Jahresbericht 2019 des Lehrstuhls für Informatik 2 (Programmiersysteme)

1 Mitarbeiterinnen und Mitarbeiter

Dipl.-Ing. (FH) Helmut Allendorf (IT-Betreuer, bis 31.03.2019), Michael Baer, M. Sc., Dipl.-Inf. Thorsten Blaß, Tobias Feigl, M. Sc., Hon.-Prof. Dr.-Ing. Bernd Hindel, Florian Jung, M. Sc., Marius Kamp, M. Sc., Hon.-Prof. Dr.-Ing. Detlef Kips, Patrick Kreutzer, M. Sc., Florian Mayer, M. Sc., Dipl.-Inf. Daniela Novac, Dr.-Ing. Norbert Oster, Akad. ORat, Prof. Dr. Michael Philippsen (Ordinarius), Prof. em. Dr. Hans Jürgen Schneider (Emeritus), Bastian Söllmann (IT-Betreuer, ab 01.04.2019), Manfred Uebler (IT-Betreuer, bis 31.03.2019), Margit Zenk (Sekretariat)

Gäste am Lehrstuhl: Dr.-Ing. Josef Adersberger, Veronika Dashuber, M. Sc., Dr.-Ing. Martin Jung (Lehrbeauftragter), Florian Lautenschlager, M. Sc., Dr.-Ing. Christopher Mutschler, Dr.-Ing. Klaudia Dussa-Zieger (Lehrbeauftragte)

2 Überblick

Der Lehrstuhl liefert ingenieurwissenschaftliche Antworten für Software-Ingenieure, die **parallele Software** im industriellen Rahmen für **Multicore-Rechner**, für daraus bestehende verteilte Systeme, sowie für vernetzte eingebettete Systeme entwickeln. Das Team arbeitet **Programm-Code-basiert**, erstellt lauffähige **Prototypen** und **evaluiert** diese quantitativ und qualitativ.
Eckpunkte unserer Forschungsthemen:
(a) Der Lehrstuhl arbeitet an **Programmiermodellen** für **heterogene** Parallelität und erzeugt dafür portablen und effizienten Code für Multicores, GPUs, Acceleratoren, Mobile Geräte, FPGAs u.ä.
(b) Die Arbeitsgruppe Programmiersysteme unterstützt die **Parallelisierung** von Software für Multicore-Rechner. Die dazu entwickelten Werkzeuge analysieren **Code-Repositories** und helfen dem Entwickler bei der **Migration** und **Refaktorisierung**.
(c) Das Team analysiert Programme und implementiert dazu **Code-Analysewerkzeuge**, die schnell, interaktiv, inkrementell und teilweise selbst parallel arbeiten. Sie finden Wettlaufsituationen, konkurrierende Ressourcenzugriffe etc. im Code und zeigen dem Entwickler Verbesserungsvorschläge punktgenau und in der Entwicklungsumgebung an.
(d) Der Lehrstuhl für Programmiersysteme **testet** parallelen Code und **diagnostiziert** Problemursachen. Die dazu vom Team entwickelten Werkzeuge erzeugen Testdaten, finden Ursachen von erratischem Laufzeitverhalten und schützen gegen **Authentizitätsangriffe**.

3 Forschung

AnaCoRe – *Analyse von Code-Repositories*: Bei der Weiterentwicklung von Software führen die Entwickler oftmals sich wiederholende, ähnliche Änderungen durch. Dazu gehört beispielsweise die Behebung von Fehlern in funktional ähnlichen Komponenten oder die Parallelisierung von sequentiellen Programmteilen. Wenn jeder Entwickler die nötigen Änderungen selbst erarbeiten muss, führt dies leicht zu fehlerhaften Programmen, weil weitere zu ändernde Stellen übersehen werden. Wünschenswert wäre stattdessen ein automatisiertes Verfahren, das ähnliche Änderungen erkennt und mit dieser Wissensbasis Software-Entwickler bei weiteren Änderungen unterstützt.

SYFEX ist ein neues Verfahren zur symbolischen Code-Ausführung, welches die Ähnlichkeit des Verhaltens zweier Code-Teilstücke bestimmt. Im Jahr 2019 wurde ein Datensatz semantisch ähnlicher Methoden aus quelloffenen Software-Archiven erstellt und veröffentlicht.

AuDeRace – *Automatische Erkennung von Wettlaufsituationen*: Moderne Software enthält immer mehr Parallelität. Durch diese Nebenläufigkeit treten etliche neue, teils schwer zu lokalisierende Fehler auf, die nicht mehr durch herkömmliche Testverfahren sicher detektiert werden können. Ob ein solcher Fehler auftritt, hängt von dem konkreten Ausführungsplan der Aktivitätsfäden ab. Dieser unterscheidet sich jedoch bei jeder Ausführung und ist auch stark von dem darunterliegenden System abhängig. Somit treten entsprechende Fehler normalerweise nicht bei jedem Testlauf auf, je nach Testsystem sogar niemals.
Das Projekt stellte bis 2019 einen Beitrag des Lehrstuhls Informatik 2 zum IZ ESI (Embedded Systems Initiative, http://www.esi.fau.de/) dar. In diesem Rahmen wurden verschiedene Ansätze zur Verbesserung der Qualität nebenläufiger Software untersucht. Zusammenfassend wurde festgestellt, dass verschiedenartige Verfahren notwendig und realisierbar sind, allerdings meistens die hohen Laufzeiten ein großes Problem darstellen. Über das ESI Projekt hinaus wurde die Verwendung von Mutationstests durch die Entwicklung eines Werkzeugs zur Äquivalenzerkennung und Testfallgeneriung verbessert. Eine entsprechende Publikation wurde eingereicht und angenommen.

AutoCompTest – *Automatisiertes Testen von Übersetzern*: Übersetzer für Programmiersprachen sind äußerst komplexe Anwendungen, an die hohe Korrektheitsanforderungen gestellt werden: Ist ein Übersetzer fehlerhaft, so generiert dieser u.U. fehlerhaften Code oder stürzt bei der Übersetzung mit einer Fehlermeldung ab. Solche Fehler in Übersetzern sind oftmals schwer zu bemerken oder zu umgehen. In der Praxis scheitert das automatisierte Testen von Übersetzern deshalb oftmals daran, dass kein zugeschnittener Programmgenerator verfügbar ist und die Entwicklung eines solchen einen zu hohen Aufwand bedeutet.
Im Jahr 2019 haben wir zusätzliche Features für das Schreiben von Sprachspezifikationen implementiert und die Effizienz unseres im Vorjahr entwickelten Programmgenerierungsverfahrens gesteigert. Durch die beiden Beiträge konnte der Durchsatz unseres Werkzeugs deutlich gesteigert werden. Des Weiteren haben wir mit Hilfe neuer Sprachspezifikationen Fehler in Übersetzern für die Programmiersprachen Lua und SQL aufgedeckt. Die Ergebnisse unserer Arbeit sind in eine Ende 2019 eingereichte (und inzwischen angenommene) wissenschaftliche Publikation eingeflossen. Neben der Arbeit an unserem Verfahren zur Programmgenerierung haben wir außerdem mit der Arbeit an einem Verfahren zur Testfallreduzierung begonnen. Das Verfahren reduziert die Größe eines zufällig generierten Testprogramms, das einen Fehler in einem Übersetzer auslöst, um die Suche nach der Ursache des Fehlers zu vereinfachen.

Holoware – *Kooperative Exploration und Analyse von Software in einer Virtual/Augmented Reality Appliance*: Der Aufwand für das Verstehen von Software umfasst in Entwicklungsprojekten bis zu 30% und in Wartungsprojekten bis zu 80% der Programmieraufwände. Die dreidimensionale Visualisierung von Software steigert das Verständnis der Sachverhalte deutlich, und damit liegt eine Nutzung von Virtual-Reality-Techniken nahe.
Im Jahr 2019 wurde der im Vorjahr erstellte Prototyp um die Darstellung dynamischen Software-Verhaltens ergänzt sowie für die kooperative (Remote-)Nutzung erweitert. Darüber hinaus wurden Commit-Nachrichten zur Anomalieerkennung ausgewertet sowie Aufrufe eines Systems nach Anwendungsfällen geclustered.
Das Papier "Towards Collaborative and Dynamic Software Visualization in VR", das auf der International Conference on Computer Graphics Theory and Applications (VISIGRAPP) 2020 angenommen wurde, zeigt die Wirksamkeit des Prototyps zur Effizienzsteigerung beim Software-Verstehen.

ORKA-HPC – *OpenMP für rekonfigurierbare heterogene Architekturen*: High-Performance Computing (HPC) ist ein wichtiger Bestandteil für die europäische Innovationskapazität und wird auch als ein Baustein bei der Digitalisierung der europäischen Industrie gesehen. Rekonfigurierbare Technologien wie Field Programmable Gate Array (FPGA) Module gewinnen hier wegen ihrer Energieeffizienz, Perfor-

mance und ihrer Flexibilität immer größere Bedeutung.
Im Jahr 2019 wurden folgende wesentlichen Beiträge geleistet: Der source-to-source Übersetzerprototyp wurde um OpenMP-Target-Outlining (incl. Smoke-Tests) erweitert. Der technische Durchstich für den ORKA-HPC-Prototypen (OpenMP-zu-FPGA-Übersetzer) wurde fertiggestellt. Eine Benchmark-Suite für die quantitative Leistungsanalyse von ORKA-HPC wurde definiert und erstellt. Der source-to-source Übersetzerprototyp wurde um das Genom für die genetische Optimierung der High-Level-Synthese durch Einstellen von HLS-Pragmas erweitert. Der TaPaSCo-Composer wurde um ein (optionales) automatisches Einfügen von Hardware-Synchronisationsprimitiven in TaPaSCo-Systeme prototypisch erweitert.

ParCAn – *Parallele Code-Analyse auf einer GPU*: Im Übersetzerbau (und auch an anderen Stellen) gibt es Analyseverfahren, bei denen Informationen solange durch einen Graph propagiert und dabei verändert werden, bis sich das Analyseergebnis als Fixpunkt einstellt. In diesem Projekt entwickeln wir den Programmrahmen ParCAn, in dem verschiedene derartige Verfahren parallel und dadurch schneller auf der Graphikkarte ablaufen können.
Im Jahr 2019 wurde ParCAn an das veränderte Ausführungsmodell neuer NVIDIA GPU-Architekturen angepasst. Mit Einführung der Volta-Architektur können Aktivitäten unabhängig Fortschritt erzielen. Jede Aktivität besitzt seinen eigenen Befehlszähler und Aufrufstapel. Vorher teilten sich Gruppen von Aktivitäten (Warps), den gleichen Befehlszähler und Aufrufstapel. Aktivitäten in einem Warp führten entweder die selbe Anweisung aus oder waren inaktiv (engl.: lock-step execution). Da jetzt Aktivitäten unabhängig voneinander Fortschritt erzielen können, besteht jetzt die Gefahr von Nebenläufigkeitsfehlern innerhalb von Warps.
Der Programmcode von ParCAn wurde nach Programmstellen durchsucht, die durch das geänderte Ausführungsmodell zu Nebenläufigkeitsfehlern führen könnten. Die Anwendung wurde entsprechend angepasst. Somit lässt sich ParCAn auch auf den neusten NVIDIA Architekturen korrekt ausführen.

RuNN – *Rekurrente Neuronale Netze (RNNs) zur echtzeitnahen Bestimmung nichtlinearer Bewegungsmodelle*: Mit wachsender Verfügbarkeit von Information über eine Umgebung (z.B. eine Sporthalle) und über die Objekte darin (z.B. Sportler in der Halle), steigt das Interesse, diese Informationen gewinnbringend zusammenzuführen (sog. Information Fusion) und zu verarbeiten. Zum Beispiel will man physikalisch korrekte Animationen (z.B. in der virtuellen Realität) von komplexen und hochdynamischen Bewegungen (z.B. in Sportsituationen) in Echtzeit rekonstruieren. Das Kernziel des Projekts ist es, zu evaluieren, wie Methoden des maschinellen Lernens zur Beschreibung von komplexen und nichtlinearen Bewegungen eingesetzt werden können. Dabei soll untersucht werden, ob RNNs die Bewegungen eines Objektes physikalisch korrekt beschreiben und bisherige Methoden unterstützen oder ersetzen können.
Im Jahr 2019 stellten wir fest, dass die verwendeten Modelle auch die menschliche Bewegung (menschliches Bewegungsmodell) vorhersagen können und dass die LSTM Modelle zur Laufzeit entweder vollständig autark oder als Stützstellen in Lokalisierungsschätzern (bspw. Pedestrian Dead Reckoning, PDR, Methoden) integriert werden können. Wir konnten auch zeigen, dass Modelle der RNN Familie in der Lage sind, Bewegungen in die Zukunft zu extrapolieren, so dass diese die Latenz der Verarbeitungskette und darüber hinaus kompensieren. Wir verbesserten die Erklärbarkeit, Interpretierbarkeit und Robustheit der hier untersuchten Modelle. Mit Hilfe eines Simulators erzeugten wir physikalisch korrekte Bewegungen, z.B. Positionen von Fußgängern, Fahrradfahrern, Autos und Flugzeugen, und demonstrierten mit diesen Daten, dass RNN Modelle zwischen unterschiedlichen Bewegungstypen interpolieren, fehlende Datenpunkte kompensieren sowie weißes und zufälliges Rauschen als solches interpretieren und Bewegungen in die Zukunft extrapolieren können. Letzteres ermöglicht die Kompensation von verarbeitungsspezifischer Latenz und ermöglicht eine Vorhersage der menschlichen Bewegung aus Funk- und Inertial-Daten in Echtzeit.

SoftWater – *Software-Wasserzeichen*: Unter Software-Wasserzeichnen versteht man das Verstecken von ausgewählten Merkmalen in Programme, um sie entweder zu identifizieren oder zu authentifizieren. Das

ist nützlich im Rahmen der Bekämpfung von Software-Piraterie, aber auch um die rechtmäßige Nutzung von Open-Source Projekten zu überprüfen. Ziel unseres Forschungsprojekts ist es, ein Wasserzeichen-Framework zu entwickeln, dessen Verfahren automatisiert (insbesondere ohne Beitrag der Programmierer zum Einbettungsprozess) beim Übersetzen des Programms Wasserzeichen hinzufügen.
Im Jahr 2019 wurde ein Ansatz auf Basis der LLVM Compiler Infrastruktur untersucht, der mittels konkolischer Ausführung (concolic execution, eine Kombination aus symbolischer und konkreter Ausführung) ein Wasserzeichen in einem ungenutzten Hardwareregister versteckt. Hierzu wurde der LLVM-Registerallokator dahingehend verändert, dass er ein Register für das Wasserzeichen freihält.

GIFzuMINTS – *Grundlagen der Informatik als Fundament eines zukunftsorientierten MINT-Studiums*: Für den Studienerfolg haben sich die in der Studieneingangsphase verorteten Lehrveranstaltungen für viele Studierende als problematische Hürde erwiesen, die letztlich häufig zum Studienabbruch führen kann. Aus diesem Grund widmen wir uns dem Ausbau der Unterstützung von angehenden Studierenden beim Übergang Schule-Hochschule sowie während der Studieneingangsphase.
Im Jahr 2019 endete das Projekt GIFzuMINTS mit einem besonderen Höhepunkt: Am 20.05.2019 besuchten uns der bayerische Staatsminister für Wissenschaft und Kunst, Bernd Sibler, und der stellvertretende Hauptgeschäftsführer der vbw bayme vbm, Dr. Christof Prechtl. Sie zeigten sich vom Stand des Projekts beeindruckt.
Bis zum Projektende wurden die entwickelten und umgesetzten Maßnahmen gründlich evaluiert und als dauerhafte Angebote etabliert. Dabei wurde das Repetitorium Informatik in ein kontinuierliches virtuelles Angebot zum Selbststudium überführt. Das an besonders begabte Studierende gerichtete Angebot der Vorbereitung auf die Teilnahme an internationalen Programmierwettbewerben wurde ausgeweitet und als Vertiefungsmodul eingerichtet.

4 Lehre

Der Lehrstuhl für Programmiersysteme bietet im Wintersemester das Pflichtmodul *Algorithmen und Datenstrukturen (AuD)* und im Sommersemester *Parallele und Funktionale Programmierung (PFP)* an. Da diese Module fakultätsübergreifend auch anderen Studiengängen (insbesondere Wirtschaftsinformatik bzw. International Information Systems, Informations- und Kommunikationstechnik, Mathematik u.v.a.) angeboten werden, erreichten die Hörerzahlen mit 721 (AuD im WS2018/19) bzw. 320 (PFP im SS2019) im Berichtszeitraum erneut nahezu den Rekordwert der Vorsemester, die sich schließlich auch in der hohen Zahl an Prüfungsanmeldungen (668 in AuD bzw. 302 in PFP) niederschlagen. In der Vertiefungsrichtung Programmiersysteme bietet der Lehrstuhl verschiedene Module zu den Themen *Übersetzerbau*, *Clustercomputing* und *Testen von Softwaresystemen* an. Die Seminare *Hallo Welt! für Fortgeschrittene* und *Machine Learning* waren ebenfalls innerhalb kürzester Zeit restlos ausgebucht.
Insgesamt betreute der Lehrstuhl für Programmiersysteme im Berichtsjahr drei Masterarbeiten, ein Masterprojekt und sieben Bachelorarbeiten.

ICPC – *International Collegiate Programming Contest an der FAU*: Seit 1977 wird der International Collegiate Programming Contest (ICPC) ausgetragen. Dabei sollen Teams aus je drei Studierenden ca. 13 Programmieraufgaben lösen. Als Erschwernis kommt hinzu, dass nur ein Computer pro Gruppe zur Verfügung steht.
Im Jahr 2019 fanden zwei lokale Wettbewerbe an der FAU statt. Ein Mannschaftswettbewerb im Wintersemester hat das Ziel, neue Studierende für die Wettbewerbe zu begeistern - es meldeten sich 30 Erlanger Teams an, mit jeweils maximal drei Studierenden. 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 2019 in Eindhoven (NL). An diesem deutschlandweit organisierten Ausscheidungskampf nahmen 27 Teams der FAU mit Studierende verschiedener Fachrichtungen teil.
Aus den besten Teams und unter Berücksichtigung der Altersbegrenzung, wurden neun Studierende ausgewählt, die für den NWERC Dreierteams bildeten. Diese erreichten dann beim NWERC in Eindhoven die Plätze 16, 41 und 52 von insgesamt 123 teilnehmenden Teams.

5 Auszeichnungen und Preise

Die folgende Veröffentlichung wurde mit dem *Best Paper Award* ausgezeichnet:
Thorsten Blaß, Michael Philippsen: GPU-Accelerated Fixpoint Algorithms for Faster Compiler Analyses In *Proceedings of the 28th International Conference on Compiler Construction* 2019. doi:10.1145/3302516.3307352

6 Publikationen

[1] Feigl T., Kram S., Woller P., Siddiqui R.H., Philippsen M., Mutschler C.: A Bidirectional LSTM for Estimating Dynamic Human Velocities from a Single IMU. In *Proceedings of the 10th International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2019. doi:10.1109/IPIN.2019.8911814

[2] Blaß T., Philippsen M.: GPU-Accelerated Fixpoint Algorithms for Faster Compiler Analyses In *Proceedings of the 28th International Conference on Compiler Construction* 2019. doi:10.1145/3302516.3307352

[3] Mayer F., Knaust M., Philippsen M.: OpenMP on FPGAs - A Survey In *OpenMP: Conquering the Full Hardware Spectrum - Proceedings of the 15th International Workshop on OpenMP (IWOMP 2019)* 2019. doi:10.1007/978-3-030-28596-8_7

[4] Kamp M., Kreutzer P., Philippsen M.: SeSaMe: A Data Set of Semantically Similar Java Methods In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR 2019)* 2019. doi:10.1109/MSR.2019.00079

[5] Feigl T., Roth D., Gradl S., Wirth M., Latoschik M.E., Eskofier B., Philippsen M., Mutschler C.: Sick Moves! Motion Parameters as Indicators of Simulator Sickness In *IEEE Transactions on Visualization and Computer Graphics* 2019. doi:10.1109/TVCG.2019.2932224

[6] Blaß T., Philippsen M.: Which Graph Representation to Select for Static Graph-Algorithms on a CUDA-capable GPU In *Proceedings of the 12th Workshop on General Purpose Processing Using GPUs* 2019. doi:10.1145/3300053.3319416