UMUGUC

Usability-focused Multi-GPU Compression

Programm / Ausschreibung	Kooperationsstrukturen, Kooperationsstrukturen, Bridge Ausschreibung 2023	**Status**	laufend
Projektstart	01.09.2023	**Projektende**	31.08.2026
Zeitraum	2023 - 2026	**Projektlaufzeit**	36 Monate
Keywords	GPU; compression; API; distributed memory		

Projektbeschreibung

Die meisten der effektivsten Architekturen für die Verarbeitung großer Datenmengen basieren heute auf Clustern mit verteiltem Speicher, die mit GPUs oder anderen Beschleuniger-Hardwarekomponenten ausgestattet sind. Bei der Arbeit mit großen Datenmengen oder E/A- und bandbreitengebundenen Algorithmen auf dieser Art von Hardware kann eine Komprimierung sehr vorteilhaft sein. Hocheffiziente Implementierungen dieses Prinzips erfordern, dass Daten komprimiert im Speicher gehalten werden und erst so nah wie möglich an der eigentlichen Berechnung dekomprimiert werden, idealerweise innerhalb desselben GPU-Kerns. Dies ermöglicht nicht nur eine verbesserte Leistung, sondern kann auch die Energieeffizienz erhöhen, da weniger Kommunikation außerhalb des Chips erforderlich ist.

Die Implementierung einer groß angelegten industriellen oder wissenschaftlichen Anwendung für GPU-Cluster mit dieser Art von End-to-End-Komprimierung erfordert ein Team, das mit der Parallelisierung von verteiltem Speicher, GPU-spezifischer Optimierung und Kompressionsalgorithmen mit hohem Durchsatz vertraut ist, zusätzlich zu dem für den eigentlichen Anwendungsfall erforderlichen Fachwissen. Dies macht die Vorteile der End-to-End-Kompression für alle außer den größten Unternehmen und den am besten finanzierten und am weitesten verbreiteten Open-Source-Forschungssoftware-Stacks unerreichbar.

In UMUGUC werden wir dieses Problem angehen, indem wir eine benutzerfreundliche, deklarative API für die komprimierte Datenverarbeitung auf GPU-Clustern entwickeln, die auf dem bestehenden Celerity High-Level-C++-Laufzeitsystem basiert. Anstatt die Komprimierung und Dekomprimierung für die verschiedenen Übertragungs- und Speicheranforderungen einer Anwendung manuell zu implementieren, müssen Entwickler lediglich die funktionale Präzision und die Zugriffsanforderungen für jeden Datenpuffer angeben, der an einer bestimmten Berechnung beteiligt ist. Anhand dieser Informationen kann ein kombiniertes Meta-Programmier- und Laufzeitsystem zur Kompilierungszeit eine geeignete Komprimierungsstrategie auswählen, die bei Bedarf automatisch implementiert wird, indem alle Kernels, die mit komprimierten Daten interagieren, geändert werden - für den Anwendungsentwickler transparent.

Der wichtigste industrielle Anwendungsfall des Projekts umfasst mehrere Berechnungsschritte, die bei der Verarbeitung von

Punktwolkendaten aus der flugzeuggestützten Laserkartierung erforderlich sind. Dabei handelt es sich um sehr E/A-intensive Vorgänge, die von einer durchgängigen Komprimierung stark profitieren können, und die Struktur der betroffenen räumlichen Daten bietet Möglichkeiten für spezielle Kompressionsverfahren. Alle Implementierungsarbeiten werden auf der Industriestandard-Programmierschnittstelle SYCL basieren und auf mindestens 3 Hardware-Plattformen validiert.

Abstract

Today, the majority of the most effective architectures for large-scale data processing are based on distributed memory clusters featuring GPUs or other accelerator hardware components. When working with large data volumes or I/O and bandwidth-bound algorithms on this type of hardware, compression can be very beneficial. Highly efficient implementations of this principle require keeping compressed data in memory, and only decompressing it as close to the actual computation as possible, ideally within the same GPU kernel. Not only does this enable improved performance, it can also increase energy efficiency by reducing the need for off-chip communication.

Implementing a large-scale industrial or scientific application for GPU clusters with this type of end-to-end compression requires a team familiar with distributed memory parallelization, GPU-specific optimization, and high-throughput compression algorithms, in addition to the domain knowledge necessary for the actual use case. This puts the advantages of end-to-end compression outside the reach of all but the largest companies and most well-funded and broadly applicable open-source research software stacks.

In UMUGUC, we will address this issue by creating a user-friendly, declarative API for compressed data processing on GPU clusters, based on the existing Celerity high-level C++ runtime system. Rather than manually implementing compression and decompression for the various transfer and storage requirements of an application, developers will only be required to specify the functional precision and access requirements on each data buffer involved in a given computation. This information will allow a combined compile-time meta-programming and runtime system to select a suitable compression strategy, which will be automatically implemented as required by modifying all compute kernels interacting with compressed data -- transparently to the application developer.

The main industrial use case of the project involves several computational steps necessary in the processing of point cloud data from airborne laser mapping. These are a highly I/O-intensive operations, which can benefit greatly from end-to-end compression, and the nature of the spatial data involved provides opportunities for specialized compression schemes. All implementation work will be based on the industry-standard SYCL programming interface, and validated on at least 3 hardware platforms.

Projektkoordinator

- Universität Innsbruck

Projektpartner

- Airborne Hydro Mapping GmbH