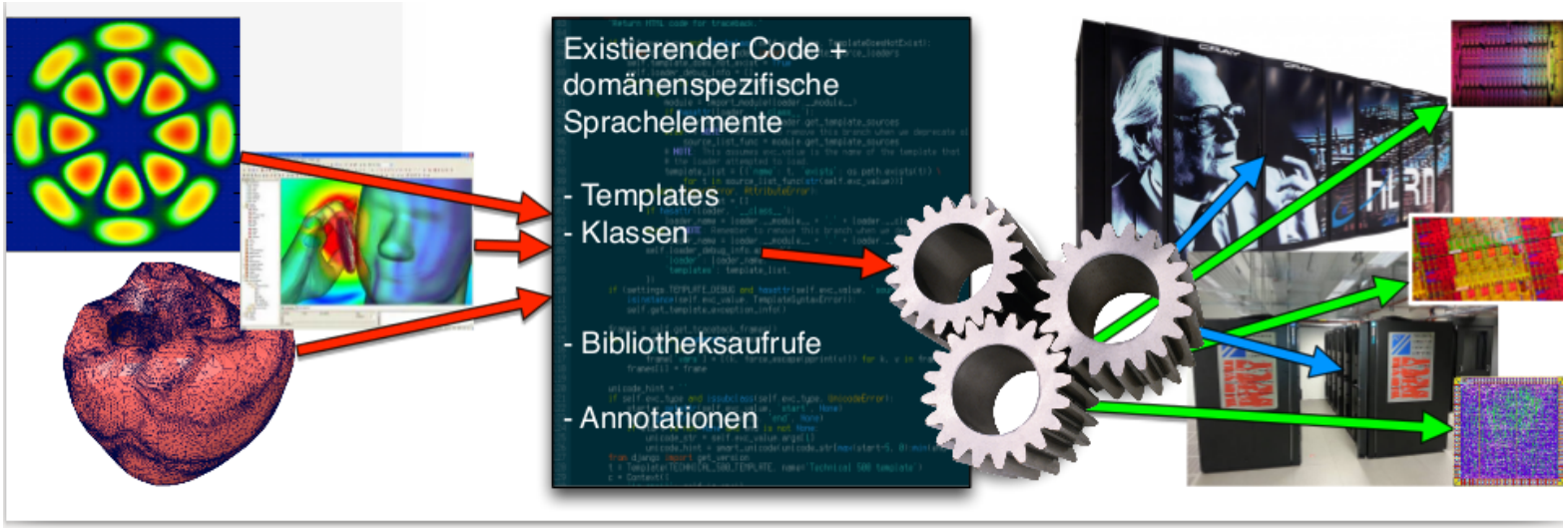
HighPerMeshes

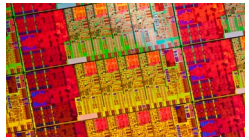


Ein Framework zur verteilten Task basierten Ausführung von iterativen Verfahren auf unstrukturierten Gittern

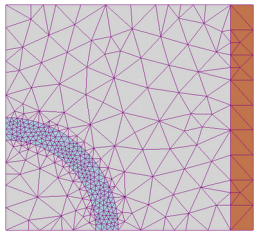
Dr. Daniel Grünewald

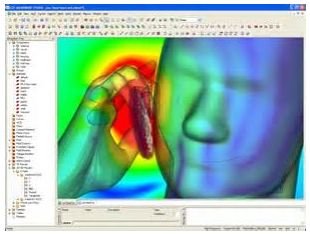
Fraunhofer
ITWM

Motivation



Komplexe, heterogene Hardware

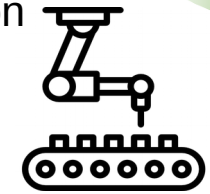




Simulationen auf unstrukturierten Gittern

Skalierbarkeit ist der Schlüssel

Skalierbarkeit
Aggregation von CPUs
Aggregation von Speicher

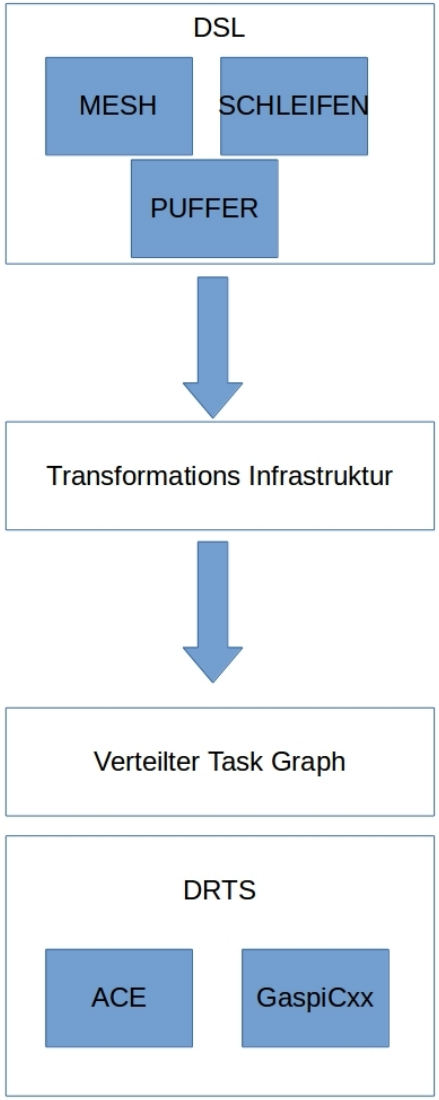


Effizienz
Vollständige Resourcen Nutzung
Niedriger Energieverbrauch

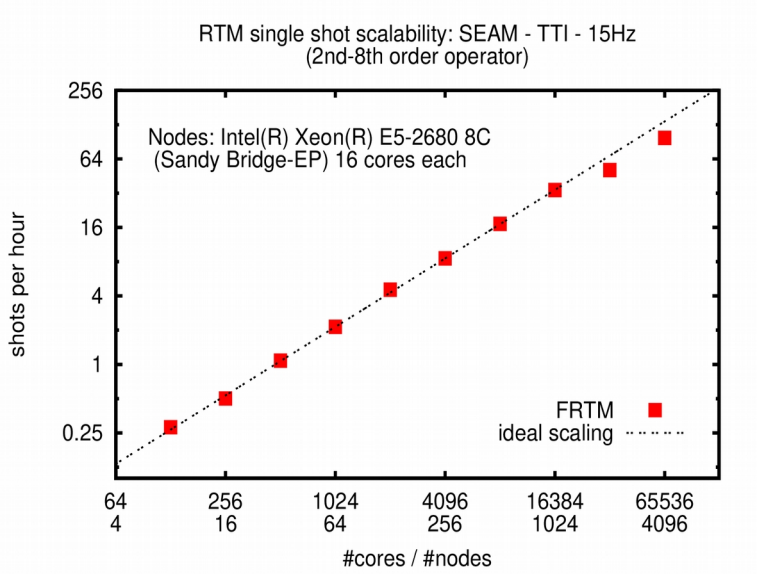
Produktivität
Komplexe Modelle
Schnelle Lösungzeit
Hoher Durchsatz

Fraunhofer ITWM

HighPerMeshes-Ansatz



- Ziel: Ein in der Praxis einsetzbares domänenspezifisches Framework zur effizienten, parallelen und skalierenden Implementierung iterativer Algorithmen auf unstrukturierten Gittern
- State of the art Komponenten: GASPI / GPI-2



- Einfache Nutzbarkeit (Domänen Experte)
- Volle Performanz

Fraunhofer ITWM

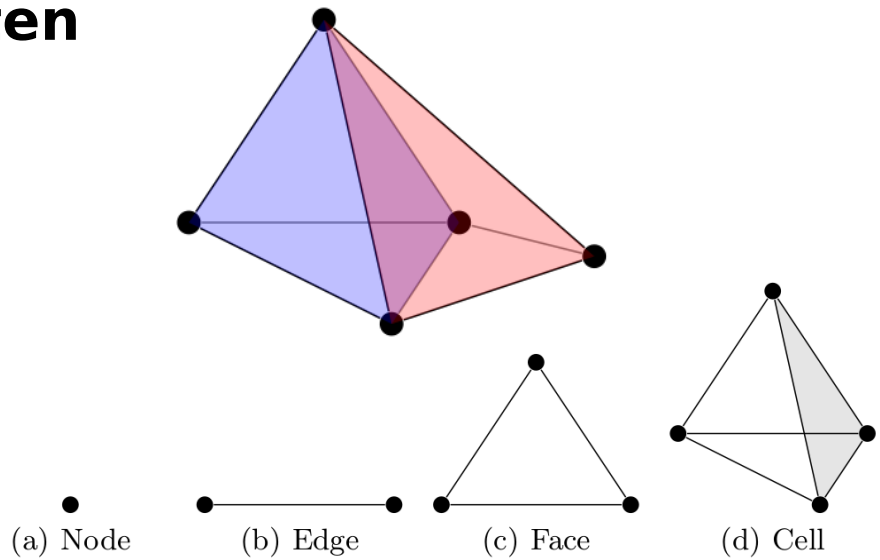
Domänenspezifische Sprache

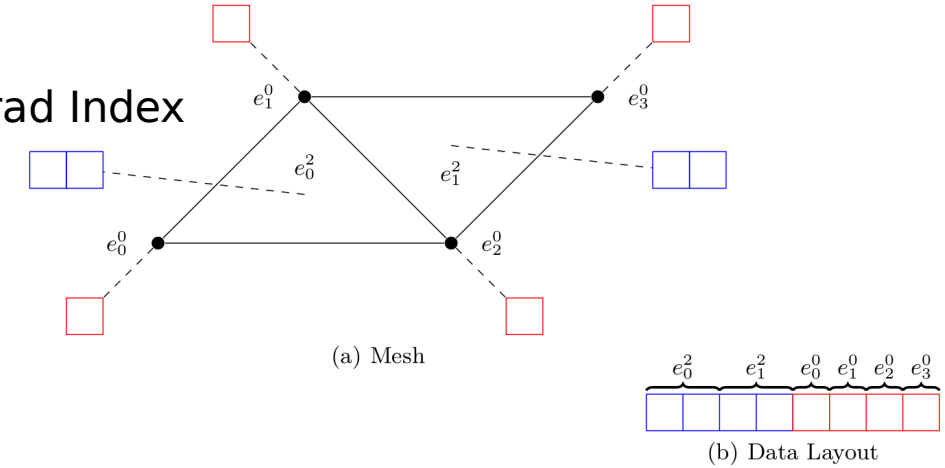
- Gitter Definition und Gitter-Daten
- Daten-Zugriffs Definition
- Gitter- und Iterations-Schleifen

Gitter-Definition und Gitter-Daten

- Abstraktes Gitter-Interface
  - Entitäten (Element,Fläche,Kante,Knoten)
  - Topologie (Nachbarschaft, Sub-Entitäten)
  - Geometrie

    → Simplex Implementierung
- DoF (Freiheitsgrade) Interface
  - Mapping Gitter Entität → Freiheitsgrad Index
- Daten-Puffer
  - Speichert lokale Frei-heitsgrade, z.B. elektrische Feld
  - Speichert Ghost Freiheitsgrade
  - Basis Typ als template Parameter



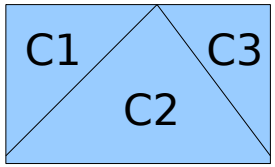


Fraunhofer ITWM

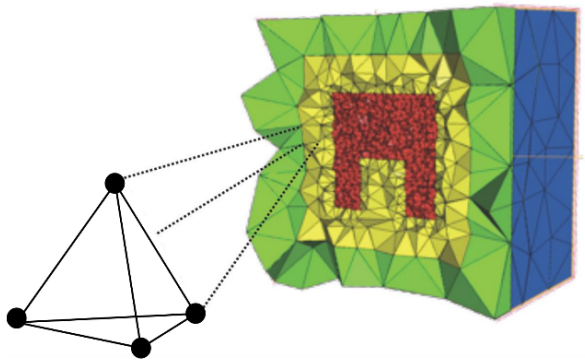
Daten-Zugriffs-Definition

- Zugriffsart auf Gitter-Daten
  - Lesen, Schreiben, Lesen und Schreiben, Akkumulieren (+=)
- Zugriffsschema
  - Mapping: Mesh Entität → Menge von Mesh Entitäten

    z.B. Nachbarzelle



C2 → {C1,C3}



→ Zugriffsdefinition, e.g. `Write(Cell(fieldE))`:

- Zugriffsschema
- Zugriffsart
- Daten-Puffer

→ Lokale Pufferansicht innerhalb der Schleife
→ Zugriff über lokale Indizes
→ Potentiell Device Spezifisches Speicher-Layout

Gitter- und Iterations-Schleifen

- Gitter-Schleifen:

```
HPM::ForEachEntity
  ( AllCells
  , std::tuple(Write(Cell(fieldE)))
  , [&](const auto &cell, auto && iterator, auto localBufferView)
      {
        /// do something useful in here
      }
  );
```

  - FOREACH Entity OF Mesh DO Schleifenrumpf + Annotation
    - Entity Range Spezifikation (Alle Zellen, Rand-Zellen, etc.)
    - Zugriffs-Definition → lokale Puffer Ansicht view
    - Schleifenrumpf: Lambda Funktion mit Entität, Iterator, lokale Puffer Ansicht
- Iterations- Schleifen
  - FOREACH Iteration OF Range DO Schleifenrumpf
  - Schleifenrumpf: Tuple aus Mesh-Schleifen

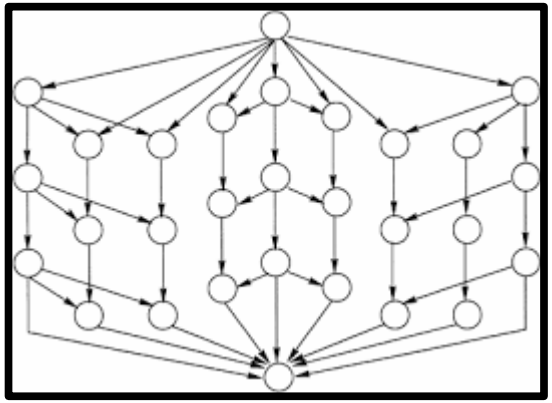
Fraunhofer ITWM

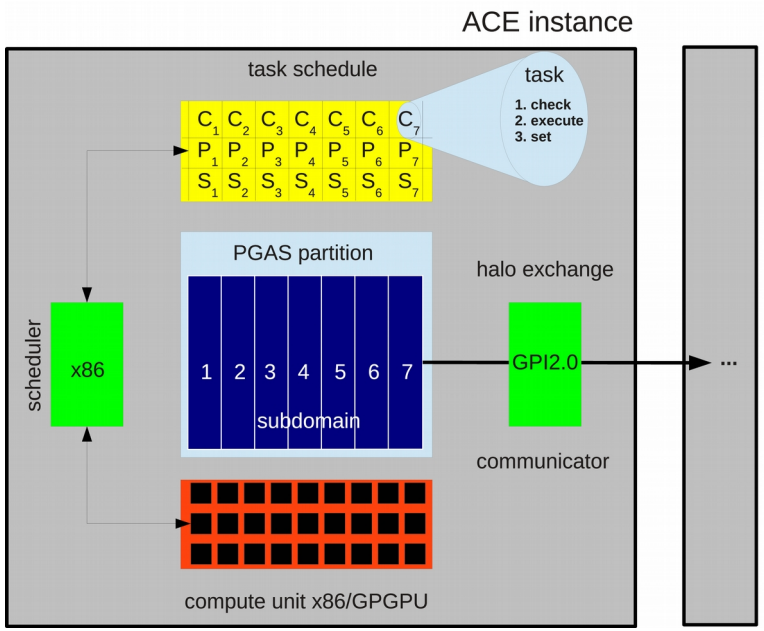
Verteiltes Laufzeitsystem

- ACE
- GASPI

ACE

- **A**synchronous **C**onstraint **E**xecution
  - Task basiertes Laufzeitsystem
  - Statischer Task Graph über alle Iterationen
    - Knoten: executables
    - Kanten: pre-conditions
    - Iterator = interner Zustand
      → Nur eine Iteration gespeichert
      → optimal für iterative Verfahren
  - Einfaches und schnelles Task scheduling
    - **pre-condition(s)** überprüfen
    - **executable(s)** ausführen
    - **post-condition(s)** setzen (++state)

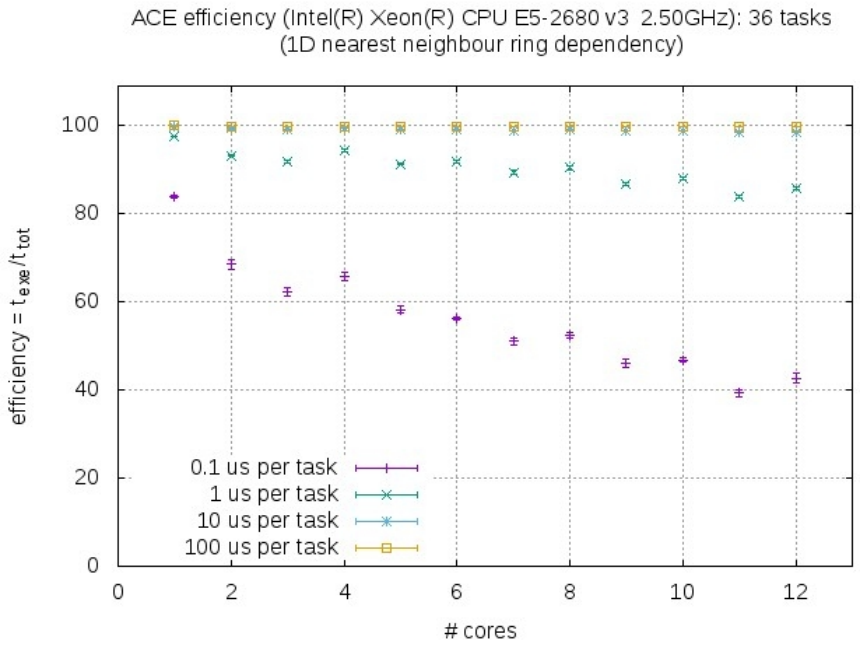


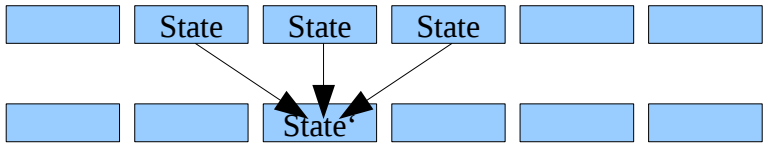


Fraunhofer ITWM

ACE – Effizienz

(Overhead = 1 - Efficiency)





Benchmark set up:

- 36 Tasks
- Abhängigkeit zu nächsten Nachbarn
- Periodische Randbedingungen
- Fixe Ausführungszeit pro Task

  0.1 us, 1 us, 10 us, 100 us
- Intel Xeon E5-2680 v3 2.5GHz (12 cores)

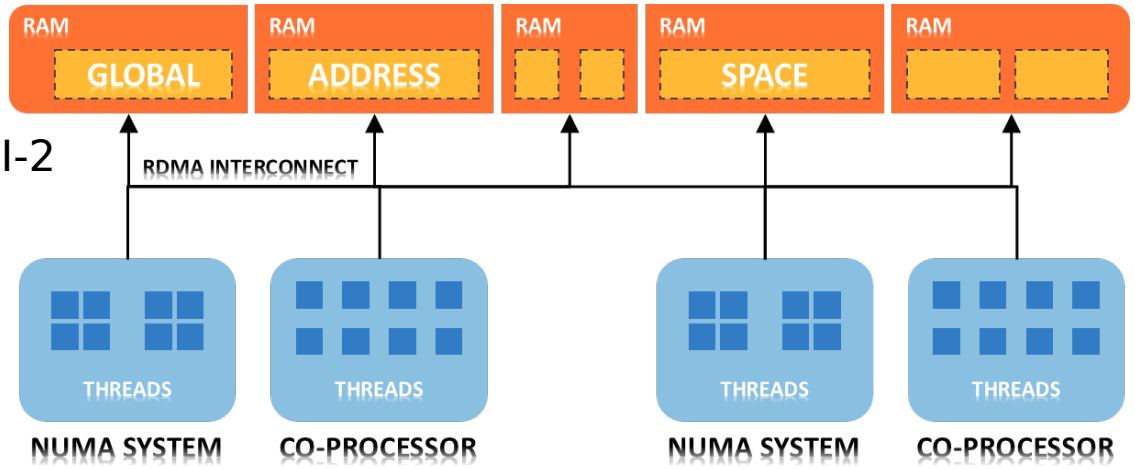
  → Ergebnisse:
  - 10 us pro Task: > 98.6% Effizienz
  - 1 us pro Task: immernoch 85% @ 12 cores

Fraunhofer ITWM

GaspiCxx

- C++ Schnittstelle für Gaspi / GPI-2
  - Einseitige echt asynchrone Kommunikation
  - Separate leichtgewichtige Synchronisierung
  - Thread Safe

    → Optimal für Task basierte Ausführungsmodelle
- Resourcen Management (RAII)
  - Segmente
  - Queues
  - Gruppen
- Kollektive Operationen



GASPI

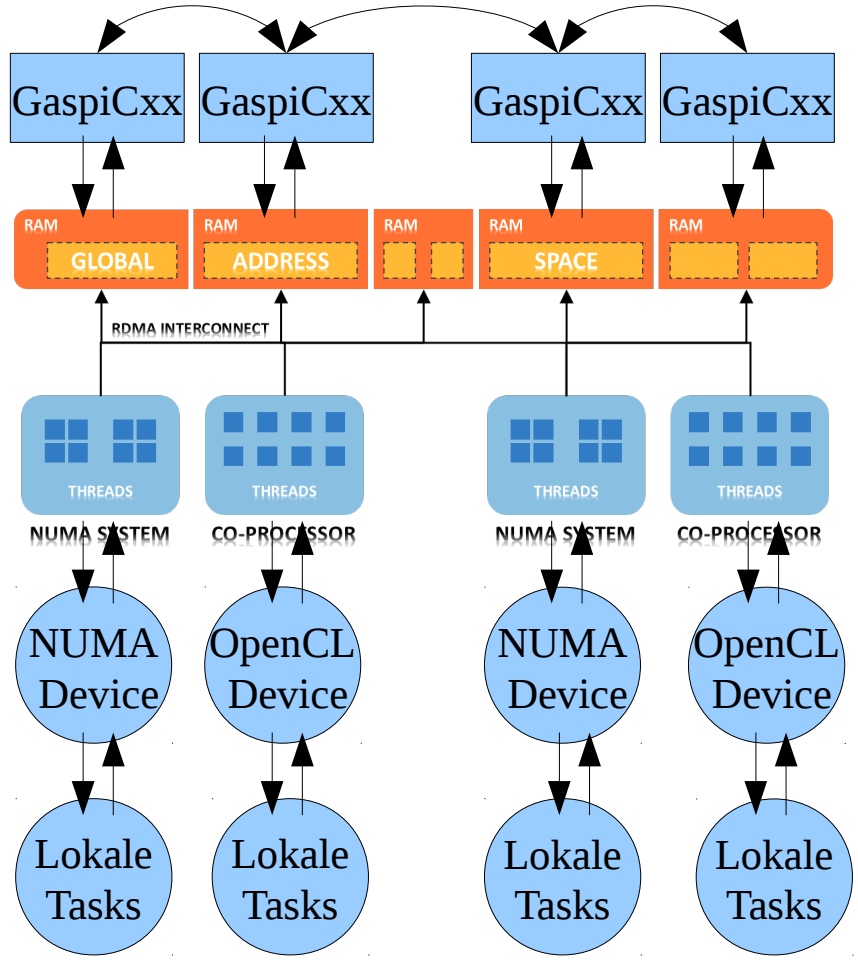
Einfache Programmierbarkeit
Volle Performanz

Fraunhofer ITWM

DRTS

- ACE: Device Erweiterung
  - Device Task Scheduling
  - Device Speicher Allokation (Datenpuffer)
  - X86 und OpenCL backend (GPU,FPGA)
- Topologiebeschreibung
  - Device Typ, Device Id
  - Ein Eintrag pro Device

    → Laufzeitsystem instantiiert konkrete Device Implementierung anhand der Topologiebeschreibung und dem Gaspi Rank



Fraunhofer ITWM

Transformations Infrastruktur

- Problem Partitionierung
- Extraktion der Datenabhängigkeiten
- Heterogenität: Source to Source Kompiler
- Aufbau des Task Graphen

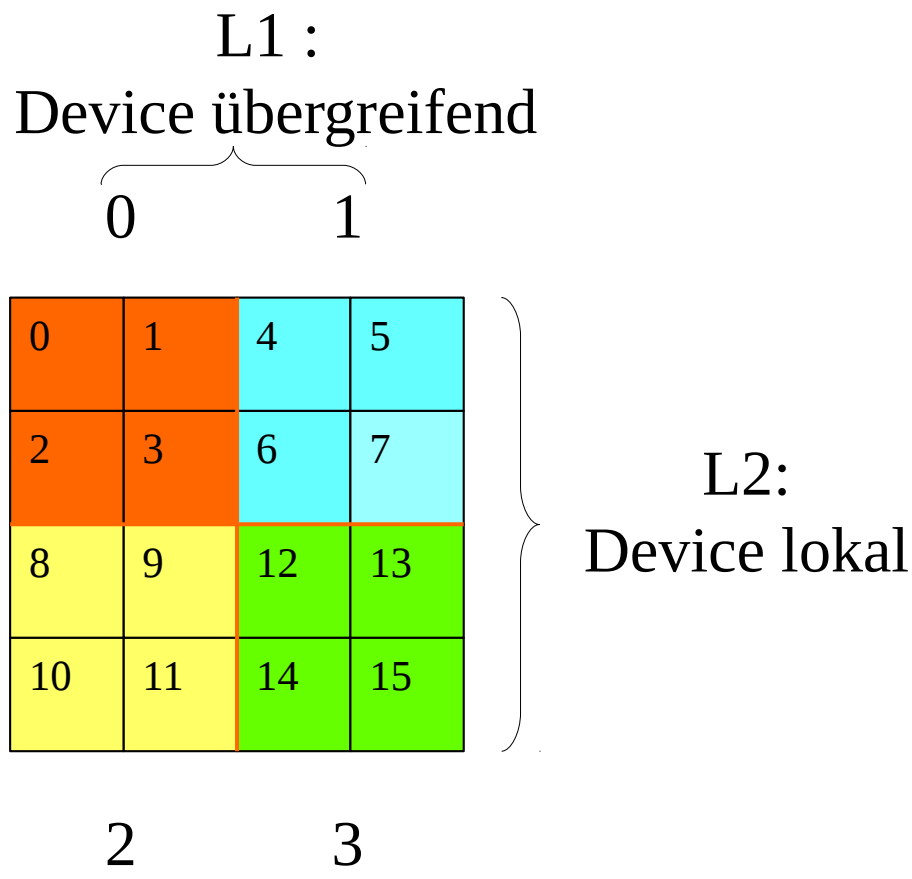
Mesh Partitionierung

- Partitionierung auf zwei Ebenen
  - Ebene 1: Device übergreifend
  - Ebene 2: Device lokal

  → Jede Mesh Entität is genau einer Partition zugeordnet
- Abstraktes Partitionierer Interface
  - METIS Implementierung

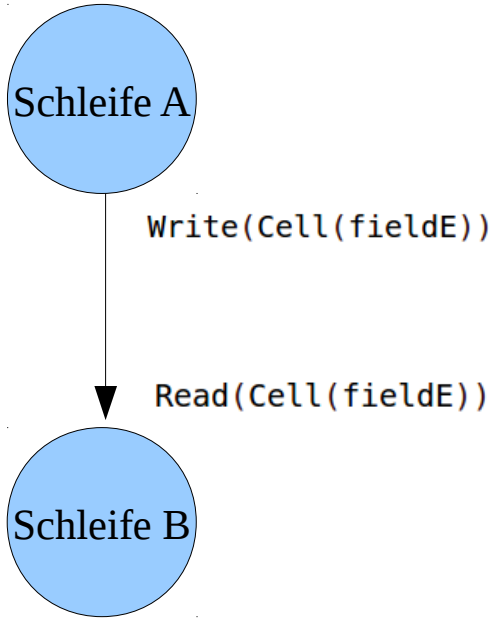
  → Mapping Entität → E2 Partition

  → Mapping E2 Partition → E1Partition
- Idee: Mesh Partitionierung: Zielgewichte werden in Einklang mit den Device Fähigkeiten gesetzt



Abhängigkeits Graph

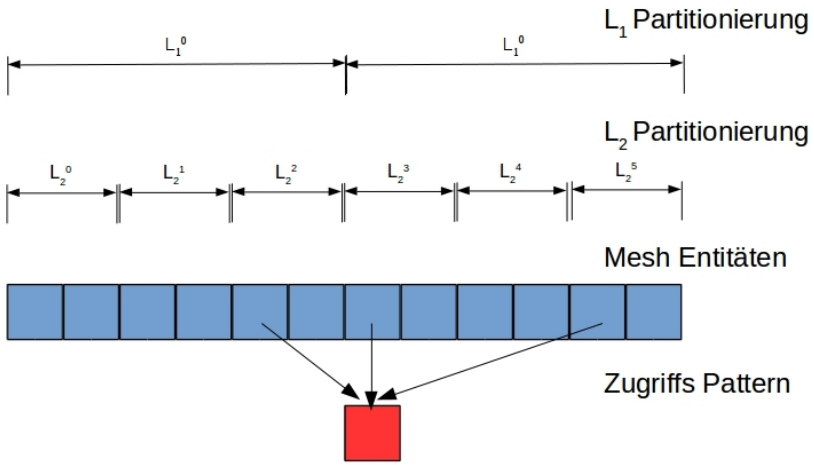
```
HPM::ForEachEntity
  ( AllCells
  , std::tuple(Write(Cell(fieldE)))
  , [&](const auto &cell, auto && iterator, auto localBufferView)
      {
        /// do something useful in here
      }
  );

HPM::ForEachEntity
  ( AllCells
  , std::tuple(Read(Cell(fieldE)))
  , [&](const auto &cell, auto && iterator, auto localBufferView)
      {
        /// do something useful in here
      }
  );
```



- Graph (Darstellung Datenabhängigkeiten zwischen Mesh loops)
  - Mesh loops als Knoten
  - Kanten: Zugriffsdefinition Feld Lesen / Schreiben
  - Generiert in iterativem Loop

Fraunhofer ITWM

Extraktion der Datenabhängigkeiten



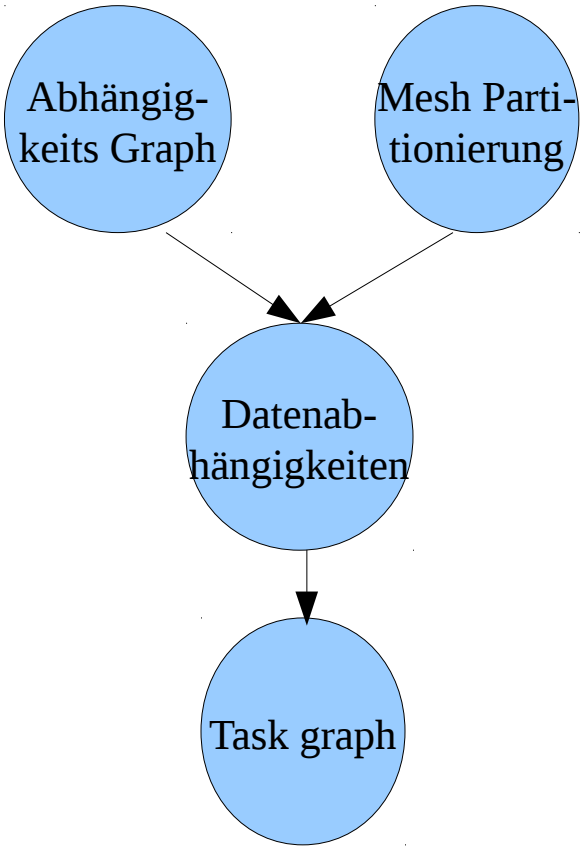
- Zugriffsschema bei gegebener Mesh Partitionierung erlaubt Extraktion von
  - Abhängigkeiten zwischen Partitionen
  - Abhängigkeiten zu nicht-lokalen Entitäten

    → Identifikation von Rand - / Halo DoFs

Task Erzeugung

- Task generiert für
  - Jede Schleife im Abhängigkeitsgraphen
  - Jede lokale Ebene 2 Partition

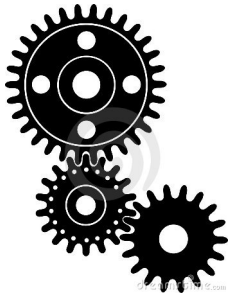
    → Äußeres Produkt aus Schleifen und Partitionen
- Pre- / Post-Conditions
  - Modelieren Datenabhängigkeiten zwischen Tasks
  - + Kommunikation bei nicht-lokalen Abhängigkeiten
- Executable:
  - Erzeugung: Mesh loop beschränkt auf lokale Elemente der gegebenen Partition



Source to source compiler

- Lambda Funktionen (Schleifenkörper) normaler C++ Quelltext
- Problem OpenCL Quelltext zur Ausführung auf OpenCL Devices benötigt

  → Source to source compiler

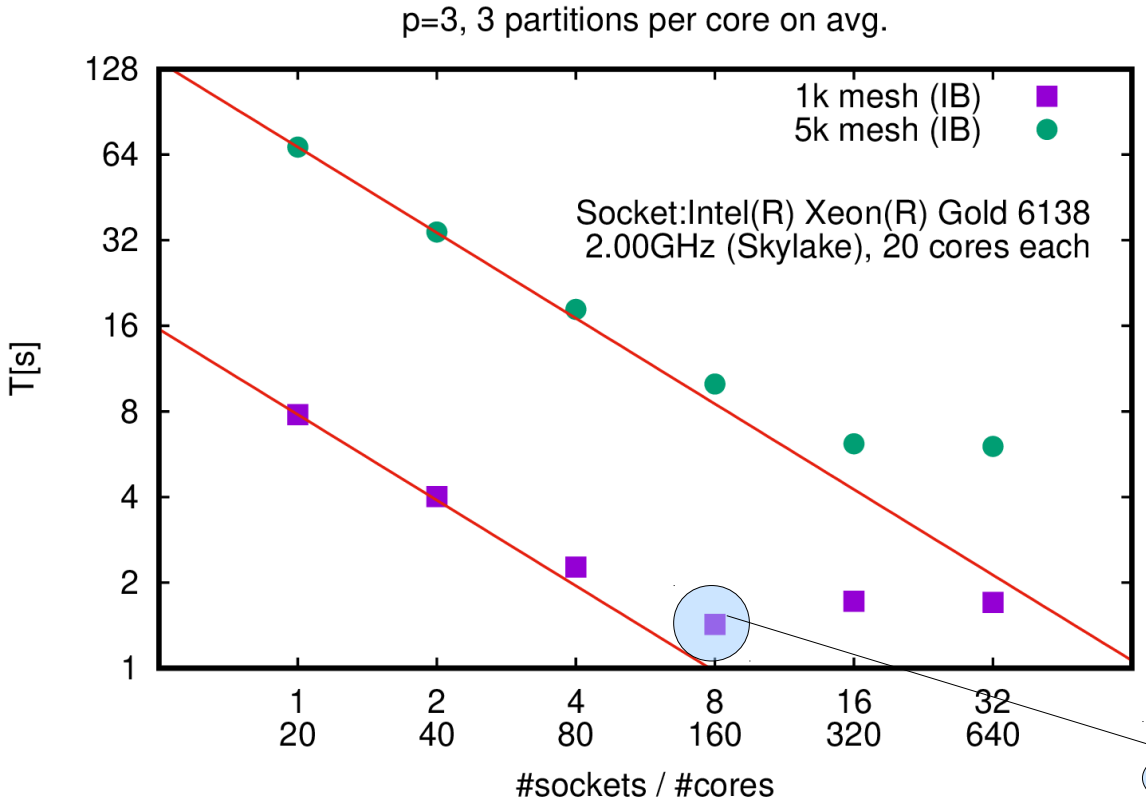


- Quelltext Transformation mittels clang
  - Identifikation der Lambda Funktionen: **clang::ast_matchers**
  - Umschreiben der Lambda Funktionen: **clang::Rewriter**
  - Code Optimierung

Fraunhofer ITWM

Ergebnisse

Speedup midg2 (DG Maxwell Gleichung)



Zusammenfassung

- HighPerMeshes
  - DSL für Matrix freie iterative Verfahren auf unstrukturierten Gittern
  - Automatische Generierung eines verteilten Task Graphen inklusive Kommunikationsprimitiven
  - Effiziente Task basierte Ausführung
    - GaspiCxx + ACE Laufzeitsystem

  → Demonstrierter Nutzen für

Vielen Dank

Fragen ?

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Fraunhofer
ITWM