Werkzeugarchitekturen

- Werkzeuge allgemein
- Grobstruktur von Werkzeugen
- Interaktive und automatische Werkzeuge
- Effizienter Entwurf von Werkzeugen
- Schnittstellenbasierter Entwurf eines universellen Monitorsystems

Werkzeugarchitekturen
Die zehn wichtigsten Fragen

- Was unterscheidet Offline- und Online-Werkzeuge?
- Was unterscheidet interaktive und automatische Werkzeuge?
- Was versteht man unter interoperablen Werkzeugen?
- Welche Werkzeugtypen gibt es zur Unterstützung der Programmierung?
- Erläutern Sie die Struktur von Werkzeugen
- Was versteht man unter Instrumentierung?
- Beschreiben Sie die Struktur von Offline-Werkzeugen
- Beschreiben Sie die Struktur von Online-Werkzeugen
- Welche allgemeinen Probleme hat der Werkzeugentwurf?
- Erläutern Sie den Ansatz eines schnittstellenbasierten Werkzeugentwurfs

Was sind Werkzeuge?

Hier: Werkzeuge in den späteren Phasen des Lebenszyklus eines parallelen Programms

- Fehlersuche
- Optimierung
- Wartung
- Produktionsbetrieb

Wir betrachten nur Werkzeuge ab dem Zeitpunkt, zu dem ein halbwegs lauffähiges Programm vorliegt

Was sind Werkzeuge?

Begriffe

- Offline-Werkzeuge

  Zeigen Informationen zum Programm nach dessen Ende oder zumindest deutlich verzögert an
- Online-Werkzeuge

  Zeigen Informationen zum Programm gleichzeitig zum Ablauf an

Was sind Werkzeuge?

Begriffe...

- Interaktive Werkzeuge

  Gestatten eine direkte Interaktion mit dem Programm beobachten/manipulieren
- Automatische Werkzeuge

  Bearbeiten ohne Benutzereinwirkung das Programm zur Laufzeit beobachten/manipulieren

- Interaktiv und automatisch nur bei Online-Werkzeugen von Bedeutung
  - Ausnahmen sind denkbar, z.B.: automatische Änderung der Parallelisierung aufgrund gemessenen Programmverhaltens

Was sind Werkzeuge?

Begriffe...

- Integrierte Werkzeugumgebungen

  Mehrere Werkzeuge unter einer einheitlichen GUI vereint und meist gemeinsam nutzbar

- Interoperable Werkzeuge

  Unabhängig voneinander entworfene und lauffähige Werkzeuge, die nun zusammenarbeiten

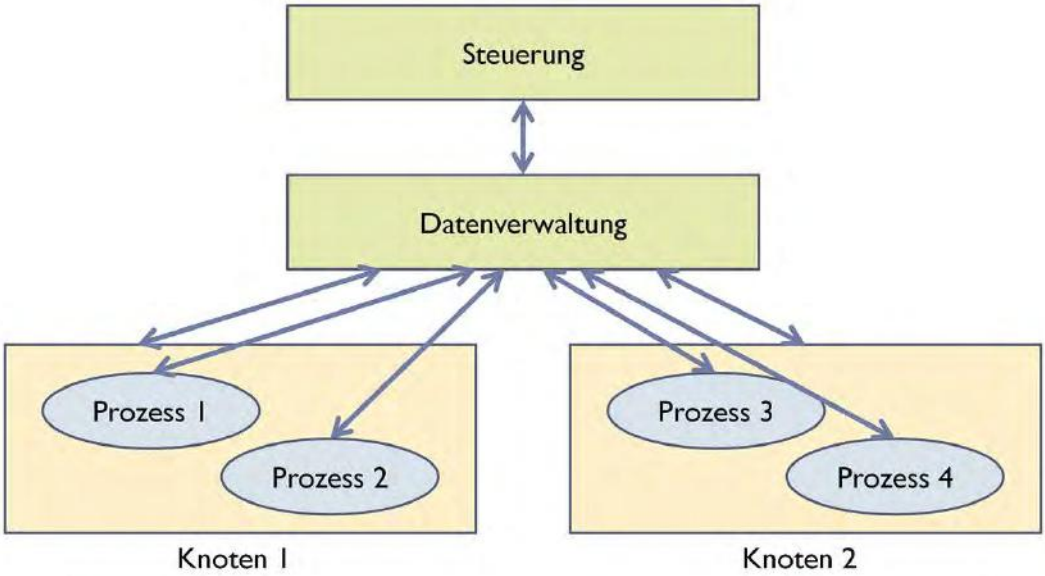
  Sehr schwer zu realisieren; Forschungsziel

Was sind Werkzeuge?

Hier nicht betrachtete Werkzeuge

- Werkzeuge zur Code-Erstellung
- Werkzeuge zur Parallelisierung
  Interaktives Parallelisieren von Daten und Codebereichen
- Werkzeuge zur Gebietszerlegung der Daten
  Gittergeneratoren bei numerischen Anwendungen

Struktur von Werkzeugen



Struktur von Werkzeugen...

Abstraktes Strukturbild klar

Aber: Realität sehr komplex

Gründe:

- Programm ist verteilt, also muss auch das Werkzeug verteilte Komponenten haben
  - Werkzeug ist MPI-Programm?
- Die einzelnen Komponenten laufen auch auf Knoten des Rechnersystems
  - Zusatzlast, Ausfallsicherheit

Struktur von Werkzeugen...

Begriffe

- Monitor (knotenlokal)

  Komponenten auf einem Rechnerknoten, die HW und SW des Knotens überwachen können
- Monitorsystem

  Summe der Komponenten im System zur Beobachtung und Manipulation von HW und SW der Zielmaschine

Struktur von Werkzeugen...

Begriffe...

- Instrumentierung

  Einbringen von zusätzlichem Code, der die Erfassung von Daten steuert und die Beeinflussung des Programms gestattet
  - Sourcecode-Instrumentierung

    Primitivste Variante: Einbau von `printf(...)`
  - Bibliotheks-Instrumentierung

    Z.B. mittels der PMPI-Schnittstelle in MPI
  - Binärcode-Instrumentierung

    Dynamisch eingebaute Sprünge in den Code des Monitorsystems

Struktur von Werkzeugen...

Offline-Werkzeuge

- Erfassung von Kenndaten zur Laufzeit des Programms
- Visualisierung nach Programmende
- Keine Beeinflussungsmöglichkeit des Programmlaufs
  - Aber nachträglich des Programms

- Überwachung wird aktiviert und Monitorsystem speichert Spurdaten ab (*trace*)
- Visualisierung der Spur nach Programmende

Struktur von Werkzeugen...

Online-Werkzeuge

- Erfassung von Kenndaten zur Laufzeit
- Sofortige Visualisierung
- Sofortige Beeinflussung des Programmlaufs möglich

- Direkte Interaktion zwischen Steuerung und Datenverwaltung
- Optional Generierung von Spurdaten

Struktur von Werkzeugen...

Interaktive (Online-)Werkzeuge

- Benutzungsschnittstelle
  (graphisch oder Kommandozeile)
- Sofortige Darstellung
  (Skalierung der Darstellung schwierig)
- Steueranweisungen des Benutzers an Programm weiterleiten
  (Problem der zeitlichen Nähe)
- Nicht bei Stapelverarbeitung einsetzbar

Struktur von Werkzeugen...

Automatische (Online-)Werkzeuge

- Keine Benutzerinteraktion vorgesehen
- Steuerung bewertet Situation aufgrund von Heuristiken
- Steuerung manipuliert laufendes Programm und startet/stoppt einzelne Überwachungen

Struktur von Werkzeugen...

Allgemeine Probleme

- Skalierbarkeit der Datenerfassung

  Wie kann ich von den vielen Prozessoren die Daten effizient einsammeln?
- Konsistenz der Daten

  Sind sie alle zum selben Zeitpunkt entstanden?
- Skalierbarkeit der Darstellung

  Wie kann ich von den vielen Prozessen und Threads die Ergebnisdaten übersichtlich darstellen?
- Beeinflussungsfreiheit

  Das Programm soll nicht im Ablauf gestört werden
- Dynamik im Ablauf

  Variierende Knotenmenge / Prozessmenge

Werkzeuge

Die typischen interaktiven Werkzeuge:

- **Fehlersuche (*debugging*)**
- **Leistungsanalyse (*performance analysis*)**
- Programmlaufvisualisierung
- Ergebnisvisualisierung
- Ablaufsteuerung (*computational steering*)
- *Problem Solving Environments*

Werkzeuge...

Die typischen automatischen Werkzeuge

- **Ressourcenverwaltung**
- Lastausgleich
- Sicherungspunktverwaltung
- Fehlertoleranzmechanismen

Werkzeuge zur Fehlersuche

Zweck

- Erkennen von Fehlerzuständen
- Auffinden von Fehlerursachen

Probleme

- Anzahl der überwachten Prozesse
- Nichtdeterminismus im Ablauf
- Beeinflussungsfreie Beobachtung

Werkzeuge zur Leistungsanalyse

Zweck

- Visualisierung wichtiger Leistungsdaten
- Erkennen von Leistungsengpässen

Probleme

- Erfassung der Daten produziert selber Last
- Zusatzlast minimieren oder herausrechnen
- Abweichende Abstraktionsebenen der Programmierung und der Meßdatenerfassung

Werkzeuge zur Programmlaufvisualisierung

Zweck

- Darstellung des Ablaufs beim Nachrichtenaustausch
  Wer kommuniziert wann mit wem
  Leichte Erkennung von Verklemmungen

Probleme

- Skalierung der graphischen Darstellung
- Zeitnähe

Werkzeuge zur Ergebnisvisualisierung

Zweck

- Visualisierung wichtiger Datenstrukturen
  Vor allem bei numerischen Anwendungen

Probleme

- Falls online: Datenkonsistenz
  Z.B. alle Daten aus derselben Iterationsstufe des Programms
- Falls online: Datenmenge

Werkzeuge zur Ablaufsteuerung

Zweck

- Manipulation algorithmischer Kenngrößen zur Laufzeit

  Z.B. Algorithmus konvergiert schlecht; dann Korrektur einzelner Parameter
- Wichtig bei langlaufenden Programmen

Probleme

- Wie bei Online-Ergebnisvisualisierung und Fehlersuche zusammen

Problem Solving Environments

Zweck

- Gruppe von Werkzeugen für einen bestimmten Einsatzzweck
  Z.B. für Anwendungen der Strömungsmechanik
- Umfaßt Gittergenerator, Ergebnisvisualisierer, Ablaufsteuerungskomponente etc.

Probleme

- Vielfältig, deshalb noch kaum Vertreter dieser Gruppe

Werkzeuge zur Ressourcenverwaltung

Zweck

- Zuteilung von parallelen Programmen zu Mengen von Knoten aufgrund definierter Strategien
- Verwaltung der Anfragen um Rechenzeit

Probleme

- Erfassen der Lastverhältnisse im System
- Berechnen einer optimalen Zuteilung (NP-hard)

Werkzeuge zum Lastausgleich

Zweck

- Korrektur von Ungleichbelastungen beliebiger Ressourcen (meist CPU) zur Laufzeit
- Erzielt optimale Ressourcennutzung
  = meist minimale Programmlaufzeit

Probleme

- Welche lasterzeugende Komponente soll verlagert werden
- Wann? Wie?

Werkzeuge zur Sicherungspunktverwaltung

Zweck

- Bei langlaufenden Programmen automatische Abspeicherung von Sicherungspunkten
- Nach z.B. Rechnerausfall Fortsetzung des Programms vom Sicherungspunkt aus

Probleme

- Konsistente Sicherungspunkte hinsichtlich z.B. nicht abgeschlossener Kommunikationen
- Wiederanlauf mit veränderter Knotenzahl

Werkzeuge zur Fehlertoleranz

Zweck

- Automatisches Tolerieren von Knotenausfällen
- Programm läuft auf anderer/kleinerer Konfiguration automatisch weiter
- Meist mit Sicherungspunkten realisiert

Probleme

- Hoher Implementierungsaufwand

Erste Zusammenfassung

- Entwurf und Implementierung von Werkzeugen sehr komplex
- Offline-Werkzeuge viel einfacher zu bauen
  Aber: genügen selten unseren Anforderungen
- Online-Werkzeuge wären nicht so viel schwieriger zu entwickeln
  Aber: Monitorsysteme ***sehr*** komplex
- **Monitorsystem ist selber verteiltes Programm**

Konzepte zum effizienten Werkzeugentwurf

Wir betrachten jetzt Monitoring-Systeme für Online-Werkzeuge

Ziele:

- Getrennte Entwicklung von Monitoring-System und Werkzeug (Steuerkomponente)
- Dadurch mehr und bessere Werkzeuge in kürzerer Zeit

Warum geht das überhaupt?

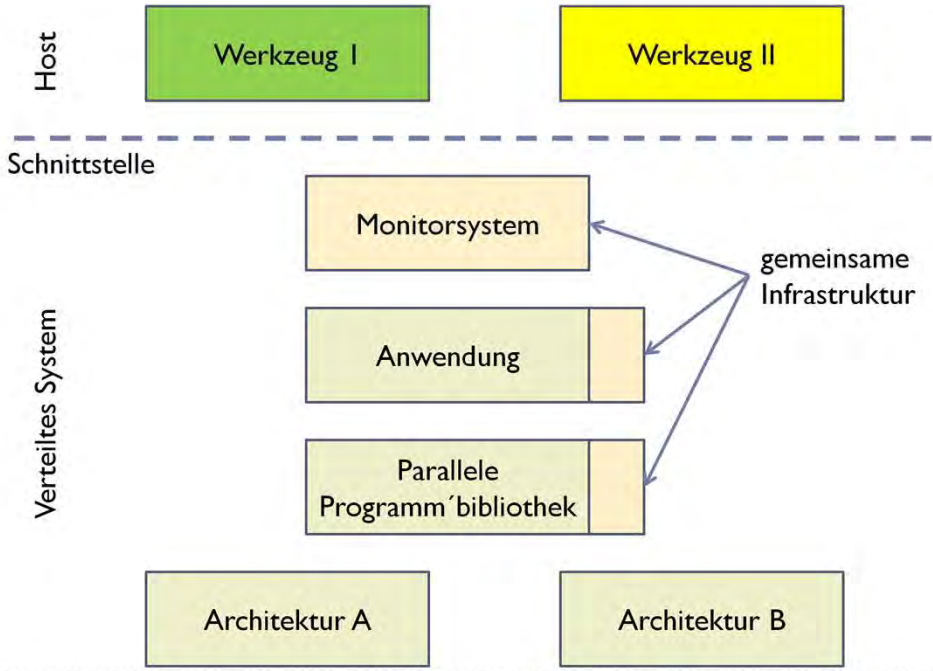
- Die meisten Werkzeuge benötigen identische Funktionen des Monitorsystems

Effizienter Werkzeugentwurf

Vorgehensweise

- Abtrennen der Steuerkomponente des Werkzeugs von der Werkzeug-Infrastruktur
- Einführung einer genormten Schnittstelle zwischen beiden Teilen
- Identifikation von Infrastrukturteilen, die vielen Werkzeugen gemeinsam sind
- Zusammenführung gemeinsamer Komponenten in einer einheitlichen Implementierung

Effizienter Werkzeugentwurf...



Struktur der Schnittstelle

Aus Sicht des Werkzeugs

- Monitorsystem ist Server, der Dienste an Objekten anbietet

Diensteklassen

- Informationsdienste (I)
- Manipulationsdienste (M)
- Benachrichtigungsdienste (B)

Objektklassen

System, Knoten, Prozesse, Threads, Nachrichten, Nachrichtenpuffer, Monitorobjekte

Arbeitsprinzip der Schnittstelle

Ereignis/Aktions-Modell

- Ereignis: interessierender Zustandsübergang im überwachten Programm
- Aktion: erwünschte Beobachtung oder Manipulation des Programms

Dienstanforderungen

- Verhalten des Monitorsystems ist durch Ereignis/Aktions-Relationen bestimmt
- Programmierung der Relationen über definierte Schnittstelle
- Aktionen jeweils ausgelöst, wenn Ereignis erkannt wird
- Leere Ereignisdefinition => unbedingte Aktion

Dienste der Schnittstelle (Beispiele)

Prozesse

I: Statische / dynamische Informationen

M: Erzeugung, Überwachung einrichten / aufgeben, Modifikation des Speichers, Änderung Priorität, ...

B: Terminierung, Empfang eines Signals, ...

Nachrichtenpuffer und Nachrichten

I: Inhalt des Puffers, Sender einer Nachricht, ...

M: Nachricht aus Puffer löschen, Nachricht markieren, ...

B: Eintrag einer Nachricht in einen Puffer, Empfang einer markierten Nachricht, ...

Programmierung der Werkzeuge

Vorgehensweise

- Einzelwerkzeuge setzen Diensteanforderungen an Monitorsystem ab und programmieren es somit für ihre Zwecke
- Bekommen Daten und Rückmeldungen zurück

Wichtiger Aspekt

- Verschiedene Werkzeuge benutzen gleiche Ereignisdefinitionen und ähnliche Aktionen

Beispiel: Leistungsanalyse

```
thread_has_started_lib_call([p_2],"MPI_SEND"):
    pt_integrator_start(pt_i_1)
    pt_counter_add(pt_c_1,$par5)
```

Wenn Prozeß p_2 einen Sendeaufruf startet: aktiviere einen integrierenden Zähler und addiere die Nachrichtenlänge ($par5) auf einen Zähler

```
thread_has_ended_lib_call([p_2]),"MPI_SEND"):
    pt_integrator_stop(pt_i_1)
```

Wenn der Prozeß den Aufruf beendet: halte den Zähler an

Werkzeugarchitekturen
Zusammenfassung

- Uns interessieren Werkzeuge für die späteren Lebenszyklusphasen der parallelen Programme
- Wir unterscheiden Offline- und Online-Werkzeuge
- Wir unterscheiden interaktive und automatische Werkzeuge
- Typisch: Fehlersuche online, Leistungsanalyse offline
- Online-Werkzeuge haben eine komplexe Struktur wegen der verwendeten Monitorsysteme
- Eine geeignete Schnittstellendefiniton trennt das Monitorsystem von den Werkzeugen und vereinfacht so die Implementierung der Werkzeuge