Hochleistungsrechnen: Einblicke in Leistungsanalyse und Speichersysteme der Zukunft

Berufungsvortrag Ernst-Moritz-Arndt-Universität Greifswald

Julian M. Kunkel

Deutsches Klimarechenzentrum / Universität Hamburg

30. Juni 2014

Gliederung

Hochleistungsrechnen – Motivation

Wissenschaftliche Anwendungen haben einen hohen Bedarf an

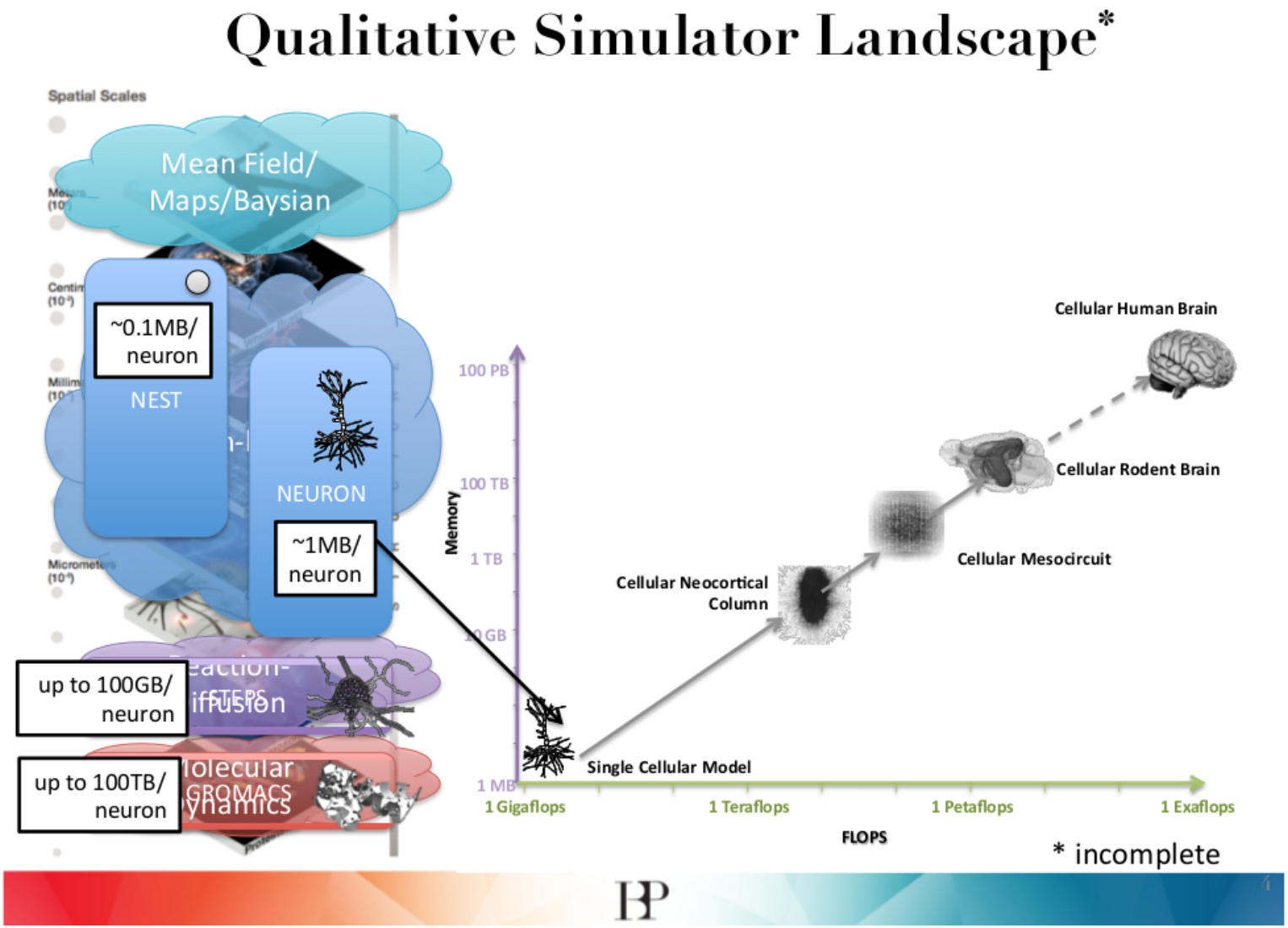
- Rechenzeit
- Arbeitspeicher
- Speicherkapazität

Ein „normaler” PC/Server ist meist nicht leistungsfähig genug

⇒ Parallele Nutzung vieler Prozessoren bzw. Server

- Moorsches „Gesetz“ erhöht Parallelität und nicht (mehr) Frequenz

Relevanz am Beispiel des Human Brain Projekts



Quelle: „Simulation Codes in the Human Brain Project“, Prof. Felix Schürmann, 2014

Titan: Zweitschnellster Rechner der Welt

Titan – Frontansicht [*Quelle: Wikipedia, Titan_(supercomputer)*]

- 200 Racks auf 400 m^2
- 18.688 Rechenknoten:
  - 1 AMD Opteron 16-core CPU
  - 1 Nvidia Tesla GPU
  - 32 GByte RAM
- Max. Rechenleistung: 27 PFlop/s
- Hauptspeicher: 710 TByte
- Speicherkapazität: 40 PByte
- Preis: $ 100 Millionen
- Leistungsaufnahme: 8.2 MW

Parallele Programmierung

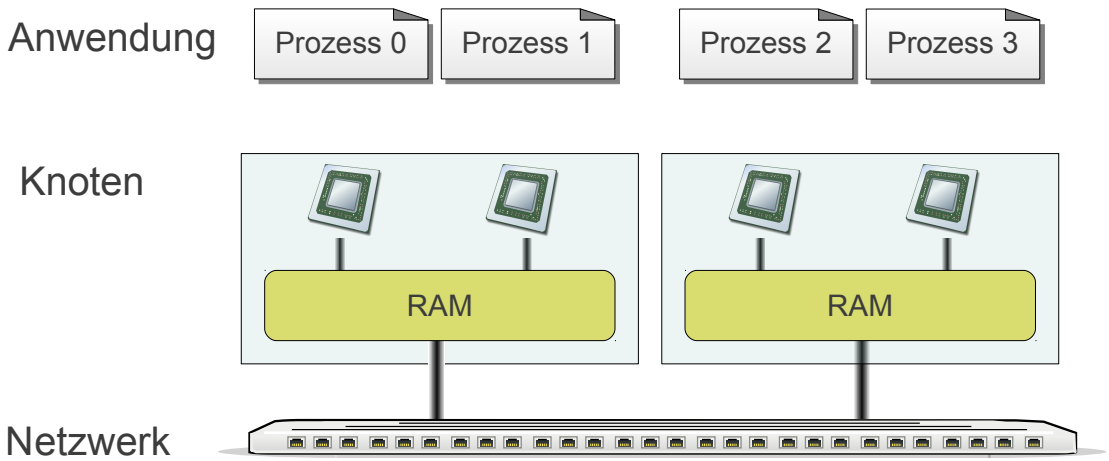
Ziel

Viele Rechenkerne kooperieren bei der Problemlösung

Vorgehen

Parallele Bearbeitung von Daten und/oder Aufgaben

- Kooperation erfordert Koordination und Datenaustausch
- Programmiermodell definiert formale Spezifikation

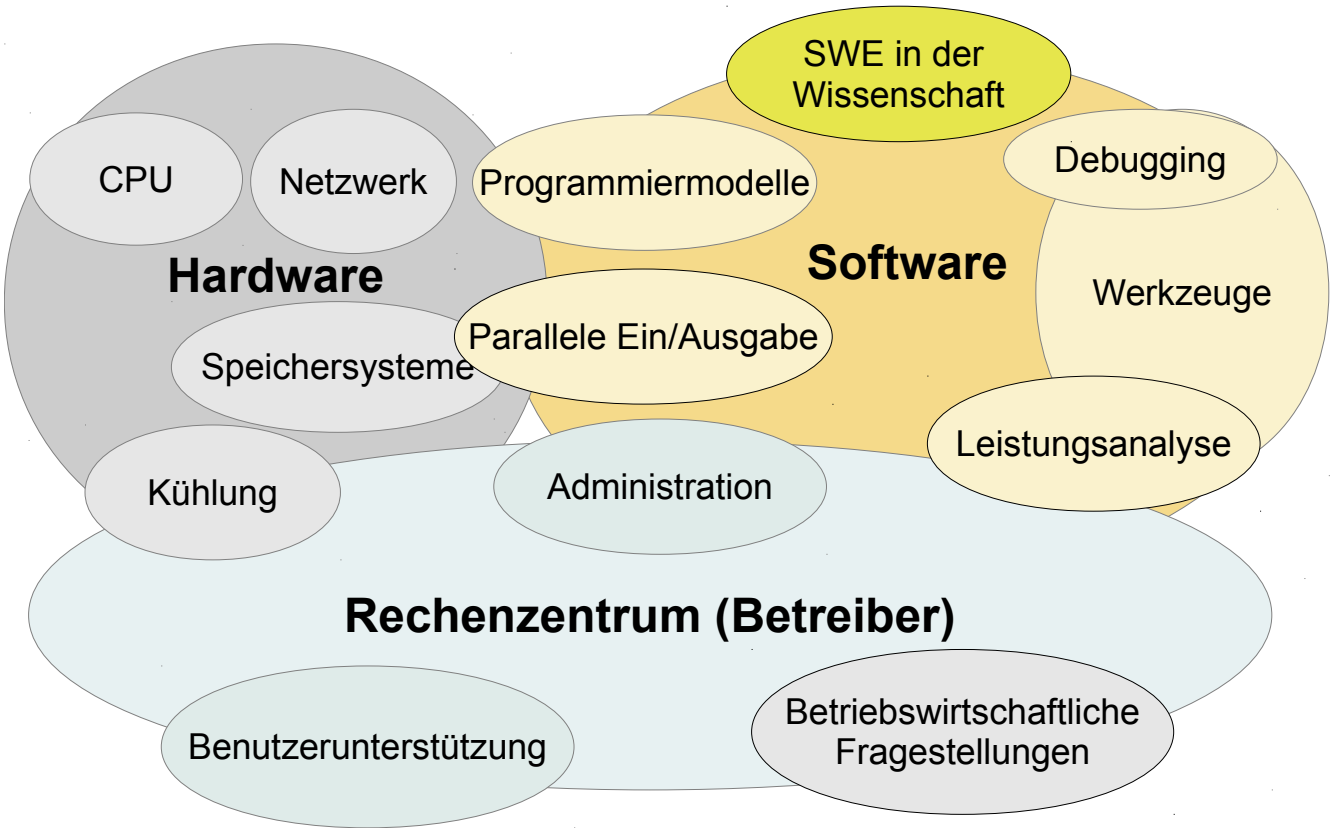


Anwendung mit vier Prozessen auf zwei Knoten

Definition: Hochleistungsrechnen

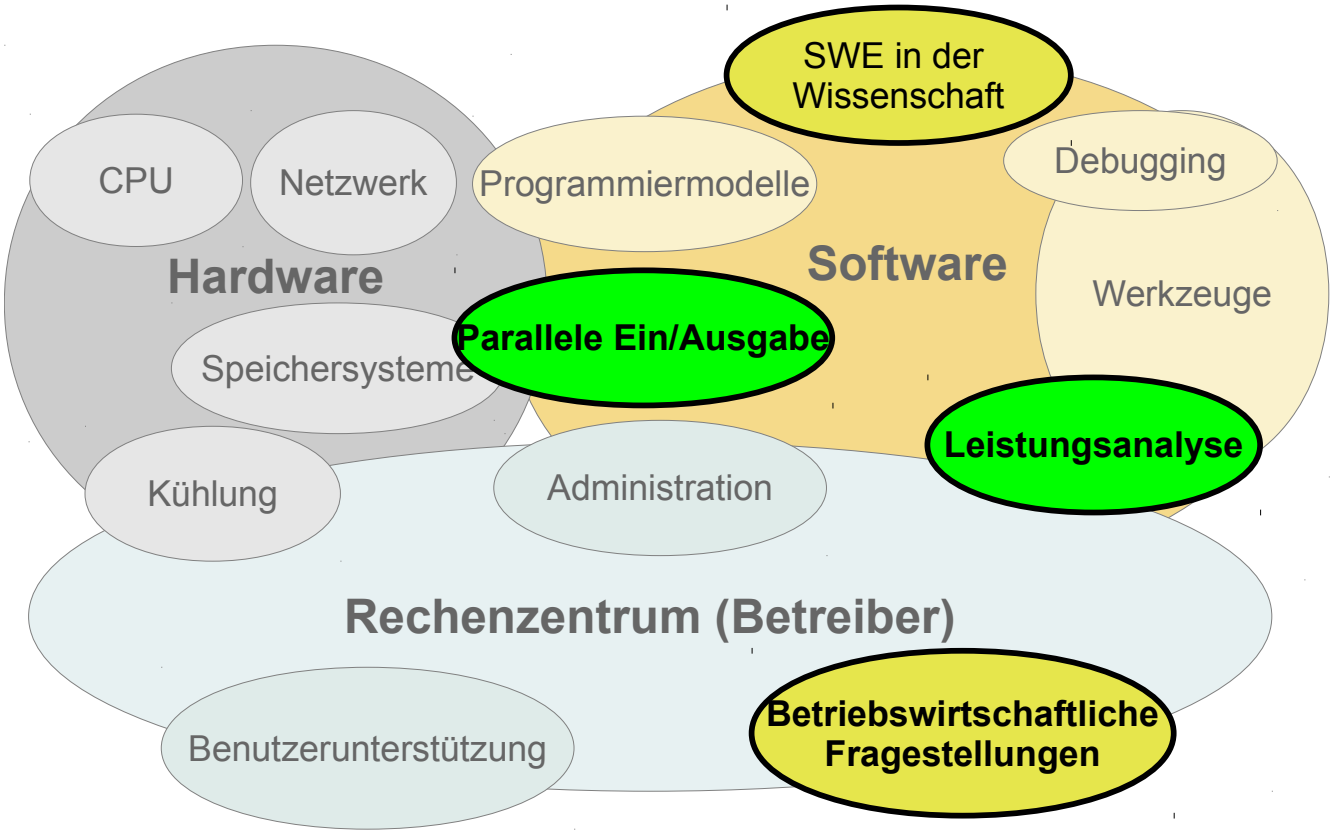
- Programmierung von Anwendungen mit hohem Ressourcenbedarf
- Disziplin der Herstellung und Nutzung von Supercomputern

Ausgewählte Teilgebiete



Forschungsprofil

- Parallele Ein-/Ausgabe & Leistungsanalyse
- Softwarentwicklung in der Wissenschaft & betriebswirtschaftliche Fragen



Leistungsanalyse und Optimierung

Motivation

Anwendungen nutzen oft nur wenig vorhandener Rechenleistung

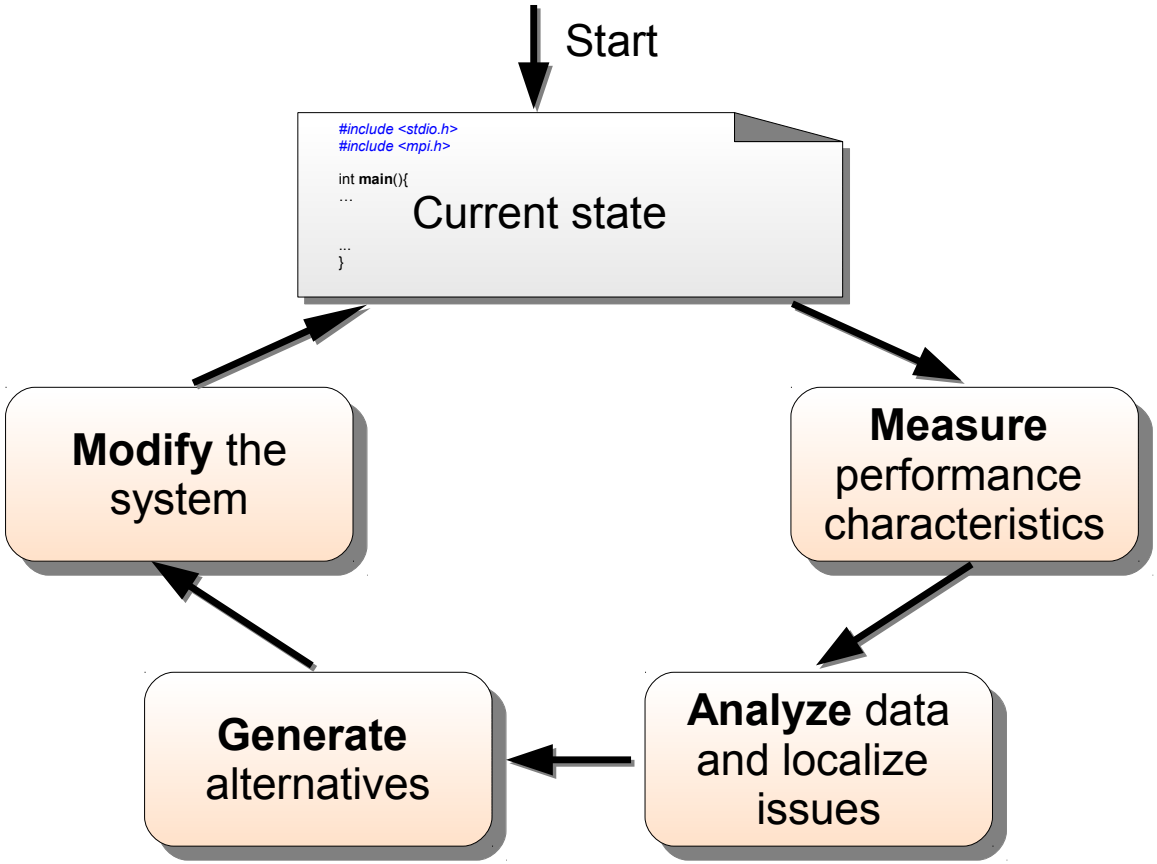
Ursachen

- Ineffiziente Berechnung, Kommunikation oder E/A
- Algorithmus/Code passt nicht zur Hardware

Optimierungsziele

- Rechenzentrum wünscht effiziente Nutzung von Ressourcen
- Anwender optimieren um komplexere Fragestellungen innerhalb von Zeiteinschränkungen lösen zu können

Tuning: Vorgehensweise zur Leistungsoptimierung

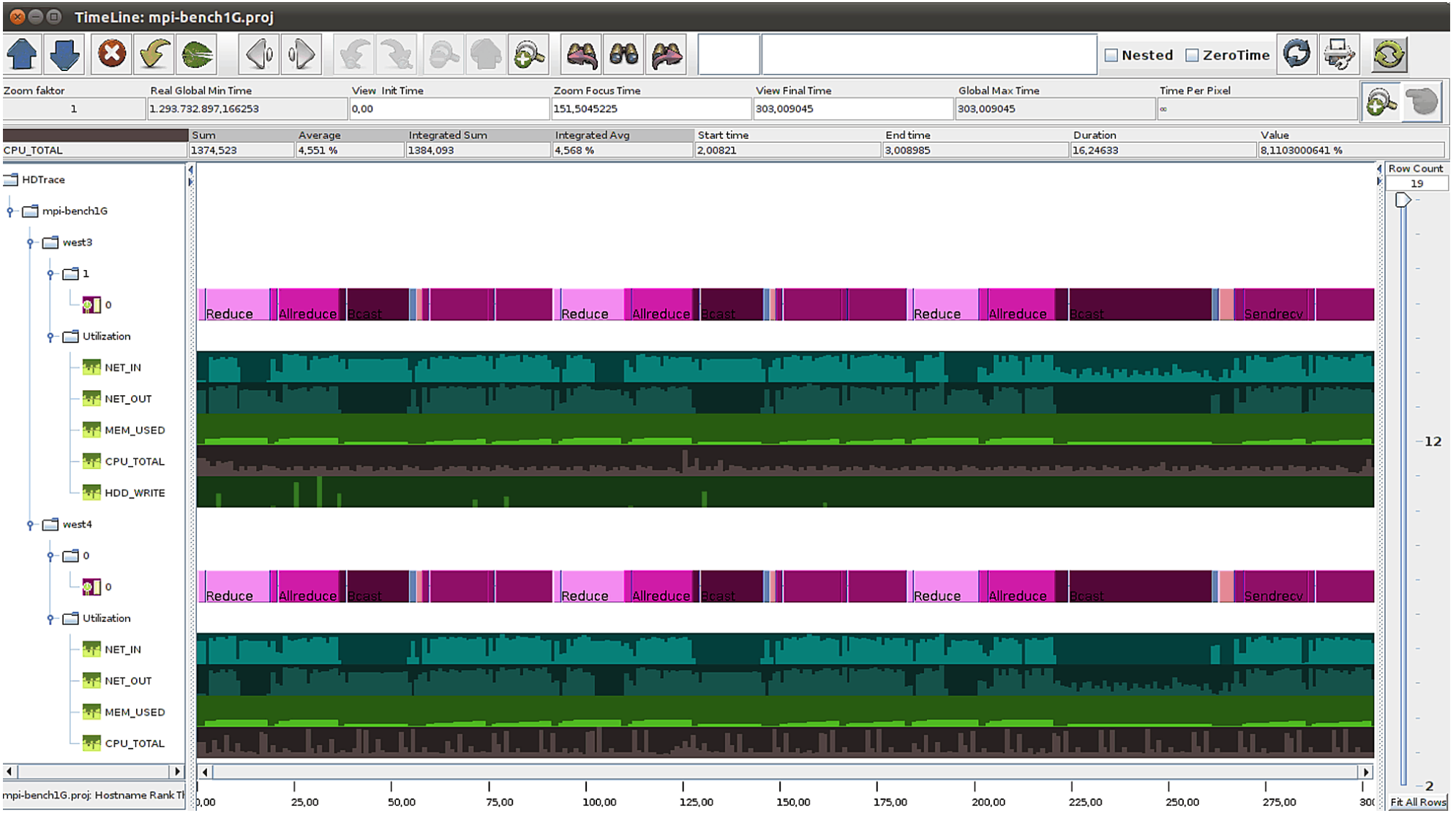


Traditionelles Vorgehen – Iterative Optimierung

Leistungsanalyse mit HDTrace

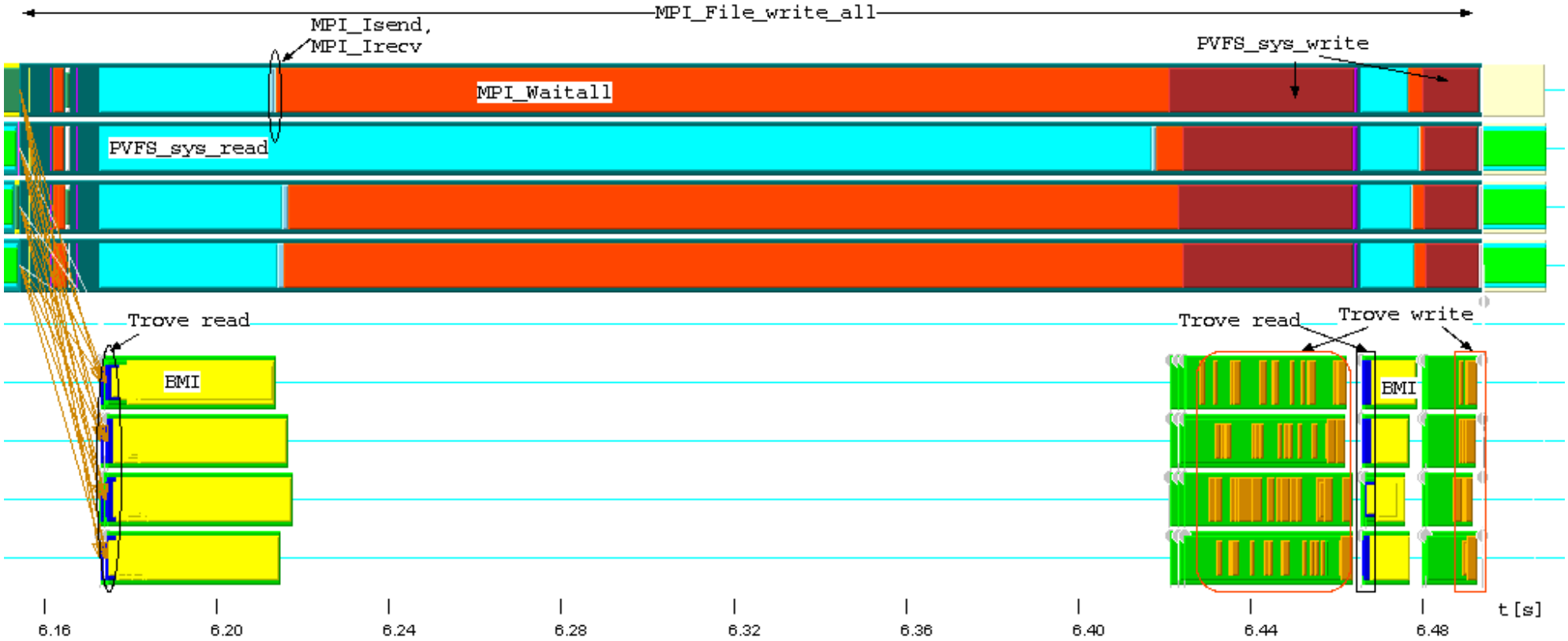
- Werkzeug für die Erfassung und Analyse von „Spurdaten“
  - Spurdaten == Zeitliche Abfolge von Programmaktivitäten
- Betrachtung von Anwendungs- und Systemverhalten
- Forschungsvehikel für Analysemöglichkeiten
  - Statistiken von Betriebsystem und Dateisystem
  - Energieverbrauch von Anwendungen
  - Aktivitäten von parallelen Dateisystemen
  - MPI-Interna (Kollektive Operationen, Datentypen und E/A-Zugriffe)
  - Alternative Visualisierungsmöglichkeiten

Visualisierungswerkzeug Sunshot



Sunshot Timelines und Statistiken

Leistungsanalyse von PVFS mittels Spurdaten



Client-Aktivität und ausgelöste Server-Aktionen

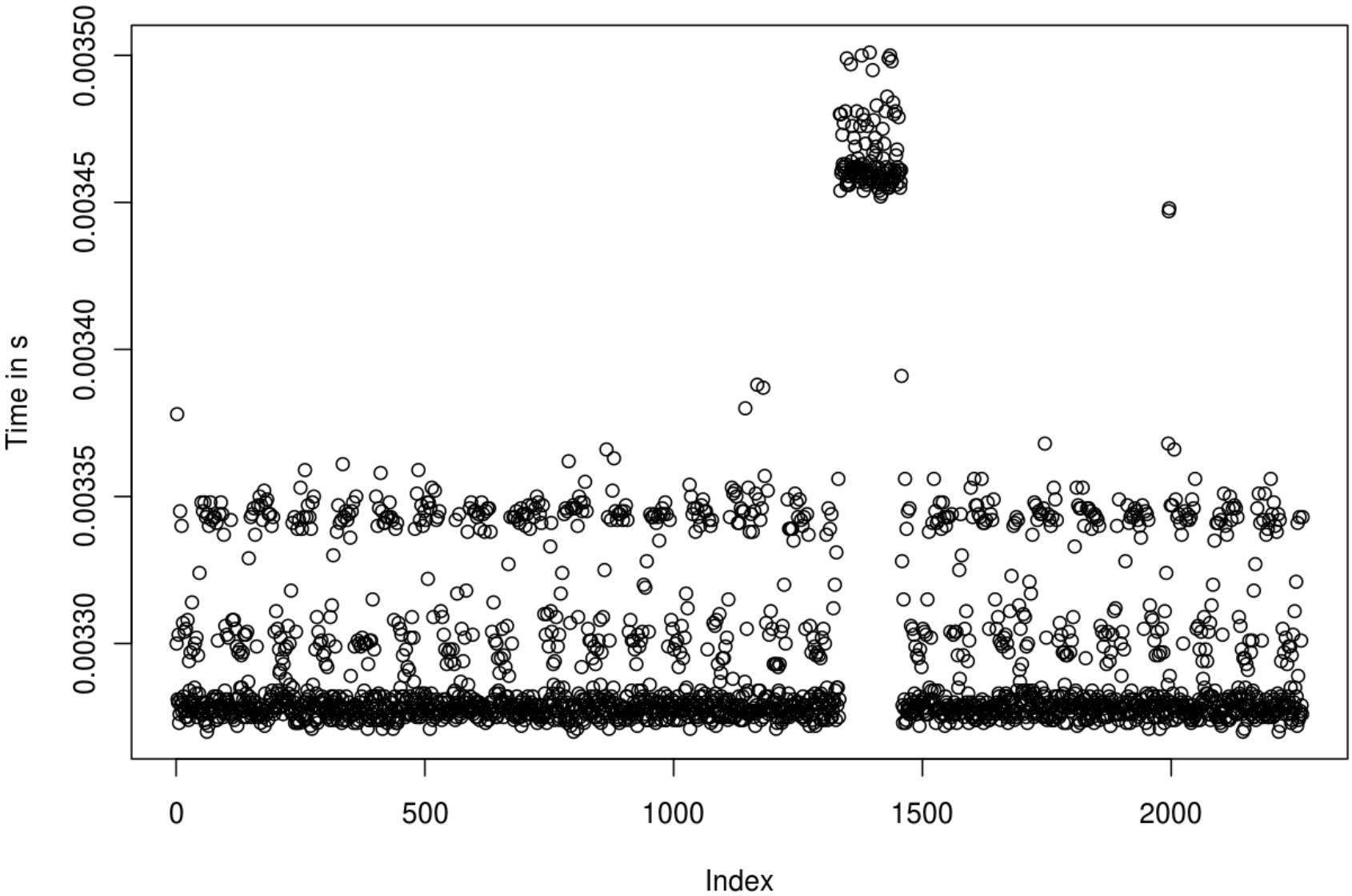
Leistungsbewertung

Erklärung von beobachteter Leistung ist nicht trivial

Einflussfaktoren

- Hardware
- Softwarekomplexität
  - Betriebsystemeinflüsse
  - Beteiligung mehrerer Schichten
- Interaktionen von Optimierungen
- Gemeinsame Nutzung von Ressourcen
- Größe des Systems und lange Programmlaufzeiten

Beispiel: MPI Punkt-zu-Punkt-Kommunikation



Intra-Sockel Einzelmessungen für 8 MiB Nachrichten

Simulation von parallelen Anwendungen

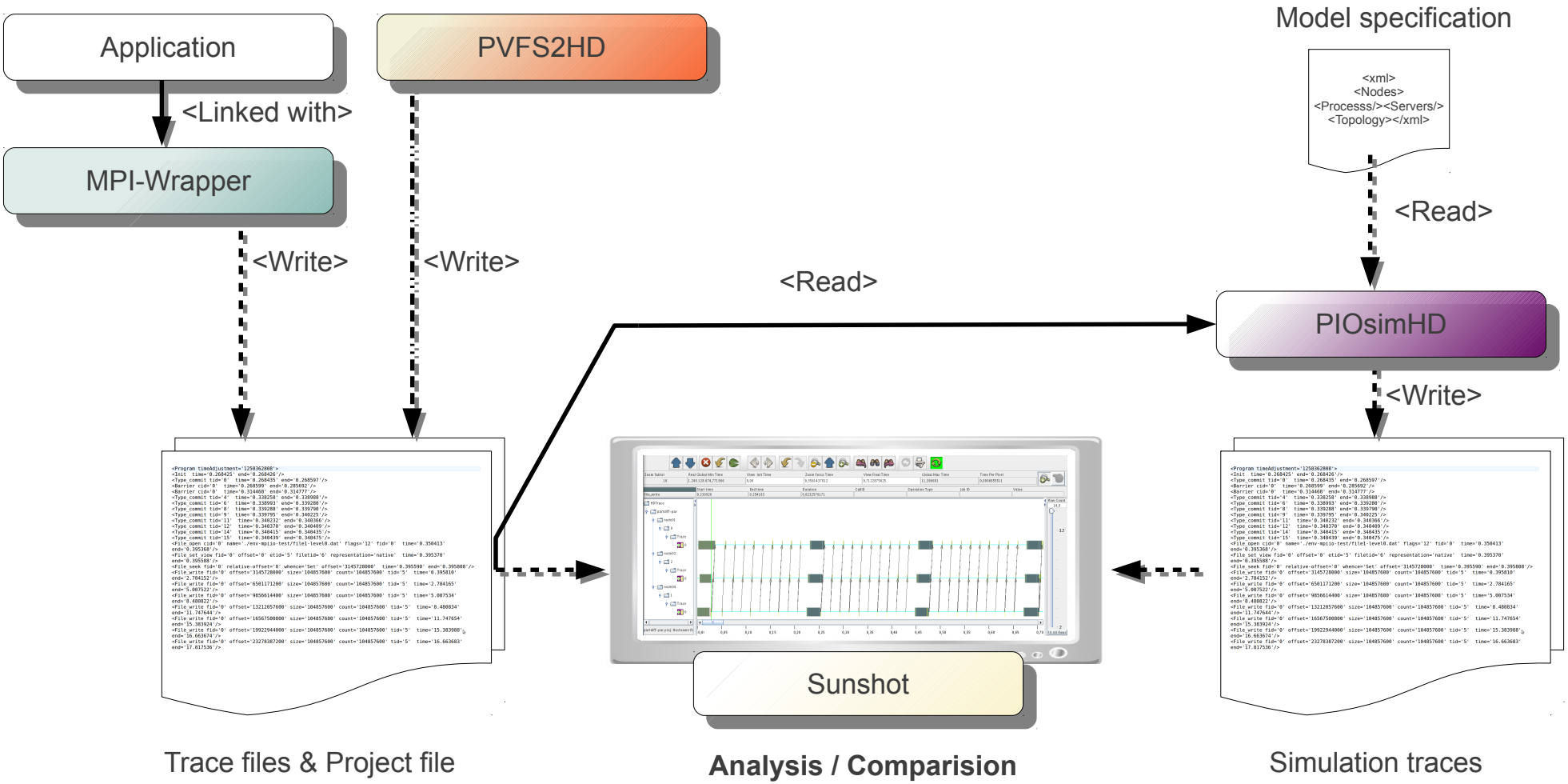
Ziele der virtuellen Forschungsumgebung PIOsimHD

- Lokalisierung von Leistungsengpässen und Ursachen
- Extrapolation von Leistungsfähigkeit für künftige Systeme
- Evaluation und Optimierungen des E/A-Pfades und MPI

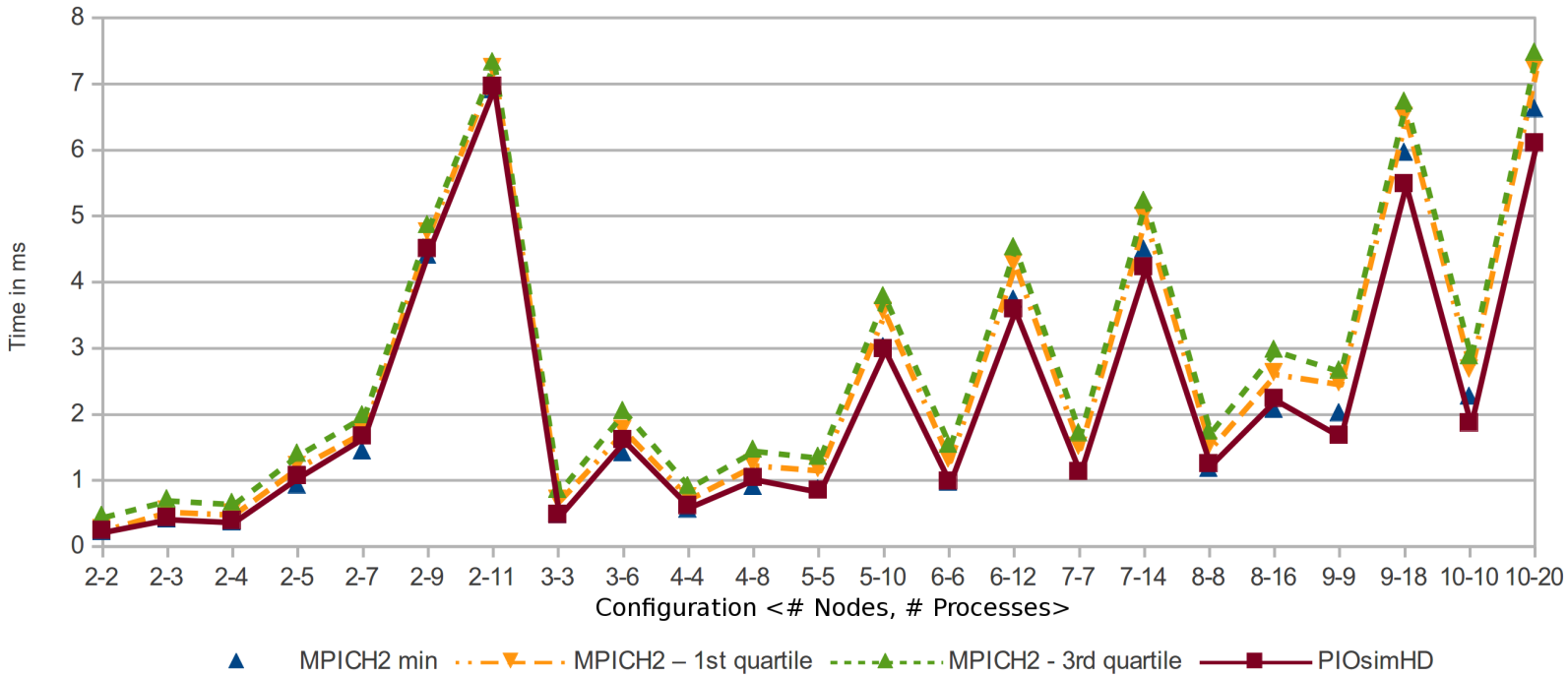
Funktionsübersicht

- Modularer ereignisorientierter Simulator
- Berücksichtigt elementare Hardware- und Software-Charakteristika
- Trace/Replay von realen Anwendungen
- Visualisierung mit Sunshot

Trace/Replay und Leistungsbewertung

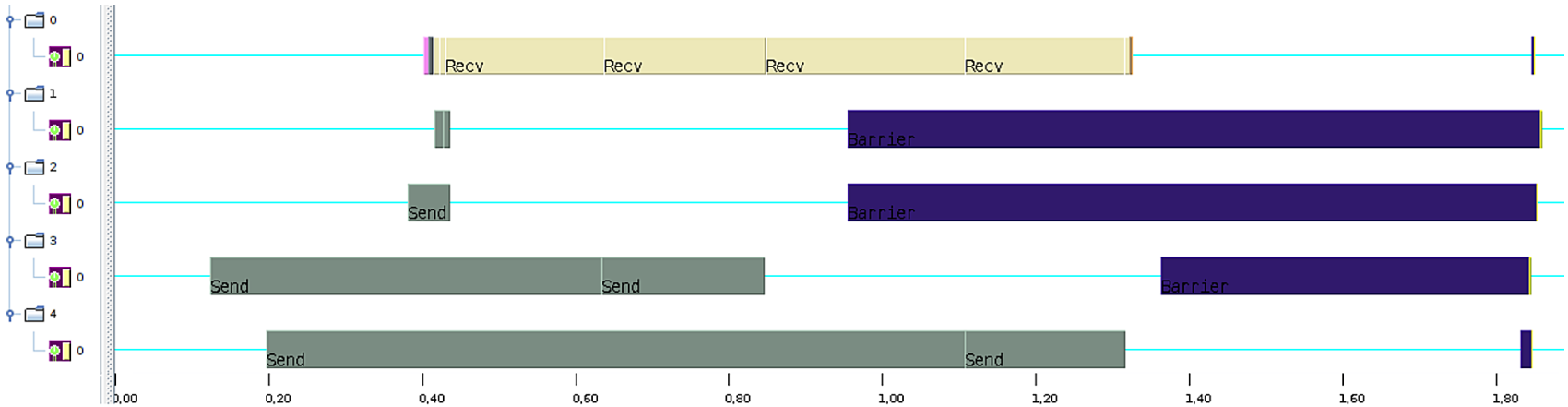


Approximation eines kollektiven Algorithmus

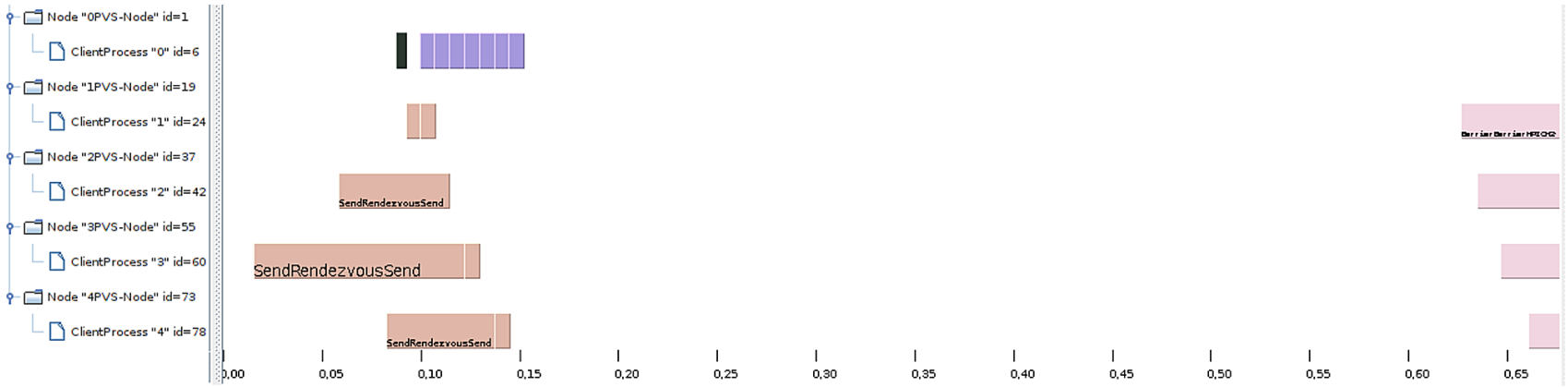


Datenaustausch mit `MPI_Allgather()` für 10 KiB Daten

Leistungsbewertung mit Hilfe von Simulation



(a) Beobachtung

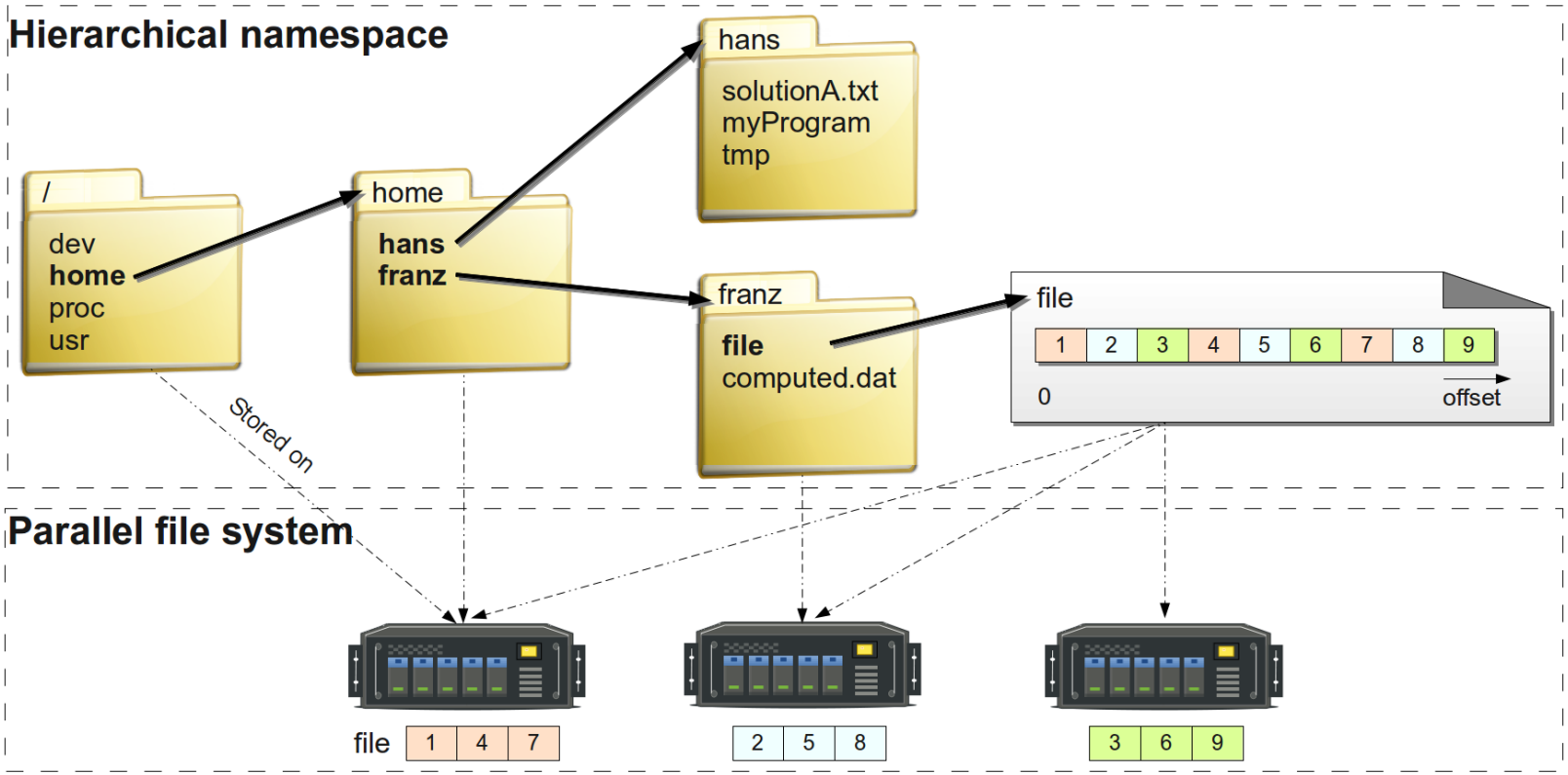


(b) Simulation

Endphase eines Jacobi-PDE-Lösers

Parallele Dateisysteme

- Verteilen Daten eines Namensraums über viele Server
- Ziel: Aggregierte Leistungsfähigkeit aller Server
- Parallel: Gleichzeitiger Zugriff mehrerer Prozesse auf eine Datei



Exemplarischer hierarchischer Namensraum

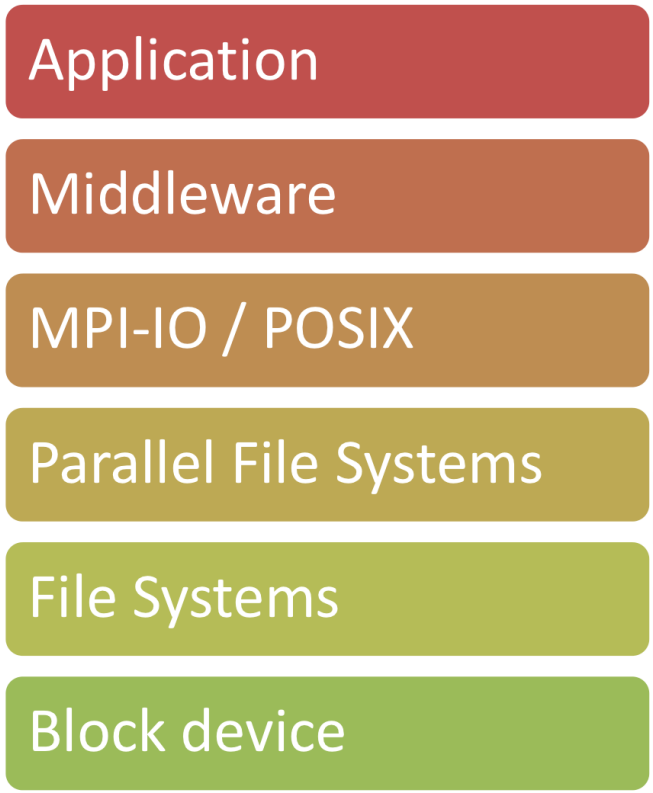
Herausforderungen im HPC-E/A-Stack

- Co-Existenz verschiedener Zugriffsparadigmen
  - Dateien (POSIX, ADIOS, HDF5), SQL, NoSQL
- Abstraktionsebene ist sehr niedrig
  - Datei ist ein Array von Bytes

  $\Rightarrow$ Communities implementieren ähnliche Features
- Reimplementation von Funktionen in Schichten

  $\Rightarrow$ Unvorhersehbare Interaktionen und Ressourcenverschwendung
- Verlust von semantischer Information

  $\Rightarrow$ Suboptimale Steuerung und Leistung
- Ungenügende „Leistungsportabilität“
  - Schicht muss für alle Systeme optimiert werden
- Management von vielen Petabyte an Daten?



Beispielhafter E/A-Stack

Semantische Kluft beim Dateizugriff

Fehlende Informationen im Dateisystem

- Datentypen
- Zugriffsemantiken
- Benötigte Parallelität
- Wert von Daten
- Lebenszyklus

Die Charakteristika können auch innerhalb einer Datei variieren

Potenzieller Nutzen der Informationen in Speichersystemen

- Leistungssteigerung: Auto-tiering, Caching
- Vereinfachung des Datenmanagements
- Datenkonsistenz sicherstellen

Manuelle Leistungsoptimierungen

- Viele Möglichkeiten um die Leistung zu steigern
  - Mittels APIs: `posix_fadvise()`, HDF5 properties
  - Kommandozeilenwerkzeuge: `lfs setstripe`
  - Setup/Initialisierung von Dateisystemen
- Optionen sind von technischer Natur
  - Beispiel: read-ahead-window von 256 KiB
  - $\Rightarrow$ Systemspezifisch, nicht portabel
- Leistungsverluste zwingen uns Optimierungen zu verwenden
  - Sind 10% der Leistung akzeptabel nur weil eine Option falsch ist?

BMBF-Projekt: SIOX

Motivation

- Mangel an Werkzeugen für die Bewertung von Anwendungs-E/A
- Vorhandene Optimierungen müssen manuell gesteuert werden

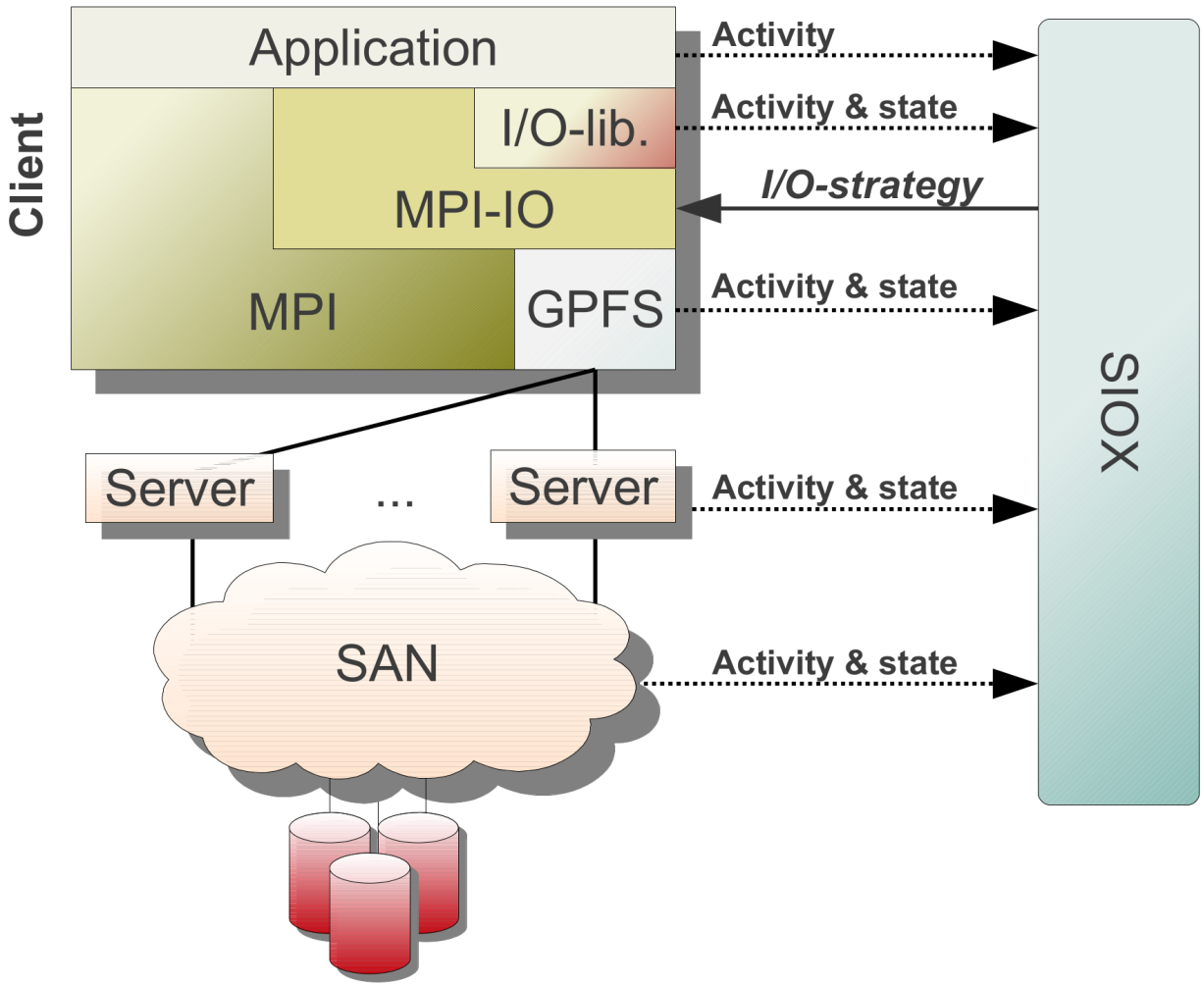
Ziele von SIOX

- Monitoring von E/A-Leistung im Produktivsystem
- Automatische Bewertung beobachteter Leistung
- Aufdeckung unbekannter Zusammenhänge
- Erlernen steuerbarer Optimierungen (Maschinelles Lernen)

Kooperationspartner

- Universität Hamburg, HLRS Stuttgart, ZIH Dresden, Exascale10

Abstrakte Funktionsweise



Integration von SIOX in den E/A-Stack

Beispiel-Optimierung: Pre-fetching

- Pre-fetching: Daten von langsamen Medium vor Bedarf einlesen
- Anwendung von `posix_fadvise()` zum Pre-fetching
  - Teilt dem Betriebsystem mit gewisse Daten in RAM zu laden
- Testprogramm liest Daten, simuliert Verarbeitungszeit
  - 100 μs und 10 ms für 20 KiB bzw. 1000 KiB stride
  - Programm: Normale Ausführung und manuelles Pre-fetching
  - SIOX Plug-in kann ohne Code-Modifikation Pre-fetchen

Resultate

Experiment	20 KiB stride	1000 KiB stride
Normale Ausführung	97.1 μs	7855.7 μs
Eingebettetes fadvise	38.7 μs	45.1 μs
Mit SIOX ausgeführtes fadvise	52.1 μs	95.4 μs

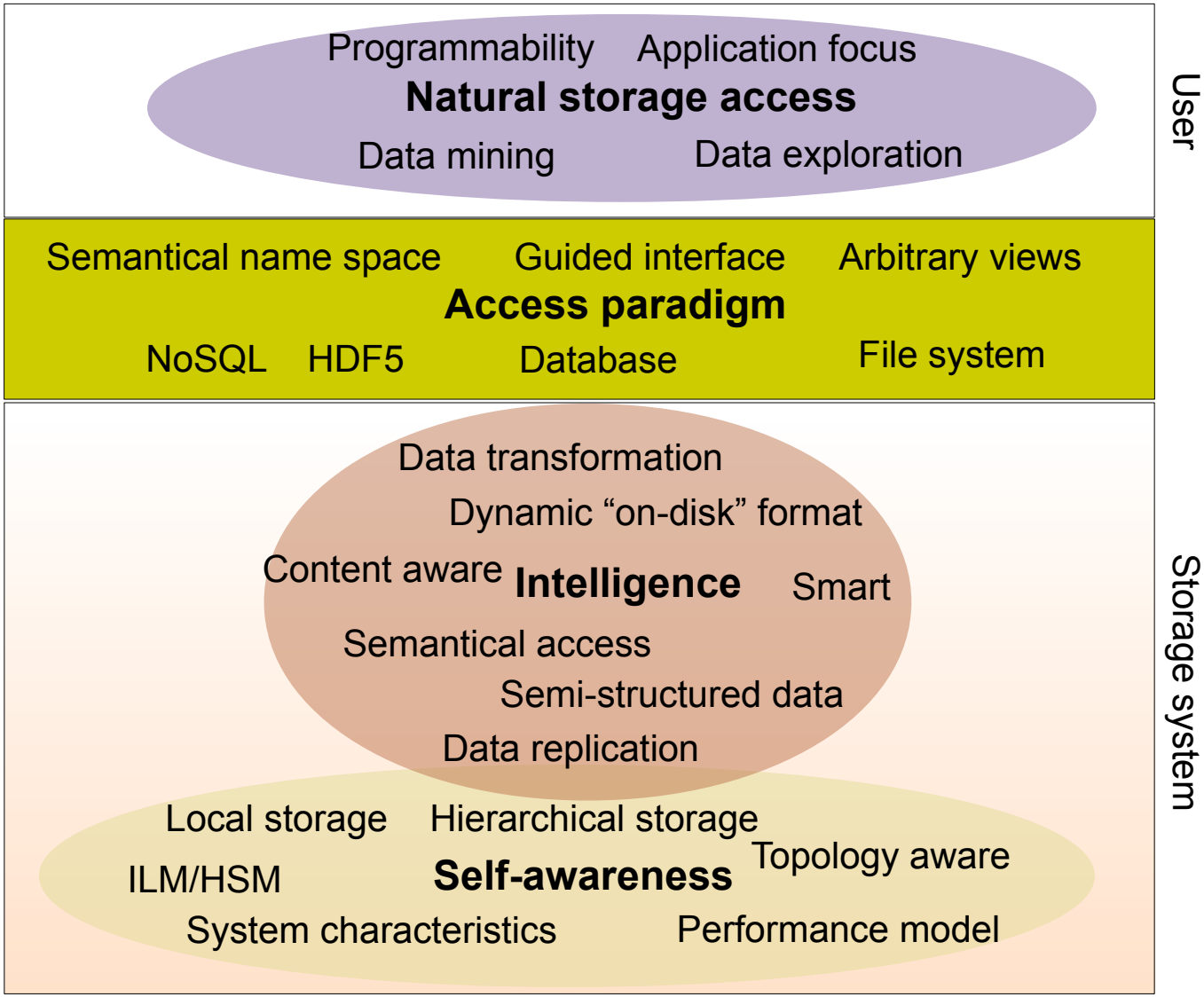
Benötigte Zeit um einen 1 KiB Datenblock zu lesen.

Kritische Nutzerfragen

- Warum muss ich mich soviel mit Datenformaten beschäftigen?
  - Das erinnert daran, das Speicherformat von C-structs festzulegen
- Warum müssen Daten zwischen Zugriffsparadigmen konvertiert werden?
  - HDF-Datei speichern und mittels SQL zugreifen?
- Wo sind meine Forschungsergebnisse mit folgenden Eigenschaften?
  - Leider muss ich mich nach den festen (Ordner)-Strukturen richten
- Warum muss ich Hardware-spezifische Leistungsoptionen setzen?
  - Erinnert an manuelle Code-Optimierung anstelle Compiler-Nutzung

„Ich würde mich gerne auf wissenschaftliche Fragen konzentieren“

Persönliche Vision Zukünftiger Speichersysteme



Die Datenspeicherung der Zukunft Beginnt Heute

- Exascale10 entwickelt neuartigen Speicher
  - Arbeitsgruppe von OpenSFS und EOFS (Lustre)
- Internationale/Globale Kollaboration
  - Data-type aware storage
  - Multiple “views” to the same data (BigData)
  - Guided interfaces instead of technical hints
  - Data “format” handled by storage system
  - Multi-tiering support
  - Intelligent monitoring
  - Feedback to optimization and “what-if” analysis
  - Integrates active storage concept
  - Post-processing handled by file system
- Erste Design-Dokumente wurden publiziert

EXASCALE IO

`http://www.exascale10.com`

Zusammenfassung

- Hochleistungsrechnen ist ein spannendes Forschungsfeld
- Supercomputer sind komplexe Systeme
- Aktuelle Systeme arbeiten oft nicht optimal
  - Die ganzheitliche Betrachtung von Hardware, Software und Nutzung ist für eine nachhaltige Optimierung notwendig
- Bedarf an Semi-automatische Optimierungswerkzeugen
  - SIOX bündelt Erfassung, Analyse und Optimierung
- Revolutionäre Konzepte für Speichersysteme werden benötigt
  - Exalscale10 ist enabler für Daten-Intensive Forschung

Backup Folien

Leistungsanalyse

Fragestellungen

1. Wie analysieren wir das Laufzeitverhalten von Anwendungen?
   - Spurdaten-Werkzeuge erfassen Aktivitäten
   - Manuelle Analyse des beobachteten Verhaltens
2. Wie bewerten wir beobachtetes Verhalten?
   - Relation von Leistungsfähigkeit und Beobachtung
   - Simulation vereinfacht theoretische Betrachtungen
3. Wie können wir die Leistungsfähigkeit von Hardware erfassen?
   - Benchmarks erzeugen definierte Lasten
   - Herstellerangaben als Referenz
4. Welche Optimierungen sollten am System durchgeführt werden?
   - Zielgerichtet durch Nutzungsstatistiken und Benutzerdialog

Programmiermodell Nachrichtenaustausch

Message Passing Interface (MPI)

- Bibliothek für C und Fortran
- Expliziter Datenaustausch und Synchronisation
  - Punkt-zu-Punkt-Kommunikation
    - `MPI_Send()`, `MPI_Recv()`, ...
  - Kollektive Operationen
    - `MPI_Bcast()`, `MPI_Scatter()`, ...
  - Parallele Ein-/Ausgabe

```
1 if (process == 0){
2   int res = taskA();
3   MPI_send(& res, 1, MPI_INT, 1, ...);
4 }else if (process == 1){
5   int remote_res;
6   int res = taskB();
7   MPI_recv(& remote_res, 1, MPI_INT, 0, ...);
8   taskC(res, remote_res);
9 }
```