Ein Werkzeugkasten zur Integration von Selbstadaption in zeitschrittbasierte Simulationen

GEFÖRDERT VOM

Matthias Korch

Universität Bayreuth
Institut für Informatik

9. HPC-Status-Konferenz der Gauß-Allianz
Paderborn Center for Parallel Computing (PC^2)

17. und 18. Oktober 2019

Inhalt

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

1. Projektüberblick
2. Werkzeugkasten
   - Basiswerkzeuge
   - Offline-Tuning
   - Online-Tuning
3. Demonstratoren
   - Fallbeispiel: Partikelsimulationen
4. Zusammenfassung und Ausblick

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Selbstadaption für zeitschrittbasierte Simulationstechniken auf heterogenen HPC-Systemen

- Gefördert vom BMBF seit 1. 1. 2017
- Programm „IKT 2020 – Forschung für Innovationen“
- Bekanntmachung „Grundlagenorientierte Forschung für HPC-Software im Hoch- und Höchstleistungsrechnen“

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Selbstadaption für zeitschrittbasierte Simulationstechniken auf heterogenen HPC-Systemen

- Gefördert vom BMBF seit 1. 1. 2017
- Programm „IKT 2020 – Forschung für Innovationen“
- Bekanntmachung „Grundlagenorientierte Forschung für HPC-Software im Hoch- und Höchstleistungsrechnen“
- 3 Universitäten + 1 assoziiertes Unternehmen, 6 Doktoranden, 3 Jahre

TECHNISCHE UNIVERSITÄT CHEMNITZ
Praktische Informatik, Gudula Rünger

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
Höchstleistungsrechnen, Gerhard Wellein

UNIVERSITÄT BAYREUTH
Parallele und verteilte Systeme,
Thomas Rauber und Matthias Korch
(Koordination)

MEGWARE
(assoziierter Partner)
Jürgen Gretzschel

Förderung, Partner, Ziel

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Selbstadaption für zeitschrittbasierte Simulationstechniken auf heterogenen HPC-Systemen

- Gefördert vom BMBF seit 1. 1. 2017
- Programm „IKT 2020 – Forschung für Innovationen“
- Bekanntmachung „Grundlagenorientierte Forschung für HPC-Software im Hoch- und Höchstleistungsrechnen“
- 3 Universitäten + 1 assoziiertes Unternehmen, 6 Doktoranden, 3 Jahre

☞ Simulationssoftware sollte sich soweit möglich selbst an sich verändernde heterogene HPC-Hardware und neue Eingaben (Anfangswerte, Modell) anpassen.

TECHNISCHE UNIVERSITÄT CHEMNITZ
Praktische Informatik, Gudula Rünger

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
Höchstleistungsrechnen, Gerhard Wellein

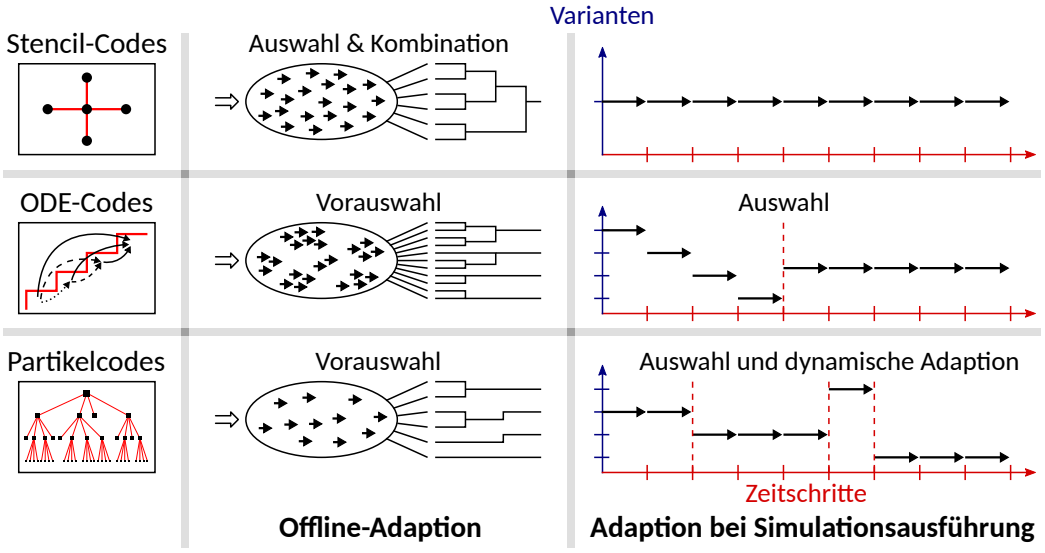
UNIVERSITÄT BAYREUTH
Parallele und verteilte Systeme, Thomas Rauber und Matthias Korch (Koordination)

MEGWARE
(assoziierter Partner) Jürgen Gretzschel

SeASiTe

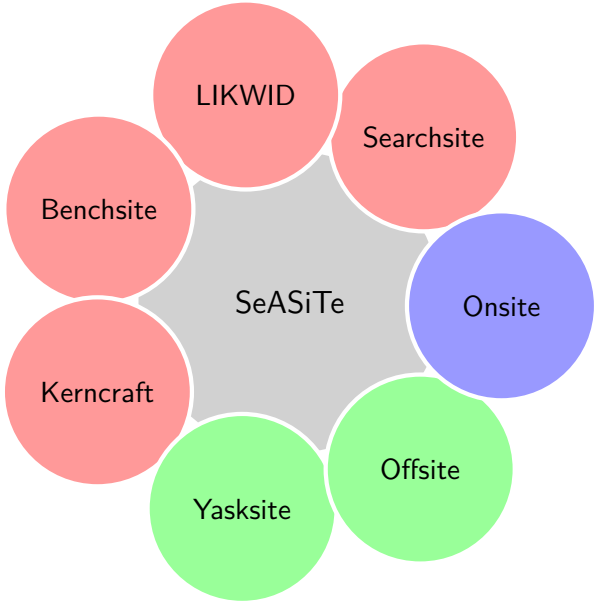
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Entwicklung und Integration selbstadaptierender Simulationsverfahren



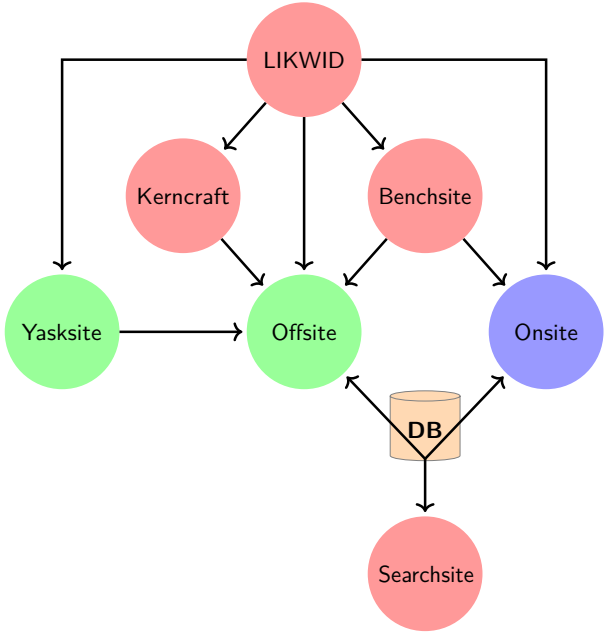
Ein Werkzeugkasten für zeitschrittbasierte Simulationen

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



Ein Werkzeugkasten für zeitschrittbasierte Simulationen

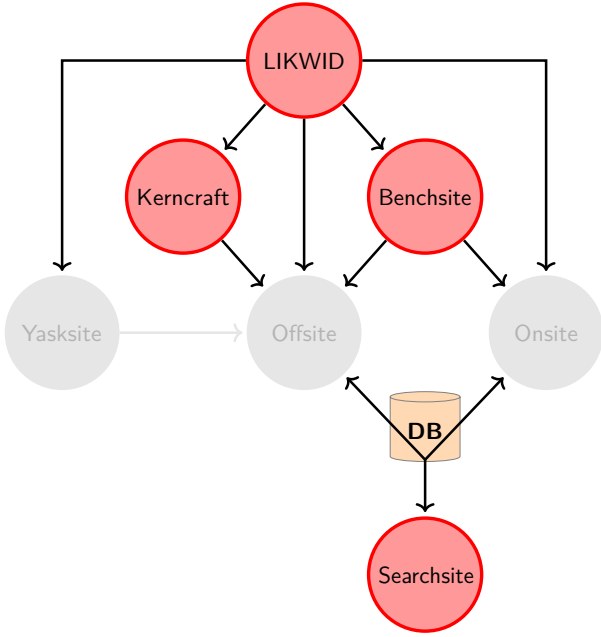
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



SeASiTe

Basiswerkzeuge

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



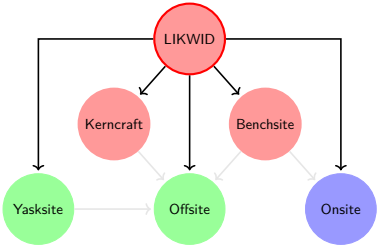
SeASiTe

LIKWID – Like I Knew What I Am Doing

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Werkzeugkasten und Bibliothek für performanceorientierte Programmierer:

- Module für Hardware Performance Monitoring von x86-CPUs.
- Kontrolle von CPU-/Prozessaffinität (OpenMP, MPI + X).
- Micro-Benchmarking mit handgeschriebenen Assembly-Kerneln.
- Manipulation der CPU/Uncore-Frequenzen oder des Prefetchers.



☞ Einfacher und *verlässlicher* Zugang zu Hardware Performance Monitoring.

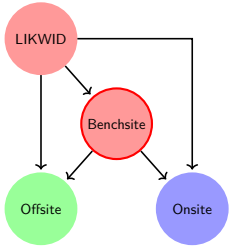
Unterstützung für ARMv8, POWER9 und Nvidia GPUs in Arbeit.
Nächste Version für Mitte November zur SC19 Konferenz geplant.

Benchsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Zusätzlich zu den in LIKWID enthaltenen Micro-Benchmarks (*likwid-bench*) bietet Benchsite eine Sammlung unabhängiger Benchmark-Tools, die in den SeASiTe-Workflow integriert wurden:

- im Rahmen von SeASiTe erstellte Benchmarks für OpenMP-Operationen,
- die Intel MPI Benchmarks,
- spezielle kombinierte Benchmarks für häufig auftretende Kommunikationsmuster.



☞ In Kombination mit ECM-Modell: Laufzeitvorhersagen für Programme auf Multicore- [ISC 2018] und Multicore-Cluster-Systemen [ICPE 2019].

Kerncraft

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

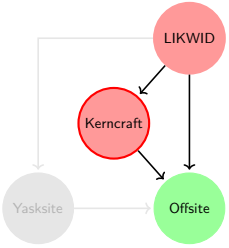
Loop Kernel Analysis and Performance Modeling Toolkit

KERNCRAFT

- Statische Codeanalyse zur Datenwiederverwendung und Cachingbedarf
- Automatisierte Erstellung des Roofline- und ECM-Modells (Execution-Cache-Memory)

Erweiterungen während des SeASiTe Projektes:

- Komplexe Zahlen (`complex_t` Datentyp)
- Erkennung von Pragmas (OpenMP, SIMD, ...)
- Multidimensionale Datentypen in einem 1D-Array



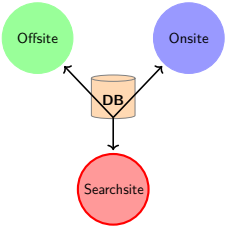
Das phänomenologische ECM-Modell nutzt LIKWIDs Hardware Counter zur Laufzeit.

Searchsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

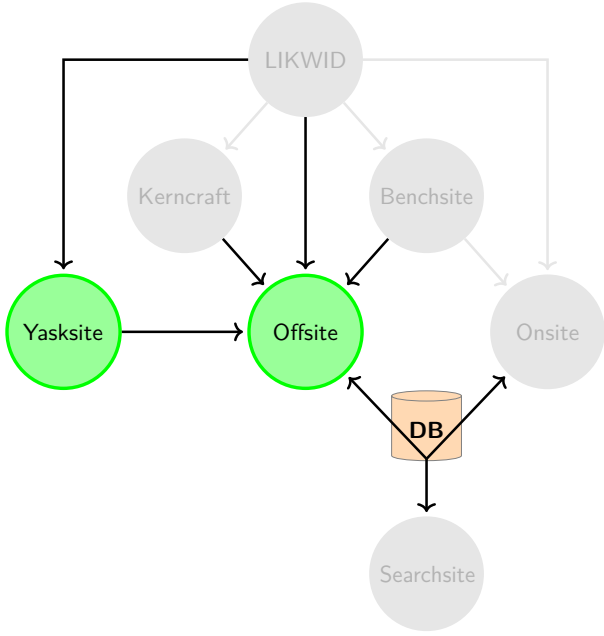
Server-Applikation zur Online-Parametersuche:

- Auf OpenTuner basierende Sammlung gängiger Autotuning-Suchalgorithmen (Nelder-Mead, Patternsearch, ...) inklusive Unterstützung von Meta-Techniken, multikriterieller Optimierung und der Möglichkeit zur Ergänzung um eigene Suchverfahren
- Standalone-Server-Applikation, die über eine austauschbare Schnittstelle mit Offsite und Onsite kommunizieren kann
- Clientseitige Konfiguration der Online-Suche
- Effiziente Durchmusterung auch großer Suchräume



Werkzeuge für Offline-Tuning

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

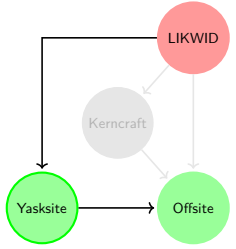


Yasksite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Stencil-Framework basierend auf Intels YASK, das einfache Erzeugung hoch-optimierter Stencil-Kernel erlaubt:

- Ermöglicht einfache Generierung und Ausführung vieler verschiedener Stencil-Anwendungen.
- Fügt eine Schicht zur modellbasierten Performance-Vorhersage basierend auf dem ECM-Modell hinzu. Die Erweiterung des Modells unterstützt einen großen Teil der Performance-Optimierungsmöglichkeiten von YASK.

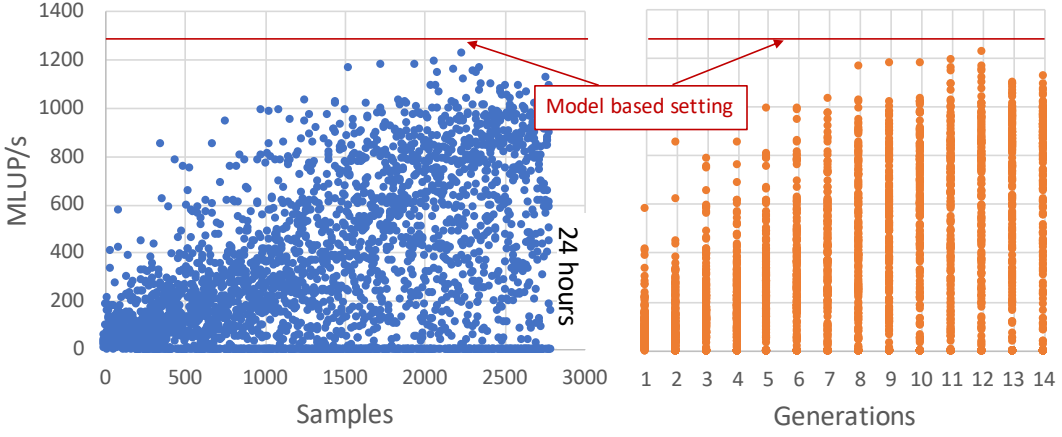


- Erweiterung des Maschinenbeschreibungsformats für Kerncraft unter Verwendung von LIKWID und anderer Benchmarks.
- Ermöglicht Offline-Tuning basierend auf Performance-Modell.
- Ermöglicht optimale Parameterauswahl für Optimierungsstrategien wie z. B. räumliches und zeitliches Blocking.

Yasksite – Beispiel

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

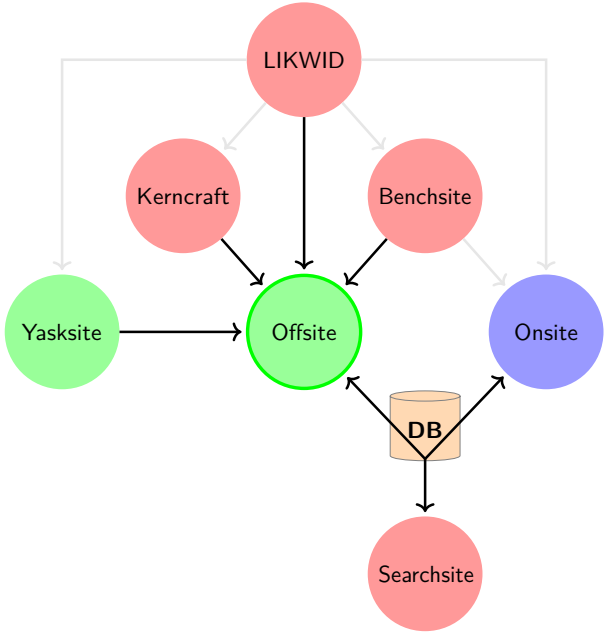
Parameter-Tuning für zeitliches Blocking des ISO3DFD-Stencils (Radius 4, doppelte Genauigkeit). Vergleich von YASKs genetischem Algorithmus mit modellbasiertem Ansatz. Das Experiments wurde auf einer einzelnen NUMA-Domäne (10 Threads) eines Cascade Lake 6248 (20 Kerne) mit aktiviertem Sub-NUMA-Clustering (SNC) durchgeführt.



SeASiTe

Offsite – Einbettung in den Werkzeugkasten

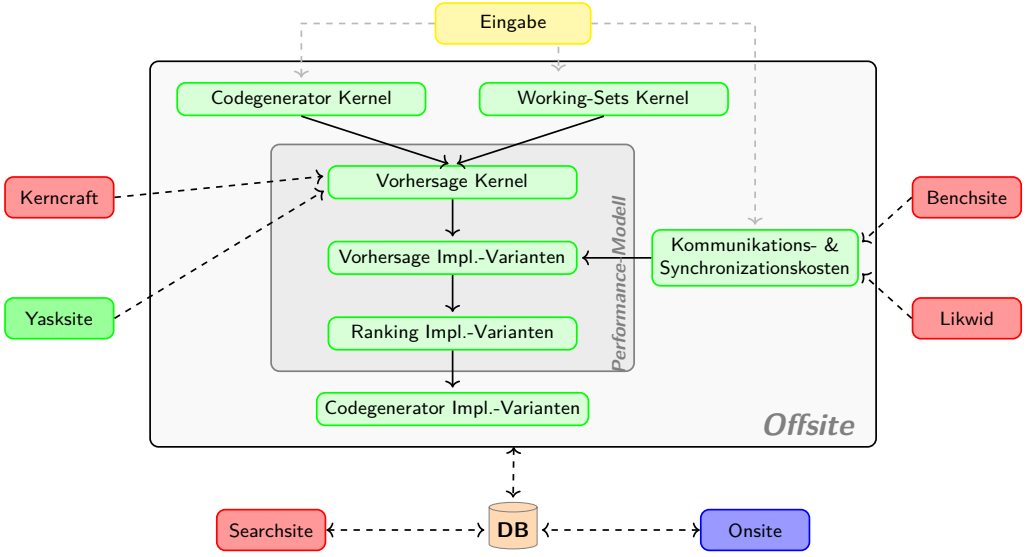
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



SeASiTe

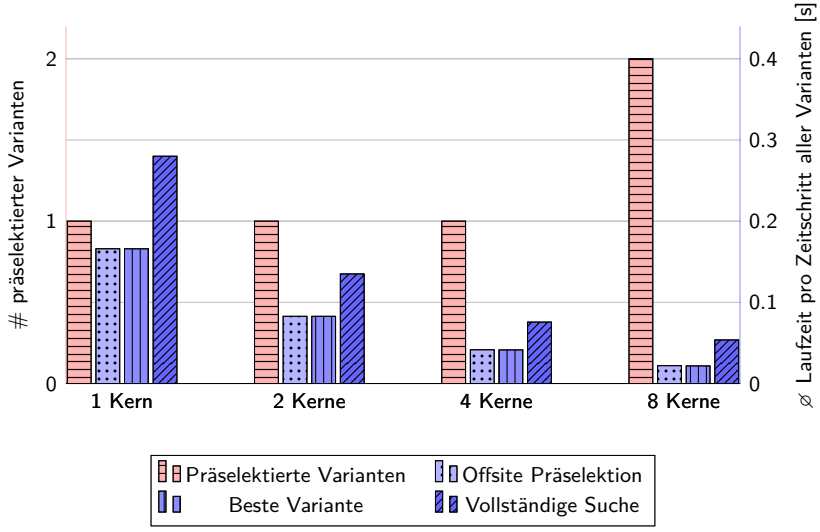
Offsite – Innere Struktur

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



Offsite – Tuning des Anfangswertproblems *InverterChain*

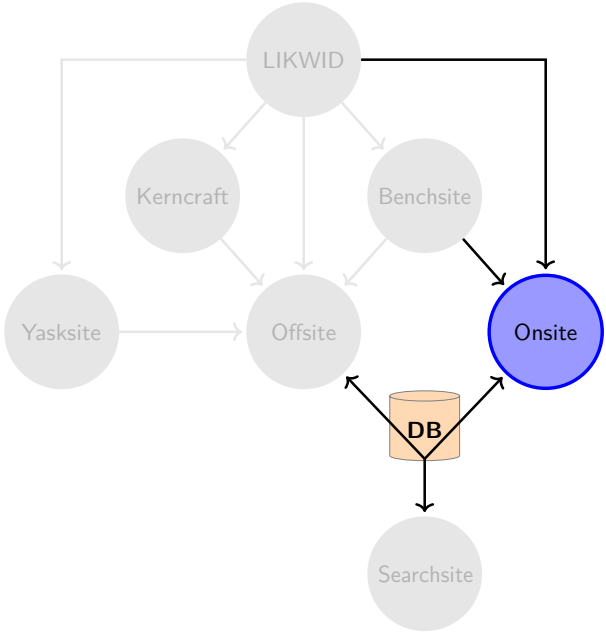
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



Overhead pro getesteter Variante beim Tuning für verschiedene Kernzahlen bei *Radau IIA(7)* und $n = 1.000.000$ auf einem Sockel eines Haswell-Systems.

Werkzeuge für Online-Tuning

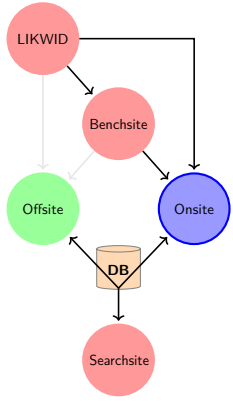
GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung



Onsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

- Nutzung der von Searchsite bereitgestellten Optimierungsalgorithmen.
- Rückkoppelung zur Offline-Phase.
- Erfassung von Konfigurationen und Meßwerten in Datenbank.
- Einfache Instrumentierung von C/C++-Code mittels Makros.
- Erfassung der Rechnertopologie mittels *hwloc*.
- Nutzerdefiniertes Profiling und Monitoring mittels LIKWID.
- Nutzerdefinierte Metriken.



Onsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

- Nutzung der von Searchsite bereitgestellten Optimierungsalgorithmen.
- Rückkoppelung zur Offline-Phase.
- Erfassung von Konfigurationen und Meßwerten in Datenbank.
- Einfache Instrumentierung von C/C++-Code mittels Makros.
- Erfassung der Rechnertopologie mittels *hwloc*.
- Nutzerdefiniertes Profiling und Monitoring mittels LIKWID.
- Nutzerdefinierte Metriken.

```
int blocksize = 128;




for(int i = 0; i < N; i++)
{

    work(blocksize);
}
```

Onsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

- Nutzung der von Searchsite bereitgestellten Optimierungsalgorithmen.
- Rückkoppelung zur Offline-Phase.
- Erfassung von Konfigurationen und Meßwerten in Datenbank.
- Einfache Instrumentierung von C/C++-Code mittels Makros.
- Erfassung der Rechnertopologie mittels *hwloc*.
- Nutzerdefiniertes Profiling und Monitoring mittels LIKWID.
- Nutzerdefinierte Metriken.

```
int blocksize = 128;

REGISTER_LOOP("LOOP");









LA_FOR("LOOP", int i = 0, i < N, i++)
{


    work(blocksize);
}
```

Onsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

- Nutzung der von Searchsite bereitgestellten Optimierungsalgorithmen.
- Rückkoppelung zur Offline-Phase.
- Erfassung von Konfigurationen und Meßwerten in Datenbank.
- Einfache Instrumentierung von C/C++-Code mittels Makros.
- Erfassung der Rechnertopologie mittels *hwloc*.
- Nutzerdefiniertes Profiling und Monitoring mittels LIKWID.
- Nutzerdefinierte Metriken.

```
int blocksize = 128;

REGISTER_LOOP("LOOP");
REGISTER_POLICY("LOOP", "ENERGY");
REGISTER_OBJECTIVE("LOOP",
                   "MINIMIZE_ENERGY_EDP");
REGISTER_PARAMETER("LOOP",
            LOOP_ADAPT_CPU_FREQUENCY);




LA_FOR("LOOP", int i = 0, i < N, i++)
{


    work(blocksize);
}
```

Onsite

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

- Nutzung der von Searchsite bereitgestellten Optimierungsalgorithmen.
- Rückkoppelung zur Offline-Phase.
- Erfassung von Konfigurationen und Meßwerten in Datenbank.
- Einfache Instrumentierung von C/C++-Code mittels Makros.
- Erfassung der Rechnertopologie mittels *hwloc*.
- Nutzerdefiniertes Profiling und Monitoring mittels LIKWID.
- Nutzerdefinierte Metriken.

```
int blocksize;

REGISTER_LOOP("LOOP");
REGISTER_POLICY("LOOP", "ENERGY");
REGISTER_OBJECTIVE("LOOP",
                   "MINIMIZE_ENERGY_EDP");
REGISTER_PARAMETER("LOOP",
            LOOP_ADAPT_CPU_FREQUENCY);
REGISTER_PARAMETER("LOOP","blocksize",
            LOOP_ADAPT_SCOPE_MACHINE,
            NODEPARAMETER_INT, 0, 4096);

LA_FOR("LOOP", int i = 0, i < N, i++)
{
    blocksize =
      GET_PARAMETER("LOOP","blocksize");
    work(blocksize);
}
```

Demonstratoren

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

1. Stencil-Berechnungen
   - regulär, gut verstanden, mit ECM-Modell vorhersagbar
   - ☞ Offline-Tuning mit Yasksite
2. Partikelsimulationen
   - aufgrund Partikelbewegung irregulär, nicht vorhersagbar
   - ☞ Online-Tuning mit Onsite
3. ODE-Verfahren
   - ODE-System kann sehr unterschiedliche Eigenschaften haben (regulär/irregulär, dicht-/dünnbesetzt, ...)
   - unterschiedliche Anwendungsszenarien
   - ☞ Offline-Tuning mit Offsite, Online-Tuning mit Onsite, Kombination möglich

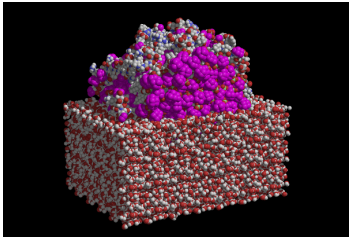
Demonstratoren

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

1. Stencil-Berechnungen
   - regulär, gut verstanden, mit ECM-Modell vorhersagbar
   - ☞ Offline-Tuning mit Yasksite
2. **Partikelsimulationen**
   - **aufgrund Partikelbewegung irregulär, nicht vorhersagbar**
   - ☞ **Online-Tuning mit Onsite**
3. ODE-Verfahren
   - ODE-System kann sehr unterschiedliche Eigenschaften haben (regulär/irregulär, dicht-/dünnbesetzt, ...)
   - unterschiedliche Anwendungsszenarien
   - ☞ Offline-Tuning mit Offsite, Online-Tuning mit Onsite, Kombination möglich

Demonstratoren – Fallbeispiel Partikelsimulation

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

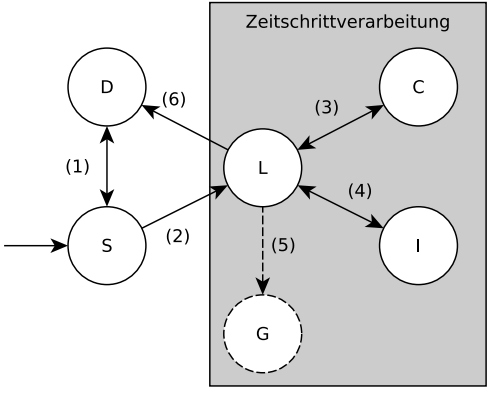


- Die meiste Rechenzeit einer Partikelsimulation benötigt die Berechnung von langreichweitigen Partikelwechselwirkungen.
- Betrachtet werden verschiedene baum- und gitterbasierte Partikelsimulationsmethoden.
- Aufgrund von Partikelbewegungen kann sich die Partikelverteilung über mehrere Zeitschritte verändern.
- Das Tuning der Berechnungsmethoden wird durch ein Online-Tuning mit Performance-Monitoring unterstützt.

[ICPE 2018], [CANA 2019]

Partikelsimulation – Komponentenbasierte Implementierung

- Hauptkomponenten des Demonstrators für Partikelsimulationen:
  D - Daten, **S** - Simulationssetup, **G** - graphische Ausgabe, **L** - Simulationsschleife,
  C - Wechselwirkungsberechnung, **I** - Integration und Positionsaktualisierung



(1) Daten anfordern und bereitstellen

(2) Ausführungsparameter und Daten übermitteln

(3) Partikelwechselwirkungen bestimmen

(4) Integrationsschritt und Aktualisierung von Partikelpositionen

(5) (Optional) Graphische Ausgabe von Partikeldaten und Performance-Metriken

(6) Speichern von aktuellen Partikeldaten

- Kontrollfluss der Berechnung eines Simulationszeitschrittes: L - C - L - I - L
- Komponenten C und I können einzeln optimiert werden.

Partikelsimulation – Verwendete Werkzeuge

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

- Verteilung von Simulationskomponenten
  - Auslagerung von rechenintesiven Komponenten, insbesondere C und I [ISPDC 2018]
  - Verwendung der **Kommunikationsbibliothek SCDC** ermöglicht Kommunikation zwischen Simulationskomponenten über TCP, UDS und innerhalb des gleichen Prozesses
  - Nutzung mehrerer HPC-Rechenknoten für MPI-basierte Komponenten möglich
- Integrierung von Werkzeugen
  - **LIKWID** als Monitoringwerkzeug für relevante Simulationskomponenten
  - **Searchsite + Onsite** als Iterationstreiber der Simulationsschleife L, u.a. zum Tunen von Parametern und Selektieren von Berechnungsmethoden
- Integrierung existierender Softwarebibliotheken
  - **scikit-learn** für Partikelsystemanalysen und Ranking von Berechnungsmethoden
  - **ScaFaCoS** für MPI basierte Methoden zur Berechnung von Partikelwechselwirkungen (C)

Zusammenfassung und Ausblick

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Status:

- Konzeption eines Werkzeugkastens bestehend aus:
  - Basiswerkzeugen für Performancemonitoring, Benchmarking, Erstellung von Performancemodellen und Bereitstellung von Suchstrategien
  - Werkzeugen für Offline-Tuning (mit Codegenerierung und modellbasierter (Vor-)Auswahl von Konfigurationen)
  - Werkzeugen für Online-Tuning (mit einfach in bestehende Anwendung integrierbarem API und empirischer Suche und Monitoring als Bestandteil der Simulation)
- Arbeiten zur Implementierung und Integration der Werkzeuge laufen.
- Vorabeiten zu Demonstratoren (u. a. komponentenbasierte heterogene Implementierung von Partikelsimulationsverfahren) laufen.

Ausblick:

- Weiterentwicklung und Implementierung der Selbstadaptionskonzepte und -werkzeuge
- Schwerpunkt: Integration der Werkzeuge
- Umsetzung der Demonstratoren