Karlsruher Institut für Technologie

Institut für Programmstrukturen
und Datenorganisation (IPD)

Lehrstuhl Prof. Dr.-Ing. Snelting

Invasives Rust

Bachelorarbeit von

Hermann Krumrey

an der Fakultät für Informatik

Erstgutachter:	Prof. Dr.-Ing. Gregor Snelting
Zweitgutachter:	Prof. Dr. rer. nat. Bernhard Beckert
Betreuende Mitarbeiter:	Dipl.-Inform. Andreas Zwinkau

Bearbeitungszeit: 1. Januar 1990 – 31. Dezember 2000

Zusammenfassung

Hier kommt eine kurze Zusammenfassung hin

Inhaltsverzeichnis

1 Einführung

Die Parallelisierung von Rechnern liegt immer mehr im Fokus der derzeitigen technologischen Entwicklung. Dies erfordert andere Paradigmen, welche effektiv die neuen Architekturen ausnutzen können. Ein möglicher Ansatz ist hierbei das invasive Computing. Dieses ermöglicht es dem Programmierer, die Ressourcennutzung eines Programms feiner zu kontrollieren.

Das IRTSS Betriebssystem, welches eine beispielhafte Implementierung für ein invasives System bietet, unterstützt derzeit die Verwendung der Programmiersprachen C, C++ als auch X10. Hiermit wird die Einführung der Programmiersprache Rust als weitere unterstützte Sprache vorgeschlagen. Diese Sprache hat einige wünschenswerte Merkmale, welche sie als interessant für den Gebrauch im invasiven Computing macht.

2 Grundlagen und Verwandte Arbeiten

2.1 Rust

Rust ist eine Programmiersprache, welche von Mozilla Research entwickelt wird [?]rustWikipediaDe). Die Entwicklung der Sprache begann als persönliches Projekt des Mozilla-Mitarbeiters Graydon Hoare und wird seit 2009 von Mozilla untersützt. 2010 erschien die erste öffentliche Version der Sprache, im Jahr 2015 wurde die erste stabile Version veröffentlicht.

2.1.1 Motivation

Eine der Kernziele der Rust Programmiersprache ist die Sicherheit bezüglich Speicherzugriffen. Dies wird mithilfe des sogenannten "Ownership"-modells erreicht. Dieses Modell ermöglicht es, fehlerhafte Speicherzugriffe und Pufferüberläufe zu vermeiden, ohne dabei von einem Garbage Collector Gebrauch zu machen. Dies macht Rust zu einer interessanten Alternative zu Low-Level Sprachen wie C, welche nicht sicher bezüglich der Speicherverwaltung sind, als auch zu Sprachen mit Garbage Collector wie X10.

Ein weiteres Ziel der Sprache ist die Geschwindigkeit. Rust soll vergleichbare Geschwindigkeiten zu anderen Low-Level Sprachen wie C erreichen und obendrein effizientes Einbinden mit anderen Sprachen ermöglichen.

Nebenläuigkeit ist ein weiteres Hauptziel der Sprache.

Rust bietet Abstraktionen, welche im Gegensatz zu Sprachen wie Python, nicht oder nur gering die Geschwindigkeit des Programm beinflussen.

2.1.2 Grundlegende Eigenschaften der Sprache

Die Programmiersprache Rust lehnt sich syntaktisch an andere C-artige Sprachen an, unterscheidet sich aber hierbei in einigen Aspekten. So sind in ‘if‘/‘else‘ Blöcken beispielsweise die runden Klammern beispielsweise Optional, so wie es auch in Sprachen wie Python der Fall ist.

2.1.3 Architektur/Compiler

Der offizielle Rust Compiler rustc

Zum Abhängigkeitsmanagement und Distribution wurde das Tool ‘cargo‘ entwickelt.

2.2 SPARC

Sparc

2.3 Invasives Computing

Invasives Computing ist ein paralleles Programmiermodell, welches es ermöglicht, temporär Ressourcen auf einem Parallelrechner zu beanspruchen und anschließend wieder freizugeben.

3 Entwurf und Implementierung

Hier sollten vorkommen: Die Rust Bindings, Der Bau des Cross-Compilers octorust, Die Verbesserungen und Abstraktionen in octolib

4 Evaluation

Hier wird evaluiert!

Vergleich mit C

Vergleich mit X10 Garbage Collector triggern

5 Fazit und Ausblick

Fazit!

Literaturverzeichnis

Erklärung

Hiermit erkläre ich, Hermann Krumrey, dass ich die vorliegende Bachelorarbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Ort, Datum	Unterschrift

Danke

Danksagung kommt hierher