FINALER BERICHT

Richtlinie zum „Software-Sprint"

UseAI.rs – Hochperformante KI-Bibliothek in Rust

Schlussbericht

Zuwendungsempfänger: Lucca Ochsner
Förderkennzeichen: 16IS25S20

---

1. Vorwort

Die Motivation für dieses Projekt wurzelt in der persönlichen Entwicklung vom IT Servicedesk Mitarbeiter zum IT Manager. Die anfängliche Begeisterung, das Hobby zum Beruf zu machen, wich schnell der Erkenntnis, dass die täglichen Aufgaben in der IT Administration mein technisches Potenzial nicht ausschöpften. Aus der Notwendigkeit heraus, nach Datenverlusten robuste Lösungen zu entwickeln, begann ich vor sechs Jahren, mein hobby mit dem ziel eine eigene Infrastruktur aufzubauen verbant. Dieses private Projekt entwickelte sich zu einer umfangreichen Codebasis.

Mit wachsender beruflicher Verantwortung und dem Aufkommen von KI Technologien wurde ein grundlegendes Problem deutlich: Bestehende Infrastrukturen sind oft nicht auf die enormen Rechenlasten von KI Anwendungen ausgelegt. Eigene Experimente mit KI Algorithmen zeigten, dass es sich hierbei weniger um Magie als vielmehr um ein anspruchsvolles Optimierungsproblem handelt. Die Vision entstand, die eigene Hobby Infrastruktur um ein KI Modul zu erweitern und damit eine grundlegende Herausforderung anzugehen.

Die Entscheidung, dieses Vorhaben zu professionalisieren, war ein Wendepunkt. Es ging nicht länger um ein Hobby, sondern um die Schaffung einer Low Level Grundlage für eine europäische KI Infrastruktur. Ziel ist es, in Europa entwickelte Modelle zu integrieren und so zu skalieren, dass etablierte Forschungsinstitute eine sichere und souveräne technologische Basis erhalten, um KI verantwortungsvoll zum positiven Nutzen für die Gesellschaft einzusetzen.

---

2. Beitrag des Projektes zu den Zielen der Förderinitiative „Software Sprint"

Der Umfang eines solchen Infrastrukturprojekts

Ein Infrastrukturprojekt dieser Größenordnung wirft komplexe technische und gesellschaftliche Fragen auf. Künstliche Intelligenz ist mehr als nur optimierte Algorithmen und leistungsfähige Hardware. Regulatorische Rahmenbedingungen wie der europäische AI Act, die Datenschutz Grundverordnung (DSGVO) und die Prinzipien der Data Governance setzen einen hohen Standard für KI Systeme, die in Europa betrieben werden.

Datenschutz darf kein nachträglicher Gedanke sein, sondern muss von Beginn an architektonisch verankert werden. Data Governance erfordert die lückenlose Nachvollziehbarkeit der Datenverarbeitung, insbesondere bei sensiblen Informationen. Der EU AI Act fordert je nach Risikoklasse eines KI Systems Transparenz- und Nachweispflichten. Eine Plattform, die sowohl für die Forschung als auch für den industriellen Einsatz geeignet sein soll, muss diese Anforderungen fundamental berücksichtigen können.

Um diese hochgesteckten Ziele zu erreichen, ist eine solide technische Grundlage unerlässlich. Das bloße Hinzufügen von Werkzeugen zu bestehenden Systemen birgt erhebliche Risiken. Aus diesem Grund wurde UseAI.rs als modulare, GPU-beschleunigte Bibliothek in Rust entwickelt. Sie dient als Fundament für die größere Vision einer europäischen KI Infrastruktur, die Datensouveränität und Performance in sich vereint.

Technische Herausforderungen und Durchbrüche

Ein Projekt mit solch technischem Tiefgang ist selten frei von Hindernissen. Eine frühe Herausforderung stellten CUDA Treiber dar, die aufgrund ihrer Aktualität Inkompatibilitäten aufwiesen. Es gelang jedoch, die Treiber noch vor der Veröffentlichung offizieller stabiler Versionen anzupassen. Die dadurch gewonnene Zeit wurde effizient umgewidmet, um die Implementierung von NCCL (NVIDIA Collective Communications Library) weiter voranzutreiben als ursprünglich geplant.

Eine weitere wichtige Erkenntnis war die Schwierigkeit, ein derart abstraktes und technisches Projekt verständlich zu vermitteln. UseAI.rs adressiert das Themenfeld Infrastruktur des Software Sprints direkt. Die durch CubeCL erreichte Hardware-Agnostik ermöglicht es Forschenden, ihre bestehende GPU Hardware herstellerunabhängig und optimal zu nutzen. Die Gewährleistung einer vollständig lokalen Datenverarbeitung ist insbesondere für die Forschung mit sensiblen Daten von entscheidender Bedeutung. Die Veröffentlichung als Open Source Projekt unter der "PolyForm Perimeter License 1.0.0" schafft Transparenz und ermöglicht die Auditierbarkeit des Codes durch Dritte.

---

3. Ausführliche Darstellung der Ergebnisse in der Förderphase

UseAI.rs als KI-Workbench

UseAI.rs ist eine in Rust entwickelte Open Source KI Workbench, die als Fundament für praktische Anwendungen in Forschungsumgebungen dient. Das Projekt baut auf den Bibliotheken Burn und CubeCL auf und nutzt einen JSON basierten Konfigurationsstandard. Diese Konfigurationsdateien definieren alle notwendigen Schritte, um Daten zu transformieren und komplexe KI-Operationen auszuführen. CubeCL stellt dabei die hardwarenahen operativen Werkzeuge bereit, um ein leistungsfähiges und praktisches KI-Framework zu realisieren. Das Tooling ermöglicht es, verschiedene Modelle miteinander interagieren zu lassen und experimentelle Forschungsansätze flexibel umzusetzen.

Für die wissenschaftliche Arbeit liegt der Fokus auf der Implementierung von Reinforcement Learning Algorithmen wie Q-Learning und Deep Q-Networks. Zusätzlich sind Ensemble Learning Algorithmen geplant, wobei in der ersten Iteration Gradient Based Decision Trees (GBDT) implementiert werden.

Erreichte Komponenten

Die modulare Architektur von UseAI.rs wurde als mehrschichtiges System konzipiert, das verschiedene Abstraktionsebenen für unterschiedliche Anwendungsfälle bietet. Die höchste Ebene bildet das **UseAI Tool (CLI)**, das einen Konfigurationsmanager und ein Staging-Werkzeug umfasst. Es ermöglicht Forschenden, die Features von UseAI.rs mit eigenen Daten zu erproben, ohne dass Rust Kenntnisse erforderlich sind. Für die Interaktivität wurde das Tool mit Inquire implementiert und wird durch umfassende Anleitungen für den produktiven Einsatz ergänzt. Als technische Voraussetzungen werden lediglich Rust und eine MongoDB Instanz benötigt.

Darunter liegt die **Store Komponente**, die eine einfache, synchrone Speicherschicht auf Basis von MongoDB bereitstellt. Sie dient dem Ablegen von Konfigurationen und Berechnungsergebnissen (Outputs), welche für nachgelagerte Analysen (Downstream) zur Verfügung stehen.

Die **Stage Komponente** ist für die Datenaufbereitung zuständig. Sie verarbeitet und strukturiert abstrakte Informationen so, dass sie von nachfolgenden Verarbeitungsschichten effizient genutzt werden können. Um die Leistung zu optimieren, werden bestimmte Operationen mittels der Crossbeam Bibliothek auf der CPU parallelisiert.

Das Herzstück bildet die **Calculator Komponente**. Sie wandelt die vorbereiteten Daten in unsere Tensoren um, die eigentlichen Datenstrukturen für die GPU Berechnungen. Aktuell werden diese Tensors mit dem CUDA Backend ausgeführt. Das Design der Komponente wurde von Grund auf für Reinforcement Learning (RL) ausgelegt. Für besonders komplexe RL Operationen werden zudem eigene GPU Kernels mithilfe von CubeCL implementiert, um maximale Performance zu gewährleisten.

Technische Durchbrüche und Beiträge zum Open-Source-Ökosystem

Ein zentraler Erfolg des Projekts ist der Beitrag zum Open Source Ökosystem, insbesondere die Integration von NCCL in CubeCL (https://github.com/tracel-ai/cubecl/pull/1057). Diese Implementierung verbessert die Multi GPU Kommunikation fundamental und wird ein Kernbestandteil der kommenden UseAI Version v0.2 sein. Sie ermöglicht eine deutlich effizientere Skalierung von KI Workloads über mehrere Grafikprozessoren hinweg. Die Implementierung GPU-beschleunigter Algorithmen, wie des Bitonic Sort, diente als wichtiger Proof of Concept, um die Parallelisierungsfähigkeiten der Architektur zu validieren.

Aus der Projektarbeit ist zudem ein spezialisierter Parser hervorgegangen, der zur einfachen Visualisierung von Rust Backend Runtimes entwickelt wurde. Dieses Werkzeug wird in den kommenden Wochen veröffentlicht und erlaubt es, komplexe Backend Architekturen transparent darzustellen und besser zu verstehen.

Dokumentation und Transparenz

Eine umfassende Dokumentation befindet sich in Arbeit, um Transparenz und Nachvollziehbarkeit zu gewährleisten. *The Hitchhiker's Guide to UseAI.rs* (https://use-airs.github.io/Use-Ai.rs/book/blog/index.html) dokumentiert architektonische Entscheidungen und frühe Entwicklungsphasen. Detaillierte Anleitungen zur GPU Architektur und Integration sind in kurze auch im Größerem umfang im Verzeichnis `docs/` zu finden.

Nicht erreichte Meilensteine

Die ursprünglich geplante vollständige Implementierung von GBDT und Deep Q-Networks musste zurückgestellt werden. Im Laufe des Projekts wurde deutlich, dass die Treiberprobleme hardwarenah behandelt werden mussten, weswegen die Priorität auf tiefere, fundamentale Implementierungen gelegt wurde. Diese Entscheidung erwies sich als richtig, da eine robuste hardwarenahe Grundlage für fortschrittliche Algorithmen in der ersten Iteration mehr Wert schafft als oberflächliche Implementierungen. Die Umsetzung der genannten Algorithmen ist für das erste Quartal 2026 geplant, wobei die nun vorhandene stabile Infrastruktur eine deutlich effizientere Entwicklung ermöglichen wird. Dabei zeigen die Bitonic und Regix Sort Algorythmen klar, dass man Konkurenzfähige KI erwarten kann und zwar bereits in der ersten Iteration.

---

4. Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Zielgruppe: Ein Workspace für alle Forschungsgebiete

UseAI.rs richtet sich primär an Forschende aus allen Disziplinen, die aus ihren Daten neue Erkenntnisse gewinnen möchten. Die Workbench wurde bewusst generisch gestaltet, um eine breite Palette von Forschungsansätzen zu unterstützen:

- **Robotik und KI-Forschende**, die performante Reinforcement Learning Frameworks für Echtzeit Anwendungen ohne Cloud Latenzen benötigen. Insbesondere Q-Networks und Deep Q-Networks ermöglichen es, komplexe Entscheidungsprozesse effizient zu trainieren.
- **Data Scientists aus diversen Fachbereichen**, die flexible Werkzeuge für experimentelle Ansätze suchen.
- **Embedded Entwickler in der Forschung**, die Machine Learning Inferenz auf ressourcenbeschränkten Geräten durchführen müssen.
- **Rust Entwickler in der akademischen Forschung**, die ML Funktionalität in ihre Projekte integrieren wollen, ohne auf Python-Bindings angewiesen zu sein.

Nutzen

Der zentrale Nutzen von UseAI.rs liegt in der **vollständigen Datensouveränität** über Forschungsdaten und Modelle. Die **Performance** wird durch GPU-Beschleunigung ohne einen Lock in Effekt auf einen bestimmten Hardware Hersteller (Vendor Lock in) maximiert. Gleichzeitig ist der Code durch die Nutzung von CubeCL **portabel** und kann auf verschiedenen GPU-Backends ausgeführt werden.

Als Open Source Projekt ist der Code vollständig **auditierbar**, ermöglicht Beiträge aus der Community und erleichtert die Integration in bestehende Forschungsprojekte. Zudem bietet es Studierenden eine Lernplattform für reale GPU Programmierung und moderne Rust Entwicklung im Hochleistungsbereich.

Weiterentwicklung

Für das erste Quartal 2026 sind folgende Entwicklungen geplant:

- Implementierung von Gradient Boosted Decision Trees mit einer zu XGBoost kompatiblen API
- Fertigstellung von Q-Learning und Deep Q-Networks für Reinforcement Learning
- Weitere GPU-optimierte Operationen und Shared-Memory-Implementierungen

Die langfristige Verwertung und Weiterentwicklung wird durch die in privater Initiative gegründete Q-Network GmbH sichergestellt. Ziel ist es, UseAI.rs als hochperformante Forschungsplattform zu etablieren, die akademischen Forschenden nachhaltig zur Verfügung steht. Darüber hinaus wurde bereits ein erstes Forschungscluster aufgebaut, das die praktische Erprobung und Weiterentwicklung der Plattform ermöglicht.

Persönliche Weiterentwicklung

Die Arbeit an UseAI.rs hat meine technischen Fähigkeiten erheblich erweitert. Ich habe ein tiefgreifendes Verständnis für GPU Architekturen entwickelt, einschließlich Warp-Level Parallelisierung, Shared Memory und Speicherhierarchien. Die Entwicklung mit CubeCL erforderte eine intensive Auseinandersetzung mit Compile Zeit Codegenerierung und dem Makro System von Rust. Der Umgang mit komplexen Type Constraints, Lifetime Management in GPU Kontexten hat meine Rust Expertise vertieft.

---

5. Kurze Darstellung relevanter Lernerfahrungen

Die erste Projektphase war geprägt von einem intensiven Lernprozess, bei dem die verwendeten Bibliotheken durch iteratives Programmieren in der Tiefe ausgelotet wurden. In diesem Prozess wurden zwangsläufig auch Entscheidungen getroffen, die sich später als suboptimal erwiesen und korrigiert werden mussten. Dieser iterative Ansatz war jedoch notwendig, um die Grenzen und Möglichkeiten von CubeCL vollständig zu verstehen.

Wichtige Erkenntnisse entstanden durch intensive Auseinandersetzung mit CUDA, NCCL, SIMD und weiteren hardwarenahen Konzepten. Diese Technologien wurden nicht nur oberflächlich betrachtet, sondern deren fundamentale Prinzipien wurden durchdrungen, um fundierte architektonische Entscheidungen treffen zu können.

---

6. Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer*innen

- **Code-Repository:** https://code.q-network.org/Q-Research/UseAIrs
- **NCCL Integration (Pull Request):** https://github.com/tracel-ai/cubecl/pull/1057
- **Projektdokumentation:** *The Hitchhiker's Guide to UseAI.rs* unter https://use-airs.github.io/Use-Ai.rs/book/blog/index.html bietet Einblicke in Entscheidungsprozesse und die Entwicklungsgeschichte.
- **Institutionelle Einbindung:** Informationen zur langfristigen Verwertung und institutionellen Einbindung finden sich auf https://q-network.org/.
- **Live-Infrastruktur:** Die Website https://q-network.org/research bietet Einblicke in die Live-Forschungsinfrastruktur mit Echtzeit-GPU-Metriken und Benchmarks.

---

7. Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Der Projektverlauf war geprägt von einer außerordentlich hohen Arbeitsintensität. Die robuste Implementierung der GPU-Infrastruktur und die Beherrschung der CUDA Schnittstelle zu Rust erwiesen sich als aufwendig, insbesondere da neue CUDA Treiber Breaking Changes mit sich brachten.

Um die gesetzten Meilensteine zu erreichen, war eine durchgehende Arbeit mit maximaler Intensität notwendig, die nur durch die Verbindung von professionellem Anspruch und persönlicher Leidenschaft für das Thema zu bewältigen war. Die Förderung ermöglichte es, dieses ambitionierte Projekt in einem konzentrierten Zeitraum voranzutreiben und eine solide technische Grundlage zu schaffen. Entsprechend wurde die Arbeis Zeit restlos genutzt, wobei der Akltuelle stand nicht erreicht worden wäre wenn ich nicht mit großer freude im rahmen meines Hobbys meine Arbeit am projekt mindestens verdoppelt hätte.

---

8. Kurze Darstellung von relevanten Arbeiten bei anderen Stellen

Die Arbeit an UseAI.rs stand in engem Austausch mit anderen Open Source Projekten. Insbesondere profitierte das Vorhaben vom **CubeCL Projekt** (https://github.com/tracel-ai/cubecl), einem Framework für hardware agnostische GPU-Programmierung. UseAI.rs nutzt CubeCL nicht nur als Abhängigkeit, sondern trägt aktiv zu dessen Weiterentwicklung bei.

Der wesentlichste Beitrag in diesem Zusammenhang war die **Integration von NCCL in CubeCL** (https://github.com/tracel-ai/cubecl/pull/1057). Diese von mir entwickelte Implementierung verbessert die Multi GPU Kommunikation erheblich und ist bei uns verfügbar. Der Code befindet sich bereits in der Stage Branch und wird nach weiteren iterationen veröffentlicht.

Zudem wurde das **Burn Framework** (https://github.com/tracel-ai/burn) analysiert, um dessen Design Patterns für High Level APIs zu verstehen. Obwohl Burn ebenfalls CubeCL nutzt, liegt dessen Fokus auf Deep Learning, während UseAI.rs Reinforcement Learning und klassische ML-Algorithmen priorisiert.

Die Implementierung der Algorithmen basierte auf der Analyse von Fachliteratur zu GPU beschleunigten Sortieralgorithmen und Warp Level Primitives, insbesondere aus der NVIDIA CUDA Dokumentation und wissenschaftlichen Veröffentlichungen.

Zusammenfassung und Ausblick

UseAI.rs hat in der Förderphase eine solide technische Basis als KI Workbench für Forschende geschaffen. Es ist eine GPU beschleunigte, hardware agnostische Infrastruktur für Machine Learning in Rust entstanden, die auf den Prinzipien von Datensouveränität und Performance basiert. Die Integration von NCCL in CubeCL stellt einen wesentlichen Beitrag zum Open Source Ökosystem dar, von dem auch andere Projekte profitieren werden.

Während nicht alle ursprünglich geplanten Algorithmen vollständig implementiert wurden, ist die geschaffene Grundlage nachhaltig und strategisch wertvoller als eine Sammlung halbfertiger Features. Die robuste GPU-Infrastruktur ermöglicht nun eine effiziente Entwicklung höherer algorithmischer Schichten.

Die anstehende Dokumentation und die Veröffentlichung als Open Source Projekt sichern die langfristige Nutzbarkeit und fördern Beiträge aus der Community. Mit der Verwertung durch die Q-Network GmbH ist die institutionelle Basis für die kontinuierliche Weiterentwicklung gelegt. Die noch offenen Features werden im ersten Quartal 2026 umgesetzt, um UseAI.rs zu einer noch leistungsfähigeren Plattform für die europäische Forschungslandschaft auszubauen.

Lucca Ochsner