Karlsruher Institut für Technologie

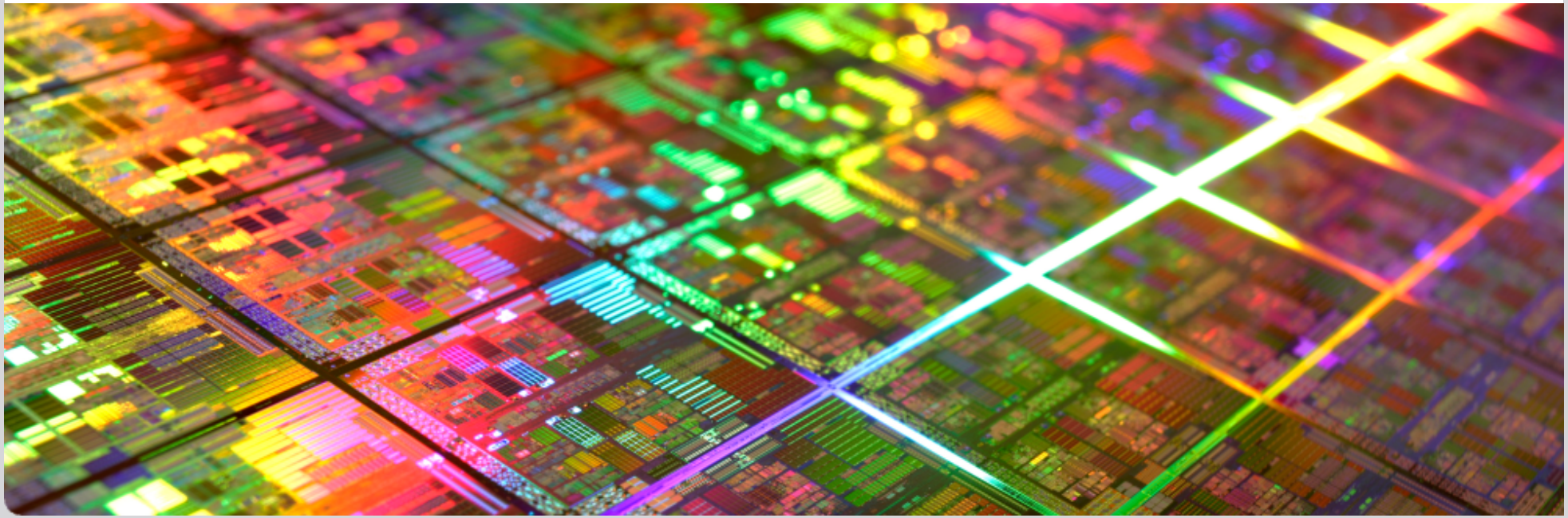
envelope

ENVELOPE - Effizienz und Zuverlässigkeit: Selbstorganisation in HPC-Systemen

Aktueller Stand

A. Brinkmann, W. Karl, S. Lankes, M. Schulz, C. Trinitis

8. Oktober 2018



Partner

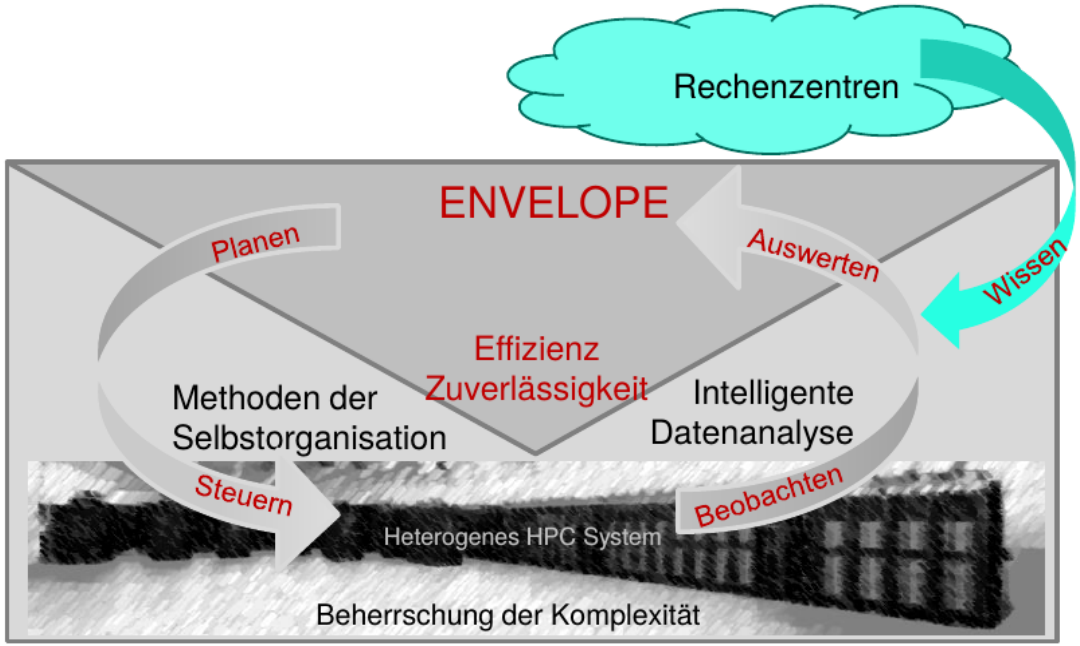
- **Karlsruher Institut für Technologie**
  - Institut für Technische Informatik
  - Prof. Dr. Wolfgang Karl (Koordination)
- **Technische Universität München**
  - Lehrstuhl für Rechnerarchitektur und Parallele Systeme
  - Prof. Dr. Martin Schulz, Dr.-Ing. Carsten Trinitis
- **Johannes Gutenberg-Universität Mainz**
  - Zentrum für Datenverarbeitung
  - Prof. Dr.-Ing. André Brinkmann
- **RWTH Aachen University**
  - Institute for Automation of Complex Power Systems
  - Prof. Antonello Monti, Ph.D., Dr. rer. nat. Stefan Lankes
- **Assoziierte Partner**
  - Leibniz Rechenzentrum
    - PD Dr. rer. nat. Josef Weidendorfer
  - MEGWARE, ParTec

Ziele

- Erhöhung der Zuverlässigkeit und Ausfallsicherheit in HPC-Systemen
- Effiziente Nutzung der zur Verfügung stehenden Ressourcen im Hinblick auf
  - Ausführungszeit
  - Energieeffizienz
- Verbergen der Komplexität heterogener HPC-Systeme vor dem Anwender

Ansatz

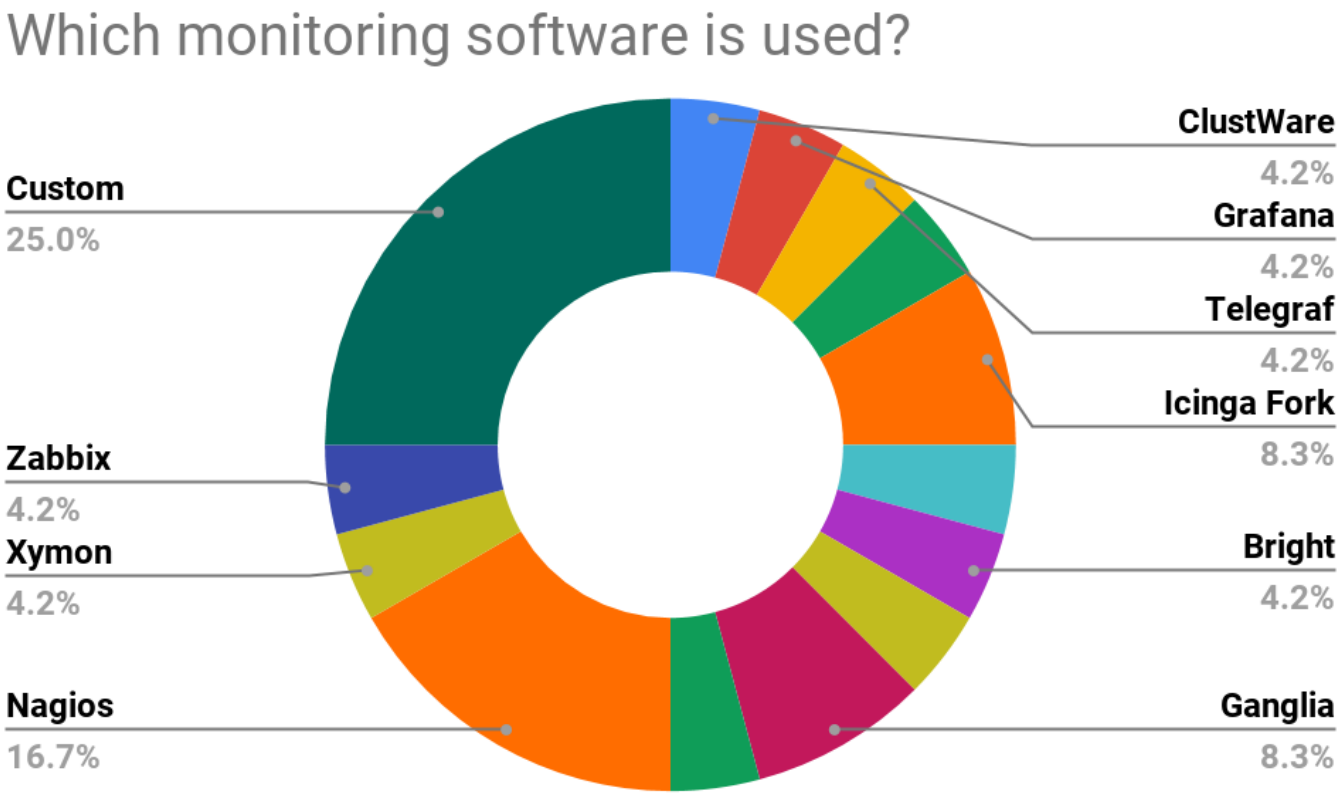
- Betrachtung des Systems auf mehreren Ebenen
  - Lokale Betrachtung der Rechenknoten
  - Globale Sichtweise des Systems
  - Techniken auf Anwendungsebene
- Einsatz von Methoden der Selbstorganisation
  - Vergleich von system- und anwendungsbasierten Strategien



Ansatz

- Erhöhung der Zuverlässigkeit und Ausfallsicherheit
  - Proaktive Vorhersage von Knotenausfällen mittels Machine-Learning Modellen
  - Datenhaltungskomponenten zur Migration und Checkpointing
    - Anwendungsgesteuert
    - Containerbasiert
    - Unikernels
- Effiziente Nutzung der vorhandenen Ressourcen
  - Dynamische Abbildungsentscheidung mit unterschiedlichen Optimierungszielen
- Verbergen der Komplexität heterogener Systeme
  - Task-basiertes Laufzeitsystem mit Bibliotheksansatz

Rechenzentrumsumfrage

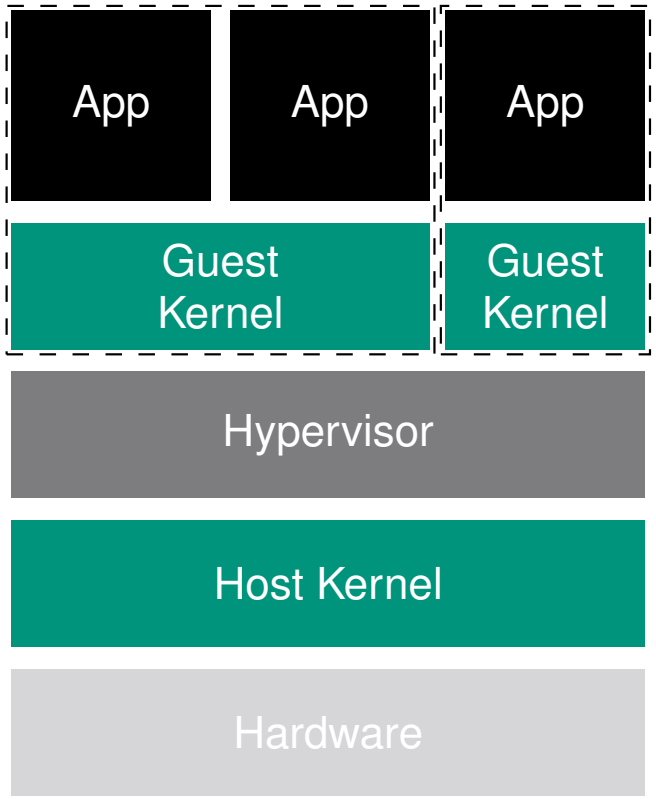


Prädiktor

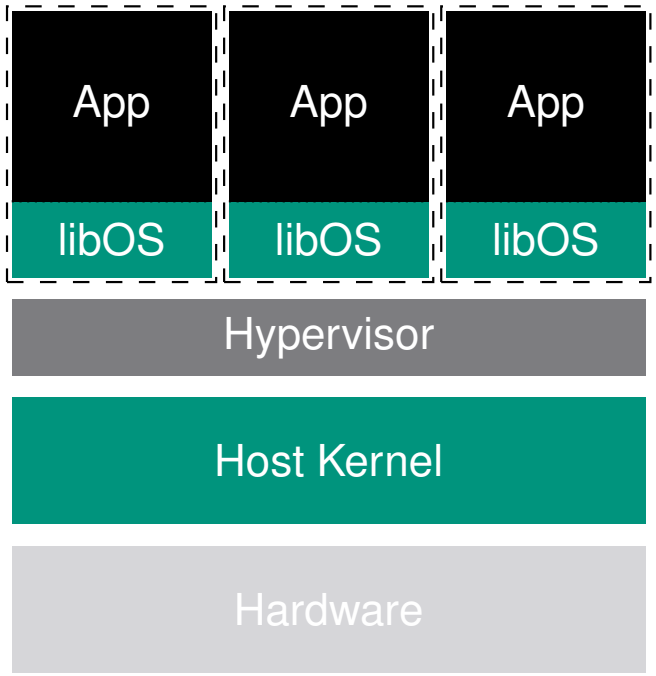
Ziel: Erstellung eines Prädiktors zur proaktiven Vorhersage von Ausfällen der Rechenknoten

- Datensammlung aus MOGON I & MOGON II an der JGU Mainz
  - IPMI (Spannung, Temperaturen), OS (CPU, Last, Speicher), Job-/Knotenzustände
- Komprimierte Daten: ~2 TB für 1 Jahr und über 500 Knoten
- Tagliche Daten: ~5 GB für 500 Knoten
- MOGON I & II benutzen Ganglia, RRD Datenbasis und Custom Daemons für die Datensammlung

Checkpointing & Migration von Unikernels



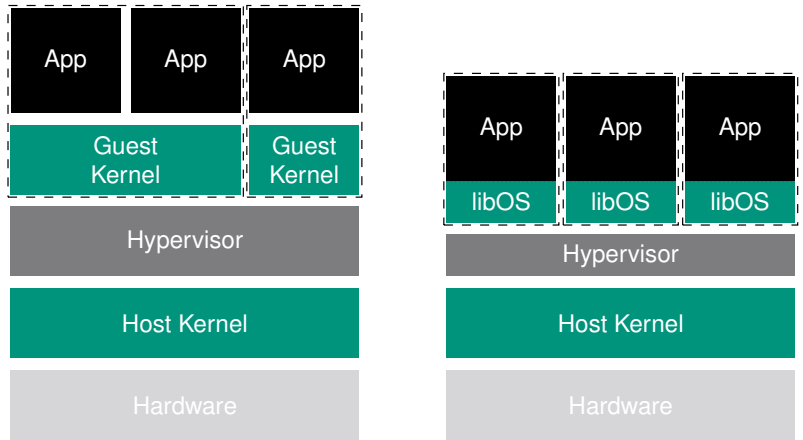
Klassische VMs



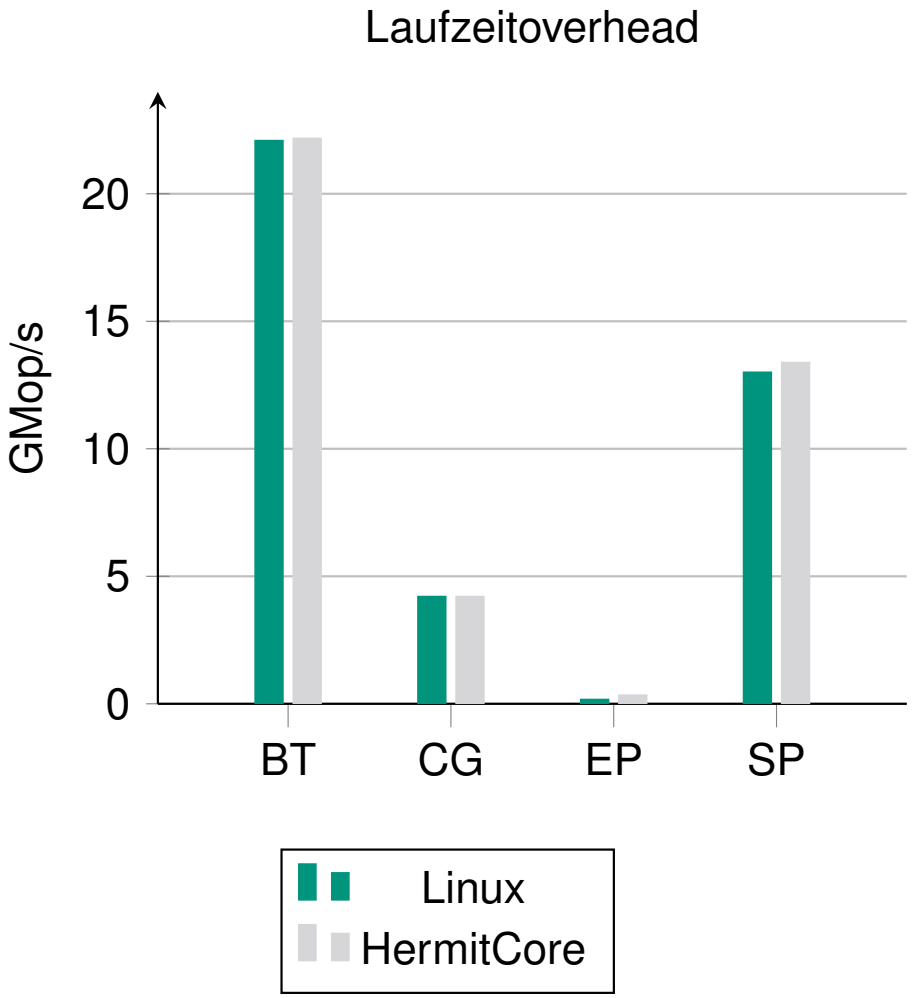
Unikernels

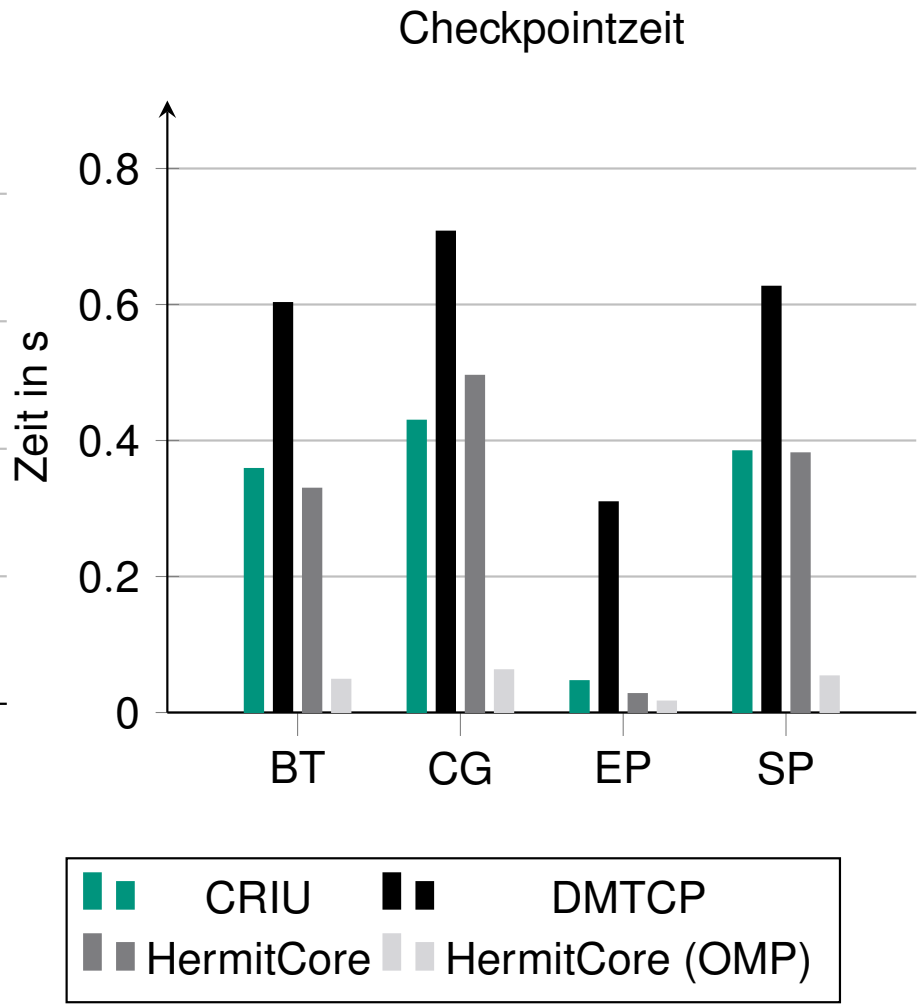
Checkpointing & Migration von Unikernels

- Nur **eine** Anwendung pro Gast
- Geteilter Adressraum zwischen Anwendung und libOS
- Hypervisor kennt den vollständigen Gaststatus
  - Eine Seitentabelle
  - Checkpointing/Migration per Page-Walk im Hypervisor
  - Klassische Parallelisierung möglich (z. B. OpenMP)
- Nutzung des Unikernels Hermitcore (https://hermitcore.org)

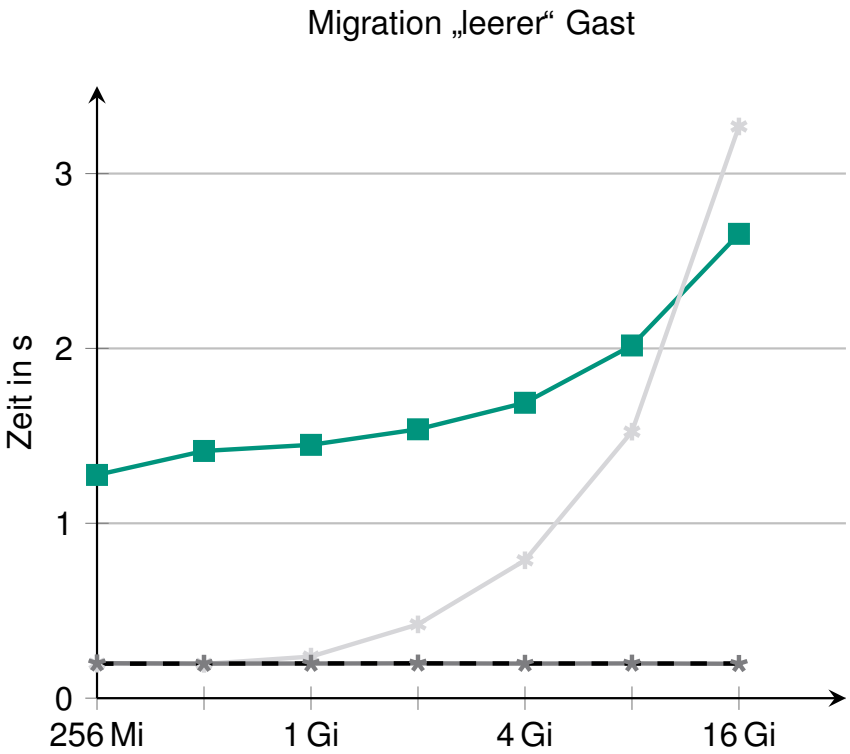


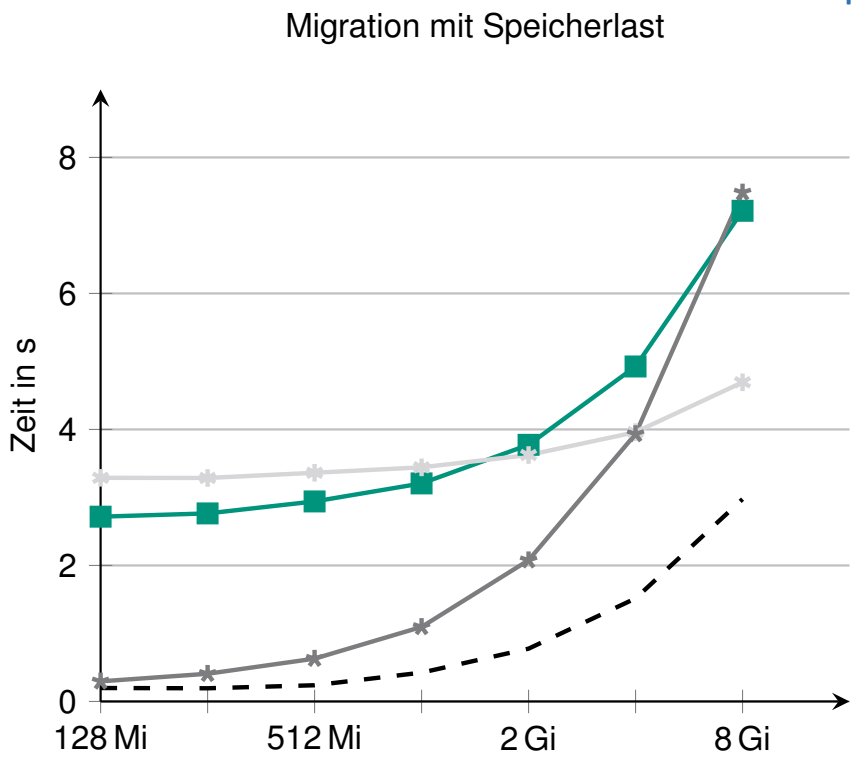
Checkpointing & Migration von Unikernels





Checkpointing & Migration von Unikernels



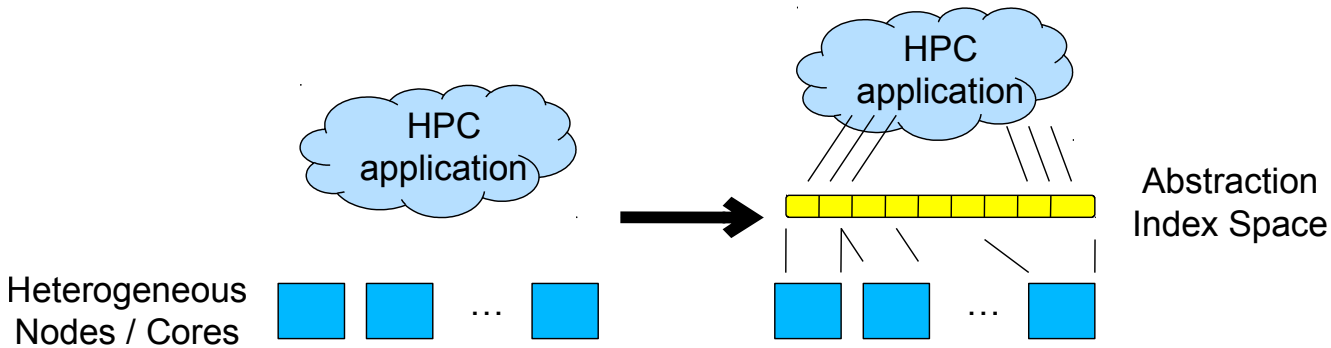


QEMU — HermitCore — HermitCore (ODP) - - - HermitCore (free-list)

Datenerhaltung auf Anwendungsebene

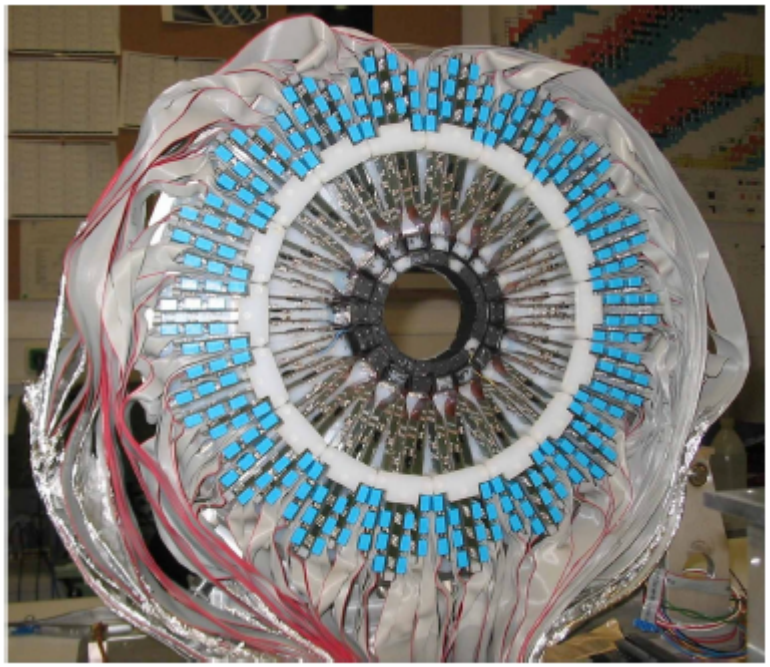
LAIK, eine Bibliothek zur dynamischen Datenverteilung

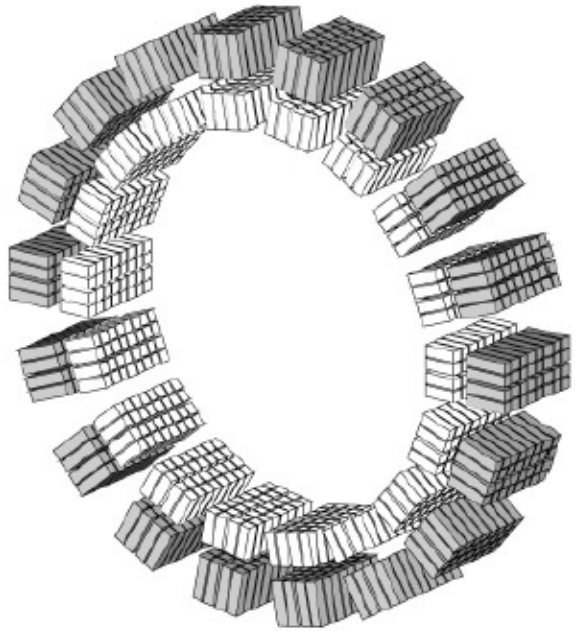
- Ermöglicht die Repartitionierung bei Ausfällen
- Einfache Portierung von existierenden Code
- Bewahrt die Skalierbarkeit von Anwendungscodes
- Erlaubt die inkrementielle Portierbarkeit von MPI-Code
- Kontrolliert die (dynamische) Verteilung und die Kommunikation von Daten
- Datenaffinität: stellt Lokalität benötigter Daten sicher
- Deklariert, wann / welche Daten aktualisiert werden müssen



Anwendungsportierung: MLEM

kleiner Tier-PET-Scanner
MADPET-II

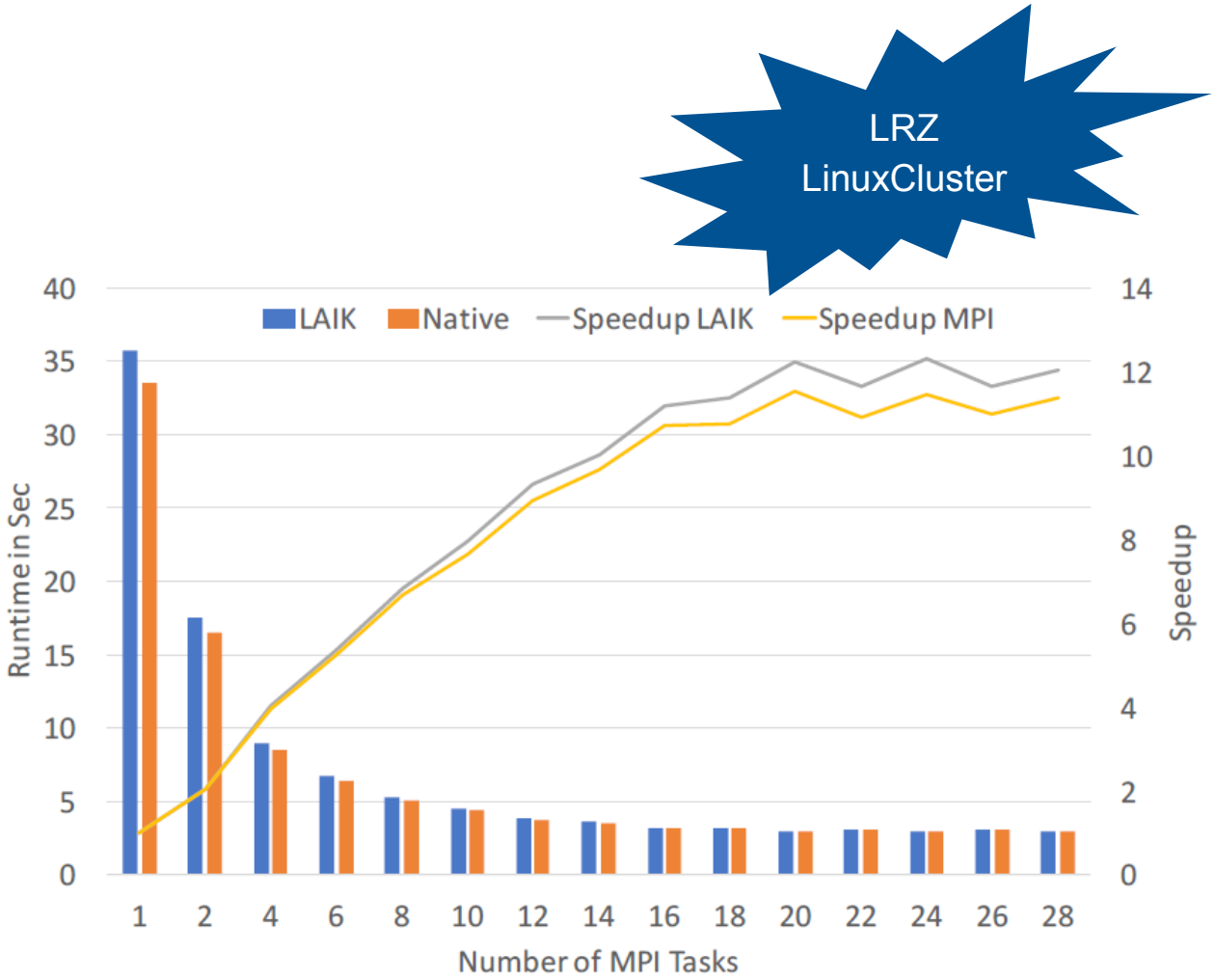




1152 Detektoren, 662976 Lines-of-Response
Sichtfeld 140 x 140 x 40 voxels, insgesamt 784000 voxels

Ergebnisse

envelope



Dai Yang, Josef Weidendorfer, Tilman Küstner, Carsten Trinitis and Sibylle Ziegler. Enabling Application-Integrated Proactive Fault Tolerance. Par-Co 2017, Bologna, Italy.

Zusammenfassung und Ausblick

- Framework zur Erhöhung der Fehlertoleranz
  - Anwendungsgesteuert ⇒ LAIK
  - Anwendungstransparent ⇒ Container, VMs, Unikernels
- Verfeinerung der Techniken zur Vorhersage von Ausfällen
  - Symptombasierte Fehlererkennung
  - Auswertung der Sensoren mit Hilfe von Machine Learning
- Weitere Informationen
  - Projektseite: `http://envelope.itec.kit.edu`
  - LAIK & Co.: `https://github.com/envelope-project/laik`
  - Migration von Unikernels: `https://hermitcore.org`

Backup

Symptombasierte Fehlererkennung

Aktueller Stand:

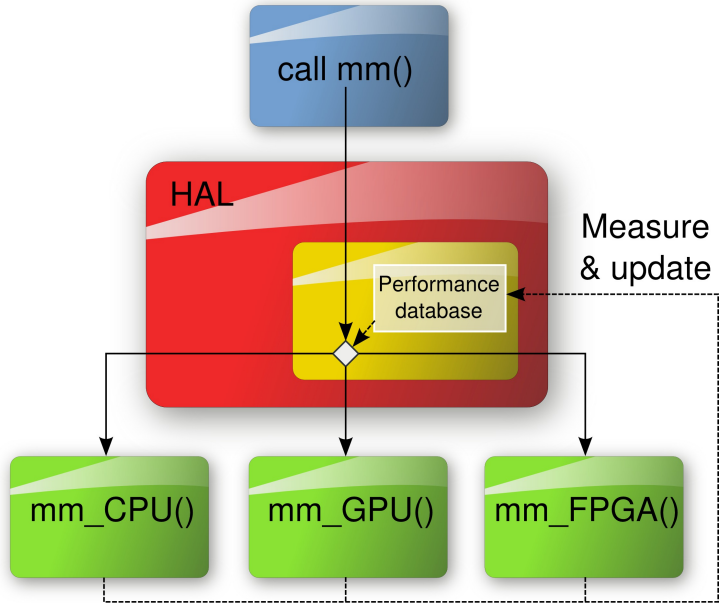
- Untersuchung verschiedener Fehlerfälle
  - Verringerung der CPU-Frequenz
  - Veränderung von Schleifenvariablen
  - Speicherleaks
  - Auslastung der Festplatte
  - Auslastung der ALU

Task-basiertes Laufzeitsystem

- Bisher: Task-Abbildungsentscheidung aufgrund von Ausführungszeiten
- Ziel: Betrachtung des Energieverbrauchs bei der Abbildungsentscheidung
  - Entscheidung über Gewichtung

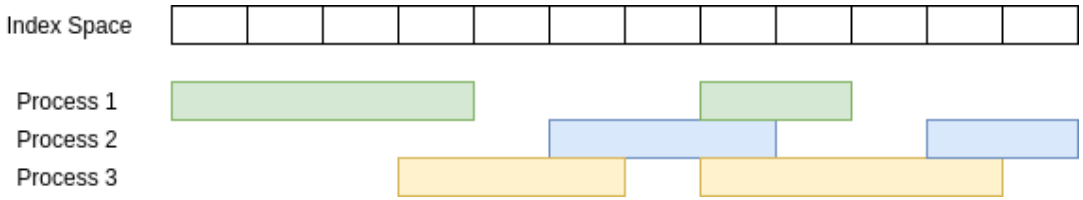
$\Rightarrow$ Performance-Datenbank um Energieverbrauch erweitert

  - RAPL-Counter für Intel CPUs
  - NVML-Counter für NVIDIA GPUs

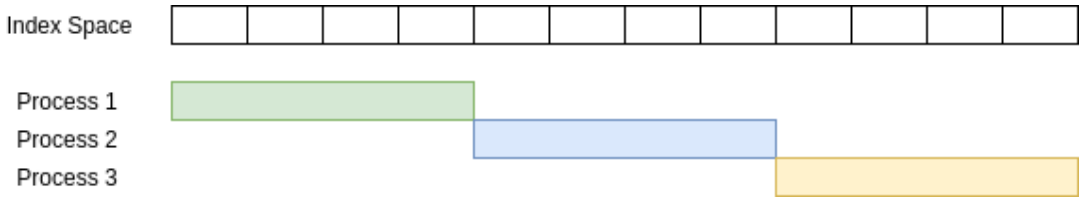


Partitionierungsbeispiel

- Allgemeine Partitionierung: mehrere Prozesse pro Index



- Disjunktive Partitionierung



- Benutzerdefinierte Partitionsalgorithmen werden unterstützt

MLEM Algorithmus

$$f_j^{(q+1)} = \frac{f_j^{(q)}}{\sum_{l=1}^{n} a_{lj}} \sum_{i=1}^{n} \frac{g_i}{\sum_{k=1}^{m} a_{ik} f_k^{(q)}}$$

Schritte:

- Vorwärtsprojektion $h = Af$
  - $A = (a_{ij}) \in \mathbb{R}^{nxm}$ Wahrscheinlichkeitsmatrix
  - $f \in \mathbb{R}^m$ Bildvektor
- Korrelation $r = \frac{g_i}{h_i}$
  - $g \in \mathbb{R}^n$ Bildvektor Scanner-Ausgabe
- Rückwärtsprojektion $c = A^T r$
- Update $f^{(q+1)_j = f_j^{(q)} c_j}$

Erhöht die Wahrscheinlichkeit $p(g|f^{(q)})$ von g mit der gegebenen aktuellen Schätzung $f^{(q)}$