Jahresbericht 2014 des Lehrstuhls für Informatik 2 (Programmiersysteme)

Anschrift: Martensstr. 3, 91058 Erlangen
Tel.: +49-9131-85-27621
Fax: +49-9131-85-28809
E-Mail: info@i2.informatik.uni-erlangen.de

Ordinarius:
Prof. Dr. Michael Philippsen
Honorarprofessor:
Hon.-Prof. Dr.-Ing. Bernd Hindel
Hon.-Prof. Dr.-Ing. Detlef Kips
Emeritus:
Prof. em. Dr. Hans Jürgen Schneider
Sekretariat:
Margit Zenk
Wiss. Mitarbeiter:
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Daniel Brinkers
Dipl.-Inf. Georg Dotzler
Marius Kamp, M. Sc. (ab 01.01.2015)
Demian Kellermann, M. Sc. (bis 30.04.2014)
Dr.-Ing. Stefan Kempf (bis 30.09.2014)
Dipl.-Math. Jakob Krainz
Andreas Kumlehn, M. Sc.
Dr.-Ing. Christopher Mutschler
Dr.-Ing. Norbert Oster
Norbert Tausch, M. Eng.
PD Dr. Ronald Veldema
Dipl.-Inf. Tobias Werth
IT-Betreuer:
Dipl.-Ing. (FH) Helmut Allendorf
Manfred Uebler
Gast:
Dr.-Ing. Josef Adersberger
Dipl.-Inf. Samir Al-Hilank
Dipl.-Inf. Ralf Ellner
Dr.-Ing. Martin Jung
Florian Lautenschlager, M. Sc.
Dr.-Ing. Stephan Otto
Dipl.-Inf. Mykola Protsenko

Externes Lehrpersonal:
Dr.-Ing. Klaudia Dussa-Zieger
Dr.-Ing. Martin Jung
Dr.-Ing. Stephan Otto

Der 1972 gegründete Lehrstuhl Informatik 2 (Programmiersysteme) wird seit April 2002 von Prof. Dr. Michael Philippsen (als Nachfolger von Prof. Dr. em. H.-J. Schneider) geleitet. Eng mit dem Lehrstuhl assoziiert ist die Professur für Didaktik der Informatik, deren Forschungsarbeiten separat dargestellt sind.

1 Forschungsschwerpunkte

Im Mittelpunkt der **Programmiersystemforschung** des Lehrstuhls stehen parallele und verteilte Systeme und deren Programmierung sowie Programmiersysteme für eingebettete und mobile Systeme. Software (und deren Erstellung) für solche Systeme sollte nicht komplexer, aber genauso portabel, wartbar und robust sein, wie heute schon für Einprozessorsysteme und Arbeitsplatzrechner. Langfristiges Ziel ist es, den Anwendungen die verfügbare Rechen- und Kommunikationsleistung möglichst ungebremst zur Verfügung zu stellen bzw. aus begrenzten Systemen ein Maximum herauszuholen. Ein besonderer Arbeitsschwerpunkt sind Programmiersysteme für Multicore-Rechner, da deren unausweichliche Verbreitung ebenso wie die preisgünstige Verfügbarkeit von sehr leistungsstarker paralleler Spezialhardware im Massenmarkt (z.B. Grafik-Karten) kaum abschätzbare Auswirkungen auf die Software-Landschaft haben werden. Forschungsergebnisse werden stets an eigenen Prototypen und Demonstratoren praktisch evaluiert.

Wichtige Forschungsfelder

- **Vorhandenes Parallelisierungspotential ausschöpfen.** Da die Taktraten von Mehrkernrechnern kaum noch steigen, dafür aber deren Kernanzahl anwachsen wird, muss das Parallelisierungspotential existierender Software erkannt und ausgeschöpft werden, um an den Leistungssteigerungen der Hardware teilzuhaben. Außer vielleicht in Nischen führt kein Weg an einem Umstieg auf Parallelverarbeitung vorbei. Deshalb entwickelt der Lehrstuhl Werkzeuge, die dem Programmierer interaktiv beim Re-Engineering bestehender Anwendungen helfen, und erarbeitet Architekturmuster für neu zu entwickelnde Software-Projekte, die eine Skalierbarkeit für und damit eine Toleranz gegen wachsende Kernanzahlen aufweisen.
- **Hochleistungsanwendungen portabel machen.** Anwendungsprogrammierer erreichen nur dann bestmögliche Laufzeiten, wenn sie die Überwindung der La-

tenzzeiten und die explizite Kommunikation zwischen unterschiedlichen Komponenten der Systemarchitektur manuell angehen, ihren Code mit Hardware-nahen "Tricks" spezifisch für eine Architektur optimieren und ihre Applikation per Hand in mehrere Teile zerlegen, von denen manche z.B. auf die Grafikkarte ausgelagert werden. Der Lehrstuhl erforscht, wie durch Anhebung des Abstraktionsniveaus der Programmierung die Produktivität gesteigert und die Portabilität verbessert werden kann, indem Code so übersetzt wird, dass verschiedene Teile auf heterogenen Systemkomponenten nebenläufig ausgeführt werden und dass die Datenkommunikation dazwischen transparent für den Entwickler erfolgt. Eine wichtige Frage dabei ist es, wie der Programmierer sein Wissen über bestehende Lokalitätsbeziehungen programmiersprachlich ausdrücken kann, damit es effizienzsteigernd nutzbar bzw. damit Lokalität schon im Design sichtbar ist. Auf dem Weg zu diesem Ziel werden im Rahmen von Re-Engineering-Projekten Details der Hardware-Architektur vor dem Anwendungsentwickler verborgen, z.B. hinter Bibliotheksschnittstellen oder in domänenspezifischen Programmierspracherweiterungen.

- **Parallelitätsgrad dynamisieren.** Hochleistungsanwendungen werden oft für eine bestimmte Prozessorzahl entwickelt. Da die benötigten Rechnerbündelknoten vom Batch-System statisch für eine feste Zeitspanne zugeteilt werden, entstehen zwangsläufig unwirtschaftliche Reservierungslöcher. Analoge Probleme treten bei vielfädigen Anwendungen auf Multicore-Rechnern auf. Der Lehrstuhl arbeitet daher an der dynamischen Anpassung des Parallelitätsgrads durch Code-Transformationen, die die Kosten der resultierenden Datenumverteilungen berücksichtigen, sowie durch Interaktion und Zusammenarbeit mit dem Betriebssystem. Die in Programmen vorgefundenen expliziten, kontrollflussorientierten Synchronisationsmaßnahmen behindern die erforderlichen Analysen, weshalb der Lehrstuhl neue und bessere Programmkonstrukte erforscht, die die bisherigen adäquat ersetzen können und die Synchronisationserfordernisse deklarativ und an den Daten orientiert ausdrücken.

- **Parallelität testbar machen.** Im Software-Engineering nimmt Testen von jeher eine wichtige Stellung ein. Stichworte wie Testüberdeckung, Testdatengenerierung, Zuverlässigkeitsbewertung etc. gehören zum Handwerk. Leider berücksichtigt die Forschung in diesem Bereich den durch Nebenläufigkeit entstehenden Indeterminismus nur unzureichend. Der Lehrstuhl arbeitet daher an Werkzeugen zur Testdatengenerierung, die bei den zugrundeliegenden Überdeckungskriterien auch Verschränkungen nebenläufiger Programmäste berücksichtigen. Dies schließt Arbeiten an Betriebssystemschnittstellen und am Ablaufplaner ebenfalls ein. Weil ferner durch die Nebenläufigkeit der Suchraum in erheblichem Maße wächst, müssen Infrastrukturen erarbeitet werden, die es ermöglichen, Massentests auf großen Rechnerbündeln auszuführen.

- **Software-Entwicklungsprozesse verbessern.** Die heute in der Industrie praktizierte Entwicklung von komplexer, geschäfts- oder sicherheitskritischer Software in global verteilten Teams verlangt nach der Einhaltung von wohldefinierten Software-Entwicklungsprozessen mit entsprechender Werkzeugunterstützung. Im Bereich der **praktischen Softwaretechnik** arbeitet der Lehrstuhl zusammen mit den **Honorar-Professoren Dr. Bernd Hindel und Dr. Detlef Kips**, die als Geschäftsführer zweier mittelständischer Software-Beratungsunternehmen über langjährige Praxiserfahrung in industriellen Software-Entwicklungsprojekten verfügen, daher an einer maschinen-ausführbaren Notation für die Modellierung von Software-Entwicklungsprozessen, wobei wegen der zum Einsatz kommenden vielfältigen Werkzeuge und Notationen, sowohl die (teil-) automatisierte Rückgewinnung von Nachverfolgbarkeitsinformation aus den Artefakten eines Entwicklungsprojekts, als auch die modellbasierte Entwicklung, Integration und Konfiguration von Softwarekomponenten betrachtet werden, wie sie insbesondere beim Entwurf eingebetteter Systeme für den Automobilbau üblich sind.

2 Forschungsprojekte

2.1 Design for Diagnosability

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Andreas Kumlehn, M. Sc.
Dr.-Ing. Josef Adersberger
Florian Lautenschlager, M. Sc.
Beginn: 15.5.2013
Förderer:
IuK Bayern
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

Viele Software-Systeme verhalten sich während der Testphase oder sogar im Regelbetrieb im negativen Sinne auffällig. Die Diagnose und die Therapie solcher Laufzeitanomalien ist oft langwierig und aufwändig bis hin zu unmöglich. Mögliche Folgen bei der Verwendung des Software-Systems sind lange Antwortzeiten, nicht

erklärbares Verhalten oder auch Abstürze. Je länger die Folgen unbehandelt bleiben, desto höher ist der entstehende wirtschaftliche Schaden.

”Design for Diagnosability” beschreibt eine Werkzeugkette mit Modellierungssprachen, Bausteinen und Werkzeugen, mit denen die Diagnosefähigkeit von Software-Systemen gesteigert wird. Mit dieser Werkzeugkette werden Laufzeitanomalien schneller erkannt und behoben – idealerweise noch während der Entwicklung des Software-Systems. Unser Kooperationspartner QAware GmbH bringt ein Software EKG ein, mit dem die Exploration von Laufzeit-Metriken aus Software-Systemen, visualisiert als Zeitreihen, möglich ist.

Das Forschungsprojekt Design for Diagnosability erweitert das Umfeld dieses bestehenden Software-EKG. Die Software-Blackbox misst minimal-invasiv technische und fachliche Laufzeitdaten des Systems. Die Speicherung der erfassten Daten erfolgt in Form von Zeitreihen in einer neu entwickelten Zeitreihendatenbank. Das Konzept dieser Zeitreihendatenbank ist darauf ausgelegt, eine Vielzahl an Zeitreihen äußerst effizient hinsichtlich Speicherplatzbedarf und Zugriffszeiten zu speichern.

Die Zeitreihen werden mit der Time-Series-API analysiert, z.B. mittels einer automatisierten Strategie zur Erkennung von Ausreißern. Die Time-Series-API bietet Grundbausteine, um weitere Strategien zur Identifikation von Laufzeitanomalien in Zeitreihen umzusetzen.

Die aufgeführten Werkzeuge werden in Kombination mit dem bestehenden Software-EKG zum Dynamic Analysis Workbench ausgebaut, um eine zeitnahe Diagnose und Behebung von Laufzeitanomalien zu ermöglichen. Hierzu sind Diagnosepläne vorgesehen, die einen Software-Entwickler unterstützen, um schneller und zuverlässiger die Laufzeitanomalie einzugrenzen und zu erkennen.

Das Ziel der Werkzeugkette ist die Qualität von Software-Systemen, insbesondere der Kennzahlen Mean-Time-To-Repair sowie Mean-Time-Between-Defects, zu erhöhen.

2.2 Effiziente Software-Architekturen für verteilte Ereignisverarbeitungssysteme

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Christopher Mutschler
Laufzeit: 15.11.2010–15.5.2016
Förderer:
Fraunhofer Institut für Integrierte Schaltungen

Funkortungssysteme, auch bekannt als Real-Time Location Systems (RTLS), geraten immer mehr in den Fokus der Logistik, Produktion und vieler weiterer Prozesse. Diese Systeme liefern wertvolle Informationen über den Aufenthaltsort von beteiligten Objekten zur Laufzeit. Damit können Prozesse verfolgt, analysiert und optimiert werden. Neben den Forschungsbereichen an der Basis von Ortungssystemen, wie robuste und störsichere Ortungstechnologien oder Verfahren zur hochgenauen Positionsbestimmung, rücken mehr und mehr Methoden in den Vordergrund, die aus Positionsdatenströmen wertvolle Informationen für weitere Verarbeitungsstufen gewinnen. In diesem Kontext erforscht das Projekt Verfahren zur Ereignisdetektion in Positionsdatenströmen zur Laufzeit.

2011 wurde damit begonnen, auftretende Ereignisse in Lokalisierungssystemen zu erkennen und vorherzusagen. Hierfür werden Ereignisströme zur Laufzeit analysiert und ausgewertet. Somit konnten Modelle erlernt werden, um Ereignisse aus Ereignisströmen zu prädizieren.

2012 wurden für die Laufzeitanalyse von Positionsdatenströmen mehrerer Methoden entwickelt, um Ereignisse mit möglichst geringer Latenz detektieren zu können. Hierbei können einzelne Teilereignisse durch sog. Ereignisdetektoren dazu verwendet werden, höhere Zusammenhänge in den Daten hierarchisch zusammenzusetzen. Hierdurch wird die Komplexität der einzelnen Detektionskomponenten drastisch reduziert. Diese werden somit wartbarer und durch die Ausnutzung paralleler und verteilter Rechnerstrukturen wesentlich effizienter. Es ist nun möglich, Ereignisse in den Positionsdatenströmen innerhalb von nur einigen hundert Millisekunden zu erkennen.

2013 wurde die Verzögerung v.a. verteilter Ereignisverarbeitungssysteme weiter minimiert und ein spezielles Migrationsverfahren entwickelt, das die Verteilung von Softwarekomponenten im laufenden Betrieb so modifizt, dass möglichst wenig zeitliche Verluste durch Netzwerkkommunikation auftreten. Des Weiteren wurde ein spekulatives Verfahren puffernder Ereignisverarbeitungssysteme zur Optimierung erarbeitet. Überschüssige Systemressourcen werden effizient verwendet, um die Latenz in Ereignisverarbeitungssystemen auf ein Minimum zu reduzieren. Es wurde außerdem ein repräsentativer Datensatz (mit Sensor- und Positionsdatenströmen sowie manuell eingefügten Ereignissen) sowie eine praxisrelevante Aufgabenstellung veröffentlicht.

2014 wurden grundlegende Verfahren zur Bewältigung von Unsicherheiten (bezüglich der Definition von Ereignisdetektoren) und Unschärfe (im Sinne von Ungenauigkeiten innerhalb von Ereignissen an sich) untersucht. Im Rahmen des Forschungsprojekts wurde ein vielversprechender Ansatz verfolgt, der ohne den üblichen Determinismus ereignisverarbeitender Systeme auskommt und stattdessen parallele Berechnungspfade verfolgt und dabei durch die parallel Betrachtung mehrerer möglicher Zustände eine robustere und genauerere Ereignisverarbeitung erreicht.

Das Projekt stellt einen Beitrag des Lehrstuhls Informatik 2 zum IZ ESI (http://www.esi-anwendungszentrum.de) dar.

2.3 Embedded Systems Institute

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Tobias Werth
Dr.-Ing. Christopher Mutschler
Andreas Kumlehn, M. Sc.
Dr.-Ing. Norbert Oster
Demian Kellermann, M. Sc.
Beginn: 1.9.2007

Das im September 2007 als Interdisziplinäres Zentrum an der Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) gegründete "ESI - Embedded Systems Institute" hat sich die fächerübergreifende Koordination und Organisation der Forschung, Lehre und Weiterbildung im Bereich Eingebetteter Systeme zum Ziel gesetzt.

Über das ESI werden an der Universität vorhandene Kompetenzen mit den Interessen, Aktivitäten und Zielen der einschlägigen Großindustrie und des Mittelstands auf dem Gebiet des Entwurfs Eingebetteter Systeme vernetzt.

Unternehmen erhalten durch das ESI Zugriff auf neueste Forschungsergebnisse sowie die Möglichkeit, gemeinsam Entwicklungsprojekte durchzuführen, Kontakte zu knüpfen und Kooperationspartner zu finden. Das ESI bündelt die Kompetenzen der Lehrstühle und macht sie für Kooperationsprojekte nutzbar. Aktuelle Forschung lässt sich damit schneller in Produkte umsetzen. Beschleunigt wird auch der Aufbau gemeinsamer Forschung. Schließlich dient das ESI auch als Schnittstelle zum frühzeitigen Zugriff auf Studierende und fachlich qualifizierten Nachwuchs.

Der Lehrstuhl Informatik 2 (Prof. Dr. Michael Philippsen) gehört zu den aktiv beteiligten Gründungsmitgliedern des ESI und führt in diesem Rahmen Forschungsprojekte durch.

Weitere Informationen finden Sie auch unter http://www.esi.uni-erlangen.de sowie http://www.esi-anwendungszentrum.de

2.4 ErLaDeF - Embedded Realtime Language Development Framework

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
PD Dr. Ronald Veldema
Dipl.-Inf. Thorsten Blaß
Dipl.-Math. Jakob Krainz
Dipl.-Inf. Daniel Brinkers
Beginn: 1.1.2012
Kontakt:
PD Dr. Ronald Veldema
Tel.: +49-9131-85-27622
Fax: +49-9131-85-28809
E-Mail: ronald.veldema@fau.de

ErLaDeF ist unsere Testumgebung für die Erforschung neuer Programmiersprachen und Compiler-Techniken. Unser Zielbereich ist die Infrastruktur, die nötig ist, um Programmieren von eingebetteten parallelen Systemen (insbesondere im Echtzeitbereich) zu vereinfachen.

Im Bereich der eingebetteten Systeme und der Echtzeit-Software gibt es feste Grenzen für den Verbrauch an Betriebsmitteln wie Hauptspeicher oder Rechenzeit. Ein typischer Steuerrechner z. B. darf für eine Regelungsaufgabe im Allgemeinen nicht beliebig viel Zeit verbrauchen, um eine Fehlfunktion des gesteuerten Systems zu verhindern. Die Hardware-Entwicklung der letzten Jahre hat dazu geführt, dass vermehrt Mehrkern-Prozessoren in eingebetteten Systemen eingesetzt werden. Um die damit verbundene Leistungssteigerung auszunutzen, ist es daher nötig, die Steuerungs- und Systemsoftware, die auf diesen eingebetteten Systemen laufen soll, ebenfalls zu parallelisieren, ohne dabei die Anforderungen an Echtzeitfähigkeit und Resourcenverbrauch zu verletzen.

Wir erforschen verschiedene Strategien, um diese Parallelisierung von Software in den Griff zu bekommen: Vereinfachung der Fähigkeiten von Programmiersprachen, automatische Parallelisierung, Bibliotheken mit Entwurfsmustern für die parallele Programmierung, tiefgehende Analyse im Übersetzer, Model Checking und Beschleunigung von Übersetzeranalyse bis hin zur interaktiven Nutzung.

Parallelisierung von Programmen zur Laufzeit

Aktuell konzentrieren wir uns bei der automatisierten Parallelisierung auf Laufzeit-Parallelisierung. Das zu parallelisierende Programm wird während seiner Ausführung auf Schleifen untersucht, die parallel ausführbar sind. Unser Ansatz besteht darin, lauf-

zeitintensive Schleifen zunächst in zwei Durchläufen auf ihre Parallelisierbarkeit zu untersuchen. Die Analysedurchläufe werden parallel ausgeführt. Im ersten Analysedurchlauf werden die Adressen aller Schreibzugriffe auf den Hauptspeicher in einer gemeinsam genutzten Datenstruktur gespeichert. Zugriffe auf diese Datenstruktur müssen nicht synchronisiert werden, wenn sichergestellt wird, dass bei konkurrierenden Schreibzugriffen überhaupt ein Wert geschrieben wird. Im zweiten Durchlauf wird für jeden Speicherzugriff (auch lesende!) überprüft, ob eine Datenabhängigkeit zu einem Schreibzugriff besteht. Falls keine Datenabhängigkeiten gefunden werden, wird die Schleife parallel ausgeführt - anderenfalls sequentiell.

Im Jahr 2014 konnten wir das Verfahren so verbessern, dass mit der Ausführung der ersten Schleifeniterationen zusammen mit der Analyse begonnen werden kann. Um das zu ermöglichen, musste die Analyse so verbessert werden, dass sie auch dann noch zuverlässig funktioniert, wenn sich die Daten (durch die gleichzeitige Ausführung) ändern. Durch diese Verbesserung konnten wir den Overhead für nicht-parallelisierbare Schleifen erheblich senken. Außerdem haben wir eine Programmiersprache entwickelt, die diejenigen Schleifen wie den beschriebenen zur Laufzeit parallelisiert, welche der Programmierer als Kandidaten zur Parallelisierung markiert hat. Eine Analyse prüft zuvor, ob die Schleifen illegale Konstrukte enthalten, mit denen der Parallelisierer noch nicht umgehen kann. Ansonsten wird Code erzeugt, der die Schleifen zur Laufzeit inspiziert und dann potenziell parallel ausführt.

Skriptsprache Pylon für eingebettete Systeme

Pylon ist eine statisch getypte Skriptsprache und kommt zwecks einfacher Erlernbarkeit mit wenigen Schlüsselwörtern aus. Ein Großteil der Komplexität (z.B. Typinferenz) ist in den Übersetzer verschoben. Der Programmierer muss sich keine Gedanken über Datentypen machen. Trotzdem bleibt das Programm statisch typisiert. Alleine aus dem Wert einer Zuweisung kann der Übersetzer den gerade benötigten Datentyp ableiten (Duck-Typing). Da die Sprache zudem implizit parallel ist, erfordert die Parallelisierung einer Anwendung kein Expertenwissen.

Beim Entwurf der Sprache wurde darauf geachtet, auf Sprachkonstrukte zu verzichten, welche die Analysierbarkeit erschweren (z.B. Zeiger, Vererbung, Polymorphie usw.), bzw. diese durch leichter analysierbare Alternativen zu ersetzen, die in früheren Projekten erarbeitet wurden. Der Fokus liegt darauf, den Anwender schon bei der Erstellung von robustem Programmcode bestmöglich zu unterstützen und Fehler statisch zu melden, die bei anderen Sprachen erst zu Laufzeit auftreten.

Analyse

Objektorientierte Sprachen bieten im Allgemeinen die Möglichkeit, mittels Konstruktoren dynamisch Objekte zu erstellen. Der hierfür benötigte Speicher wird vom Laufzeitsystem angefordert. Im Unterschied zu Desktop-Systemen ist bei eingebetteten Systemen der Speicher noch sehr limitiert. Eine häufige Verwendung des new-Operators

kann dazu führen, dass zur Laufzeit nicht genügend Speicher zu Verfügung steht und das Programm unerwartet beendet wird.

In diesem Berichtszeitraum wurde daher eine Analyse entwickelt, um dieses Problem bereits zur Entwicklungszeit zu erkennen und an den Entwickler rückzumelden.

Dazu ermittelt die Analyse die Lebensspanne einer Referenz auf ein Objekt. Existiert keine Referenz mehr auf dieses Objekt, kann das Objekt aus dem Speicher entfernt werden. Wir setzen hierzu das aus der automatischen Speicherbereinigung als Reference Counting (RC) bekannte Verfahren statisch und interaktiv während der Entwicklung eines Programmes ein, um Fehler zum frühestmöglichen Zeitpunkt zu erkennen. Wurde ein Objekt detektiert, das keine Referenzen mehr hat, muss der Programmierer es durch gezieltes Einfügen von Anweisungen löschen (z.B. delete) oder wiederverwenden. Damit kann der Speicherverbrauch einer Anwendung bereits zur Entwicklungszeit abgeschätzt werden. Die weitgehend sprachunabhängige Analyse basiert auf Pylon und dem im Vorjahr entwickelten parallelen Propagationsframework für Prädikate.

Im Jahr 2014 haben wir auch unser Werkzeug zur interaktiven Programmanalyse weiterentwickelt, um größere Code-Bestände zu analysieren und die Analyseergebnisse für spätere Verwendung aufzubewahren. Dadurch wird auch Code präzise analysierbar, der (vorab analysierte) größere Bibliotheken benutzt.

Das Projekt stellt einen Beitrag des Lehrstuhls Informatik 2 zum IZ ESI dar, siehe auch http://www.esi.uni-erlangen.de .

2.5 Graphen und Graphtransformationen

Projektleitung:
Prof. em. Dr. Hans Jürgen Schneider
Beginn: 1.10.2004
Kontakt:
Prof. em. Dr. Hans Jürgen Schneider
Tel.: +49-9131-85-27620
Fax: +49-9131-85-28809
E-Mail: hans.juergen.schneider@fau.de

Graphen werden an vielen Stellen als intuitives Hilfsmittel zur Verdeutlichung komplizierter Sachverhalte verwendet. Außerhalb der Informatik trifft dies z.B. auf die Chemie zu, wo Moleküle graphisch modelliert werden. Innerhalb der Informatik werden beispielsweise Daten- bzw. Kontrollflussdiagramme, Entity-Relationship-Diagramme oder Petri-Netze zur Visualisierung sowohl von Software- als auch von Hardware-Architekturen verwendet. Graphgrammatiken und Graphtransformationen kombinieren

Ideen aus den Bereichen Graphentheorie, Algebra, Logik und Kategorientheorie, um Veränderungen an Graphen formal zu beschreiben.

Die Kategorientheorie ist ein attraktives Hilfsmittel, äußerst unterschiedliche Strukturen in einer einheitlichen Weise zu beschreiben, z.B. die unterschiedlichen Modelle für asynchrone Prozesse: Petri-Netze basieren auf gewöhnlichen markierten Graphen, Statecharts verwenden hierarchische Graphen, die parallele logische Programmierung kann mit Hilfe sogenannter Dschungel graphentheoretisch interpretiert werden, und die Aktorsysteme lassen sich als Graphen darstellen, deren Markierungsalphabet eine Menge von Termgraphen ist.

In letzter Zeit haben wir uns auf einen theoretischen Aspekt konzentriert.

Unsere Arbeiten zu den Graphtransformationen basieren auf Konzepten der Kategorientheorie. Der sogenannte Doppelpushout-Ansatz stellt eine Produktion durch zwei Morphismen dar, die an einem gemeinsamen Interface-Graphen ansetzen. Das eine Pushout fägt die linke Seite der Produktion in den Kontext ein, das andere die rechte Seite. Wenn wir einen Ableitungsschritt tatsächlich konstruieren wollen, müssen wir auf der linken Seite ein Pushout-Komplement finden, was als nachteilig empfunden wird. Raoult hat 1984 vorgeschlagen, die Graphersetzung durch ein einzelnes Pushout zu beschreiben; der Ansatz wurde dann von Loewe ausführlich diskutiert, wobei die Untersuchungen weitgehend auf injektive Morphismen beschränkt blieben. Unter dieser Voraussetzung sind die Ansätze äquivalent. Jedoch führen einige Anwendungen, z.B. die Termersetzung, zu nichtinjektiven Morphismen. Wir haben diese Fälle detailliert untersucht und konnten zeigen, dass sie auch in diesem Fall äquivalent sind, solange die Einbettung vernünftige Eigenschaften hat.

2.6 International Collegiate Programming Contest an der FAU

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Tobias Werth
Dipl.-Inf. Daniel Brinkers
Dipl.-Math. Jakob Krainz
Beginn: 1.11.2002
Kontakt:
Dipl.-Inf. Tobias Werth
Tel.: +49-9131-85-28865
Fax: +49-9131-85-28809
E-Mail: tobias.werth@fau.de

Die Association for Computing Machinery (ACM) richtet seit Jahrzehnten den International Collegiate Programming Contest (ICPC) aus. Dabei sollen Teams aus je drei Studenten in fünf Stunden neun bis elf Programmieraufgaben lösen. Als Erschwernis kommt hinzu, dass nur ein Computer pro Gruppe zur Verfügung steht. Die Aufgaben erfordern solide Kenntnisse von Algorithmen aus allen Gebieten der Informatik und Mathematik, wie z.B. Graphen, Kombinatorik, Zeichenketten, Algebra und Geometrie.

Der ICPC wird jedes Jahr in drei Stufen ausgetragen. Zuerst werden innerhalb der Universitäten in lokalen Ausscheidungen die maximal drei Teams bestimmt, die dann zu den regionalen Wettbewerben entsandt werden. Erlangen liegt seit dem Jahr 2009 im Einzugsbereich des Northwestern European Regional Contest (NWERC), an dem u.a. auch Teams aus Großbritannien, den Benelux-Staaten und Skandinavien teilnehmen. Die Sieger aller regionalen Wettbewerbe der Welt (und einige Zweitplatzierte) erreichen die World Finals, die im Frühjahr des jeweils darauffolgenden Jahres (2015 in Marrakesh, Marokko) stattfinden.

Im Jahr 2014 fanden zwei lokale Wettbewerbe an der FAU statt. Im Wintersemester wurde ein Mannschaftswettbewerb ausgetragen mit dem Ziel, neue Studierende für die Wettbewerbe zu begeistern - es meldeten sich 24 Erlanger Teams an. Jedes Team bestand aus maximal drei Studenten. Außerdem nahmen noch 35 Teams von anderen europäischen Universitäten teil. Im Sommersemester fand vor dem Wettbewerb zum wiederholten Mal das Hauptseminar ”Hallo Welt! - Programmieren für Fortgeschrittene” statt, um Studierende verschiedener Fachrichtungen in Algorithmen und Wettbewerbsaufgaben zu schulen. Der Wettbewerb im Sommersemester diente danach der Auswahl der studentischen Vertreter der FAU für den NWERC 2014. Insgesamt nahmen an dem deutschlandweit organisierten Ausscheidungskampf 19 Teams der FAU mit Studenten verschiedensten Fachrichtungen teil. Aus den besten Teams wurden neun Studenten ausgewählt, die für den NWERC Dreierteams bildeten.

Das beste Team der FAU löste beim nordwesteuropäischen Wettbewerb NWERC 2014 in Linköping punktgleich mit dem Siegerteam aus Dänemark neun Aufgaben und erreichte damit den zweiten Platz. Somit stellt die FAU das einzige deutsche Team bei den Weltmeisterschaften im Mai 2015 in Marokko. Sowohl das zweite als auch das dritte Erlanger Team konnte in Linköping sechs Aufgaben lösen und landete auf dem 17. bzw. 24. Platz von 96 Teams.

2.7 OpenMP/Java

Projektleitung:
Prof. Dr. Michael Philippsen

Beteiligte:
PD Dr. Ronald Veldema
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Laufzeit: 1.10.2009–1.10.2015

JaMP ist eine Implementierung des bekannten OpenMP Standard für Java. JaMP erlaubt es (unter anderem) Schleifen zu parallelisieren ohne sich mit der low-level Thread API von Java befassen.zu müssen Eine Parallele Schleife hätte in JaMP folgende Form:

```
class Test {
...void foo(){
.....//#omp parallel for
.....for (int i=0;i<N;i++) {
.........a[i]= b[i]+ c[i]
.....}
...}
}
```

JaMP implementiert im Moment die Funktionalität von OpenMP 2.0 und Teile der Spezifikation 3.0 (z.B. die collapse clause). Die aktuelle JaMP Version erzeugt reinen Java Code und ist auf jeder JVM (die Java 1.5+ unterstützt) lauffähig. Die neueste Version kann sogar CUDA fähige Hardware zur Ausführung von Schleifen verwenden, wenn der Schleifenrumpf eine Transformation nach CUDA möglich macht. Ist die Transformation nicht möglich, wird nebenläufiger Code für gängige Multicore Prozessoren erzeugt. JaMP unterstütz auch die gleichzeitige Nutzung von mehreren Maschinen und Acceleratoren. Dieses wurde durch die Entwicklung von zwei Abstraktionsbibliotheken ermöglicht. Die untere Abstraktionsschicht bietet abstrakte Recheneinheiten, die von den eigentlichen Berechnungseinheiten wie CPUs und GPUs und ihrem Ort in einem Rechnerbündel abstrahieren. Eine weitere Abstraktionsschicht baut auf dieser Schicht auf und bietet Operationen zur Verwaltung partitionierter und replizierter Arrays. Ein partitioniertes Array wird dabei automatisch über die abstrakten Berechnungseinheiten verteilt, wobei die Geschwindigkeiten der einzelnen Berechnungseinheiten berücksichtigt werden. Welcher abstrakte Array-Typ für ein Array in einem Java-Programm konkret eingesetzt wird, wird vom JaMP-Übersetzer bestimmt, der erweitert wurde, um ein Programm entsprechend zu analysieren.

Im Jahr 2014 wurde eine JaMP-Version für Android 4.0 entwickelt, die bisher nur das SIMD-Konstrukt von OpenMP unterstützt.

2.8 PATESIA - Parallelisierungstechniken für eingebettete Systeme in der Automatisierungstechnik

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Marius Kamp, M. Sc.
Beginn: 1.6.2009
Förderer:
ESI-Anwendungszentrum
Kontakt:
Dipl.-Inf. Georg Dotzler
Tel.: +49-9131-85-27624
Fax: +49-9131-85-28809
E-Mail: georg.dotzler@fau.de

Dieses im Jahr 2009 gestartete Projekt befasst sich mit der Refaktorisierung und Parallelisierung von Anwendungen aus der Automatisierungstechnik. Die Programme laufen dabei auf speziellen eingebetteten Systemen. Diese Hardware bildet einen Industriestandard und kommt weltweit zum Einsatz. Da auch in eingebetteten Systemen zunehmend Multicore-Architekturen eingesetzt werden, muss bestehende, sequentielle Software für diese neuen Architekturen parallelisiert werden, um einen Zuwachs an Leistung zu gewinnen. Da die Programme typischerweise in der Industrie zur Regelung von Prozessen und zur Fertigungsautomatisierung eingesetzt werden, haben sie einen langen Lebenszyklus, um die Investitionskosten für die Unternehmen gering zu halten. Aufgrund der langen Einsatzzeit der Programme werden diese oftmals nicht mehr durch die ursprünglichen Entwickler gepflegt. Weiterhin wurde für die Programme oftmals ein hoher Aufwand betrieben, um eine zuverlässige Funktionsweise zu gewährleisten. Erweiterungen an der Software werden daher nur zögerlich vorgenommen.

Eine Migration dieser Altanwendungen auf neue Systeme und ihre anschließende Parallelisierung kann daher nicht von Hand vorgenommen werden, da sich dies als zu fehlerträchtig erweist. Es sind also Werkzeuge nötig, die diese Aufgaben automatisch vornehmen bzw. den Entwickler bei der Migration und Parallelisierung unterstützen.

Erforschung von Parallelisierungstechniken

Zur Parallelisierung von bestehenden Anwendungen aus der Automatisierungstechnik wurde ein spezieller Übersetzer entwickelt, der zunächst Programme aus der Automatisierungstechnik auf ihre automatische Parallelisierbarkeit untersuchte. Hierbei zeigte

sich, dass eine automatische Parallelisierung mit existierenden Techniken nicht effizient durchführbar ist. Deshalb konzentriert sich dieses Teilprojekt auf zwei Schritte. Im ersten Schritt wurde eine Datenabhängigkeitsanalyse entwickelt, die potentielle kritische Abschnitte in einem parallelen Programm erkennt und diese dem Entwickler vorschlägt und deren Schutz in den Code einbaut. Die Analyse erlaubt es, atomare Blöcke zu identifiziert, die sehr gut mit den atomaren Blöcken übereinstimmen, die ein Experte im Code platzieren würde. Es wurde außerdem nachgewiesen, dass die Laufzeit nur unmaßgeblich beeinträchtigt wird, falls das Verfahren in seltenen Fällen kritische Abschnitte annimmt, die tatsächlich gar nicht existieren oder kritische Abschnitte ermittelt, die größer als tatsächlich notwendig sind.

Im zweiten Schritt wurden die bestehenden Verfahren Software-Transaktionaler-Speicher (STM) und Lock-Inferenz zur Implementierung atomarer Blöcke verfeinert und verbessert. In dem neuen Ansatz verwendet ein atomarer Block STM, solange Lock-Inferenz keine feingranulare Synchronisation garantieren kann, und wechselt zur Laufzeit von STM zu Lock-Inferenz, sobald feingranulare Synchronisation mit Locks möglich ist. Damit wird jeder atomare Block feingranular synchronisiert, wobei der Laufzeitaufwand von STM minimiert wird. Um die Effektivität des kombinierten Verfahrens zu steigern, wurden bestehende Lock-Inferenz-Algorithmen verbessern, um in mehr Fällen als bisher feingranular synchronisierte atomare Blöcke mit Lock-Inferenz zu implementieren. Es konnte gezeigt werden, dass das kombinierte Verfahren zur Implementierung atomarer Blöcke die Ausführungszeiten gegenüber einer reinen Umsetzung mit STM oder Lock-Inferenz um einen Faktor zwischen 1.1 und 6.3 beschleunigt. Obwohl feingranulare Synchronisation im Allgemeinen zu besserer Leistung als eine grobgranulare Lösung führt, gibt es Fälle, in denen eine grobgranulare Implementierung die gleiche Leistung zeigt. Daher wurde ein Laufzeitmechanismus für STM entwickelt, der ebenfalls mit der kombinierten Technik funktioniert. Dieser Mechanismus startet mit einer kleinen Anzahl an Locks, d.h. mit einer grobgranularen Synchronisierung, bei der Zugriffe auf unterschiedliche gemeinsame Variablen durch die selbe Schlossvariable synchronisiert werden. Falls dieses grobgranulare Locking dazu führt, dass zu viele Zugriffe auf unterschiedliche Variablen auf das selbe Lock warten müssen, dann erhöht die entwickelte Technik graduell die Anzahl an Locks. Dies erhöht die Granularität des Lockings, sodass unabhängige Zugriffe häufiger nebenläufig ausgeführt werden können. Dieser Laufzeitmechanismus zur dynamischen Anpassung der Locking-Granularität führt zu einer bis zu 3.0 mal besseren Laufzeiten als eine statische grobgranulare Lösung.

Das Teilprojekt wurde im Jahr 2014 abgeschlossen.

Erforschung von Migrationstechniken

Unsere Forschung zur Migration von Altanwendungen bestand ursprünglich aus dem Ansatz, veraltete Code-Konstrukte durch ein spezielles Werkzeug automatisch durch

besseren Code ersetzen zu lassen. Die zu ersetzenden Code-Konstrukte und der verbesserte Code wurden dabei von Programmierern in einer speziell entwickelten Beschreibungssprache spezifiziert. Hierbei stellte sich jedoch die Handhabbarkeit dieses Werkzeugs für unerfahrene Entwickler als zu schwierig heraus.

Daher wurde mit der Entwicklung eines neues Werkzeugs begonnen, das zu ersetzende Code-Sequenzen automatisch aus Software-Versionsarchiven erlernt, diese dann generalisiert und Verbesserungen vorschlägt. Der Ansatz basiert darauf, dass zwei Versionen eines Programms miteinander verglichen werden. Unser Werkzeug extrahiert dabei, welche Änderungen sich zwischen den beiden Versionen ergeben haben und leitet daraus generalisierte Muster aus zu ersetzenden Code-Sequenzen und die dazugehörigen Code-Verbesserungen automatisch ab. Diese Muster werden in einer Datenbank gespeichert und können anschließend von unserem Werkzeug dazu verwendet werden, analoge Änderungen für den Quellcode anderer Programme vorzuschlagen.

Im Jahr 2014 wurde ein neues Verfahren zur symbolischen Code-Ausführung entwickelt, das die Anzahl falscher Vorschläge reduziert. Abhängig von der Anzahl und Generalität der Muster in der Datenbank kann es ohne das neue Verfahren zu unpassenden Vorschlägen kommen. Um die unpassenden Vorschläge zu verwerfen, wird das semantische Verhalten des Vorschlags mit dem semantischen Verhalten des Musters aus der Datenbank verglichen. Weichen beide zu sehr voneinander ab, wird der Vorschlag aus der Ergebnismenge entfernt. Die Besonderheit des Verfahrens besteht darin, dass es auf herausgelöste Code-Teilstücke anwendbar ist und keine menschliche Vorkonfiguration benötigt.

Die aktuellen Ergebnisse von unserem Werkzeug SIFE sind hier[1] zu finden (letztes Update: 2014-05-09).

Teile des Projekts werden im Rahmen des ”ESI-Anwendungszentrum”[2] gefördert.

2.9 Softwareleitstand

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Josef Adersberger
Norbert Tausch, M. Eng.
Laufzeit: 1.11.2009–31.12.2015
Förderer:
Bundesministerium für Wirtschaft und Technologie

[1] URL: https://www2.cs.fau.de/staff/dotzler/SIFE_results.tar.gz
[2] URL: http://www.esi-anwendungszentrum.de/

Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

Prototypische Entwicklung eines neuartigen Werkzeugs zur Qualitätsabsicherung bei der Softwareentwicklung.

Moderne Softwaresysteme werden sowohl fachlich, technisch als auch organisatorisch zunehmend komplexer: So steigt die Anzahl und der Vernetzungsgrad der zu realisierenden Anforderungen pro System stetig, die technischen Vorgaben z.B. an den Verteilungsgrad und die Zuverlässigkeit der Systeme werden komplexer und die Softwareentwicklung selbst findet zunehmend in global verteilten Teams und mit wachsendem Zeitdruck statt. Aus diesen Gründen wird es auch zunehmend schwieriger, Softwareentwicklungsprojekte fachlich, technisch und organisatorisch zu steuern.

Als Softwareleitstand bezeichnen wir ein Werkzeug, das leitenden Projektrollen wie dem Projektleiter, dem Softwarearchitekten, dem Anforderungsarchitekten und dem Entwicklungsleiter eine hohe Transparenz und damit verbesserte Steuerbarkeit von Softwareentwicklungsprojekten ermöglicht.

Transparenz herrscht dann, wenn sowohl Zusammenhänge zwischen den vielfältigen Erzeugnissen eines Softwareentwicklungsprojekts als auch deren Eigenschaften schnell und gesamtheitlich zugänglich sind und entsprechend dem individuellen Informationsbedarf eines Projektbeteiligten aufbereitet sind.

Der Softwareleitstand ist ein Werkzeug, das den Zugang zu den Zusammenhängen (Traceability) und den Eigenschaften (Metriken) der Erzeugnisse von Softwareentwicklungsprojekten vereinheitlicht. Damit kann die Effizienz von Softwareentwicklungsprojekten maßgeblich gesteigert werden. Es sollen Erzeugnisse des Softwareentwicklungsprojekts (Artefakte) und ihre Zusammenhänge (Relationen), sowie zu den Artefakten zuordenbare Metriken zentral erfasst, integriert und analysiert werden können. Die entsprechenden Analysen werden in Form von Visualisierungen des Artefaktgraphen mitsamt den zugeordneten Metriken und Regelprüfungen durchgeführt.

Das Projekt Softwareleitstand wird in Kooperation des Lehrstuhls mit der QAware GmbH München durchgeführt. Die ersten 20 Projektmonate wurden aus Mitteln des BMWi gefördert.

Die Umsetzung des Softwareleitstands erfolgt dabei in zwei Arbeitssträngen, die auch den beiden Subsystemen des Werkzeugs entsprechen: Der Integration Pipeline, die Traceability Informationen und Metriken aus verschiedensten Werkzeugen der Softwareentwicklung zusammen sammelt sowie dem Analysis Core (Analysekern), der eine gesamtheitliche Auswertung der integrierten Daten ermöglicht.

Die Integration Pipeline wird durch den Projektpartner QAware GmbH entwickelt. Dabei wurde im bisherigen Projektverlauf zunächst eine Modellierungssprache für Traceability Informationen in Kombination mit Metriken (TraceML) definiert. Die Sprache besteht dabei aus einem Meta-Modell sowie einer Modellbibliothek zur einfachen Definition von angepassten Traceability Modellen. Aufbauend auf der TraceML wurde das Integration Pipeline Framework auf Basis des Eclipse Modeling Projekts entwickelt. Dabei wird sowohl das Eclipse Modeling Framework zur Abbildung der Modelle und Metamodelle, als auch die Modeling Workflow Engine zur Modelltransformation und Eclipse CDO als Modell-Repository verwendet. Auf Basis des Integration Pipeline Frameworks wurden dann eine Reihe von gängigen Werkzeugen der Softwareentwicklung wie z.B. Subversion, Eclipse, JIRA, Enterprise Architect und Maven angebunden.

Der Analysekern wird durch den Lehrstuhl entwickelt. Zentrales Thema ist dabei die Konzeption und Realisierung einer domänenspezifischen Sprache für die graph-basierte Traceability-Analyse. Das Ziel dieser Traceability Query Language (TracQL) ist es, den Aufwand zur Umsetzung von Traceability-Analysen zu reduzieren. Dies ist derzeit der Hauptkritikpunkt von Industrie und Wissenschaft, der die Umsetzung der Traceability hemmt. TracQL erleichtert dabei sowohl die Extraktion als auch die Transformation der Traceability-Daten, so dass diese dann mittels kurzer funktional formulierter Graph-Traversierungen analysiert werden können. Die Sprache baut auf der multi-paradigmen Sprache Scala auf und wurde bereits mehrfach in realen Industrieprojekten zur Analyse erfolgreich eingesetzt.

Im Berichtszeitraum erweiterten wir die Modularität der Sprache um sie sowohl strukturell als auch operational anpassbar und erweiterbar zu machen. Dies erhöht nicht nur die Ausdrucksstärke der Sprache, sondern verbessert auch die Wiederverwendbarkeit bereits erstellter Traceability-Analysen.

2.10 Übersetzerunterstützte Parallelisierung für Mehrkern-Architekturen

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Tobias Werth
Beginn: 1.3.2007
Kontakt:
Dipl.-Inf. Tobias Werth
Tel.: +49-9131-85-28865
Fax: +49-9131-85-28809
E-Mail: tobias.werth@fau.de

Die Entwicklung von schnelleren und immer effizienteren Rechnerarchitekturen ist in den letzten Jahren an verschiedene Grenzen gestoßen. Althergebrachte Techniken trugen nicht mehr oder nur noch wenig zur Beschleunigung der Hardware bei. Grundprobleme sind dabei das auseinander driftende Verhältnis der Latenzen von Speicher und CPU und die Abwärme bei steigenden Taktfrequenzen. Als Lösung drängen sich homogene und heterogene Mehr- und Vielkern-Architekturen auf, die durch verringerte Taktfrequenzen und mehrschichtige Speicherhierarchien einen Großteil der genannten Problematik vermeiden. Unter Umständen wird mittels Spezialisierung einzelner Komponenten die Rechenleistung weiter erhöht. Aktuelle Zielarchitekturen sind dabei vor allem Grafikkarten mit Hunderten von Recheneinheiten und der Intel XeonPhi-Prozessor, der auf einem Board 60 Kerne inkl. Hyperthreading zur Verfügung stellt.

Während sich datenparallele Probleme mit Hilfe der neuen Architekturen meist relativ einfach beschleunigen lassen, ist die Umsetzung von auf Arbeitspaketen basierenden (sog. Task-parallelen) Probleme Gegenstand der aktuellen Forschung. Die Schwierigkeit ergibt sich dabei meist durch die Irregularität des entstehenden Task-Baumes und der damit verbundenen unterschiedlichen Ausführungsdauer. Dadurch ergeben sich die folgenden Fragestellungen: Welcher Kern führt welche Arbeitspakete in welcher Reihenfolge aus? Wann werden Arbeitspakete vom aktuellen Rechenknoten auf einen anderen Rechenknoten verschoben? Welche Daten gehören zu einem Arbeitspaket, können mehrere Kerne/Rechnerknoten parallel auf die Daten zugreifen? Wie müssen die Daten von verschiedenen Rechnerknoten wieder zusammengeführt werden? In welcher Form kann der Übersetzer in Zusammenarbeit mit dem Laufzeitsystem selbständig Arbeitspakete erzeugen und verteilen?

Im Jahr 2011 wurde in diesem Projekt das Programmiermodell Cilk für die heterogene CellBE-Architektur (ein PowerPC-Kern (PPU) mit acht SPU ”Koprozessoren” zur Ausführung paralleler Berechnungen) implementiert und erweitert. Die CellBE-Architektur bietet sich aufgrund ihrer enormen Leistung auf einem einzelnen Chip als Zielarchitektur an. Um ein Arbeitspaket in der heterogenen Architektur verschieben zu können, haben wir das Cilk-Programmiermodell um ein weiteres Schlüsselwort ergänzt. Dazu entwickelten wir eine Quell-zu-Quelltext-Transformation, die aus einem Quelltext den entsprechenden Quelltext sowohl für die PPU als auch für die SPU erzeugen kann. Desweiteren wurden die entsprechenden Daten zu den Arbeitspaketen per DMA-Transfer passend in die lokalen Caches der Koprozessoren verschoben und nach Abarbeitung eines Teilbaumes auf den entfernten Koprozessoren der Speicher mit Hilfe eines automatischen Speicherbereinigers wieder freigegeben.

Im Jahr 2012 wurden Grafikkarten (GPUs) als weitere Zielarchitektur ins Auge gefasst, die heutzutage eine weitaus höhere Rechenleistung im Gegensatz zu normalen Prozessoren anbieten. Diese Rechenleistung kann durch die Programmierung mit Cuda

(NVidia) oder OpenCL (AMD) aufgrund ihrer Struktur für datenparallele Probleme relativ einfach abgerufen werden. Weitaus schwieriger ist die Umsetzung Task-paralleler Anwendungen, die im weiteren Verlauf des Projekts untersucht werden sollen. Dazu werden verschiedene Lastausgleichsalgorithmen entworfen, prototypisch umgesetzt und miteinander verglichen werden. Im Jahr 2012 wurde ein erstes Verfahren mit mehrstufigen Warteschlangen und dem Prinzip der Arbeitsspende (work donation) entworfen.

Im Jahr 2013 wurden neben der Weiterentwicklung der Lastausgleichsalgorithmen für die Grafikkarte mit dem Intel XeonPhi-Prozessor die Gruppe der Zielarchitekturen weiter vergrößert. Der XeonPhi-Prozessor stellt mit seiner Vielzahl an Kernen und der großen Registerbreite und der damit verbundenen Vektorisierbarkeit neue Herausforderung an die Algorithmen zur Verteilung der Arbeitspakete. Konkret wurde das Cilk-Verfahren für den XeonPhi erweitert und angepasst, um automatisch während der Quellcode-Transformation Funktionen zu verschmelzen und die Möglichkeiten zur (automatischen) Parallelisierung durch den Intel-Compiler zu erhöhen. Dabei wurden mehrere Analysen implementiert, die nicht nur die Anzahl verschmelzbarer Funktionen erhöhen, sondern darüber hinaus auch das Auseinanderlaufen von Kontrollflusspfaden in den verschmolzenen Funktionen verhindern (bzw. zumindest abfangen).

Im Jahr 2014 wurde die vorhandene Implementierung der Lastausgleichsalgorithmen für den Intel XeonPhi-Prozessor so erweitert, dass die Arbeit auf mehrere XeonPhi-Prozessoren verteilt werden kann. Im Gegensatz zum Arbeitsdiebstahl, der weiterhin auf einer einzelnen Instanz des XeonPhi zur Lastverteilung zwischen den Kernen verwendet wird, wird die Arbeit aktiv an andere Prozessoren abgegeben. Mit einer neu entwickelten Annotation werden zur Abarbeitung des Arbeitspakets notwendige Daten gekennzeichnet und mit verschoben. Die Herausforderung dabei ist das Verschmelzen der Daten an Synchronisationspunkten. Desweiteren wurde begonnen, Cilk in den clang-Übersetzer des LLVM-Frameworks einzubauen, um automatisiert den CUDA-Code zur Ausführung auf Grafikkarten zu erzeugen. Dieser Code soll dabei auf ein leichtgewichtiges Laufzeitsystem zur Ordnung und Ausführung der Arbeitspakete aufsetzen. Dazu werden Analysen implementiert, die Divergenz soweit möglich vermeiden sollen.

2.11 WEMUCS - Methoden und Werkzeuge zur iterativen Entwicklung und Optimierung von Software für eingebettete Multicore-Systeme

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Demian Kellermann, M. Sc.
Dr.-Ing. Norbert Oster
PD Dr. Ronald Veldema

Laufzeit: 15.10.2012–30.11.2014
Förderer:
IuK Bayern
Kontakt:
Prof. Dr. Michael Philippsen
Tel.: +49-9131-85-27625
Fax: +49-9131-85-28809
E-Mail: michael.philippsen@fau.de

Für eingebettete Systeme werden Multicore-Prozessoren immer wichtiger, da diese hohe Rechenleistung bei niedrigem Energieverbrauch ermöglichen. Die Entwicklung von paralleler Software für diese Systeme stellt jedoch neue Herausforderungen für viele Branchen dar, da existierende Software und Werkzeuge nicht für Parallelität entworfen wurden. Die effiziente Entwicklung, die Optimierung und das Testen von Multicore-Software, speziell für eingebettete Systeme mit hohen Zuverlässigkeits- und Zeitanforderungen, sind offene Probleme.

Im Verbundprojekt WEMUCS[3] wurden im 2-jährigen Verlauf des Projekts Techniken für die effiziente und iterative Entwicklung, die Optimierung sowie den Test von Multicore-Software durch neue Werkzeuge und Methoden geschaffen. Hierfür wurden mehrere Technologien und innovative Werkzeuge zur Modellierung, Simulation, Visualisierung, Tracing und zum Test entwickelt und zu einer Werkzeugkette integriert. Diese wurden in Fallstudien in der Automobilbranche, der Telekommunikation und der Automatisierungstechnik evaluiert und iterativ weiter entwickelt.

Während es für klassische Single-Core-Applikationen gut erforschte Methoden zur Erzeugung von Testfällen und bewährte Maße und Hierarchien für deren Überdeckung gibt, sind ebenfalls nötige Methoden für mehrkernfähige Anwendungen noch nicht etabliert. Gerade durch das Zusammenspiel gleichzeitig ablaufender Aktivitäten können sich aber erst Fehlerzustände ergeben, die durch das isolierte Testen jeder einzelnen, beteiligten Aktivität nicht identifiziert werden können. Als Teil des WEMUCS-Projektes (genauer: Arbeitspaket AP3[4]) haben wir ausgehend von einem Fallbeispiel ein generalisierbares Verfahren entwickelt (im Folgenden "Testing-Pipeline" genannt), um derartige Wechselwirkungen von parallelen Aktivitäten systematisch zu lokalisieren und etwaige Auswirkungen auf den Programmablauf durch Testmethoden gezielt zu untersuchen.

Zum Zweck der praktischen Erprobung der im AP3 entwickelten Testing-Pipeline, einschließlich der automatischen Parallelisierung von bislang sequentiellem Code, wurde vom Projektpartner Siemens eine Gepäckförderanlage, wie man sie von Flughäfen kennt, vollständig (inkl. Code) modelliert. Das modellierte Fallbeispiel ist so ge-

[3] URL: http://www.multicore-tools.de/
[4] URL: http://www.multicore-tools.de/de/test.html

staltet, dass man unterschiedlich große Anlagen, d.h. u.a. mit unterschiedlich vielen Förderbändern für Zufuhr bzw. Abtransport, automatisiert generieren kann. Die Hardware der Förderanlage wird dabei durch das Simulationsgerät SIMIT von Siemens emuliert, während die Steuerungssoftware (in der Sprache: AWL) auf einer Software-basierten SPS ausgeführt wird.

Im ersten Schritt der Testing-Pipeline wird der AWL-Code durch ein vom Lehrstuhl für Programmiersysteme im Berichtszeitraum und im Vorjahr entwickeltes Werkzeug in die für Menschen verständlichere Programmiersprache HLL konvertiert und die in AWL noch sequentiell ausgeführte Abschnitte in einem direkt angeschlossenen zweiten Schritt automatisch in parallel ausgeführte HLL-Einheiten überführt. Für eine exemplarische Gepäckförderanlage mit acht Zufuhr- und acht Abtransportförderbändern sowie einem ringförmigen Zwischenband aus mehreren geraden und gewundenen Segmenten hat unser Werkzeug 11.704 Zeilen sequentiellen AWL-Code automatisch in 34 KB parallelen HLL-Code transcodiert.

Im dritten Schritt der Testing-Pipeline wird der HLL-Code analysiert und durch ein weiteres, im Projektverlauf vom Lehrstuhl entwickeltes Werkzeug in ein Testmodell überführt. Dieses Testmodell stellt den interprozeduralen Kontrollfluss der nebenläufigen Unterprogramme zusammen mit potentiellen und für den Test relevanten Thread-Wechseln als hierarchische UML-Aktivitäten dar (derzeit als XMI-Dokument, z.B. für den Import in Enterprise Architect von Sparx Systems). Für die vorangehend beschriebene Gepäckförderanlage hat unser Werkzeug ebenfalls automatisch 103 Aktivitätsdiagramme (mit 1.302 Knoten und 2.581 Kanten) erzeugt.

Nach einem optionalen Schritt, bei dem das Testmodell von einem Tester manuell angepasst werden kann (z.B. um Prioritäten zu ändern oder zusätzliche Prüfschritte einzubauen), kann das Modell dann in die MBTsuite des Projektpartners sepp.med GmbH geladen werden. Dieses umfangreich konfigurierbare Werkzeug dient zur Generierung von Testfällen, die eine möglichst vollständige Überdeckung des Modells anstreben. Für die exemplarische Gepäckförderanlage wurde u.a. auf einem haushaltsüblichen Standard-PC innerhalb von lediglich sechs Minuten eine hochgradig optimierte Testfallmenge generiert, die mit nur 10 Testfällen rund 99

Zusammen mit dem Projektpartner sepp.med GmbH demonstrieren wir im Projekt dass und wie zwei Export-Module für die MBTsuite entwickelt werden können, die die generierten Testfälle einerseits für den menschlichen Tester als Tabellendokument ausgibt, bei dem jeder Testfall ein Tabellenblatt füllt, dessen Spalten und Zeilen anschaulich den nebenläufigen Ablauf und die Thread-Wechsel zeigen, sowie andererseits zur weiteren Verarbeitung als ausführbare Testfallmenge (d.h. hier als Java-Programm für den nun folgenden Schritt). Die ausführbaren Tests bestehen aus mehreren Testfällen, bei dem jeder Testfall aus mehreren Testschritten und jeder Testschritt aus Steueranweisungen besteht, so dass ein Testlauf einer jederzeit eindeutig reproduzierbaren Pro-

grammausführung des zu testenden, nebenläufigen HLL-Codes entspricht. Werden diese Testfälle gestartet, dann schließt sich die Werkzeugkette, indem der HLL-Code in einem von Lehrstuhl entwickelten HLL-Emulator nach der Vorgabe im Testfall kontrolliert ausgeführt und dabei gleichzeitig ein detailliertes Protokoll der Testausführung zum Zweck der Visualisierung generiert wird.

Dieses Protokoll kann schließlich durch ein von der sepp.med GmbH entwickeltes Plug-In für Enterprise Architect wie eine zusätzliche Schicht/Ansicht "über" das Testmodell gelegt werden, so dass der Tester für jeden einzelnen Testfall bzw. für die gesamte Testfallmenge graphisch nachvollziehen kann, welche Abläufe getestet wurden und wo gegebenenfalls ein Fehler aufgetreten ist: Die "graphische Spur" endet in diesem Fall genau an der betroffenen HLL-Anweisung und lässt sich z.B. "rückwärts" untersuchen, um die Ursache des Fehlschlags zu identifizieren.

Das am Fallbeispiel prototypisch realisierte Verfahren stellt damit einen wesentlichen Beitrag zum Testen nebenläufigen Codes auf eingebetteten Systemen dar. Es ist ein Beitrag des Lehrstuhls Informatik 2 zum IZ ESI[5].

3 Publikationen 2014

– Al-Hilank, Samir ; Jung, Martin ; Kips, Detlef ; Husemann, Dirk ; Philippsen, Michael: Using Multi Level-Modeling Techniques for Managing Mapping Information . In: ACM/IEEE (Hrsg.) : Proceedings of the 1st Int. Workshop on Multi-Level Modelling, ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems (1st Int. Workshop on Multi-Level Modelling, ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems Valencia, Spain Sept. 28 - Oct. 3). Aachen : CEUR-WS, 2014, S. 103-112. (CEUR Workshop Proceedings Bd. 1286)

– Brinkers, Daniel ; Philippsen, Michael ; Veldema, Ronald: Simultaneous inspection: Hiding the overhead of inspector-executor style dynamic parallelization . In: Springer (Hrsg.) : Proceedings of the International Workshop on Languages and Compilers for Parallel Computing (LCPC 2014) (International Workshop on Languages and Compilers for Parallel Computing (LCPC 2014) Hillsboro, OR, USA 15.-17.09.2014). 2014, S. -.

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Combining Lock Inference with Lock-Based Software Transactional Memory . In: Cascaval, Calin ; Montesinos, Pablo (Hrsg.) : Proceedings of the 26th International Workshop on

[5] URL: http://www.esi.uni-erlangen.de

Languages and Compilers for Parallel Computing (LCPC 2013) (26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) Santa Clara, California, USA 25.-27.09.2013). Berlin : Springer-Verlag Berlin Heidelberg, 2014, S. 325-341. (Lecture Notes in Computer Science (LNCS) Bd. 8664) - ISBN 978-3-319-09966-8

– Kempf, Stefan: Compiler and Runtime Techniques to Identify and Optimize Atomic Blocks in Parallel Programs . Göttingen : Cuvillier Verlag, 2014. Zugl.: Erlangen, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 2014. - 161 Seiten.

– Lautenschlager, Florian ; Adersberger, Josef ; Kumlehn, Andreas ; Philippsen, Michael: Design for Diagnosability . In: Java Magazin (Software-&-Support-Verlag, Frankfurt am Main) (2014), Nr. 5, S. 44-50, ISSN 1619-795X

– Mutschler, Christopher ; Philippsen, Michael: Adaptive Speculative Processing of Out-of-Order Event Streams . In: ACM Transactions on Internet Technology (TOIT) 14 (2014), Nr. 1, S. 4:1-4:24, ISSN 1557-6051

– Mutschler, Christopher: Latency Minimization of Order-Preserving Distributed Event-Based Systems . München : Verlag Dr. Hut, 2014. Zugl.: Erlangen, Friedrich-Alexander-Universität Erlangen-Nürnberg, Diss., 2014. - 229 Seiten. ISBN 978-3-8439-1472-7

– Mutschler, Christopher ; Loeffler, Christoffer ; Witt, Nicolas ; Edelhäußer, Thorsten ; Philippsen, Michael: Predictive Load Management in Smart Grid Environments (Best Paper Award) . In: ACM (Hrsg.) : Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (8th ACM International Conference on Distributed Event-Based Systems Mumbai, India 26.05. - 29.05.2013). 2014, S. 282-287. - ISBN 978-1-4503-2737-4

– Tausch, Norbert ; Philippsen, Michael: A Modular and Statically Typed Effectful Stack for Custom Graph Traversals . In: Tichy, Matthias ; Westfechtel, Bernhard (Hrsg.) : Proceedings of the 8th International Workshop on Graph-Based Tools (GraBaTs 2014) (8th International Workshop on Graph-Based Tools (GraBaTs 2014) York, UK 25.07.2014). 2014, S. -. (Electronic Communications of the EASST, Nr. 68)

4 Studien- und Abschlussarbeiten

– Bachelor Thesis: Vectorization of recursive parallel programs. Bearbeiter: Patrick Kreutzer (beendet am 31.01.2014); Betreuer: Dipl.-Inf. Tobias Werth; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Entwurf und Realisierung einer Programmierschnittstelle zur Software-Diagnose basierend auf aspekt-orientierten Paradigmen. Bearbeiter: Pascal Wagner (beendet am 04.04.2014); Betreuer: Andreas Kumlehn, M. Sc.; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Abbildung der LLVM-Zwischensprache (Bitcode) auf die Zwischensprache des LS2-Programmanalyse-Frameworks.. Bearbeiter: Andreas Grünwald (beendet am 08.09.2014); Betreuer: Dipl.-Inf. Thorsten Blaß; PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Master Thesis: Entwicklung eines Werkzeugs zum Vergleich von Code-Fragmenten durch symbolische Ausführung. Bearbeiter: Marius Kamp (beendet am 01.10.2014); Betreuer: Dipl.-Inf. Georg Dotzler; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Integration of new SIMD-Features into the Android Dalvik VM. Bearbeiter: Christian Cardello (beendet am 02.10.2014); Betreuer: Dipl.-Inf. Georg Dotzler; PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Entwurf und Umsetzung einer Bibliothek zur automatisierten Analyse von Zeitreihen im Umfeld der Software-Diagnose. Bearbeiter: Marc Rosenbauer (beendet am 30.10.2014); Betreuer: Andreas Kumlehn, M. Sc.; Prof. Dr. Michael Philippsen

– Master Thesis: Optimierungen für ein CUDA-basiertes JavaScript-System. Bearbeiter: Eugen Meissner (beendet am 01.11.2014); Betreuer: PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Realisierung eines Serialisierungsmechanismus für tiefe Modelle. Bearbeiter: Florian Gerdes (beendet am 1.11.2014); Betreuer: Dipl.-Inf. Samir Al-Hilank; Dr.-Ing. Martin Jung; Hon.-Prof. Dr.-Ing. Detlef Kips

– Bachelor Thesis: Synchronisationsfreie Graphstruktur auf der GPU. Bearbeiter: Daniel Wust (beendet am 6.11.2014); Betreuer: Dipl.-Inf. Thorsten Blaß; PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Bachelor Thesis: CudaMPI: Nachrichtenaustausch zwischen Rechnerknoten über Cuda und MPI. Bearbeiter: Matthias Huth (beendet am 10.11.2014); Betreuer: PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Berichtgenerierung für tiefe Modelle. Bearbeiter: Valerie Wiedemann (beendet am 17.11.2014); Betreuer: Dipl.-Inf. Samir Al-Hilank; Dr.-Ing. Martin Jung; Hon.-Prof. Dr.-Ing. Detlef Kips

– Bachelor Thesis: Entwurf und Implementierung einer Lastverteilung auf mehrere XeonPhi-Prozessoren im Cluster. Bearbeiter: Christian Eichler (beendet am 18.11.2014); Betreuer: Dipl.-Inf. Tobias Werth; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Measurement and Analysis of Runtime-Metrics in a Continuous Integration Environment. Bearbeiter: Victor Simon (beendet am 25.11.2014); Betreuer: Andreas Kumlehn, M. Sc.; Prof. Dr. Michael Philippsen

– Master Thesis: Entwurf und Evaluation eines Ansatzes zur approximativen Ereignisverarbeitung auf Sensordatenströmen. Bearbeiter: Christoffer Löffler (beendet am 1.12.2014); Betreuer: Dr.-Ing. Christopher Mutschler; Prof. Dr. Michael Philippsen