Jahresbericht 2013 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 (ab 01.05.2013)
Wiss. Mitarbeiter:
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Daniel Brinkers
Dipl.-Inf. Georg Dotzler
Demian Kellermann, M. Sc. (ab 01.04.2013)
Dipl.-Inf. Stefan Kempf
Dipl.-Math. Jakob Krainz
Andreas Kumlehn, M. Sc.
Dipl.-Inf. Christopher Mutschler
Dr.-Ing. Norbert Oster
Dipl.-Inf. Mykola Protsenko (ab 15.05.2013)
Norbert Tausch, M. Eng.
PD Dr. Ronald Veldema
Dipl.-Inf. Tobias Werth
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
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 oder FPA-Hardware) 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 Latenzzeiten 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 Soft-

ware 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 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 bestehende Software-EKG. Die Software-Blackbox misst minimal-invasiv technische und fachliche Laufzeitdaten des Systems. Das Werkzeug FindPerformanceBugs ermöglicht eine automatisierte Erkennung von Laufzeitanomalien. Das Bindeglied zwischen dem observierten Software-System und der Werkzeugkette bildet die Performance Modeling Language (PML), eine Modellierungssprache für die performance-relevanten Eigenschaften der Systemkomponenten. Die PML wird einerseits zur Vermessung des Systems durch die Software-Blackbox verwendet. Andererseits bildet sie zusätzlich die Informationsgrundlage für die Runtime Anomaly Diagnosis Language (RADL), welche die Beschreibung von Regeln zur Erkennung der Laufzeitanomalien des Software-Systems erlaubt.

2.2 Effiziente Software-Architekturen für verteilte Ereignisverarbeitungssysteme

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Christopher Mutschler
Laufzeit: 15.11.2010–15.5.2014
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. Auf der 7. Intl. Konf. für verteilte ereignisbasierte Systeme (DEBS) wurde außerdem ein repräsentativer Datensatz (mit Sensor- und Positionsdatenströmen sowie manuell eingefügten Ereignissen) sowie eine praxisrelevante Aufgabenstellung veröffentlicht.

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:
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Dipl.-Inf. Tobias Werth
Dipl.-Inf. 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 durch Sprach-Features

Die Vereinfachung der Parallelitätsfähigkeiten der Programmiersprache ermöglicht es, in einer Übersetzeranalyse die existierenden Fehler (insbesondere Nebenläufigkeitsfehler und Verletzungen von Echtzeitbedingungen) zu finden. Diese Vereinfachung bildet gleichzeitig eine Grundlage für die anderen Ansätze, die wir in diesem Projekt verfolgen. Im Jahr 2013 haben wir Alternativen zu Polymorphismus und Vererbung untersucht, die Analysen weiter vereinfachen könnten. Wir haben außerdem alternative Mechanismen zur Fadensynchronisation untersucht, wie u.a. transaktionalen Speicher, implizite Synchronisation oder fadenübergreifende Funktionsaufrufe.

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, laufzeitintensive 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 Spei-

cherzugriff (auch lesende!) überprüft, ob eine Datenabhängigkeit zu einem Schreibzugriff besteht. Falls keine Datenabhängigkeiten gefunden werden, wird die Schleife parallel ausgeführt - anderenfalls sequenziell. Die Analyse kann parallel zu einer modifizierten sequenziellen Ausführung stattfinden.

Im Jahr 2013 konnten wir die Analyse verfeinern, indem wir in der Analyse beachten, dass die Speicherzugriffe der sequenziellen Ausführung der ersten Schleifen vor den parallelen Ausführungen der restlichen Schleifen stattfindet. Dadurch ist in mehr schwierigen Fällen eine Parallelisierung möglich. Zusätzlich sorgt das dafür, dass falls eine Schleife nicht parallel ausgeführt werden kann, unsere Analyse nur geringen Overhead verursacht.

Entwurfsmuster für parallele Programmierung

Eine Bibliothek von Entwurfsmustern der parallelen Programmierung erlaubt es einem Programmierer, bekannte Strategien auszuwählen und bei ihrer Implementierung auf erprobten, effizienten Code zurückzugreifen. Wir erforschen, welche Entwurfsmuster für parallele Programmierung und insbesondere Kommunikation in parallelen Programmen es überhaupt gibt, und wann diese angewendet werden können. Aus unserer bereits über 30 verschiedene Kommunikationsmuster enthaltenden Bibliothek versuchen wir in diesen Projekt für einen gegebenen Anwendungsfall automatisiert das zu seinen Anforderungen passende Entwurfsmuster zu selektieren. In 2013 haben wir diese Bibliothek um NUMA-taugliche Kommunikationskanäle zur Verwendung auf Network-on-Chip Prozessoren erweitert.

Model Checking

Beim Model Checking werden alle Ausführungsmöglichkeiten von Aktivitätsfäden betrachtet, um zu bestimmen, ob ein Nebenläufigkeitsfehler auftreten kann. Die Fehler, die ein Model Checker finden kann, sind u. A. Wettlaufssituationen und Verklemmungen. Ein Model Checker kann garantieren, dass ein Fehler gefunden wird, benötigt aber viel Zeit für die Analyse des Programms. Im Jahr 2013 haben wir weitere Fortschritte bei der Entwicklung eines allgemeinen, sprachunabhängigen Model Checkers gemacht. Wir haben zwei neue Wege entdeckt, wie der Model Checker durch Verkleinerung des Zustandsraums beschleunigt werden kann. Zum Ersten können wir jetzt die in einen Programm existierende Lokalität ausnutzen. Zum Zweiten haben wir das Data-Centric-Synchronization Paradigma übernommen, um grobgranular statt feingranular zu arbeiten.

Interaktive Programmanalyse

Um sicherzustellen, dass Fehler im Programmdesign möglichst schon früh im Entwicklungsprozess gefunden werden, ist es nötig, Fehler möglichst schon während des Editierens des Programms zu finden. Dazu muss die verwendete Analyse so schnell sein, dass interaktiver Einsatz möglich ist. Wir arbeiten an zwei Ansätzen, dies zu verwirklichen.

Unser erster Ansatz basiert darauf, die Probleme der Programmanalyse durch verbesserte Algorithmen zu lösen. Ein wichtiges Hilfsmittel ist dabei eine verzögerte Analyse, bei der ein Programmteil erst dann analysiert wird, wenn die zugehörigen Analyseergebnisse benötigt werden. Die Analyse soll auch inkrementell ablaufen, d.h. bei kleinen Änderungen im Programm-Code soll nur möglichst wenig des Programms neu analysiert werden.

Dazu zerlegen wir ein Programm rekursiv in Teile und berechnen anschließend, welche Auswirkungen und Effekte diese Teile während einer Ausführung des Programms haben; diese Auswirkungen und Effekte speichern wir zusammenfassend in symbolischen Beschreibungen, die dann dazu verwendet werden, um die Fehler beim Zusammenspiel der zugehörigen Teile des Programms zu finden, um die Auswirkungen von größeren Teilen des Programms zu beschreiben und um bei Änderung eines Bestandteils des Programms nicht das komplette Programm neu analysieren zu müssen, sondern stattdessen die symbolischen Beschreibungen aller unveränderten Programmteile wieder zu verwenden.

In 2013 lag der Schwerpunkt der Forschung auf der Erstellung von sowohl präzisen als auch kompakten Datenstrukturen zur Repräsentation der Effekte und Auswirkungen von Programmteilen, sowie der Erarbeitung von effizienten Algorithmen zur Erstellung und Nutzung dieser Datenstrukturen.

Unser zweiter Ansatz basiert darauf, die Programmanalyse selbst zu parallelisieren und dadurch auf interaktive Geschwindigkeit zu beschleunigen. Im Jahr 2013 haben wir begonnen grundlegende Übersetzeranalysen in eine datenparallele Darstellung zu bringen. Dazu ist ein Rahmenprogramm für Prädikatspropagation in der Entwicklung. Diese datenparallelen Algorithmen sind dann einfach auf viele verschiedene Multi-Core Architekturen und GPUs portierbar.

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 der 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 (2014 in Jekaterinenburg, Russland) stattfinden. Im Jahr 2013 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 19 Erlanger Teams an. Jedes Team bestand aus maximal drei Studenten. Außerdem nahmen noch 30 Teams von anderen europäischen Universitäten teil.

Im Sommersemester fand 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 der Auswahl der studentischen Vertreter der FAU für den NWERC 2013. Insgesamt nahmen an dem deutschlandweit organisierten Ausscheidungskampf 14 Teams der FAU mit Studenten verschiedensten Fachrichtungen teil. Aus den besten Teams wurden neun Studenten ausgewählt, die für den NWERC Dreierteams bildeten (ein zehnter Student wurde als Ersatzmann ausgewählt). Das beste Team der FAU löste beim nordwesteuropäischen Wettbewerb NWERC in Delft sieben Aufgaben und erreichte damit eine Bronzemedaille als drittbestes deutsches Team. Sowohl das zweite als auch das dritte Erlanger Team konnte sechs Aufgaben lösen und landete auf dem 20. bzw. 25. Platz von 92 Teams.

2.7 InThreaT - Inter-Thread Testing

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Norbert Oster
Beginn: 1.1.2012
Kontakt:
Dr.-Ing. Norbert Oster
Tel.: +49-9131-85-28995
Fax: +49-9131-85-28809
E-Mail: norbert.oster@fau.de

Zur Beschleunigung von Rechensystemen setzen Prozessor-Hersteller schon lange nicht mehr auf steigende Taktraten - im Gegenteil: Die absolute Taktzahl sinkt, stattdessen werden in Prozessoren immer mehr unabhängige Recheneinheiten (cores) verbaut. Dafür müssen die Entwickler nun umdenken: Sie bekommen ihre Applikationen nur dann performanter (effizienter), wenn sie ihre Programme so modularisieren, dass unabhängige Codeabschnitte nebenläufig ausgeführt werden. Leider sind heutige Systeme schon funktional so komplex geworden, dass selbst die Entwicklung für eine sequentielle Ausführung noch nicht fehlerfrei gelingt - die Parallelisierung auf mehrere Rechenkerne fügt der System-Konzeption eine weitere nicht-funktionale Komplexitätsdimension hinzu. Zwar hat die Forschung auf dem Gebiet der Softwaretechnik eine Vielzahl von Qualitätssicherungsmaßnahmen hervorgebracht, da aber die zunehmende Verbreitung von Mehrkernsystemen noch verhältnismäßig neu ist, fehlen bislang wirksame Verfahren zum Testen nebenläufiger Applikationen.

Das vorliegende Projekt hat zum Ziel, diese Lücke durch Bereitstellung eines automatisierten Testsystems zu schließen. Dazu bedarf es zunächst einer Testkriterienhierarchie, die Überdeckungsmaße speziell für das Konzept der Nebenläufigkeit bereitstellt. Vergleichbar z.B. der Verzweigungsüberdeckung für sequentielle Programme, die die Ausführung jedes Programmzweigs im Test fordert (z.B. die Bedingung eines if-statements sowohl wahr als auch falsch erzwingt - auch dann, wenn es keinen expliziten else-Zweig gibt), muss ein fundiertes Testendekriterium für nebenläufige Applikationen die systematische Ausführung aller relevanten Verschränkungen fordern (z.B. alle Reihenfolge-Kombinationen die auftreten können, wenn zwei Fäden einen gemeinsamen Speicherbereich verändern dürfen). Eine Testkriterienhierarchie schreibt dem Tester zwar vor, welche Eigenschaften seine ”fertige” Testfallmenge vorweisen muss, hilft dem Tester aber nicht bei der Identifikation der einzelnen Testfälle. Dabei genügt es nicht, wie im Falle sequentieller Tests, das Augenmerk auf die Testfälle allein zu richten: Testszenarien für parallele Module müssen zusätzlich Steuerungsinformationen zur gezielten Ausführungskontrolle der TUT (Threads Under Test) enthalten.

Im Jahr 2012 ist ein Framework für Java entstanden, das diese gezielte Ablaufsteuerung für TUT automatisch generiert. Der Tester muss lediglich die Byte-Code-Dateien seiner Applikation bereitstellen, weitere Details wie z.B. Quellcode oder Einschränkungen auf bestimmte Testszenarien kann er optional angeben, er muss sie aber nicht zwangsweise mühsam einpflegen. Der Ansatz verwendet Aspekt-Orientierte Programmierung, um die für typische Nebenläufigkeitsfehler verantwortlichen Speicherzugriffe (Lesen bzw. Schreiben von Variablen) mittels automatisch generierten Advices zu umschließen. Werden die Aspekte ins SUT (System Under Test) eingewoben, dann werden Variablenzugriffe zur Ausführungszeit abgefangen und die ausführenden Threads in der Ablaufsteuerung "geparkt", bis das gewünschte Testszenario erreicht ist, und dann kontrolliert in der gewünschten Reihenfolge zur weiteren Ausführung reaktiviert. Zur Demonstration wurden einige naive Ablaufsteuerungen umgesetzt, die individuelle Variablenzugriffe z.B. gezielt abwechselnd verschiedenen Threads erlauben.

Der 2012 begonnene Prototyp des InThreaT-Framework wurde im Jahr 2013 zum Eclipse-Plugin umgebaut. Damit ist der Ansatz Multi-Projekt-fähig geworden und die erforderliche Funktionalität integriert sich nahtlos in die vertraute Entwicklungsumgebung (IDE) des Entwicklers bzw. des Testers. Darüber hinaus verringert sich für den Tester dadurch auch der Konfigurationsaufwand, der nunmehr ebenfalls intuitiv in der gewohnten IDE erfolgt - z.B. die Auswahl und Persistenz der im Test zu beobachtenden Verschränkungspunkte, welche schließlich Grundlage der automatische Variation der zu testenden Verschränkungen sind. Für die automatische Exploration der relevanten Verschränkungen bedarf es außerdem einer Infrastruktur zur Anreicherung von Testfällen mit Steuerungsinformationen zwecks kontrollierter (Wieder-) Ausführung einzelner Tests. Im Jahr 2013 wurde exemplarisch ein solcher Ansatz für JUnit untersucht und implementiert, bei dem einzelne Testmethoden und/oder ganze Testklassen mit geeigneten Annotationen versehen werden.

2.8 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 Standards 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 verwenden um Schleifen auszuführen, wenn der Schleifenrumpf eine Transformation nach CUDA möglich macht. Ist die Transformation nicht möglich, wird nebenläufiger Code für gängige Multicore Prozessoren erzeugt. JaMP unterstützt 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 um partitionierte und replizierte Arrays zu verwalten. Ein partitioniertes Array wird dabei automatisch über die abstrakten Berechnungseinheiten verteilt, wobei die Geschwindigkeiten der einzelnen Berechnungseinheiten berücksichtigt werden. Welcher abstrakte Arraytyp 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 Berichtszeitraum 2013 haben wir untersucht, wie man die Verwendung von Java-Objekten in parallelem OpenMP-Code optimieren kann. Es hat sich gezeigt, dass wir den Sprachumfang leicht einschränken müssen, indem wir Vererbung bei Objekten verbieten, die im parallelen Code eingesetzt werden. Das stellt sicher, dass zur Laufzeit keine anderen Typen als zur Übersetzungszeit vorliegen. Wir benutzen diese Eigenschaft, um Objekte in Arrays auf direkte Weise einbetten zu können. Dadurch wurde die Kommunikation mit den Recheneinheiten der GPU enorm beschleunigt und auch die Laufzeit auf den Berechnungseinheiten wurde leicht gesenkt.

2.9 PATESIA - Parallelisierungstechniken für eingebettete Systeme in der Automatisierungstechnik

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dipl.-Inf. Stefan Kempf
Dipl.-Inf. Georg Dotzler
Dipl.-Inf. Thorsten Blaß
Beginn: 1.6.2009
Förderer:
ESI-Anwendungszentrum
Kontakt:
Dipl.-Inf. Stefan Kempf
Tel.: +49-9131-85-27624
Fax: +49-9131-85-28809
E-Mail: stefan.kempf@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 wurde der Fokus vorerst daraufhin verschoben, Entwicklern Rückmeldung über schwer parallelisierbare Code-Teile zu geben und ihnen die anschließende Parallelisierung selbst zu überlassen. Die korrekte Synchronisation der parallelen Ablaufstränge wurde jedoch durch den Übersetzer vorgenommen. Als Synchronisationstechniken wurden dabei atomare Blöcke und transaktionaler Speicher gewählt, die innerhalb eine prototypischen Ausführungsumgebung implementiert wurden. Im Jahr 2013 wurde schließlich eine für Programme aus der Automatisierungstechnik wirksame Parallelisierungstechnik entwickelt, die Program-Slicing und Graphfärbealgorithmen verwendet, um aus einem sequentiellen Programm parallel ausführbare Ablaufstränge zu extrahieren. Ein grafisches Werkzeug zeigt die Ergebnisse dieser Analyse an, so dass Entwickler Stellen, an denen eine Parallelisierung nicht möglich ist, leicht erkennen und refaktorisieren können, so dass ein erneuter Durchlauf der Analyse zu besseren Resultaten führt. Da die extrahieren Ablaufstränge mit Hilfe von kritischen Abschnitten synchronisiert werden, wurde zusätzlich eine weitere neuartige Übersetzeranalyse zur Implementierung kritischer Abschnitte entwickelt. Dieses Verfahren kombiniert Software-Transactional-Memory und Lock-Inferenz, die die beiden bisher hauptsächlich genutzten Techniken zur Übersetzung kritischer Abschnitte bilden. Dazu teilt unser Übersetzer kritische Abschnitte in mehrere Code-Sequenzen auf und wählt für jede Sequenz diejenige Technik, die zu einer feingranulareren Synchronisierung führt. Da Lock-Inferenz in der Regel geringere Aufwände zur Laufzeit als STM besitzt, aber oftmals zu grobgranularer Synchronisierung führt, haben wir ersteres Verfahren um Optimierungen ergänzt, die dazu führen, dass mehr kritische Abschnitte feingranular mit Lock-Inferenz implementiert werden können, als es bisher möglich war.

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 erkennt 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 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, ana-

loge Änderungen auf dem Quellcode anderer Programme vorzuschlagen. Im Jahr 2013 wurde diese automatische Mustergenerierung aus Software-Versionsarchiven vervollständigt. Unser Werkzeug extrahiert weiterhin alle Änderungen aus den Software-Archiven. Da die in den Änderungen enthaltene Information viel zu detailreich ist, wurde ein neuer Vorverarbeitungsschritt hinzugefügt, der die relevanten abstrakten Informationen aus den Änderungen extrahiert, um diese für spätere Optimierungen zu nutzen. Danach werden in einem neuen Verarbeitungsschritt alle ermittelten abstrakten Änderungen als Zeichenkette kodiert und mit dem Needleman-Wunsch-Algorithmus untereinander verglichen. Haben zwei verglichene Zeichenketten hohe Ähnlichkeitswerte, bedeutet dies, dass die dazugehörigen im Quellcode vorgenommenen Änderungen annähernd identisch sind und somit zur gleichen Gruppe von Code-Modifikationen gehören. Diese Gruppen werden dann wie bisher klassifiziert und weiterverarbeitet, um abschließend Muster aus zu ersetzenden und verbesserten Code-Sequenzen zu generieren.

Teile des Projekts werden im Rahmen des http://www.esi-anwendungszentrum.de/ gefördert.

2.10 Softwareleitstand

Projektleitung:
Prof. Dr. Michael Philippsen
Beteiligte:
Dr.-Ing. Josef Adersberger
Norbert Tausch, M. Eng.
Laufzeit: 1.11.2009–31.12.2014
Förderer:
Bundesministerium für Wirtschaft und Technologie
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 Software-

entwicklung 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 vielerlei 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 und wurde von 2009 bis 2012 aus Mitteln des BMWi gefördert. Die Projektlaufzeit ist November 2009 bis Dezember 2014. 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 wurde die Ausdrucksstärke der Sprache durch Verbesserung von Modularität und statischer Typisierung erhöht.

2.11 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 auseinanderdriftende Verhältnis der Latenzen von Speicher und CPU und die Abwärme bei steigenden Taktfrequenzen.

Als Lösung drängen sich homogene und heterogene Mehrkern-Architekturen auf, die dem Programmierer enorme Leistung zur Verfügung stellen. Durch verringerte Taktfrequenzen tritt ein Großteil der genannten Problematik nicht auf, die hohe Leistung wird durch Vervielfältigung der Ressourcen erreicht. Somit sind zum Beispiel bei niedrigerem Energieverbrauch mehr Rechenoperationen pro Zeiteinheit möglich. Unter Umständen wird mittels Spezialisierung einzelner Komponenten die Rechenleistung weiter erhöht. Durch eine mehrschichtige Speicherhierarchie mit vielen Zwischenspeichern soll zum Beispiel das Problem der Latenz verkleinert werden.

Aus Mehrkern-Architekturen die volle Leistung abzurufen stellt sich als große Schwierigkeit für den Programmierer heraus. Die hohe Rechenkapazität kann er nur dann erreichen, wenn er Expertenwissen sowohl in der Domäne der Anwendung, als auch für die konkrete Architektur besitzt.

Gegenstand der Forschung sind dabei unter anderem die folgenden Fragestellungen: Welche Unterstützung kann der Übersetzer dem Programmierer beim Entwickeln von Anwendungen für verschiedenen Mehrkern-Architekturen bieten? Wie viel Kontextwissen ist notwendig, damit der Übersetzer sinnvolle Entscheidungen bei der Parallelisierung auf die einzelnen Kerne trifft? Welchen Anteil der zur Verfügung stehenden Rechenkapazität kann der Programmierer mit vertretbarem Aufwand erreichen, ohne Detailwissen über die Eigenheiten der einzelnen Architekturen besitzen zu müssen? Wie müssen geeignete Werkzeuge zum Auffinden von Fehlern und Flaschenhälsen in der Anwendung auf Mehrkern-Architekturen aussehen?

Ziel dieses Projektes ist es, diese Fragen anhand einer eingeschränkten Anwendungsdomäne zu beantworten und mögliche Lösungswege aufzuzeigen. Als Domäne wird das Lattice-Boltzmann-Verfahren herangezogen, das vor allem in der Strömungssimulation angewandt wird. Durch seine Gitterstruktur und eine überschaubare Anzahl an Datenabhängigkeiten zwischen den einzelnen Zellen lässt sich das Verfahren relativ einfach parallelisieren, so dass sich die Forschung auf die oben genannten Fragestellungen konzentrieren kann.

Die heterogene CellBE-Architektur bietet sich aufgrund ihrer enormen Leistung auf einem Chip als Zielarchitektur an. Sie besteht aus einem PowerPC-Kern (PPU) und acht sogenannten Synergistic Processing Units (SPUs), die Berechnungen parallel ausführen können. Das Programmiermodell Cilk für die CellBE-Architektur wurde weiterentwickelt, um eine stabile und effiziente Ausführung auf den SPUs zu ermöglichen. Dabei wurden vor allem Möglichkeiten geschaffen, die das Finden von Fehlern beim Stehlen von Funktionen auf entfernten SPUs erleichtern. Außerdem wurde die Quell-zu-Quellcode-Transformation überarbeitet, um leichter den entsprechenden Code für PPU und SPU erzeugen zu können.

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) Paralleliserung durch den Intel-Compiler zu erhöhen. Dabei wurden mehrere Analysen implementiert, die nicht nur die Anzahl verschmelzbarer Funktionen zu erhöhen, sondern darüber hinaus auch das Auseinanderlaufen von Kontrollflusspfaden in den verschmolzenen Funktionen verhindern (bzw. zumindest abfangen).

2.12 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
Beginn: 15.10.2012
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 werden Techniken für effiziente und iterative Entwicklung, Optimierung sowie Test von Multicore-Software durch neue Werkzeuge und Methoden geschaffen. Hierfür werden mehrere Technologien und innovative Werkzeuge zur Modellierung, Simulation, Visualisierung, Tracing und zum Test entwickelt und zu einer Werkzeugkette integriert. Diese werden in Fallstudien in der Automobilbranche, der Telekommunikation und der Automatisierungstechnik evaluiert und iterativ weiter entwickelt.

Als Teil des WEMUCS-Projektes entwickeln wir neue Modellierungsmöglichkeiten für parallele Abläufe, um die mit der Parallelisierung neu auftretenden Problemstellungen zu untersuchen und während des Testens reproduzierbar zu machen. In diesem Zusammenhang wird auch untersucht, welche Analysen schon zur Übersetzungszeit angewendet werden können, um mögliche Nebenläufigkeiten zu identifizieren und an die Testschnittstelle weiterzugeben.

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

3 Publikationen 2013

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Combining Lock Inference with Lock-Based Software Transactional Memory . In: Springer (Hrsg.) : Proceedings of the 26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) (26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) Santa Clara, California, USA). 2013, S. -.

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Compiler-Guided Identification of Critical Sections in Parallel Code . In: De Bosschere, Koen ; Jhala, Ranjit (Hrsg.) : Proceedings of the 22nd International Conference on Compiler Construction (International Conference on Compiler Construction Italy, Rome 21.-22.03.2013). 2013, S. 204-223. - ISBN 978-3-642-37050-2

– Kempf, Stefan ; Veldema, Ronald ; Philippsen, Michael: Reduktion von False-Sharing in Software-Transactional-Memory . In: GI (Hrsg.) : Proceedings of the 25th Workshop on Parallel Systems and Algorithms (PARS 2013) (25th Workshop on Parallel Systems and Algorithms (PARS 2013) Erlangen, Germany 11.-12.04.2013). 2013, S. 70-79.

– Lautenschlager, Florian: Design for Diagnosability: Wie mache ich Software diagnostizierbar? In: the coaches (Veranst.) : German Testing Day 2013 (German Testing Day 2013 Munich, Germany 12.11.2013). 2013, S. -.

– Loeffler, Christoffer ; Mutschler, Christopher ; Philippsen, Michael: Evolutionary Algorithms that use Runtime Migration of Detector Processes to Reduce Latency in Event-Based Systems . In: IEEE Computer Society (Hrsg.) : Proceedings of the 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013) (2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013) Torino, Italy June 25-27, 2013). 2013, S. 31-38. - ISBN 978-1-4673-6382-2

– Mutschler, Christopher: Apparatus, Method, and Computer Program for Processing Out-of-Order Events . Schutzrecht EP13153525.4 Patentanmeldung (31.01.2013)

– Mutschler, Christopher ; Philippsen, Michael: Distributed Low-Latency Out-of-Order Event Processing for High Data Rate Sensor Streams . In: IEEE Computer Society (Hrsg.) : Proceedings of 27th International Parallel and Distributed Processing Symposium (27th IEEE International Parallel & Distributed Processing Symposium (IPDPS) Boston, Massachusetts, USA May 20-24, 2013). 2013, S. 1133-1144. - ISBN 978-0-7695-4971-2

– Mutschler, Christopher ; Witt, Nicolas ; Philippsen, Michael: Do Event-Based Systems have a Passion for Sports? (Best Poster/Demo Audience Award) . In: ACM (Hrsg.) : Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (7th ACM International Conference on Distributed Event-Based Systems Arlington, Texas, USA 29.06. - 03.07.2013). 2013, S. 331-332. - ISBN 978-1-4503-1758-0

– Mutschler, Christopher ; Philippsen, Michael: Dynamic Low-Latency Distributed Event Processing of Sensor Data Streams . In: Gesellschaft für Informatik e.V. (Hrsg.) : Proceedings of the 25th Workshop on Parallel Systems and Algorithms (PARS 2013), ISSN 0177-0454 (25th Workshop on Parallel Systems and Algorithms (PARS 2013) Erlangen, Germany 11.-12.04.2013). 2013, S. 5-14.

– Mutschler, Christopher ; Philippsen, Michael: Reliable Speculative Processing of Out-of-Order Event Streams in Generic Publish/Subscribe Middlewares . In: ACM (Hrsg.) : Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (7th ACM International Conference on Distributed Event-Based Systems Arlington, Texas, USA 29.06. - 03.07.2013). 2013, S. 147-158. - ISBN 978-1-4503-1758-0

– Mutschler, Christopher ; Philippsen, Michael: Runtime Migration of Stateful Event Detectors with Low-Latency Ordering Constraints . In: IEEE (Hrsg.) : Proceedings of the 2013 IEEE International Conference on Pervasive Computing and Communications Workshops (9th International Workshop on Sensor Networks

and Systems for Pervasive Computing San Diego, CA, USA 18.-22.03.2013). 2013, S. 609-614. - ISBN 978-1-4673-5075-4

– Mutschler, Christopher ; Ziekow, Holger ; Jerzak, Zbigniew: The DEBS 2013 Grand Challenge . In: ACM (Hrsg.) : Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems (7th ACM International Conference on Distributed Event-Based Systems Arlington, Texas, USA 29.06. - 03.07.2013). 2013, S. 289-294. - ISBN 978-1-4503-1758-0

– Otto, Stephan ; Edelhäußer, Thorsten ; Witt, Nicolas ; Völker, Matthias ; Voll, David ; Mutschler, Christopher: Apparatus, Method and Computer Program for Providing a Virtual Boundary . Schutzrecht EP13156961.8 Patentanmeldung (27.02.2013)

– Philippsen, Michael ; Tillmann, Nikolai ; Brinkers, Daniel: Double inspection for run-time loop parallelization . In: Rajopadhye, S. ; Strout, M. Mills (Hrsg.) : Proceedings of the 24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) (24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) Fort Collins, Colorado, USA 08.-10.09.2011). Berlin : Springer-Verlag Berlin Heidelberg, 2013, S. 46-60. (Lecture Notes in Computer Science (LNCS) Bd. 7146) - ISBN 978-3-642-36035-0

– Veldema, Ronald ; Philippsen, Michael: Language and Runtime Techniques for better Model Checking Efficiency of Parallel Programs . In: Springer (Hrsg.) : Proceedings of the 26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013), Poster Session (26th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2013) Santa Clara, California, USA 25.09. - 27.09.2013). 2013, S. -.

– Werth, Tobias ; Schreier, Silvia ; Philippsen, Michael: CellCilk: Extending Cilk for heterogeneous multicore platforms . In: Rajopadhye, S. ; Strout, M. Mills (Hrsg.) : Proceedings of the 24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) (24th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2011) Fort Collins, Colorado, USA 08.-10.09.2011). Berlin : Springer-Verlag Berlin Heidelberg, 2013, S. 91-105. (Lecture Notes in Computer Science (LNCS) Bd. 7146) - ISBN 978-3-642-36035-0

– Wolf, Carolin ; Dotzler, Georg ; Veldema, Ronald ; Philippsen, Michael: Object Support for OpenMP-style Programming of GPU Clusters in Java . In: IEEE Computer Society (Hrsg.) : Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA 2013)

(27th International Conference on Advanced Information Networking and Applications Workshops Barcelona, Spain 25.03.-28.03.2013). 2013, S. 1405-1410. - ISBN 978-1-4673-6239-9

4 Studien- und Abschlussarbeiten

– Master Thesis: Konzeption und prototypische Realisierung eines Integrationsframeworks für Entwicklungswerkzeuge. Bearbeiter: Julia Mailova (beendet am 24.4.2013); Betreuer: Dr.-Ing. Martin Jung; Hon.-Prof. Dr.-Ing. Detlef Kips

– Master Thesis: Entfernung überflüssiger Thread-Synchronisationen in NGAPL. Bearbeiter: Philipp Weissmann (beendet am 26.4.2013); Betreuer: PD Dr. Ronald Veldema; Prof. Dr. Michael Philippsen

– Diplomarbeit: Re-Engineering des Werkzeugs .gEAr zum Eclipse-Plugin für die aktuelle Sprachversion von Java. Bearbeiter: Waldemar Krawtschuk (beendet am 15.08.2013); Betreuer: Dr.-Ing. Norbert Oster; Prof. Dr. Michael Philippsen

– Bachelor Thesis: Evaluation und prototypische Implementierung eines Parsers für Ereignisbeschreibungssprachen in Echtzeitlokalisierungssystemen. Bearbeiter: Cerny Patrick (beendet am 11.11.2013); Betreuer: Dipl.-Inf. Christopher Mutschler; Prof. Dr. Michael Philippsen; Dr.-Ing. Thorsten Edelhäußer

– Bachelor Thesis: Entwicklung einer adaptiven HMI-Schnittstelle zur Erkennung von Gesten auf Basis hochpräziser Echtzeitlokalisierung. Bearbeiter: Dennis Salzner (beendet am 2.12.2013); Betreuer: Dipl.-Inf. Christopher Mutschler; Dr.-Ing. Stephan Otto; Prof. Dr. Michael Philippsen

– Master Thesis: Entwicklung eines Werkzeugs zur Identifizierung vergleichbarer Code-Modifikationen in Software-Archiven. Bearbeiter: Christoph Romstöck (beendet am 3.12.2013); Betreuer: Dipl.-Inf. Georg Dotzler; Prof. Dr. Michael Philippsen

– Studienarbeit: Entwurf und Implementierung eines Visualisierungs- und Explorationskonzeptes für multi-relationale Pseudographen am Beispiel der Software-Traceability. Bearbeiter: Peter Kranz (beendet am 12.12.2013); Betreuer: Norbert Tausch, M. Eng.; Prof. Dr. Michael Philippsen