Designing a ROCm-Aware MPI Library for AMD GPUs: Early Experiences

Seminar Supercomputer: Forschung und Innovation

Niclas Schroeter

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2022-01-18

Gliederung (Agenda)

1 Motivation

2 Grundlagen

3 Design

4 Evaluation

5 Zusammenfassung

6 Literatur

Motivation

- Ende von Moore's Law ist in Sicht
- Leistungssteigerung soll trotzdem nicht enden
- Bisheriger Ansatz: Mehr Kerne pro Prozessor
- Lösung ist für den HPC-Bereich nicht optimal

Motivation

- Einsatz von Spezialhardware
  $\Rightarrow$ Beschleuniger
- Meist GPUs
- Besser im parallelen Rechnen als herkömmliche CPUs

CPU vs GPU

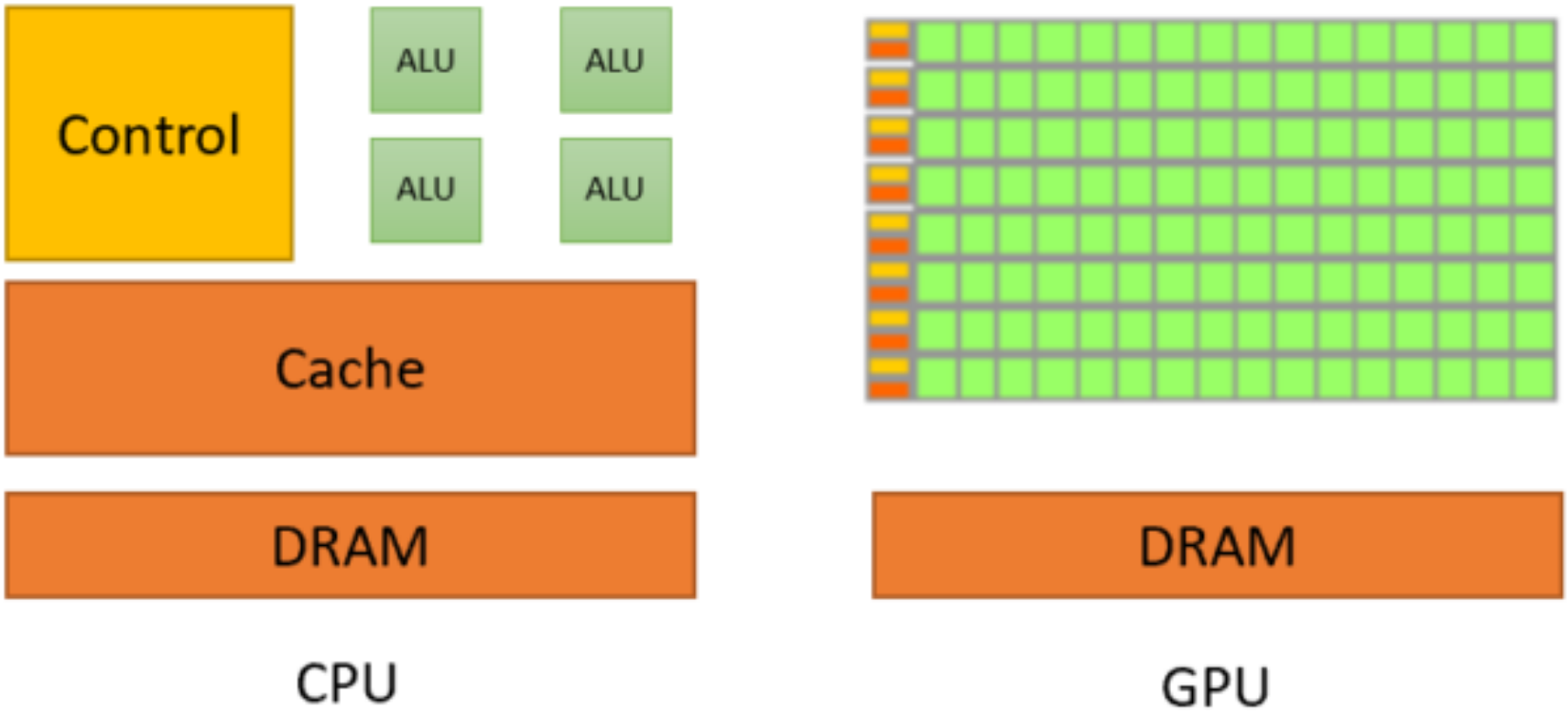


Abbildung: Schematischer Vergleich von CPUs und GPUs[Gup20]

Nvidia und CUDA

- Die meisten GPUs in HPC-Systemen stammen von Nvidia
  $\Rightarrow$ Anwendungen darauf ausgerichtet
- Nvidia bietet API zur Verwendung von GPUs in herkömmlichen Programmen (GPGPU) namens CUDA
- Nvidia GPU + CUDA als defacto Standard für die Nutzung von GPUs im HPC-Bereich

Kommende Exascale-Systeme

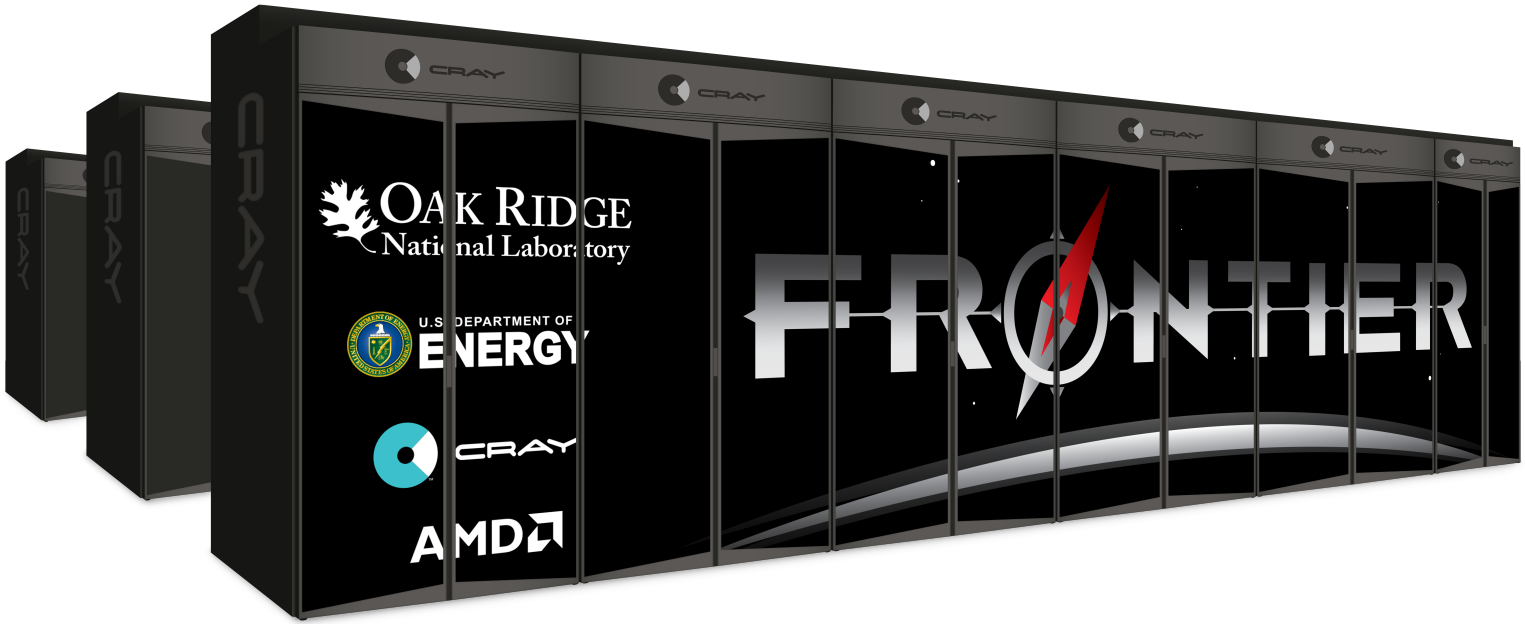


Abbildung: Pressrelease zu Frontier[McC19]

Problem

- Software auf Nvidias GPUs und CUDA zugeschnitten
- CUDA läuft nicht auf AMD GPUs
- Software muss verändert bzw. erweitert werden, um (gut) auf der neuen Hardware zu laufen

Ziel der Autoren

- Design für Kommunikationsmiddleware
  $\Rightarrow$ neue MPI Runtime
- Unterstützung für CUDA und ROCm
- Effiziente Kommunikation durch Nutzung der diversen Features der AMD GPUs und ROCm

MPI

- Message Passing Interface
- Programmiermodell für parallele Programme
- Basiert auf Nachrichtenaustausch
- Kommunikation kann in vielen Formen stattfinden
  $\Rightarrow$ Point-to-Point, kollektiv etc.

MPI + GPUs

- MPI ursprünglich nur zur Kommunikation zwischen Prozessen auf CPUs
- Heutzutage erweitert auf Nachrichtenaustausch zwischen GPUs
  $\Rightarrow$ GPU-aware MPI
- Durch Verbreitung von CUDA meist eher CUDA-aware MPI

MPI Beispiel

```
1 MPI_Init(NULL, NULL);
2 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
3 //Berechne irgendwas...
4 //Tausche Zwischenergebnisse aus:
5 MPI_Isend(buffer, [...], otherRank, [...]);
6 MPI_Recv(receiveBuffer, [...], otherRank, [...]);
7 //Rechne weiter...
```

Listing 1: (Pseudo-)Codebeispiel für MPI

ROCm

- Radeon Open Compute
- Software Plattform zur Unterstützung von AMD GPUs
- Unterstützung für verschiedene Programmiermodelle und Interfaces (OpenMP, HIP, ...)
- Open Source im Gegensatz zu CUDA

$\Rightarrow$ ROCm dient als Plattform bzw. Interface zur Verwendung der AMD GPUs

ROCm Überblick

Abbildung: Ein Überblick über die ROCm Software Plattform[AMD22]

Grundlage der neuen MPI Runtime

- Ziel: ROCm-aware MPI Implementation
- MVAPICH2-GDR als Grundgerüst
  - Ist bereits CUDA-aware
- Wie interagiert man mit den AMD GPUs?
- Wie verhindert man Code Duplication?

Interface der AMD GPUs

- Zwei Optionen:
  - Low-level HSA API
  - High-level HIP API
- Entscheidung fällt auf HIP
  - Portabler Code, funktioniert mit Nvidia und mit AMD GPUs

HIP Beispiel

```
1 //Allokiere Speicher auf der GPU:
2 hipMalloc(&A_d, NBytes);
3 //Kopiere Daten zur GPU:
4 hipMemcpy(A_d, A_h, NBytes, hipMemcpyHostToDevice);
5 //Starte die Berechnung:
6 hipLaunchKernel([...]);
```

Listing 2: Codebeispiel für HIP

Unified Device Abstraction Interface (UDI)

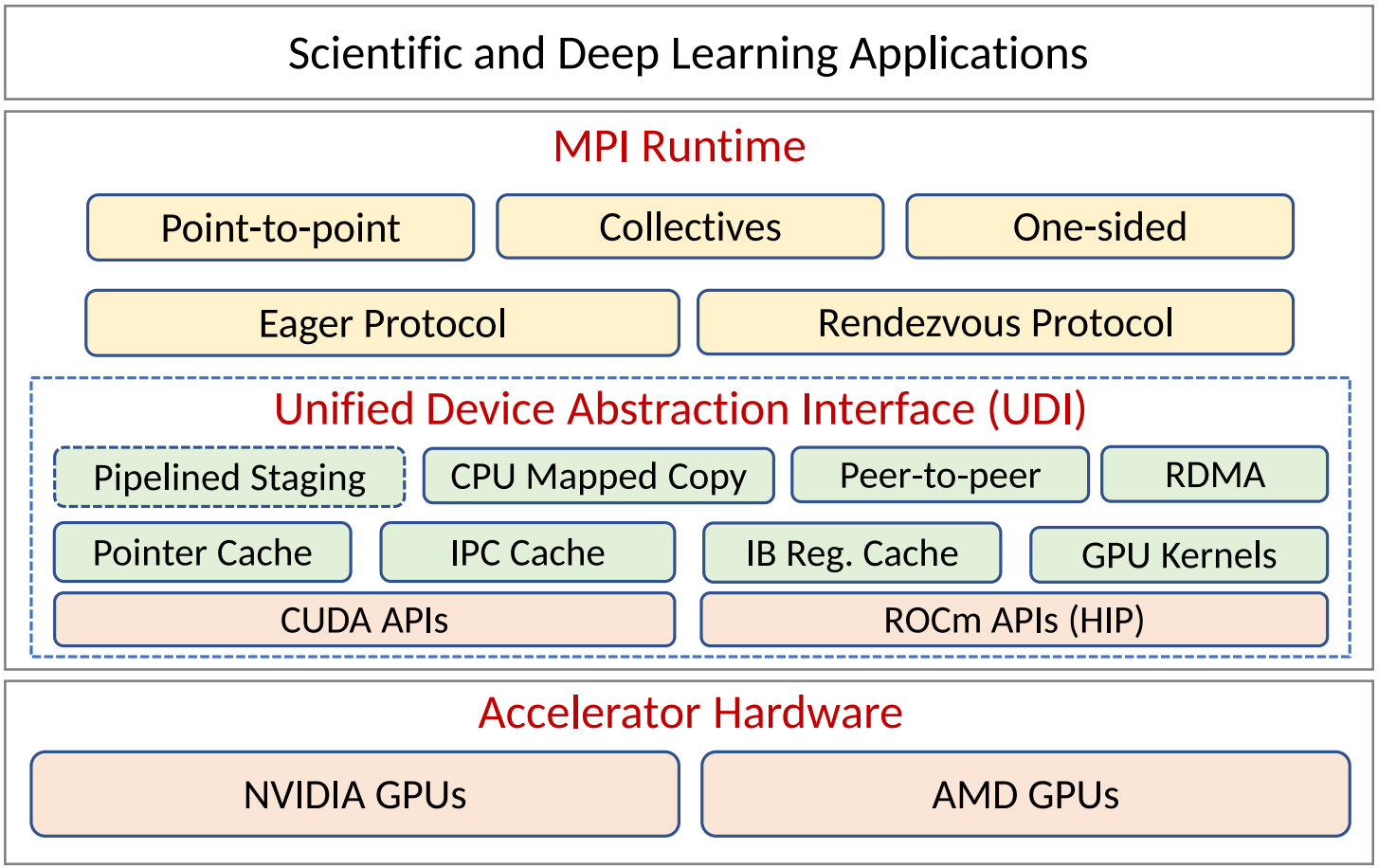


Abbildung: High-level Architektur der neuen MPI Runtime[KHC+21]

Inter Process Communication (IPC)

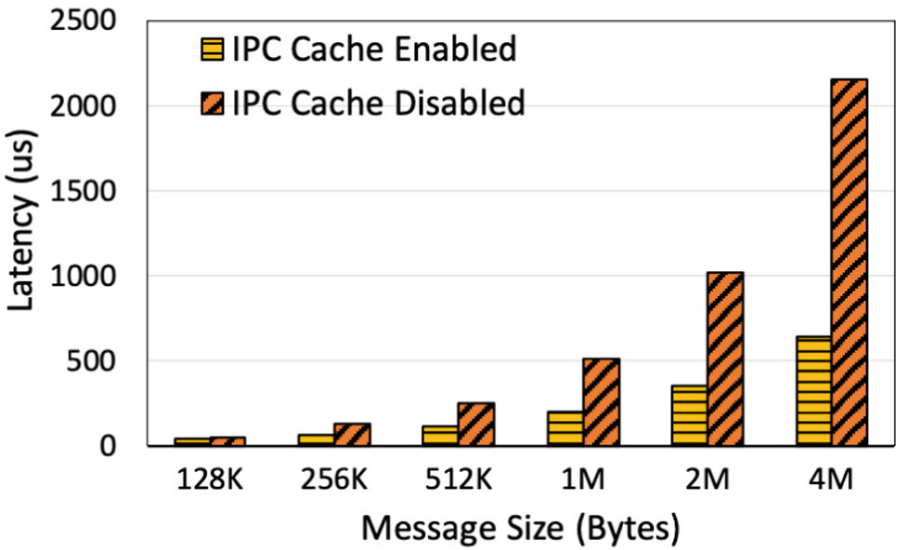
- Kommunikation von GPUs auf demselben Knoten
- Inter Process Communication Interface zur direkten Kommunikation zwischen GPUs
  - Ursprünglich von Nvidia
- CPU wird umgangen

Datenaustausch mit IPC

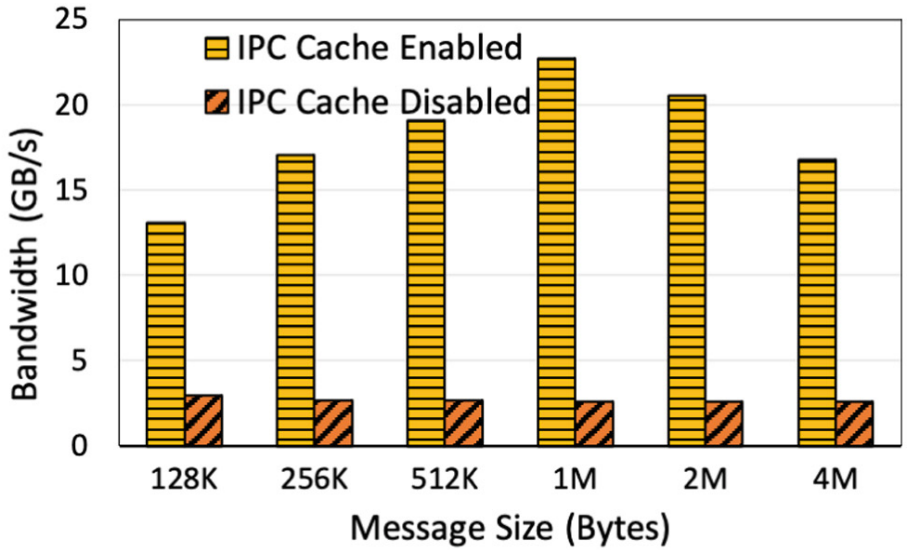
1. Sender erstellt IPC Handle
   - Enthält Informationen über den kommenden Datenaustausch
2. Handle wird an den Empfänger gesendet
3. Empfänger kann Inhalt des Speichers vom Sender in den eigenen Speicher kopieren

- IPC Handles werden beim Empfänger und beim Sender in einem Cache gespeichert
  $\Rightarrow$ Folgende Zugriffe deutlich beschleunigt

Auswirkungen von IPC



(a) Intra-Node Latency



(b) Intra-Node Bandwidth

Abbildung: Latenz und Bandbreite bei der Nutzung des IPC Caches[KHC+21]

Fertiges Design

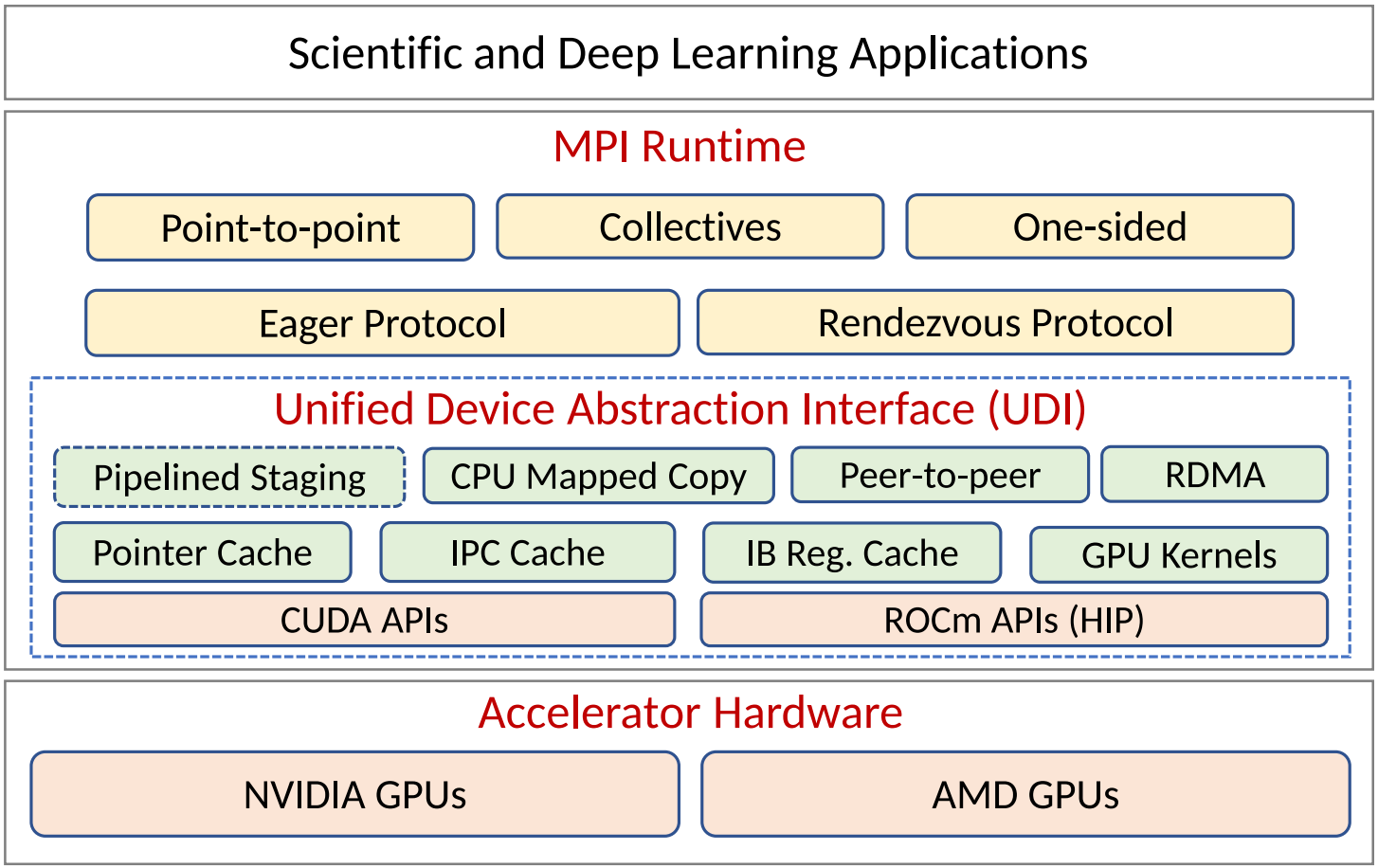
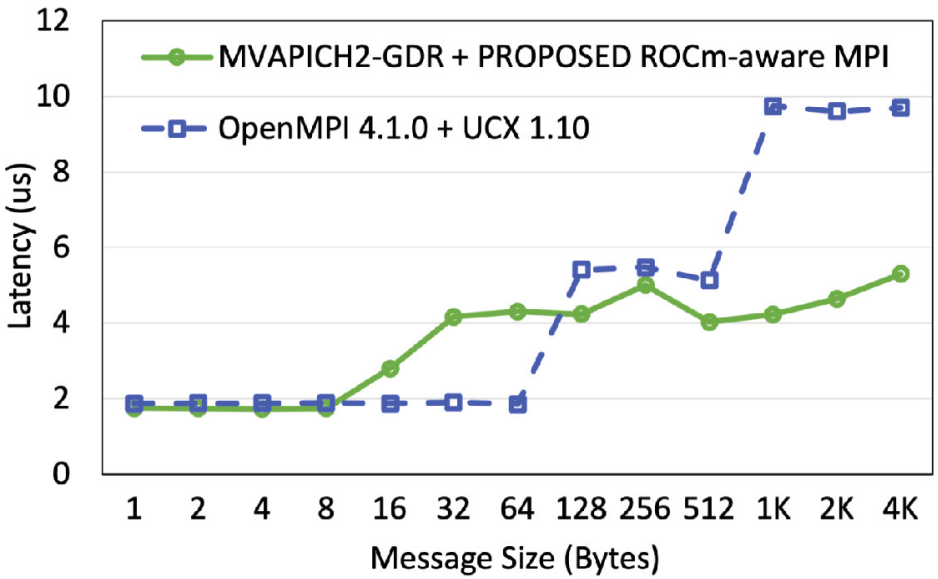


Abbildung: High-level Architektur der neuen MPI Runtime[KHC+21]

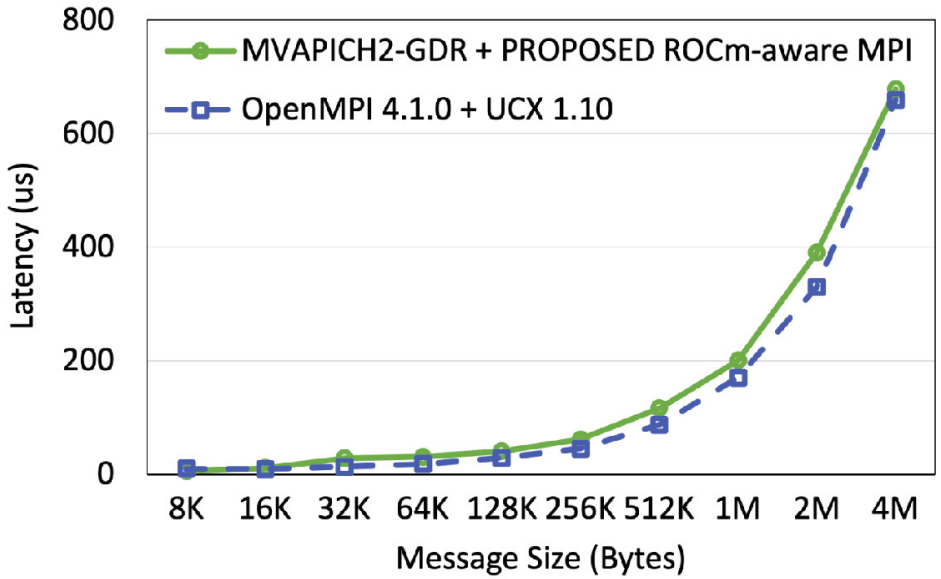
Testumgebung

- Evaluation durchgeführt auf dem Corona Cluster des LLNL
  - Genutzte Knoten: AMD EPYC 7402 CPU (24 Kerne) mit 8 MI50 AMD GPUs pro Knoten
- ROCm Version 4.1.0
- OpenMPI + UCX zum Vergleich benutzt

OSU Microbenchmarks (Latenzen)



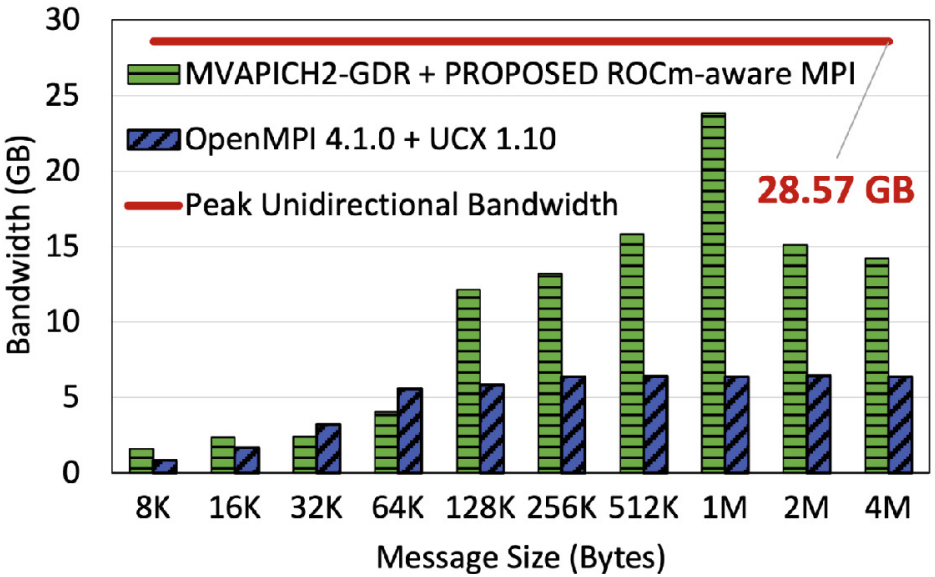
(a) Latency (Small Messages)



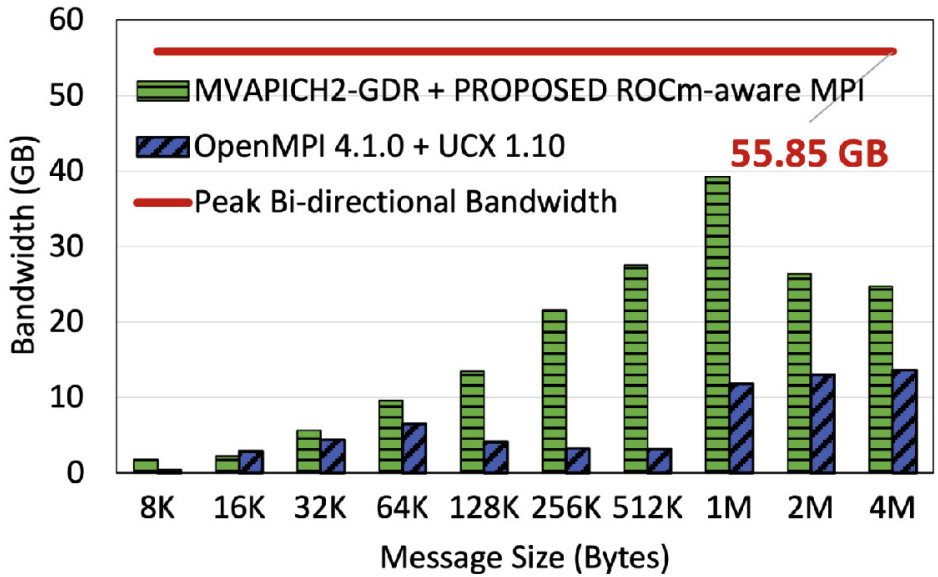
(b) Latency (Large Messages)

Abbildung: Latenzmessungen mit den OSU-Microbenchmarks[KHC+21]

OSU Microbenchmarks (Bandbreiten)



(c) Bandwidth



(d) Bi-Directional Bandwidth

Abbildung: Bandbreitenmessungen mit den OSU-Microbenchmarks[KHC+21]

Anwendungsbenchmark (rocHPCG)

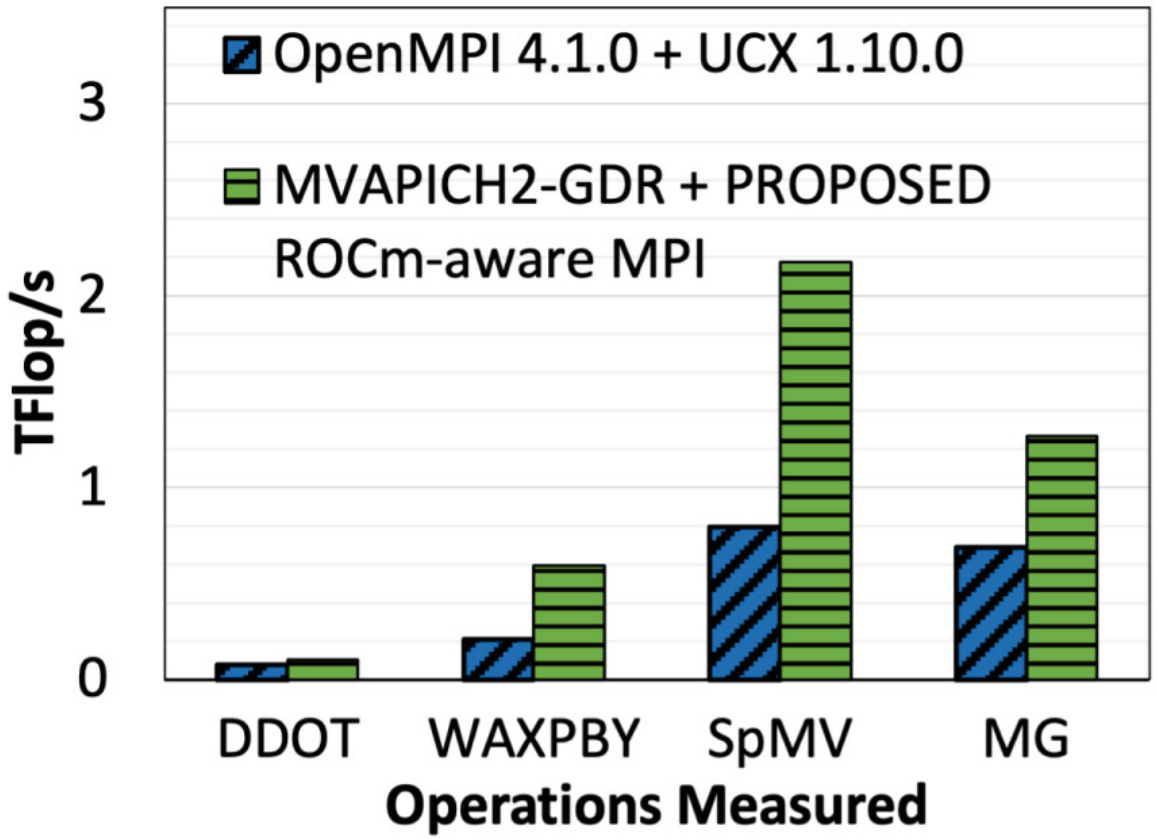


Abbildung: Vergleich der MPI Implementationen mit dem ROCm-aware High Performance Conjugate Gradients Benchmark[KHC+21]

Zusammenfassung

- Software nicht auf AMD GPUs ausgelegt
- Design einer ROCm-aware MPI Runtime
- Erste ihrer Art
- Ergebnisse der Benchmarks sind vielversprechend
  - Bessere Latenzen, Bandbreiten, TFLOP/s

Literatur

[AMD22] AMD. ROCm Documentation, 2022. last access on 13.01.2022.

[Gup20] Pradeep Gupta. CUDA Refresher: Reviewing the Origins of GPU Computing, 04 2020.

[KHC+21] Kawthar Shafie Khorassani, Jahanzeb Hashmi, Ching-Hsiang Chu, Chen-Chun Chen, Hari Subramoni, and Dhabaleswar K Panda. Designing a ROCm-Aware MPI Library for AMD GPUs: Early Experiences. In *International Conference on High Performance Computing*, pages 118–136. Springer, 2021.

[McC19] Morgan McCorkle. U.S. Department of Energy and Cray to Deliver Record-Setting Frontier Supercomputer at ORNL, 05 2019.