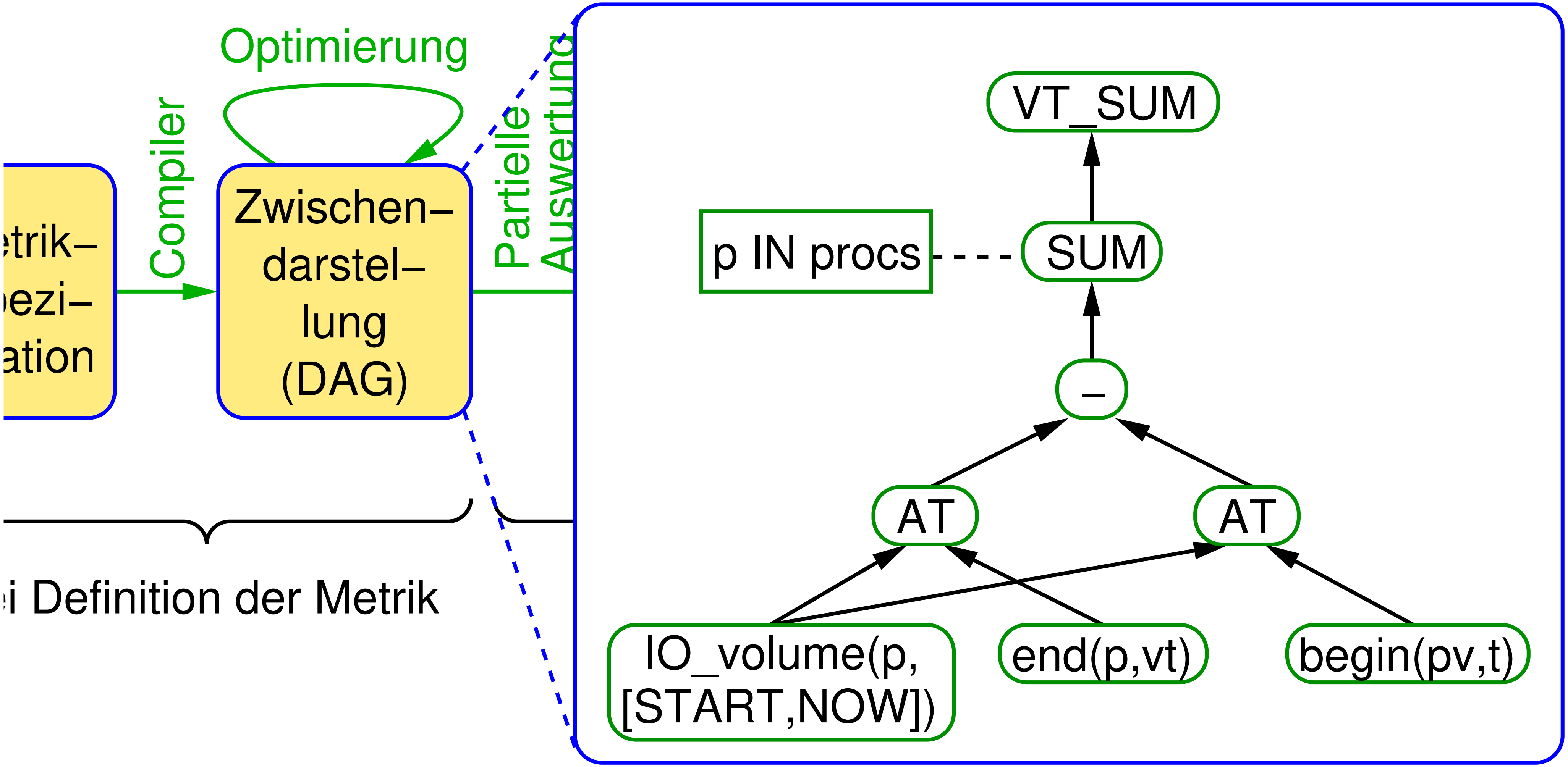
Spezifikation der Metrik

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

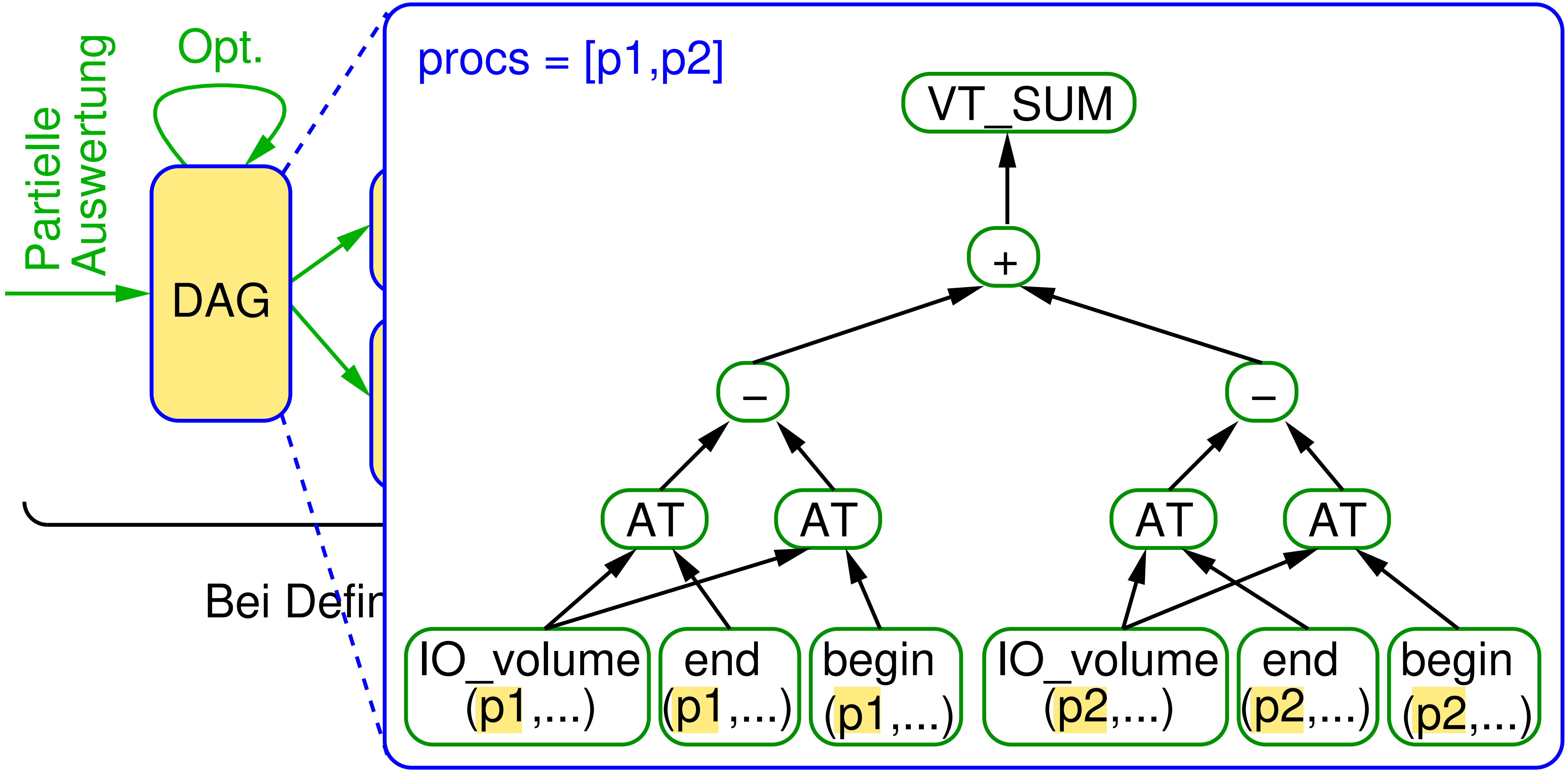
Ableitung des verteilten Auswerteschemas



Ableitung des verteilten Auswerteschemas

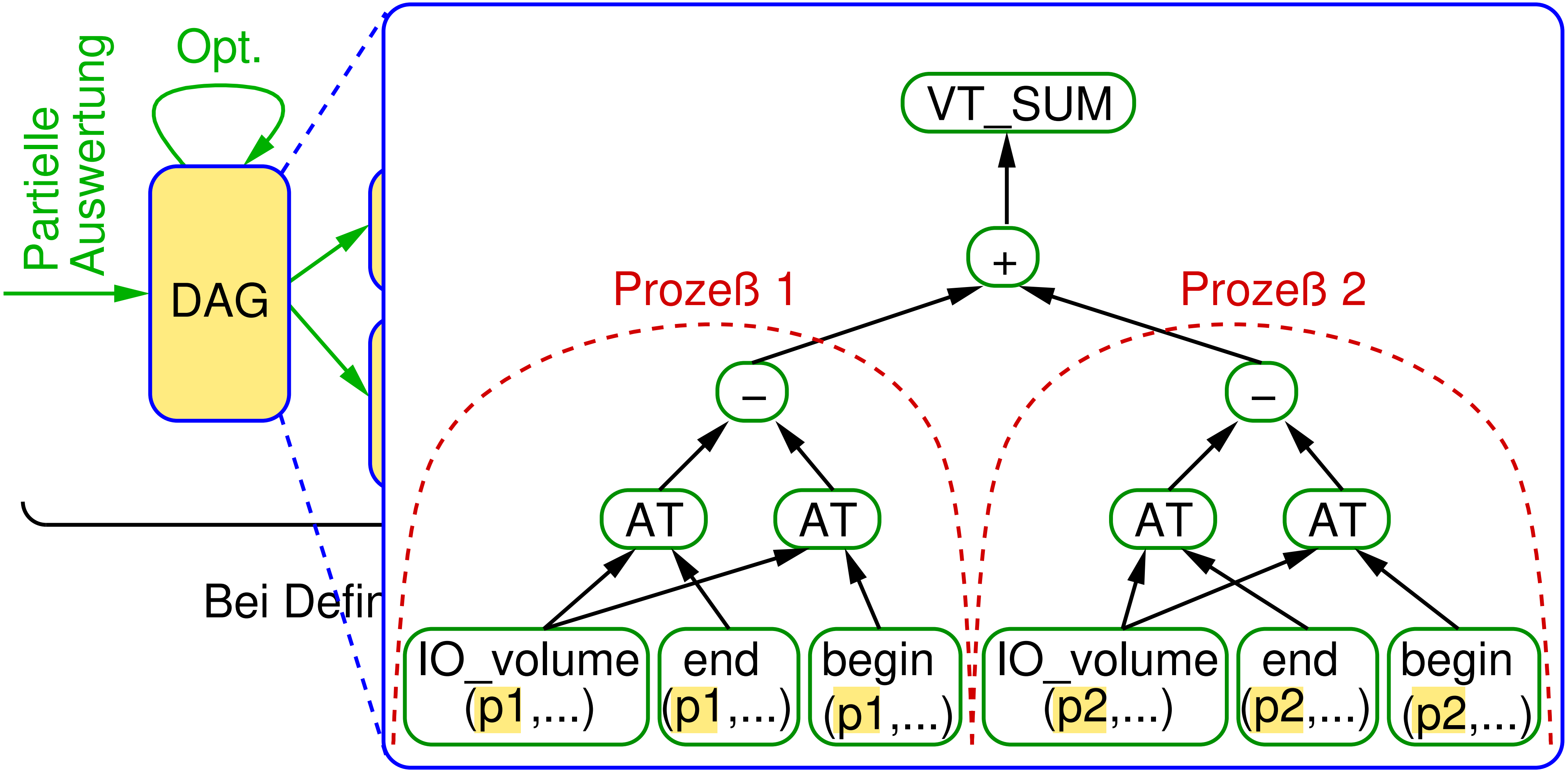


Ableitung des verteilten Auswerteschemas



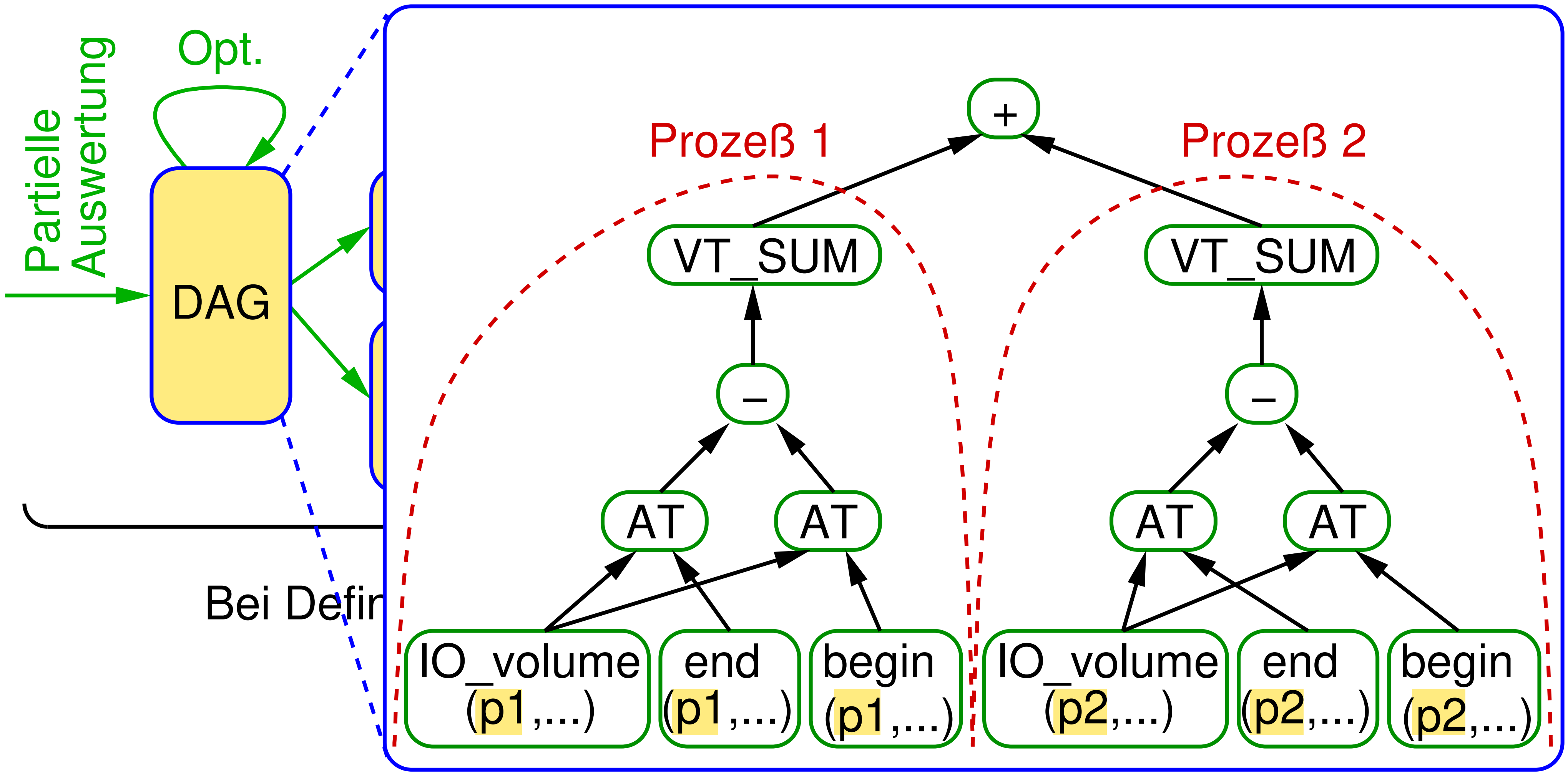
4. Verteilte Auswertung von Leistungsmetriken ...

Ableitung des verteilten Auswerteschemas



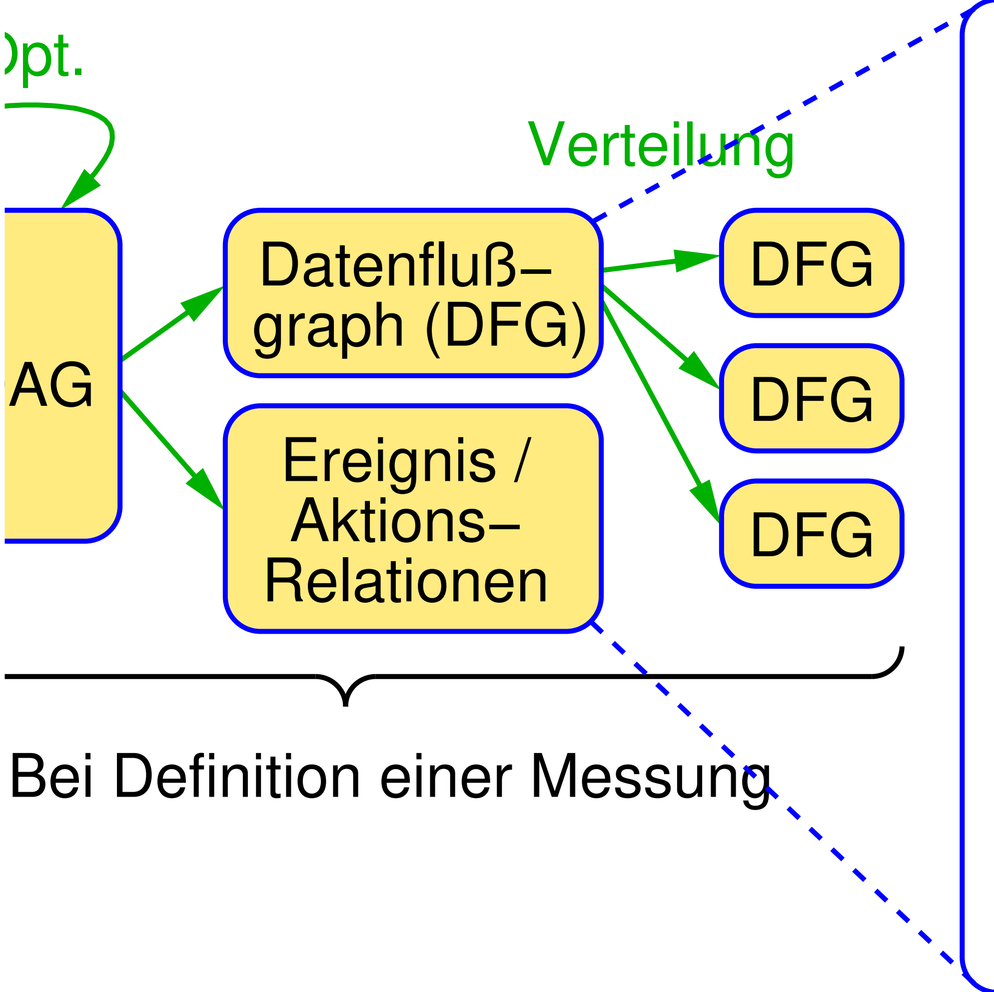
4. Verteilte Auswertung von Leistungsmetriken ...

Ableitung des verteilten Auswerteschemas



4. Verteilte Auswertung von Leistungsmetriken ...

Ableitung des verteilten Auswerteschemas



Umwandlung des DAG in Daten–strukturen für die Ausführung im OCM

- Datenflußgraphen
  - Verteilung auf OCM–Kompo–nenten
  - Interpretation durch OCM–*Plug–In*
- Ereignis/Aktions–Relationen
  - Überwachung von Ereignissen
  - Datentransport zwischen Datenflußgraphen

5. Laufende und künftige Forschungsarbeiten

Bisheriger Stand

- Automatisch verteilte Auswertung
  - OCM
    - Anbinden beliebiger Aktionen an Ereignisse
    - verteilte Vorverarbeitung nur über *Plug-Ins*
  - G-PM
    - benutzerdefinierte Datenauswertung
    - nur für Spezialfall Leistungsmetriken
    - Auswertungs-Primitive nicht erweiterbar
- Skalierbarkeit: Objektmengen, i.w. statische Eigenschaften
- Eingeschränktes Zielsystem: parallele Programme

5. Laufende und künftige Forschungsarbeiten ...

(Drahtlose) Intelligente Sensornetze

- Verteilte Vorverarbeitung der Daten ist essentiell
  - Datenmenge, Energieverbrauch
- Herausforderungen:
  - komplexe, veränderliche Topologie
    - *Routing*, Fehlertoleranz, ...
  - Adressierung der Sensoren über Eigenschaften
    - auch dynamisch veränderliche
  - Skalierbarkeit
    - „*Smart Dust*“: Tausende von Sensoren
  - einfache, effiziente Schnittstelle

5. Laufende und künftige Forschungsarbeiten ...

Flexiblere Programmier-Schnittstellen

- Realisierung durch objektorientierte Umgebung
  - Zielsystem-Objekte werden im Werkzeug durch lokale *Proxy*-Objekte repräsentiert
  - Überwachung und Datenauswertung kann im Werkzeug frei programmiert werden
  - verteilte Auswertung, ggf. durch Migration von Code
- Basiskonzepte:
  - Methodenaufrufe auf Objektmengen
  - asynchrone Auswertung (*Futures*)

  ProActive (INRIA)

  - stromverarbeitende Funktionen (Ereignisse)
  - Übersetzung und Optimierung zur Laufzeit