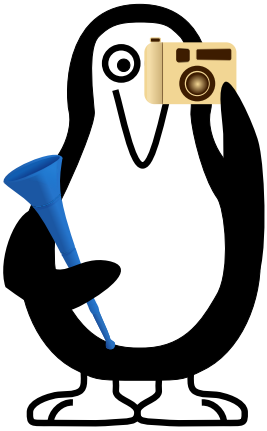
Quartl

3/2017
83. Ausgabe



Inhalt

Editorial	2
Iterationsschleife	5
ExaHyPE - an Exascale Hyperbolic PDE Engine	7
Scaling Workshop 2017: Emergent Applications	15
ISC 2017 PhD Forum	17
SPPEXA News - Workshops im Herbst 2017	19
Notiz*Notiz*Notiz	21

Das Quartl erhalten Sie online unter http://www5.in.tum.de/quartl/

Das Quartl ist das offizielle Mitteilungsblatt des *Kompetenznetzwerks für Technisch-Wissenschaftliches Hoch- und Höchstleistungsrechnen in Bayern* (KONWIHR) und der *Bavarian Graduate School of Computational Engineering* (BGCE)

Editorial

Skandal am Garchinger Forschungscampus!
Am Vormittag des Freitags, 7.7.17, sah sich Kollege B. aus der TUM-Mathematik (Name der Redaktion bekannt) einer äußerst hinterhältigen Attacke aus Fernost ausgesetzt: Eine Gruppe von Schülerinnen und Schülern oder Studierenden aus Korea oder China, so genau war das nicht auszumachen, stürmte während einer seiner Mathematik-Vorlesungen seinen Hörsaal, gaffte auf höchst impertinente Weise auf Dozent sowie Hörerinnen und Hörer und drückte, als Gipfel der Unverschämtheit, auf den einen oder anderen Auslöser. Dieses unerhörte Vorgehen (es wurden schon aus geringerem Anlass Kriege geführt!) brachte den Kollegen B. aus der Ruhe und auf die Palme; denn trotz äußerst energisch vorgetragener Proteste dauert es ein paar Minuten, bis die Horde ein Einsehen hatte und den Saal wieder verließ, sodass sich alle im Saal wieder ungestört an Theoremen ergötzen konnten. Doch der Faden war gerissen – nichts war mehr so wie zuvor, der Schaden war irreparabel. Der Untergang des Abendlandes halt.

Nachmittags erreichte mich dann eine Email des geschätzten Kollegen; formuliert in einer Tonart, wie man normalerweise allenfalls ein Beschwerdeschreiben an eine zahlungsunwillige Versicherung, nicht jedoch eine Grummel-Notiz unter sich seit Jahrzehnten kennenden Kollegen verfasst. Warum ich? Nun, Kollege B. hatte nach Ende seiner so gebeutelten Vorlesung sofort umfangreiche Recherchen veranlasst und schlussendlich erfahren, dass dieser unsägliche Bungartz den Einfall der Mongolinnen und Mongolen autorisiert habe. Un-er-hört! All das bekam ich natürlich auf's Email-Brot geschmiert. Umgehend schaltete auch ich unseren Nachrichtendienst ein. So langsam fügte sich das Bild zusammen. Richtig, morgens hatte mich ein Zuruf über den Flur ereilt, was man besagter Gruppe sagen solle, die unangemeldet aufgeschlagen sei und nun kurz eine Vorlesung besuchen wolle.
Meine Antwort: „Ja mei, sie sollen halt in einen Hörsaal gehen, leise sein, kurz schauen und dann wieder ebenso leise verschwinden." Da sind sie wieder, diese unerträgliche Leichtigkeit des Seins, dieser schlimme Pragmatismus, die Kleinigkeiten, über die man wunderbar erst mal ausgiebig in den verschiedensten Gremien diskutieren könnte, einfach mal kurz regeln.

Damit war das für mich erledigt gewesen. Was für eine krasse Fehleinschätzung! In besagter Email stand neben allerlei Lamentieren („Wir fühlten uns wie Tiere im Zoo!“ – statt dass man sich freut, dass sich mal jemand für einen und das eigene Treiben interessiert) auch konstruktiv eine Liste von Punkten, wie Kollege B. sich wünsche, dass solche Dinge in Zukunft ablaufen sollten – in etwa so: Anmeldung des Besuchs sieben Jahre im Voraus, lustvolles Absitzen der gesamten Vorlesung, absolutes Schweigegelübde, strengstes Fotografierverbot, etc. etc. Was für ein weltfremdes Gedöns. Wie viele Beweise man führen könnte in der Zeit, die für solch dämliche Emails und deren Beantwortung drauf geht – schoss es mir durch den Kopf. Aber die Erregung des Kollegen B. war wohl zu hoch, an Beweise war an diesem Freitag nicht mehr zu denken. Ein irreparabler Schaden halt.

Garniert war das alles übrigens noch mit dem Hinweis, dass es durchaus problematisch sei, wenn der Dekan der Informatik die Störung einer Lehrveranstaltung der Mathematik autorisiere. Was für ein Blödsinn – als ob der Dekan der Informatik nichts Besseres zu tun hätte, als lustvoll und heimtückisch den Lehrbetrieb der Schwesterfakultät (der er zudem auch angehört) zu stören. Der Mathematiker, das empfindsame Wesen.
Übrigens: Kaum ist die Beschwerde des einen über diesen niederträchtigen Anschlag auf die Konzentration verklungen, meldet sich der nächste zu Wort. Und wieder geht es um eine Attacke auf die Konzentration der Wissenschaftler, diesmal durch Studierende, die im Sommer doch tatsächlich auf den draußen aufgestellten Bierbänken sitzen und sich auch mal etwas lauter unterhalten.

Doch auch hier wird sofort organisierte Kriminalität vermutet – angeblich wurden Gruppen mit Animateur-ähnlichen Einpeitschern gesichtet. Alles höchst plausibel, ist doch Garching als Ziel von Pauschalreisenden stark im Kommen, seit am Ballermann neue Gesetze gelten und kaum jemand mehr Lust auf Antalya hat. Na ja, zumindest kam aus dieser Ecke noch nicht der Vorwurf, die Informatik würde absichtlich die Konzentration der Mathematik stören. Das wäre nun auch wirklich absurd, sind doch die Bierbank-nächsten Büros die des Informatik-Dekanats und des Informatik-Dekans. Aber gewundert hätte mich auch so ein Vorwurf nicht. Vielleicht würden manche Mathematiker gerne einen TUM-Campus auf Spitzbergen beziehen?

Da schaut höchstens mal ein Eisbär vorbei, und der fotografiert höchst selten. Und wenn der einen gefressen hat, stört man sich nicht mal mehr selbst. Perfekt. Nachdem die TUM ja eh gerade am Expandieren ist – Straubing, Heilbronn – erscheint Spitzbergen nicht mehr als undenkbar.

Aber wir halten fest: Wer uns besuchen will, ist auch fürderhin willkommen. Und wenn sich unsere Studierenden am Campus wohl fühlen und feiern, ist das doch schön. Nur Leichen verursachen keine Geräusche.

In meiner Antwort-Email an Kollege B. habe ich diesem übrigens für die Steilvorlage für's Quartl herzlich gedankt. Es ist schon rührend, wie sich alle bemühen, dass mir der Stoff für meine Sottisen nicht ausgeht.

Jetzt höre ich aber auf – die gesamte Quartl-Redaktion wünscht Ihnen, liebe Leserinnen und Leser, einen ruhigen und ungestörten Herbst. Zunächst aber wünschen wir viel Spaß mit dieser neuen Ausgabe Ihres Quartls!

Hans-Joachim Bungartz

Iterationsschleife

N=24
14. September 2017

Der Mathematiker Alan Turing [a] gilt als einer der Urväter der Digitalisierung. Seine Turingmaschine gilt als Ausgangspunkt für die Entwicklung moderner Computer. Einem breiten Publikum ist Turing bekannt, weil er im Zweiten Weltkrieg an der Entschlüsselung von deutschen Funksprüchen gearbeitet hat. Zuletzt erschien ein Film[b], der das Leben Alan Turing beschreibt und insbesondere auf seine Arbeit im Zweiten Weltkrieg eingeht. Turing war demnach ein Experte im Verstehen, Entwickeln und Knacken von Codes.

1952 wurde Alan Turing in seiner Heimat wegen seiner Homosexualität strafrechtlich verfolgt und schließlich zu einer Hormonbehandlung gezwungen, die seine Homosexualität „kurieren" sollte. In Folge dieser Behandlung beging Alan Turing 1954 Selbstmord. Der britische Rechtsstaat hat dieses Urteil nie formal aufgehoben oder durch einen Freispruch korrigiert. Stattdessen wurde von Königin Elisabeth II am 24. Dezember 2013 ein royal pardon ausgesprochen[c].

Der Fall zeigt die Bandbreite des Begriffs „Code" exemplarisch auf. Turing war in der Lage technische Codes zu verstehen, zu benutzen und zu knacken. An den sozialen und juristischen Codes seiner Zeit zerbrach er jedoch. Seine Rehabilitation scheiterte schließlich zunächst am demokratischen juristischen Code seines Landes, der jedoch final durch einen Rechtsakt aus dem monarchistischen Code des 17ten Jahrhunderts wieder ausgehebelt wurde. Selten erschien Monarchismus so sympathisch.

Die Bandbreite menschlicher Codes ist im Vergleich zu technischen Codes offenbar größer, und ihre Regeln und Widersprüche sind unübersichtlicher als bei technischen Codes. Franz Kafka[d] hat sich in zweien seiner Romane (Process, Das Schloß) mit diesen dunklen Seiten der menschlichen Codes beschäftigt. Experten[e] meinen, Kafka benutze in seinen Romanen eine mehrfache Verschlüsselung. Doch scheint es dem Leser von Franz Kafka doch eher so, dass Kafka ganz offen über die Dunkelheit der menschlichen Codes spricht, während der Literaturwissenschaftler Licht ins Dunkel der Literatur bringen will - und dabei das Dunkel der beschriebenen Codes mit dem Dunkel einer Verschlüsselung verwechselt.

---

[a] Alan Turing, 23. Juni 1912 – 7. Juni 1954
[b] The Imitation Game, Großbritannien, 2014
[c] http://www.bbc.com/news/technology-25495315
[d] Franz Kafka, 3. Juli 1883 – 3. Juni 1924
[e] Oliver Jahraus, Kafka, reclam, 2006

Das Versprechen der Digitalisierung scheint nun darin zu liegen, dass Computercodes eingesetzt werden können um die Grauzonen der menschlichen Entscheidung ins grelle Licht der klaren Logik zu heben. Der Begriff der künstlichen Intelligenz suggeriert dabei, dass eben nicht unmenschlich und dumm – also maschinell – entschieden wird, sondern sozusagen am menschlichen Begriff der Intelligenz ausgerichtet. Doch Optimisten übersehen dabei, dass Intelligenz ein Begriff aus der Grauzone menschlicher Codes ist. Intelligenz ist keine messbare Eigenschaft, sondern eine kulturhistorisch interessanter Beschreibungsversuch für die menschliche Fähigkeit, Probleme zu lösen. Als solcher Begriff verschleiert er in der Digitalisierung mehr als erhellen kann.

Auch hier hat Alan Turing die Grundlagen gelegt, indem er in der Beschreibung des Turing-Tests[a] davon ausging, dass die Frage, ob ein Computer zumindest sprachlich einen Menschen vortäuschen könne, etwas Wesentliches sei. Dabei ist umgekehrt die Frage doch: kann ein Mensch jemals den Zustand erreichen in dem es ihm gelingt, das Verhalten eines Computers vorzutäuschen?

M. Resch

[a] Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460

Warum es so wichtig ist, bisweilen die Kontrolle zu verlieren

Das EU-Projekt *ExaHyPE—an Exascale Hyperbolic PDE Engine* hat zum Ziel, eine Engine zu bauen, mit der Anwendungsexperten mit hinreichenden Compute-Meriten aber ohne starke HPC-Expertise einen ADER-DG-basierten Löser für Gleichungen der Bauart

$$\frac{\partial}{\partial t}\mathbf{Q} + \nabla \cdot \mathbf{F}(\mathbf{Q}) + \sum_i \mathcal{B}_i \frac{\partial \mathbf{Q}}{\partial x_i} = \mathbf{S} + \sum \delta$$

schreiben können; innerhalb eines Jahres mit Exascale-Potential. Alle Software ist Open-Source mit ausführlicher Dokumentation. Wir versuchen einen möglichst großen potentiellen Nutzerkreis anzusprechen. Das Konsortium selber demonstriert das Potential der Engine anhand eines Seismic-Risk-Szenarios und der Simulation von Gravitationswellen, wie sie von umeinander rotierenden Neutronensternen emittiert werden.

Warum denn nun Engine? Das ist der Informatikerkrankheit geschuldet, für alles und jeden Zweck ein Framework zu schreiben. Man denke an die 100+1 Frameworks zum Lösen der Poisson-Gleichung auf dem Einheitsquadrat, die aber selbstverständlich alle auch verwendet werden können, nahezu alle elliptischen Probleme dieser Welt anzugehen[1]. Solch ein Framework wollen wir nicht. Den Begriff der Engine entlehnen wir in ExaHyPE der Computerspielewelt. Ähnlich einer 3D-Grafik-Engine bauen wir einen, bzgl. der Anwendung, relativ generischen Satz an numerischen Werkzeugen, geben aber das Compute-n-Feel—ähnlich einem gewissen Grafikstil bei Spielen—fest vor. In unserem Fall heißt das: Die Engine benutzt ADER-DG gekoppelt mit Finiten Volumen als Limiter. Die Engine benutzt Code-Generierung, um für die zeitkritischen Kernroutinen von ADER-DG alles Potential aus Intel-basierten Architekturen herauszukitzeln. Die Engine benutzt dynamisch adaptive, kartesische Gitter. Während sich ExaHyPEs zwei

[1] Der Autor hat selber zwei solche Frameworks produziert . . . falls jemand Interesse hat.

Anwendungsszenarien leicht den involvierten Nutzergruppen an LMU und Frankfurts FIAS zuordnen lassen, werden die drei methodischen Kernkompetenzen durch Gruppen aus Trento (Michael Dumbser), der TUM/LRZ (Michael Bader) und aus Durham eingebracht.

Da ein Gitter in jedem gitterbasierten Löser nahezu alle methodischen Bausteine, also Algorithmen plus Daten, integriert, stellt sich bzgl. des Gitters unweigerlich die Frage, wie ein Nutzer mit den algorithmischen Bausteinen und Daten interagiert. Ich sehe im Wesentlichen zwei Möglichkeiten: Das Gitter kann als Art Datenbank fungieren, die alle Informationen sammelt oder zumindest zusammenführt, also indiziert. Die einzelnen Algorithmenschritte sind frei sich zu organisieren. Sie greifen, wann auch immer sie Daten brauchen, halt auf diese Datenbank zu. Im Finite-Volumen-Zusammenhang frage ich die Menge aller Zellen ab, iteriere dann über diese, lese ihre Werte aus, frage das Gitter, was denn adjazente Faces sind, finde deren Flüsse, und so weiter und so fort. Alternativ kann das Gitter aber auch Dreh- und Angelpunkt aller Arbeit sein und selbst die Hoheit über die Operationen und insbesondere Datenzugriffe behalten. Das ist der ExaHyPE-Way—eine Eigenschaft, die wir aus der früheren Arbeit zu unseren Peano-Codes übernommen haben. Das gesamte Gitter ist mit einem Automaten verwoben, der genau eine Aufgabe hat: Laufe über das Gitter und stelle in jedem Schritt, also in jeder Zelle zum Beispiel, sicher, dass alle dem Gitter assoziierten Daten bekannt und im besten Fall effizient verfügbar sind—im Idealfall gleich im Cache. Der User hat dann die Möglichkeit, sich in bestimmte Automatentransitionen ("ich verlasse jetzt eine Zelle und geh rüber zum Nachbarn" oder, im Multiskalenkontext, "ich gehe von einer Zelle zum Vater") einzuhängen, überlässt die Wahl einer sinnvollen, effizienten Traversierungsreihenfolge aber dem Gitter und seinem Automaten.

Ein paar Assoziationen: Der Software-Ingenieur (zumindest der, der wie der Autor mit den ersten Java-Versionen und den epochalen Gang-of-Four aufgewachsen ist) wird so ein Muster mit Visitor-Pattern und Callbacks bezeichnen: der Nutzer schreibt was passiert, falls man Gitterelemente besucht,

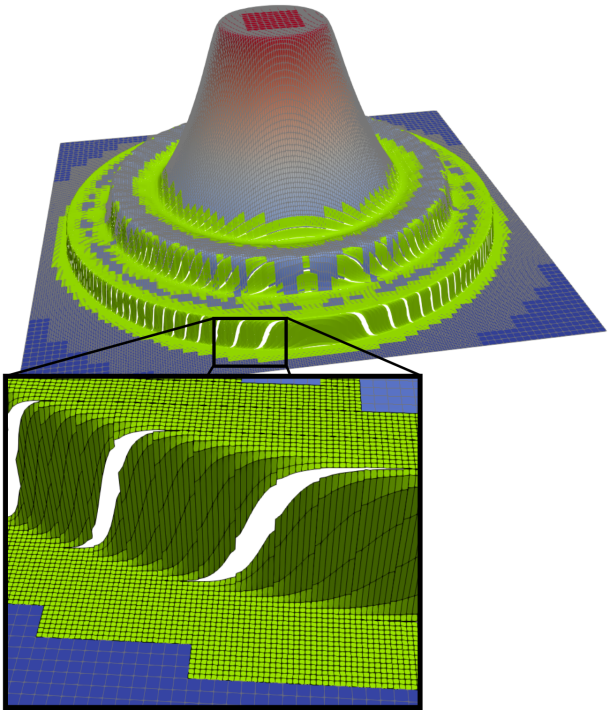


Abbildung 1: Punktexplosion mit den Euler-Gleichungen. Wir sehen deutlich Diskontinuitäten in der Lösung (als Löcher in den Bildern) und das Adaptive Mesh Refinement. Von Dominic E. Charrier (Durham).

mischt sich aber nicht in die Besuchsreihenfolge ein. Viele HPC-Menschen machen heute alles mit Tasks. Auch so können wir das Programmiermodell lesen: Der User schreibt eine Zahl von Tasks und deren Implementierung. Er spezifiziert allenfalls noch Constraints, welche Tasks auf welchen Zellkombinationen auf keinen Fall parallel laufen dürfen. Ein konkretes Gitter instantiiert dann den eigentlichen Task-Graphen, wobei wir natürlich gar nicht wirklich physisch instantiieren müssen, weil der Graph durch das Gitter

ja schon implizit da ist. Wir laufen einfach geeignet über das Gitter und füttern das System mit Tasks. Task-assembly-free könnte man das nennen. Die Assoziation mit matrixfreier Linearer Algebra drängt sich auf.

In ExaHyPE umschreiben wir Entwickler das Programmiermodell am liebsten als Hollywood-Pattern: *Don't call us, we call you*. Der Nutzer spezifiziert, wie Ergebnisse zu berechnen sind. Aber dann läuft jemand anders über das Gitter und entscheidet, wo und wann Operationen ausgeführt werden. Erst wenn tatsächlich gerechnet werden muss, ruft der entsprechend zurück und frägt, wie denn die auszuwertenden Formeln genau aussehen. "Wir rufen Sie dann an" dürfte den meisten bekannt sein - zumindest denen, die schon mal vorsingen durften.

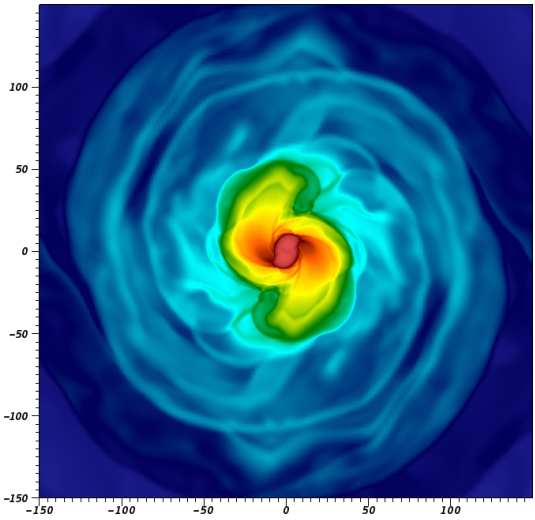


Abbildung 2: Momentaufnahme der Verschmelzung zweier Neutronensterne. Von Sven Köppel (FIAS, Frankfurt)

Und wie benutzt man es nun? Zentraler Angelpunkt eines ExaHyPE-Projektes (also eines Lösers, der die Engine benutzt) ist ein Spezifikations-File. Es ist eine Mischung aus deskriptiver Spezifikation, Domain-Specific-

Language und Konfigurationsdatei. Es definiert, wie Löser heißen, welche Pfade zu verwenden sind, für welche Architektur zu optimieren ist, welche Polynomordnung man verwenden will, welche Time-Stepping-Variante zu wählen ist, und so weiter und so fort. Der Nutzer übergibt diese Datei einem Code-Generator (wir nennen ihn Toolkit), der uns nicht nur allen möglichen Code inklusive einer `main` erzeugt (wir erinnern: in ExaHyPE gibt man die eigentliche Kontrolle über Zeitschrittwahl, Gittertraversierung, Gitterzerteilung, ... und damit den Programmfluss inkl. der `main` ab), sondern auch Gimmick wie ein fertiges Build-System (wir vermuten mal, dass sich mit Exascale viel ändern wird, aber `make`, das ist und bleibt).

Die Kernaufgabe des Nutzers ist, die eigentliche PDE zu realisieren. Unser Toolkit generiert Klassen[2] samt leerer Routinen für die Flussfunktionen, Initialbedingungen, maximalen Eigenwerte, und so weiter und so fort. Für viele Anwendungen reicht es, vier, fünf davon zu implementieren. Wenn die PDEs komplex sind, muss es nicht auch noch die Software sein. Es ist ExaHyPEs Gitter, das ab hier die Kontrolle übernimmt und beim "Löser" um geeignete Flüsse oder Eigenwerte nachfragt. Wann welche Operation auf welchem Kern auf welchem Knoten mit welchen Daten auszuführen ist, entscheidet ExaHyPE ausgehend von der Gitterkonstellation und dem ADER-DG-Schema.

Ja was bedeutet das nun? Noch nicht mal halb durch das Projekt ist es zu früh, ein Fazit zu ziehen, ob unser ExaHyPE-Ansatz und -Programmiermodell aufgeht. Ein paar Beobachtungen aber können wir schon machen:

Die funktionale Dekomposition und der Zwang, dass sich der Nutzer auf das Wesentliche, sprich die PDE, fokussiert, mögen elegant sein, aber sie sind nicht immer unproblematisch; insbesondere dann, wenn sich User, wie in einem Open-Source-Projekt üblich, tiefer in den Code wühlen und feststellen, dass die Callbacks keine übergestülpte Schicht sondern grundlegendes Design-Konzept sind. Sie ziehen sich quer durch den Code. Nach einer relativ

[2] Ja, wir haben FORTRAN 90-Unterstützung, aber weil ich FORTRAN an sich nicht mag, lass ich das hier mal unter den Tisch und in die Fußnote fallen und beschränke mich auf C++; nutze aber die Gelegenheit, einen schönen Gruß gen Italien zu schicken.

schnellen Start-Up-Phase haben ExaHyPE-Entwickler oftmals ein “ich will nun aber das und das machen” gehört. Frägt man nach, stellt man oft fest, dass diese Aussage ein “ich will das so und so machen” ist. Die Antwort, dass man auch mal loslassen müsse und manche Entscheidung zum Wie ExaHyPE zu überlassen habe, trifft nicht immer auf Verständnis. Wir zwingen User zu reflektieren, was genau in kleinen Schritten erreicht werden soll. Wir lassen den User diese aber nicht selber abwickeln. Kritisches Hinterfragen unseres Programmiermodells ist also kein kleinkindliches “ich will aber”; der Bayer dadert wohl eher sagn “des hammer oiwei scho so gmacht”. Es zeigt vielmehr, dass gerade ein so einfaches Software-Interaktionskonzept wie in ExaHyPE Umdenken auf der Benutzerseite erfordert: wie lassen sich Algorithmen in einfache, sehr kleine Tasks unterteilen und was sind die kausalen, wirklich notwendigen Zusammenhänge zwischen diesen? Nicht alle grösseren Algorithmen oder Methoden, die wir heute tagtäglich verwenden, liegen bereits so durchleuchtet und formalisiert in einem Lehrbuch vor.

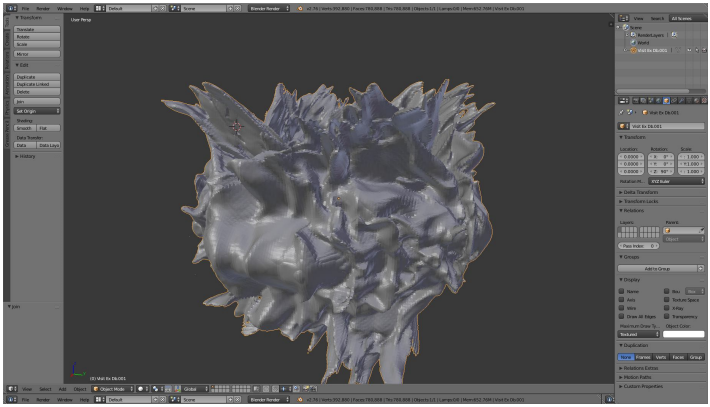


Abbildung 3: Isosurfaces in der Simulation einer Neutronensternverschmelzung. Von Sven Köppel (FIAS, Frankfurt).

All das Task- und Callback-Gedöns liest sich als Computer-Scientist-by-Training schön. In ExaHyPE hat es uns ermöglicht, schnell und abstrahierend von unseren Anwendungen eine Engine aufzusetzen, die, OpenMP und TBB

sei Dank, auch auf einem KNL alle Kerne nutzt. Ist sie dadurch schnell? Das kommt drauf an. Im ADER-DG gibt es Tasks pro Zelle, die extrem arithmetisch intensiv sind. Hier funktioniert eine Task-basierte Parallelisierung super. Bei all den anderen Tasks sind wir uns nicht sicher: Es ist frustrierend zu sehen, dass ein `parallel for` aus den Anfangstagen von OpenMP oft um Meilen schneller ist, als Task-basiertes Programmieren; zumindest wenn man wenig(er) Kontrolle und Wissen über den Inhalt der Tasks hat; Wissen und Entscheidungen, die wir ja genau dem Nutzer überlassen wollen. Es ist frustrierend zu sehen, dass die Task-Zuordnung zu Kernen mit OpenMP 4 auf dem KNL am Ende genau das ist, was man in der Numerikvorlesung als Red-Black-Colouring kennengelernt hat. All der Task-Formalismus war dann gar nicht so nötig; das hätte man auch billiger haben können. Es bleibt abzuwarten, wie schnell die Task-basierte Shared-Memory-Parallelisierung zu klassischem Performance-Engineering aufholen kann. Für uns wäre es gut wenn bald.

Und beim MPI? Das ist vielleicht der Punkt, wo unser Interaktionsansatz am stärksten punkten kann. Unser Gitter instantiiert, ohne ihn explizit aufzusetzen, einen Task-Graphen, zu dem ADER-DG Abhängigkeiten vorgibt. Sobald wir das Gitter kennen, kennen wir also den genauen Informationsfluss und können Daten im Hintergrund zur eigentlichen Berechnung verschieben. Wir haben viele Herausforderungen in ExaHyPE durch sich ständig verfeinernde Gebiete, Limiter, die mal an und dann wieder aus gehen, adaptives lokales Time-Stepping. Das kann aber alles als Load-Balancing-Challenges umgedeutet werden. Das eigentliche MPI-Datenaustauschgeschäft scheint gut zu unserem User-Ansatz zu passen.

Wo geht es hin? Wir arbeiten daran, interessante erste Rechnungen auf Maschinen wie dem SuperMUC hochzuskalieren. Dabei beobachten wir, wie sich ein Kreis schließt: Das Programmiermodell mit einem Verstecken der Ausführungsordnungen hat uns gezwungen, unsere Anwendungen zu taskifizieren und damit sicher die Entwicklung eher verlangsamt. Insbesondere konnten wir nicht altes Zeugs einfach recyceln. Die konsequente Modellierung als Tasks gibt uns nun aber zunehmend die Freiheit auf der Gitterseite, über gute Ausführungssequenzen (zeitlich und bzgl. der Core-Affinitäten sowie des Node-/Speicher-Ortes) nachzudenken und diese zu optimieren. Ich bin zuversichtlich: There is Power in Loosing Control.

Tobias Weinzierl

Alle Links kurzgefasst:
`www.exahype.eu`
`www.dur.ac.uk/tobias.weinzierl`
`www.peano-framework.org`

Scaling Workshop 2017: Emergent Applications

Das LRZ hat vom 15.-18.Mai 2017 den „Scaling Workshop 2017: Emergent Applications“ durgeführt, bei dem 6 Teams die Möglichkeit hatten, ihre Codes auf das halbe SuperMUC-Phase1-System zu skalieren.
Folgende Projekte nahmen teil:

- MPAS (D. Heinzeller, KIT) Wetter Vorhersage
- VLASOV6D (K. Reuter et al., TUM/IPP/MPCDF), Plasmaphysik
- ECHO (M. Bugli, MPA) Akkretionsscheiben
- BFPS (M. Wilczek et al., MPDS) Turbulence
- MGLET (Y. Sakai et al., TUM) Hydrodynamik
- TERRA-NEO (S. Bauer et al., LMU/TUM) Geophysik

In dem 4 tägigen Workshop wurden die Teilnehmer von der Applikationsgruppe des LRZ und den Herstellerfirmen IBM, Intel und Lenovo, sowie Allinea bei der Optimierung der Programme unterstützt. Am Dienstagabend fand ein Social-Event statt bei dem die Teilnehmer die Gelegenheit hatten, ihre Erfahrungen mit der Nutzung von SuperMUC auszutauschen. Von den 6 Projekten gelang es 5 auf 8 Inseln des SuperMUC zu skalieren und die Performance ihrer Software zu optimieren.

Als Highlight fand am Donnerstag die Verleihung des Leibniz Scaling Awards 2017 durch Prof. Kranzlmüller an Matteo Bugli vom Max-Planck-Institut für Astrophysik statt, der für seine hervorragenden Fortschritte in der Skalierung und Optimierung seines Codes ECHO während des Scaling-Workshops ausgezeichnet wurde.

Abbildung 1: Prof. Kranzlmüller (LRZ) überreicht den Leibniz Scaling Award an Dr. Matteo Bugli (Max-Planck Institut für Astrophysik)

Das Projekt von Herrn Bugli beschäftigt sich mit der Simulation von Akkretionsscheiben um Neutronensterne und Schwarze Löcher im Rahmen der relativistischen Magnetohydrodynamik. Matteo Bugli benutzte des Weiteren das Profiling-Programm Darshan dazu den IO zu optimieren und damit die Gesamt-Performance des Programms um 18% zu steigern.

Ferdinand Jamitzky

ISC 2017 PhD Forum

Im Rahmen der ISC 2017 Konferenz (19.6.2017 - 22.06.2017) in Frankfurt am Main fand am 19.6.2017 zum zweiten Mal das ISC 2017 PhD Forum statt - ein Wettbewerb bei dem Promovierende ihre laufenden Forschungsarbeiten vorstellen können.

Aus den zahlreichen Einreichungen hat ein Programmkommittee unter Leitung von Prof. Dr. Bill Gropp (University of Illinois at Urbana-Champaign) 13 Doktoranden ausgewählt, die dann im Zuge einer jeweils 4-minütigen Präsentation sowie anschließender Postersitzung ihre Arbeiten mit großer Begeisterung vorstellten.

Abbildung 1: Diskussion am Poster/ Teilnehmer und Mitglieder des Programmkommittess

Aus dem thematisch sehr breiten Spektrum interessanter Arbeiten wählte schließlich das Programmkomitee den Beitrag von Julian Hammer zum Thema „Performance Engineering by Means of Automated Performance Modeling“als besten Beitrag aus, der mit einem iPad sowie einem Buchgutschein von Springer prämiert wurde. Julian Hammer hat an der FAU Erlangen-Nürnberg Computational Engineering studiert und dort auch an seiner Doktorarbeit im Bereich HPC.

Damit geht auch im zweiten Jahr der ISC PhD Forum der Preis nach Bayern, nachdem im Vorjahr Alfredo Parra Hinojosa von der TU München das Rennen für sich entscheiden konnte. Auch in diesem Jahr wurden der Konferenzbesuch für die Doktoranden über deutlich reduzierte Konferenzgebühren sowie einen Reisekostenzuschuss durch das DFG Schwerpunktprogramm SPPEXA finanziell unterstützt.

Das ISC 2018 PhD Forum Programmkomitee freut sich bereits jetzt auf zahlreiche Einreichungen für das nächste Jahr.

Gerhard Wellein

SPPEXA News – Workshops im Herbst 2017

Der 2. Workshop-Call von SPPEXA Phase 2 traf wiederum auf großes Interesse der Projekte. Damit können wir die Sommerpause zügig hinter uns lassen und in einen spannenden Herbst mit neuen Workshops starten.

GAMM Angewandte und Numerische Lineare Algebra 2017
Der GAMM Fachausschuss für Angewandte und Numerische Lineare Algebra trifft sich zu ihrem jährlichen Workshop in Köln am **7. und 8. September 2017**. Schwerpunktthema dieses Jahr ist High Performance Computing, unter kräftiger Mitwirkung des SPPEXA Projektes EXASTEEL. Vermeidung von Kommunikation bei klassischen Lösungsverfahren, seien es Krylov, Mehrgitter oder Nichtlineare Löser, spielt eine entscheidende Rolle auf unserem Weg zu Exascale.

Parallel Programming Models – Productivity and Applications (2nd Edition)
Nach dem überaus erfolgreichen 1. Teil des Workshops im März diesen Jahres (siehe ausführlichen Artikel im Quartl 82), steht nun der angekündigte Folge-Workshop an. Wiederum treffen sich die SPPEXA Projekte MYX, ESSEX und DASH gemeinsam mit ihren japanischen und französischen Kollegen, um sich über Kollaborationen innerhalb SPPEXA, aber auch in den Partnerprogrammen in Frankreich und Japan, auszutauschen. Während der 1. Workshop in Tokyo stattfand, trifft man sich nun am **18. Oktober 2017** in Versailles, um gezielt die Zusammenarbeit mit den französischen Kollegen zu stärken.

6th Workshop on Parallel-in-Time Integration

Zeitparallelisierung eröffnet eine spannende Möglichkeiten, um Skalierbarkeit auf Exascale Rechnern zu erlangen. Der bereits 6. Workshop zu diesem Thema findet dieses Jahr von **23. bis zum 27. Oktober 2017** in Monte Verità in der Schweiz statt. SPPEXA is vor Ort vertreten durch die Projekte ExaStencils und EXASOLVERS.

Benjamin Uekermann

* Notiz * Notiz * Notiz *

Konferenzen 2018

- Society for Industrial and Applied Mathematics: SIAM Conference on Parallel Processing for Scientific Computing SIAM PP SC in Tokyo: 07.03.-10.3.2018 `https://www.siam.org/meetings/pp18/`
- Gesellschaft für angewandte Mathematik und Mechanik, International Association of Applied Mathematics and Mechanics Jahrestagung GAMM München:
  19.3.-23.3.2018 `http://jahrestagung.gamm-ev.de/`
- Society for Industrial and Applied Mathematics: SIAM Conference on Applied Linear Algebra SIAM ALA in Hong Kong (Applied Linear Algebra):
  4.5.-8.5.2018 `http://www.math.hkbu.edu.hk/siam-ala18/`

Quartl* - Impressum

Herausgeber:
Prof. Dr. A. Bode, Prof. Dr. H.-J. Bungartz, Prof. Dr. U. Rüde
Redaktion:
S. Herrmann, S. Seckler, Dr. S. Zimmer
Technische Universität München, Fakultät für Informatik
Boltzmannstr. 3, 85748 Garching b. München
Tel./Fax: ++49-89-289 18611 / 18607
e-mail: herrmasa@in.tum.de, **www:** http://www5.in.tum.de/quartl
Redaktionsschluss für die nächste Ausgabe: **01.12.2017**

* Quartel: früheres bayerisches Flüssigkeitsmaß,
→ das Quart: $1/4$ Kanne = 0.27 l
(Brockhaus Enzyklopädie 1972)