Reproduzierbarkeit

Thomas Ludwig · Beate Geyer

Die Reproduzierbarkeitskrise

Reproduzierbarkeit von Forschungsergebnissen gehört zu den grundlegenden Qualitätskriterien in der Wissenschaft. Auf Betreiben Robert Boyles wurde ab der Mitte des 17. Jahrhunderts erstmals eine Forderung nach Transparenz des wissenschaftlichen Erkenntnisprozesses in die Standards der Royal Society aufgenommen. Seit Beginn der 2010er-Jahre gewinnt dabei der Begriff der Reproduzierbarkeits- oder Replikationskrise zunehmend an Bedeutung. Ihren Ausgangspunkt nahm diese als Methodenschwäche erkannte Problematik in den Sozialwissenschaften. In der Psychologie veröffentlichte Forschungsergebnisse konnten von unabhängigen Wissenschaftlern in Replikationsstudien mehrfach nicht nachvollzogen werden. Die Wissenschaft diskutiert methodische Verbesserungen hierzu unter den Überschriften Open Data und Open Science [7]. Im vorliegenden Schlagwort konzentrieren wir uns auf die computergestützten Wissenschaften und insbesondere auf die computerbasierte Klimaforschung als ein Beispiel für die zu diskutierenden Fragestellungen.

Ebenen der Reproduzierbarkeit

Reproduzierbarkeit findet sich als Konzept auf verschiedenen Abstraktionsebenen und im Zusammenhang mit verschiedenen Methoden der wissenschaftlichen Erkenntnisgewinnung. An oberster Stelle steht die allgemeine Forderung nach der Reproduzierbarkeit wissenschaftlicher Ergebnisse durch Außenstehende. Auf diese Weise kann neuen Erkenntnissen ein höherer Wahrheitswert zugeschrieben werden. Die Deutsche Forschungsgemeinschaft hat in ihrer Denkschrift „Vorschläge zur Sicherung guter wissenschaftlicher Praxis“ [2] in

Empfehlung 7 eine Aussage zur Sicherung und Aufbewahrung von Primärdaten getroffen, die in ihrem Stand vom Jahr 2013 allerdings hinter der aktuellen Debatte hinterherhinkt und die Arbeitsweisen datenintensiver computerbasierter Forschungen nicht adäquat abdeckt. Zu stark wird hier auf empirische Forschungen abgehoben und zu wenig spiegelt sich die Vorgehensweise computerbasierter Forschungen wider. Die DFG stellt fest, dass numerische Rechnungen nur reproduziert werden können, wenn alle wichtigen Schritte nachvollziehbar sind. „Dafür müssen sie aufgezeichnet werden. Jede Veröffentlichung, die auf Experimenten oder numerischen Simulationen beruht, enthält obligatorisch einen Abschnitt „Materialien und Methoden“, der diese Aufzeichnungen so zusammenfasst, dass die Arbeiten an anderem Ort nachvollzogen werden können“ [1]. Die Umsetzung dieser Nachvollziehbarkeit wird hier nicht diskutiert. Im Bereich der computerbasierten Forschung müsste man fordern, dass auch alle erstellten und genutzten Softwarepakete in diese Materialiensammlung einbezogen werden sowie die Darstellungen, wie unter Verwendung der (aufbewahrten) Primärdaten die Ergebnisdaten erzielt wurden. Im Bereich des Hochleistungsrechnens findet seit 2016 im Rahmen der

<https://doi.org/10.1007/s00287-019-01149-2>

© Die Autoren 2019. Dieser Artikel wurde mit Open Access auf Springerlink.com veröffentlicht.

Thomas Ludwig
Deutsches Klimarechenzentrum (DKRZ), Hamburg
E-Mail: ludwig@dkrz.de

Beate Geyer
Helmholtz-Zentrum Geesthach (HZG), Geesthach
E-Mail: beate.geyer@hzg.de

Alle „Aktuellen Schlagwörter“ seit 1988 finden Sie unter:
<http://www.is.informatik.uni-wuerzburg.de/as>

jährlichen *International Conference for High Performance Computing, Networking, Storage and Analysis* ein Wettbewerb unter der Überschrift *SCC Reproducibility Initiative* statt. Studentische Teilnehmer versuchen hierbei, die Ergebnisse eines wissenschaftlichen Artikels zu reproduzieren, zu dem die Autoren alle notwendigen Informationen, Programme und Primärdaten mitgeliefert haben [3]. Auf der Ebene des Rechnersystems selbst bleibt allerdings noch die Frage bestehen, wie reproduzierbar die Ergebnisse einzelner Programmläufe sind. Im Idealfall wünschen wir uns eine bitweise Reproduzierbarkeit. Wir wollen dieses genauer analysieren.

Warum bitweise reproduzierbar?

Der Wunsch und die Forderung bitweiser Reproduzierbarkeit numerischer Berechnungen auf Hochleistungsrechnern entspringt nicht primär der Sorge um die Reproduzierbarkeit der wissenschaftlichen Erkenntnisse. Vielmehr sehen die Wissenschaftler hier handfeste pragmatische Aspekte. Zum einen ist die Fehlersuche in parallelen Programmen ein komplexer Prozess. Bei sequenziellen Programmen lassen wir ein Programm bis zum Auftreten des Fehlerzustandes laufen und wiederholen dann den Programmablauf, um uns an die Fehlerursache heranzutasten. Wir erwarten hierbei, dass sich das Programm deterministisch verhält und bei identischen Eingabedaten einen identischen Ablaufpfad einschlägt, der eine effiziente Fehlersuche ermöglicht. Dies ist bei parallelen Programmen häufig nicht mehr gegeben. Ein paralleles Programm mit deterministischem Ablauf kann bei Vorliegen eines Fehlers entweder abstürzen oder durch Überholvorgänge zu abweichenden, potenziell falschen Ergebnissen kommen. Der erste Fall ist leicht erkennbar, der zweite bedarf aufwendiger Untersuchungen. Er setzt auch voraus, dass das Programm deterministisch programmiert ist, damit Abweichungen im Ergebnis bei identischen Eingaben einen Fehlerzustand erkennen lassen. Hierzu sind von den Wissenschaftlern vielerlei Vorsichtsmaßnahmen bei der Implementierung zu treffen, die sich allesamt effizienzmindernd auswirken. Dies wird in Kauf genommen, um eine weitere Problematik abzudecken: Bei einer Änderung am Code oder der Systemumgebung kann bei einem deterministischen Programm der Einfluss der Änderungen auf das Ergebnis der Berechnungen untersucht

werden. Ist das Programm per se mit nichtdeterministischen Programmkonstrukten versehen, entfällt diese Möglichkeit.

Rechner und Zahldarstellung

Welche Effekte führen nun in einem Programm zu Nichtdeterminismus? Wir betrachten zunächst die Zahldarstellung im Prozessor und die arithmetischen Operationen. Reelle Zahlen werden durch sogenannte Gleitkommazahlen dargestellt. Üblicherweise wird der *IEEE Standard for Binary Floating-Point Arithmetic for microprocessor systems (ANSI/IEEE Std 754-1985)* eingesetzt, der die interne Darstellung und die Verfahren zur Durchführung mathematischer Operationen und der dabei auftretenden Rundungen definiert [6]. Eine Gleitkommazahl für doppelte Genauigkeit ist beispielsweise in 64 Bit abgelegt, wobei 1 Bit für das Vorzeichen der Zahl, 11 Bit für den Exponenten und 52 Bit für die Mantisse genutzt werden. Durch die Begrenztheit der Zahldarstellung im Rechner gilt das Assoziativgesetz nicht mehr. Je nach Reihenfolge der Auswertung der Terme eines Ausdrucks können sich unterschiedliche Resultate ergeben. Betrachten wir den Ausdruck $(-1 + 1) + 2^{-53} \neq -1 + (1 + 2^{-53})$, so ergibt sich für die linke Seite der Ungleichung im Prozessor der Wert 2^{-53} , für die rechte Seite jedoch der Wert 0. Probleme dieser Art treten auf, wenn besonders große mit besonders kleinen Zahlen verknüpft werden. Das Ergebnis ist dann nicht mehr mit 52 Bit Mantisse vollständig darstellbar. Je nach Anwendungsreihenfolge der Operatoren auf die Operanden entstehen also in den niederwertigsten Bits kleine Differenzen, die bei länger laufenden Berechnungen zu größeren Differenzen werden können.

Hochleistungsrechner und Reproduzierbarkeit

In einem Rechner, auf dem ein Programm als sequenzieller Code läuft, darf man im Allgemeinen davon ausgehen, dass bei identischen Eingabewerten eine bitidentische Ausgabe erfolgt. Bei Parallelrechnern werden aber die mathematischen Operationen auf viele verschiedene Prozessoren verteilt, dort parallel abgearbeitet und häufig erfolgt am Ende eine Zusammenführung von Einzelergebnissen zu einem abschließenden Ergebnis. Die bitweise Reproduzierbarkeit des einzelnen Programmablaufs unter ansonsten unveränderten Bedingungen hängt

von der Programmierweise des Wissenschaftlers ab. Mit viel Aufwand ist es möglich, diese bitweise Reproduzierbarkeit zu erzwingen. Wir haben bereits erwähnt, dass sie aus verschiedenen Gründen wünschenswert ist.

Welche Faktoren beschädigen nun die bitweise Reproduzierbarkeit? Allgemein sind es alle Operationen, die zu Zwecken der Laufzeitoptimierung die Reihenfolge der mathematischen Operationen ändern. Ein Beispiel auf Programmebene ist die Integration von dynamischem Lastausgleich. Zur besseren Auslastung aller Prozessoren werden Teile der Berechnung von überlasteten Prozessoren auf weniger ausgelastete verschoben. Die Assoziativität in der Mathematik des Programms wird aber von der Hardware nicht gewährleistet, wie oben gezeigt wurde. Ändert man die Abarbeitungsreihenfolge bei Teilen der Operationen, so kann das neue Ergebnis vom vorherigen geringfügig abweichen. Gelegentlich baut der Programmierer mit Absicht Nichtdeterminismus ein, indem z. B. Teilergebnisse in der Reihenfolge ihres Eintreffens beim koordinierenden Prozess verarbeitet werden. Dies ist effizienter, als eine feste Abarbeitungsreihenfolge zu erzwingen, jedoch verliert man dabei das deterministische Verhalten.

Eine weitere Quelle von Störungen sind Bibliotheken, die der Programmierer zu seinem Programm dazu bindet. Sie gestatten beispielsweise die Bildung von globalen Summen, jedoch ist unbekannt, auf welche Weise sie das im Detail durchführen und ob die Abarbeitungsschritte hierbei deterministisch sind. Auch die Compiler beeinflussen den Determinismus. Verschiedene Optimierungsstufen nehmen unterschiedliche Veränderungen in der Abarbeitungsreihenfolge vor und verändern dadurch das Programmergebnis. Einige der Einflussgrößen sind unter der Kontrolle des Programmierers und er kann wählen zwischen mehr Determinismus und weniger Effizienz in der Rechnernutzung. Andere Komponenten werden vom Systemverwalter beeinflusst, beispielsweise das Austauschen von Laufzeitbibliotheken, Compilern, Betriebssystemkomponenten. Bei wichtigen lang andauernden Klimamodellläufen insistieren die Wissenschaftler auf einer stabilen unveränderten Infrastruktur des Rechnersystems. Sie wollen in der Lage zu sein, Einflüsse des Rechners auf das Modellergebnis von Einflüssen des Experimentierens mit dem Klimamodell zu unterscheiden.

Beispiel Klimaforschung

Anhand zweier Beispiele soll der Einfluss des Compilers auf die Ergebnisse einer Klimasimulation gezeigt werden. Ausgangspunkt ist eine Klimamodellierung mithilfe des Modells ICON, das beim Deutschen Wetterdienst in Offenbach und dem Max-Planck-Institut für Meteorologie in Hamburg entwickelt wird. Es ist das modernste Klimamodell der deutschen Klimaforschungsgemeinschaft und basiert auf einem ikosaederförmigen Berechnungsgitter. Es werden visuelle Darstellungen von globalen Berechnungsergebnissen gezeigt, wobei die Erdkugel als Quadrat dargestellt wird. Die x -Achse entspricht hier der geografischen Länge, die y -Achse der geografischen Breite. Die Farben kodieren Werte der über die z -Achse integrierten Wasserdampfsäule für jeden berechneten Punkt der x - y -Ebene. Die simulierte Zeit beträgt 100 Modelltage, sodass sich kleine Änderungen in Zwischenwerten auf das Endergebnis auswirken können. Zum Einsatz kommen Intel-Prozessoren.

Betrachten wir zunächst die Abb. 1a und b. Sie zeigen die Ergebnisse zweier Programmläufe mit den Optimierungsstufen -O2 und -O3. Die Optimierungsstufe -O2 arbeitet ohne Vektorisierung im Prozessor, -O3 hingegen nutzt Vektorisierung. Zusätzlich werden aggressive Schleifentransformationen zur Effizienzsteigerung aktiviert, wie z. B. Fusion, Block-Unroll-and-Jam, Collapsing-IF-Statements. Beim Vergleich der beiden Endergebnisse auf visueller Basis erkennt man deutliche Unterschiede. Der Wissenschaftler muss entscheiden, ob beide Ergebnisse valide Aussagen im Sinne des modellierten Sachverhaltes darstellen.

Mit der Erhöhung der Optimierungsstufe ist natürlich auch eine Laufzeitverbesserung verbunden: In Stufe -O2 läuft der Code 10:30 min, in Stufe -O3 8:41 min, mithin ca. 17 % schneller. Die beiden Optimierungsstufen weichen zwar bezüglich der Ergebnisse voneinander ab, sind aber selber deterministisch. Jeder Programmlauf mit identischer Eingabe liefert bei -O2 bzw. bei -O3 jeweils ein bitidentisches Ergebnis. Dies ist darauf zurückzuführen, dass die Programmierer dies im Modellcode erzwingen.

Als Kontrast betrachten wir die Abb. 1c-f. Hier kommt die Compiler-Option -O3 -xAVX zum Einsatz, die die *Advanced Vector Extensions* der x86-Prozessoren aktiviert [5]. Zusätzlich verwendet diese Variante als Standard ein dynamisches

Abb. 1 Berechnung des Gewichtes der Wasserdampfsäule in einem globalen Klimamodell mit verschiedenen Compileroptionen. Abb. 1a mit Optimierungsstufe -O2 ohne Vektorisierung, Abb. 1b mit Optimierungsstufe -O3 und Vektorisierung, Abb. 1c-f mit Optimierungsstufe -O3 -xAVX und nicht bitweise reproduzierbarem Berechnungsergebnis

Datenalignment. Damit wird aber die Ausführungsoptimierung abhängig von anderen, zeitgleich stattfindenden Aktivitäten auf dem Prozessor. Der Determinismus der Ausführung geht verloren und die vier Programmläufe resultieren in vier verschiedenen Endergebnissen, die im Sinne der Klimamodellierung valide, aber eben nicht bitidentisch sind. Dass überhaupt unterschiedliche Ergebnisse

entstehen, fällt natürlich erst auf, wenn mehrfach dieselbe Berechnung durchgeführt wird.

Künftige Rechner

Künftige Rechnerarchitekturen werden die Problematik der bitweisen Reproduzierbarkeit weiter verschärfen. Auf dem Weg zu Exascale-Rechnern, die ExaFLOPS an Rechenleistung erbringen wer-

den, wird es unabdingbar sein, jede mögliche Quelle der Programmoptimierung auszunutzen. Dazu gehören Lastausgleich, hochgradig optimierte mathematische Bibliotheken und optimierte Laufzeitbibliotheken zum Nachrichtenaustausch, welche aber auch kollektive mathematische Operationen auf der Gesamtmenge aller beteiligten Prozesse unterstützen. Jede dieser Methoden wird zu einer dynamischen Anpassung der Abarbeitungsreihenfolge der mathematischen Operationen zur Laufzeit führen und damit zu Nichtdeterminismus im Programm. Zusätzlich werden die Rechner an Heterogenität gewinnen, indem GPUs oder FPGAs als Beschleunigerhardware eingesetzt werden. Für jede dieser Komponenten sowie für das Zusammenspiel aller Komponenten muss wiederum geklärt werden, welche Programmiermethodik die Reproduzierbarkeit erhält und welche sie verletzt.

Ein vieldiskutierter Aspekt in der Hardware sind Bitfehler durch Defekte oder kosmische Strahlung. Die Vergrößerung der Hauptspeicher eines Parallelrechners in den Petabyte-Bereich hinein vergrößert die Wahrscheinlichkeit einer solchen Störung. Der Nutzer darf erwarten, dass der Rechnerhersteller durch Mechanismen der Fehlererkennung und Fehlerkorrektur diese Störungen kompensiert. In keinem Fall kann aber eine vollständige Störungsfreiheit erreicht werden. Es wird stets eine gewisse Rate von Störungen nicht korrigiert werden, die sich ggf. auf die Programmkorrektheit auswirkt.

In allen Fällen gilt wieder: Eine nichtdeterministische Störung des Programmergebnisses lässt sich – wenn überhaupt – nur nachweisen, indem man das Programm mehrfach ablaufen lässt. Bei großen Modellcodes verbietet sich das aufgrund der damit verbundenen Kosten.

Ausblick

Dieses Aktuelle Schlagwort analysiert die bitweise Reproduzierbarkeit von Berechnungsergebnissen auf der Ebene des einzelnen Programms auf einem dedizierten Rechnersystem. Sie ist die Basis

für alle darauf aufbauenden Aspekte von Reproduzierbarkeit: Können wir unsere Daten und Programme an Außenstehende in einer Form weitergeben, sodass diese unsere Ergebnisse reproduzieren können? Können wir generell publizierte Forschungsergebnisse über einen definierten Zeitraum hinweg replizieren? Was ist der Nutzen eines solchen Vorgehens? Wie hoch sind die damit verbundenen Kosten? Wer organisiert die Replikationsanalysen veröffentlichter wissenschaftlicher Ergebnisse? Wir stehen einer neuen Wissenschaftsmethodik gegenüber, die sich in den kommenden Jahren weiter etablieren wird, zu der aber auch noch viel strukturierende Forschungsarbeit zu erbringen ist.

Danksagung

Wir bedanken uns bei Herrn Dr. Panagiotis Adamidis, der freundlicherweise die Compiler-Tests durchgeführt und die Visualisierungen der ICON-Ergebnisse zur Verfügung gestellt hat.

Open Access. Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz (<http://creativecommons.org/licenses/by/4.0/deed.de>) veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden.

Literatur

1. http://www.dfg.de/download/pdf/dfg_im_profil/reden_stellungnahmen/download/empfehlung_wiss_praxis_1310.pdf, letzter Zugriff: 27.1.2019
2. http://www.dfg.de/foerderung/grundlagen_rahmenbedingungen/gwp/, letzter Zugriff: 27.1.2019
3. <http://sc16.supercomputing.org/studentssc/scc-reproducibility-initiative-winner/index.html>, letzter Zugriff: 27.1.2019
4. <https://code.mpimet.mpg.de/projects/iconpublic>, letzter Zugriff: 27.1.2019
5. https://de.wikipedia.org/wiki/Advanced_Vector_Extensions, letzter Zugriff: 27.1.2019
6. https://de.wikipedia.org/wiki/IEEE_754, letzter Zugriff: 27.1.2019
7. <https://www.heise.de/newsticker/meldung/Open-Science-Forscher-brechen-aus-dem-Elfenbeinturm-aus-3630380.html>, letzter Zugriff: 27.1.2019