Gefördert durch:
Bundesministerium für Wirtschaft und Energie
aufgrund eines Beschlusses des Deutschen Bundestages

JÜLICH FORSCHUNGSZENTRUM

Forschungsvorhaben: 03ET7011J
Entwicklung von Verbrennungstechnologien im CEC für klimaschonende Energieerzeugung - Projekt 1J:
„Grundlagen - High Performance Computing von Gasturbinen-verbrennungssystemen auf Hochleistungscomputern“

Laufzeit des Vorhabens: 01.01.2013 – 31.12.2015

Abschlussbericht

Johannes Grotendorst und Bernd Körfgen

Institute for Advanced Simulation, Jülich Supercomputing Centre,
Forschungszentrum Jülich GmbH

Juni 2016

I. Kurze Darstellung zu Aufgabenstellung und Voraussetzungen

1. Aufgabenstellung

Dieses Projekt ist Teil des Verbundprojektes „Entwicklung von Verbrennungstechnologien für klimaschonende Energieerzeugung - Projekt 1: Grundlagen“.

Für die Entwicklung effizienterer Kraftwerke sind entwicklungsbegleitende rechnergestützte Simulationen von Prozessen unerlässlich. Das Ziel dieses Vorhabens ist die Bereitstellung einer für innovative massiv-parallele Rechnerarchitekturen optimierten Software, mit deren Hilfe Modellierungen von Verbrennungsprozessen aber auch Simulationen aus einer Vielzahl von anderen technischen und wissenschaftlichen Bereichen effizient durchgeführt werden können. Als Software ist hierfür das frei verfügbare Open Source Softwarepaket OpenFOAM [1] vorgesehen.

Das konkrete Ziel des Vorhabens ist es, die Simulation einer kompletten Ringbrennkammer zu ermöglichen, die voraussichtlich mehrere zehntausend Rechenkerne benötigen wird. Die Größe solcher von 500 Millionen bis 1 Milliarde Gitterpunkte umfassenden Simulationen erfordert die effiziente Nutzung moderner massiv-paralleler Rechnerarchitekturen wie beispielsweise der Blue Gene/Q Architektur von IBM [2]. Dazu stellt sich das Vorhaben vier Arbeitsziele:

1) Portierung der Software OpenFOAM auf die massiv-parallele Blue Gene/Q Architektur
2) Evaluierung der Software auf der Blue Gene/Q
3) Behebung möglicher Performance- und Skalierungsengpässe in Vorbereitung der geplanten Simulationsläufe
4) Bereitstellung der optimierten Software sowie Unterstützung des Industrie Partners Siemens AG bei der Durchführung von Simulationen

Derzeit ist eine aktuelle Version des Softwarepakets OpenFOAM für die Blue Gene/Q Rechnerarchitektur nicht verfügbar und die Größe möglicher Simulationsläufe ist auf wenige tausend Rechenkerne beschränkt. Somit kann diese Limitierung bereits durch das erste Arbeitsziel aufgehoben werden, wodurch die Durchführung größerer Simulationen für die Projektpartner ermöglicht wird.

2. Stand der Technik

Für strömungsmechanische Simulationen (CFD, Computational Fluid Dynamics) von Verbrennungsprozessen können Simulationsverfahren basierend auf der Methode der finiten Elemente (FEM) eingesetzt werden. OpenFOAM [1] ist ein CFD Softwarepaket, das aufgrund der freien Verfügbarkeit als Open Source Code sowie aufgrund seiner hohen Flexibilität durch seinen modularen Aufbau als Bibliothek vielfältig eingesetzt werden kann. Es ist in C++ geschrieben und unter Verwendung von MPI (Message Passing Interface) [3] parallelisiert. Der Anwendungsbereich erstreckt sich über viele wissenschaftliche Gebiete, angefangen von der Modellierung komplexer Flüssigkeitsströmungen bei chemischen Reaktionen, Turbulenz und Wärmeleitung bis hin zu Kontinuumsmechanik und Elektrodynamik. Das Programmpaket ist somit für eine große Anzahl von HPC-Nutzern aus verschiedenen wissenschaftlichen Gebieten von Interesse. Die OpenFOAM Software wird von der OpenFOAM Foundation [1] gepflegt, die die Weiterentwicklung des Pakets betreibt.

Im Hinblick auf die Größe der im Rahmen dieses Verbundprojekts geplanten Simulationen ist der Einsatz von modernen massiv-parallelen Höchstleistungsrechnern erforderlich. OpenFOAM ist bereits für eine Reihe von Rechnerarchitekturen (in der Hauptsache Clusterarchitekturen) portiert worden. Die Portierung einer älteren OpenFOAM Version (1.6)

auf Blue Gene/P wurde von der Victorian Life Sciences Computation Initiative (VLSCI) [4] durchgeführt, allerdings unter Verwendung der GNU Compiler Collection (GCC) [5]. Für die Blue Gene Architektur sind im Allgemeinen Compiler der IBM XL Suite [6] besser geeignet, da sie für die Blue Gene Architektur hochoptimierten Code erzeugen können.

3. Planung und Ablauf des Vorhabens

Das Vorhaben Portierung und Evaluierung des Softwarepakets OpenFOAM auf neue massiv-parallele Rechnerarchitekturen wird in drei Arbeitspakete unterteilt

Arbeitspaket 1: Codeportierung
Arbeitspaket 2: Test und Evaluierung
Arbeitspaket 3: Optimierung

Im Arbeitspaket 1 soll auf den Vorarbeiten aufbauend die aktuelle Version des Programmpakets (derzeit Version 2.1.0) im ersten Schritt des Projekts auf das Blue Gene/Q System JUQUEEN am JSC portiert werden. Hierzu sollen insbesondere die Compiler der IBM XL Compiler Suite verwendet werden.
Die Performance der OpenFOAM Installation soll im Arbeitspaket 2 getestet werden und im Hinblick auf mögliche Skalierungs- und Performanceengpässe untersucht werden. Insbesondere soll die Performance der in OpenFOAM zur Verfügung stehenden Löser für Konjugierte-Gradienten-Verfahren (Conjugate Gradient Solver) analysiert werden, die in den im Rahmen dieses Projekts geplanten Simulationen zum Einsatz kommen sollen. Anhand dieser Analysen soll eine Strategie erstellt werden, ob und gegebenenfalls wie die Effizienz dieser Routinen für die Blue Gene/Q Architektur verbessert werden kann.
Abhängig von den Resultaten der Analysen soll im dritten Arbeitspaket die Strategie zur Optimierung des Codes umgesetzt werden. Aufgrund der massiv-parallelen Blue Gene/Q Architektur, die pro Rechenkern bis zu 4 Threads ermöglicht, könnte hier beispielsweise die bestehende MPI Parallelisierung durch eine zusätzliche OpenMP Parallelisierung erweitert werden. Durch diese hybride Parallelisierungsstrategie kann die Blue Gene/Q Architektur optimal ausgenutzt werden. Zudem soll die Laufzeitumgebung für die geplanten Simulationen optimiert werden. Hierzu zählen die optimale Abbildung der Prozesse (Tasks) auf die Hardware (Mapping) sowie die Optimierung des Verhältnisses von MPI zu OpenMP Parallelisierung (Task-zu –Thread-Verhältnis pro Rechenknoten).

Arbeitspaketbeschreibung und Ressourcenplanung

AP	1.1
Name	Portierung
Beschreibung	Portierung des Softwarepakets OpenFOAM auf die IBM Blue Gene/Q Architektur
Inhalt/Ergebnis	Verfügbarkeit einer auf der IBM Blue Gene/Q Architektur lauffähigen OpenFOAM Version
Durchführung	JSC
Start	Januar 2013
Ressourcen	Personal 1 PM

AP	2.1
Name	Skalierungstests und Performance-Analyse
Beschreibung	Die Skalierungseigenschaften und die Performance von OpenFOAM werden auf der Blue Gene/Q untersucht
Inhalt/Ergebnis	Zusammenstellung von Benchmarkkonfigurationen, Durchführung von Benchmarkläufen, Performance-Analyse
Durchführung	JSC Zusammenarbeit mit Industriepartner Siemens
Start	Februar 2013
Ressourcen	Personal 2 PM

AP	2.2
Name	Optimierungsstrategie
Beschreibung	Aufbauend auf AP 2.1 Untersuchung verschiedener Ansätze zur Optimierung der Code Performance
Inhalt/Ergebnis	Entwicklung einer Optimierungsstrategie
Durchführung	JSC
Start	April 2013
Ressourcen	Personal 2 PM

AP	3.1
Name	Codeoptimierung
Beschreibung	Aufbauend auf AP 2.2 Umsetzung der Optimierungsstrategie
Inhalt/Ergebnis	Implementierung und Benchmarking
Durchführung	JSC
Start	Juni 2013
Ressourcen	Personal 6 PM

AP	3.2
Name	Simulationsunterstützung
Beschreibung	Unterstützung der Projektpartner bei der Durchführung von Simulationen
Inhalt/Ergebnis	Unterstützung beim Aufsetzen der Simulationen sowie bei Haltung und Verarbeitung von Simulationsdaten
Durchführung	JSC Zusammenarbeit mit Verbundprojektpartnern
Start	Dezember 2013
Ressourcen	Personal 1 PM

4. Zusammenarbeit

Alle Arbeiten werden in Zusammenarbeit mit dem Industriepartner Siemens AG durchgeführt. Insbesondere bei den Skalierungstests und der Performance-Analyse werden realistische und hinreichend große Modelle der Ringbrennkammer benötigt, um aussagekräftige Schlussfolgerungen für die Optimierungsstrategie ziehen zu können. Aufbauend auf den resultierenden OpenFOAM-Installationen wird das Paket allen Nutzergruppen zur Verfügung gestellt und die Projektpartner bei der Durchführung ihrer Simulationen unterstützt

II.1 Eingehende Darstellung zu den vorgegebenen Zielen und erzielten Ergebnissen

Arbeitspaket 1.1 - Portierung von OpenFOAM auf das Blue Gene/Q-System JUQUEEN

OpenFOAM Version 2.1.1 wurde mit Hilfe verschiedener GCC Compiler auf die Blue Gene/Q-Architektur portiert. Um OpenFOAM auf der Blue Gene/Q erfolgreich kompilieren und ‚linken' zu können, waren spezifische Anpassungen im ‚Make'-Prozess von OpenFOAM erforderlich. Grund dafür war das in OpenFOAM konsequent eingesetzte Konzept des ‚dynamischen Linkens'. Auf konventionellen Linux-Clustern wie JUROPA oder JURECA ist ‚dynamisches Linken' eine weit verbreitete Methode, während auf Blue Gene-Systemen ‚statisches Linken', d.h. keine Nutzung von dynamischen Bibliotheken, empfehlenswert ist. Entsprechend erforderte die Verwendung von dynamischen Bibliotheken in OpenFOAM auf JUQUEEN besondere Vorsicht und Anpassungen.

Verschiedene Anläufe wurden unternommen, um OpenFOAM auch in einer ‚statisch gelinkten' Variante zu installieren, da dies eine bessere parallele Skalierung bei großen Prozessorzahlen verspricht; außerdem hätte dies den Einsatz moderner Tools zur Programm- und Performance-Analyse erlaubt, die bei ‚dynamisch gelinkten' großen Softwaresystem erschwert sind. Da alle Tests mit den resultierenden ‚statisch gelinkten' OpenFOAM-Versionen mit Fehlern abbrachen, wurde dieser Ansatz in dieser Phase des Projekts nicht weiter verfolgt und ein erneuter Versuch auf die Vorbereitung der Optimierung verschoben!

Die resultierende OpenFOAM-Installation wurde mit verschiedenen Standard-Beispielen sowie dem von Dr. Beck, Siemens, zur Verfügung gestellten Datensatz *BC4* getestet. *BC4* ist ein Modell einer einzelnen Brennkammer mit ca. 20 Mio. Zellen und wird mit Hilfe des OpenFOAM-Lösers `rhoPimpleFoam` berechnet. Da die von Siemens im Rahmen dieses Projekts geplanten Simulationen auf Modellen wie *BC4* und von Siemens adaptierten Versionen des Lösers `rhoPimpleFoam` basieren werden, fokussierten sich die Arbeiten im JSC auf diesen Löser und Datensätze wie *BC4*.
Die Rechengeschwindigkeit der OpenFOAM 2.1.1 Installationen, die mit den drei verschiedenen GCC Compiler Versionen 4.4.6, 4.8.1 und 4.9 erzeugt wurden, wurde untersucht, aber es wurden bei keinem Compiler signifikante Vorteile festgestellt. Daher wurde schließlich die mit dem GCC 4.4.6 Compiler erstellte Version allen Nutzern zur Verfügung gestellt. Diese wurde zwischenzeitlich von verschiedenen Gruppen, z.B. vom Karlsruhe Institute of Technology (KIT), eingesetzt.

Weitere Arbeiten zu OpenFOAM auf JUQUEEN betrafen die Portierung mit zusätzlichen Compilern (Clang, XL Compiler). Diese waren nur teilweise erfolgreich: Während die serielle Performance, d.h. auf einem Prozessor, der mittels Clang und XL Compiler erzeugten OpenFOAM-Installationen ca. 10% besser war als der mit GCC kompilierten Version, traten bei parallelen Berechnungen MPI-Fehler in der OpenFOAM-Bibliothek `Pstream` auf. Diese erfordern weitere Analyse in Kooperation mit OpenFOAM-Entwicklern, da die Bibliothek `Pstream` zentrale Routinen zu Kommunikation und IO in OpenFOAM enthält, die nicht einfach an spezielle Softwareumgebungen angepasst werden können.

Im August 2013 wurde eine neue Version der Blue Gene/Q-Software, inklusive neuer MPI-Bibliotheken, auf JUQUEEN installiert. Entsprechend wurde eine neue Version von OpenFOAM 2.1.1 kompiliert und getestet, die die neue Blue Gene/Q-Software unterstützt. Diese neue Version wurde allen Nutzern zur Verfügung gestellt und ist seitdem auf JUQUEEN in Produktion.

Um weitere Testbeispiele für Untersuchungen mit OpenFOAM verfügbar zu haben, wurden auf Empfehlung von Dr. Beck, Siemens, Messungen basierend auf dem Tutorial-Beispiel *pitzDaily3D* durchgeführt. Dazu wurde mit Hilfe von OpenFOAM-Tools das Ausgangsnetz so verfeinert bis ca. 20.000 Zellen pro Prozess erreicht wurden, eine für industrielle Anwendungen typische Auflösung. Erste Tests ergaben, dass die Physik dieses Beispiels, Verbrennung bei stationärer turbulenter Strömung, zu aufwändigeren Berechnungen und damit längeren Rechenzeiten als beim Modell *BC4* führt.

Test-Rechnungen mit diesem Datensatz verteilt auf 2048 MPI-Prozesse waren nicht erfolgreich, während das Modell *BC4* trotz schlechtem Skalierungsverhalten berechnet werden konnte. Weitere Untersuchungen sind erforderlich, um klären zu können, ob die Ursache für das „Einfrieren“ des Programms numerische Instabilitäten oder Probleme beim IO in OpenFOAM sind. Der Datensatz *pitzDaily3D* führt zu deutlich mehr Dateien, die auf Festplatte geschrieben werden, als *BC4*; daher könnte dieses Beispiel eher auf Limitierungen beim IO stoßen als Modelle wie *BC4*.

Angesichts des Aufwands beim Erzeugen großer Rechennetze und deren Gebietszerlegung, ein in OpenFOAM inhärent serieller und damit langsamer Pre-Processing-Schritt, wurde der Ansatz Testdaten basierend auf allgemein verfügbaren Tutorial-Beispielen nicht weiterverfolgt. Selbst wenn alle dabei beobachteten Probleme behoben werden können, können die Schlussfolgerungen daraus nicht ohne weiteres auf die hochaufgelösten Simulationen einer kompletten Ringbrennkammer übertragen werden.

Weitere Tests und Messungen zur Laufzeit von OpenFOAM auf JUQUEEN wurden daher zurückgestellt bis realistische Modelle der kompletten Ringbrennkammer vorliegen!

Arbeitspaket 2.1 - Skalierungstests mit OpenFOAM

OpenFOAM wurden nach der Portierung und Installation auf JUQUEEN und JUROPA erfolgreich getestet. Die Messungen zur Rechengeschwindigkeit und Skalierungstests wurden mit dem Löser `rhoPimpleFoam` und dem Datensatz *BC4* durchgeführt. Die folgenden Ergebnisse beinhalten nicht die Gebietszerlegung, da dieser serielle Pre-Processing-Schritt separat ausgeführt wurde.

Genutzt wurden auf JUQUEEN die Version OpenFOAM 2.1.1, erzeugt mit den GCC Compilern, und auf JUROPA Version OpenFOAM 2.0.1, die mit dem Intel ® Compiler erstellt wurde. Die Ergebnisse für ‚starke Skalierung‘ mit 512, 1280 und 2048 MPI-Prozessen sind in Tabelle 1 angegeben. Alle Messungen wurden mit je einem MPI-Prozess je physikalischem Core gemacht.

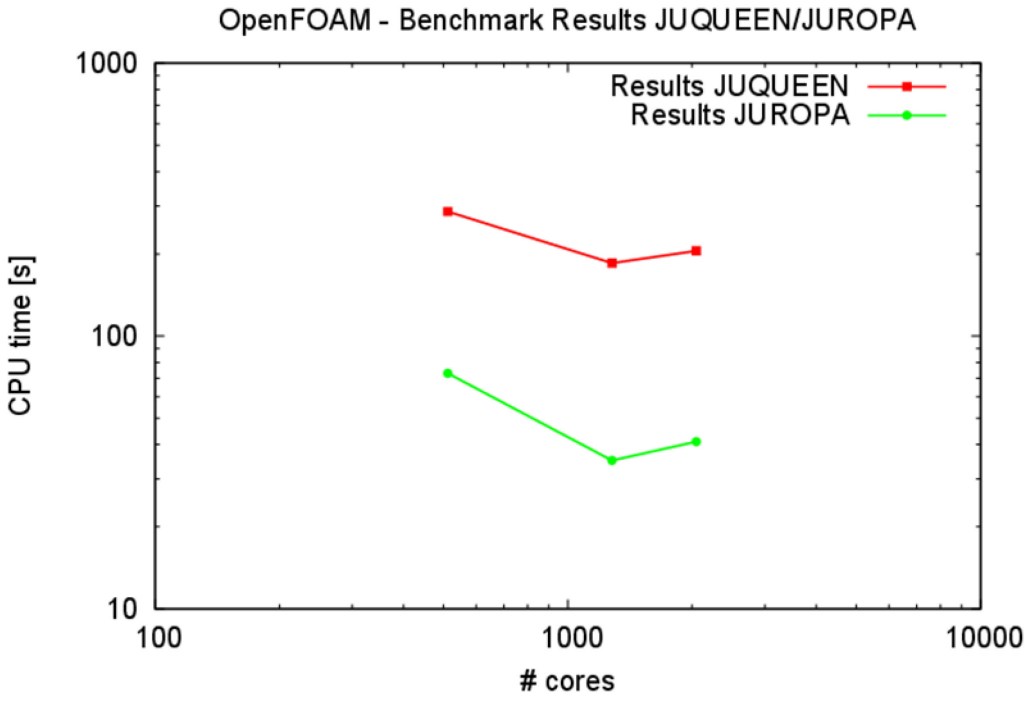
Tabelle 1: Laufzeit [Sekunden] von OpenFOAM-Rechnungen auf JUQUEEN und JUROPA.

# Prozesse	JUQUEEN	JUROPA
512	286 s	71 s
1280	185 s	32 s
2048	205 s	34 s

Die Ausführungszeiten auf JUQUEEN waren ca. 4- bis 6-mal länger verglichen mit JUROPA. Diese Beobachtung kann mit der langsamer getakteten CPU des Blue Gene/Q-Systems

erklärt werden. Die Skalierung auf beiden Architekturen ist in Abbildung 1 dargestellt. Die Laufzeit ist jeweils für 1280 MPI-Prozesse minimal. Eine größere Anzahl an Prozessen führt zu einer Verlangsamung der Rechnung. Dieses Minimum ist offensichtlich unabhängig von der Computerarchitektur und wird von der Anzahl der Zellen pro Prozess bestimmt. Wird eine gewisse Anzahl von Zellen pro Prozess unterschritten so überwiegt der parallele Overhead und die Effizienz der Rechnung sinkt. Daher werden erst größere Datensätze, d.h. mit mehr Zellen pro Prozess, eine realistische Abschätzung der Performance von OpenFOAM für die in diesem Projekt geplanten Berechnungen einer kompletten Ringbrennkammer ermöglichen.

Abbildung 1: Ergebnisse der Benchmark-Rechnungen auf JUQUEEN (rot) und JUROPA (grün) in doppelt-logarithmischer Darstellung



Arbeitspaket 2.2 – Performance-Analyse und Optimierungsstrategie

Es wurde versucht, die in Produktion befindliche Installation von OpenFOAM auf JUQUEEN so zu instrumentieren, dass ‚Hardware Performance Counter' genutzt werden können, um automatisch Performance-Daten für OpenFOAM zu sammeln. IBMs Hardware Performance Monitor (HPM) Bibliothek stellt diese Möglichkeit auf JUQUEEN bereit und funktioniert gut mit anderen Anwendungen. Leider verhinderte OpenFOAMs komplexe Datenstruktur die erfolgreiche Nutzung der HPM-Bibliothek.

Ein sogenanntes ‚Profiling' von OpenFOAM, d.h. die Messung der in den einzelnen Routinen benötigten Rechenzeit, war ein weiterer Schritt zur Entwicklung eine Optimierungsstrategie. Dieses wurde zunächst mit Hilfe des Tools `Gprof` durchgeführt, da alternative Tools entweder auf JUQUEEN nicht zur Verfügung standen oder nur ‚statisch gelinkte' Programme unterstützen. Die Ergebnisse der Messung mit dem Datensatz *BC4* auf 80 JUQUEEN-Knoten, insgesamt 1280 MPI-Prozesse, zeigt die folgende Tabelle. Dabei sind 25 Zeitschritte mit einer Schrittweite von jeweils 2.0E-6 Sekunden berechnet worden:

Routine	Aufrufe	Anteil [%] Laufzeit
operator `/=`	125	15.56
`surfaceIntegrate`	950	13.45
`surfaceInterpolationScheme::interpolate`	1	12.45
operator `dot`	251	4.96
`List<double>::List`	1550	4.54
`patchInternalField<double>`	77270	3.64
`patchInternalField<Foam::Vector<double> >`	78028	3.06

Die hier aufgeführten Routinen haben den größten Beitrag zur gesamten Rechenzeit und machen zusammen mehr als 50% der gesamten Laufzeit aus. Realistische Simulationen haben sehr viel mehr Zeitschritte, so daß die Bedeutung von Routinen, die nur selten aufgerufen werden, dort geringer sein wird.

Folgende Schlussfolgerungen kann man aus diesen Messungen ziehen:

- Der Operator `/=` berechnet die Division eines Vektorfelds durch einen Skalar. Eine Parallelisierung mittels OpenMP kann zu einer Verkürzung der Laufzeit führen.
- `surfaceIntegrate` führt eine Oberflächenintegration aus, die ebenfalls hybrid parallelisierbar ist.
- Der Operator `dot` berechnet ein Skalarprodukt und hat ebenfalls Potential für eine Optimierung durch eine OpenMP-Parallelisierung.
- `patchInternalField` führt keine Berechnungen durch (Daten-Kopien) und ist kein geeigneter Kandidat für Optimierung.

Vor der endgültigen Festlegung von Routinen, deren Optimierung bzw. hybride Parallelisierung mittels OpenMP in Angriff genommen werden kann, sind weitere Messungen mit größeren, möglichst realistischen Modellen erforderlich, da diese Modelle und ihre Berechnung zu anderen Hotspots bzgl. der Rechenlast bzw. bisher nicht beobachtbaren Effekten bei der Kommunikation oder dem IO führen könnten und so die Optimierungs-Prioritäten verschieben könnten.

Eine weitere Option zur Optimierung rechenzeitintensiver Schleifen ist deren Vektorisierung unter Ausnutzung der SIMD-Fähigkeit der A2-Prozessoren der JUQUEEN. Da die GCC-Compiler diese Vektorisierung nicht hinreichend unterstützen, müssten dafür allerdings die Probleme mit den XL-Compilern gelöst werden.

Die Ergebnisse des ‚Profiling' mit `Gprof` zeigten, dass eine detaillierte Untersuchung des Laufzeitverhaltens von OpenFOAM 2.1.1 mit verschiedenen Performance-Analyse-Tools erforderlich ist, um die Ursachen von parallelem Overhead in OpenFOAM, z.B. aufgrund von Kommunikation, Wartezeiten oder Lastungleichgewicht, zu identifizieren. Alle folgenden Messungen wurden wieder mit dem Löser *rhoPimpleFoam* und dem Datensatz *BC4* durchgeführt.

Die damals einzig verfügbare Installation von OpenFOAM war ‚dynamisch gelinkt'. Dies hatte Konsequenzen für die Durchführbarkeit der Performance-Analyse:

- Verschiedene viel benutzte Performance-Analyse Tools unterstützen keine ‚dynamisch gelinkten' Programme, so z.B. Scalasca [7] und Vampir [8], und konnten daher nicht genutzt werden.

- Sowohl die Ausführung von massiv-parallelen Rechnungen mit ‚dynamisch gelinkten' Programmen, als auch deren Analyse sind verlangsamt, da das Öffnen von dynamischen Bibliotheken „bei Bedarf" immer wieder Zeit erfordert.

- Das Konzept des ‚dynamischen Linkens' verhindert den Einsatz von Optimierungen beim ‚Linken', z.B. Interprozedurale Analyse (IPA)

Die Vorgehensweise bei der weiteren Performance-Analyse war daher wie folgt:

1. Analyse der ‚dynamisch gelinkten' OpenFOAM-Installation auf JUQUEEN und JUROPA mit den verfügbaren Tools.

2. Nochmaliger Versuch, OpenFOAM statisch zu linken, und Analyse dieser Version

Ergebnisse mit ‚dynamisch gelinktem' OpenFOAM auf JUQUEEN und JUROPA

Nur wenige Performance-Analyse Tools, wie TAU [9], HPCToolkit [10] und Open|SpeedShop, unterstützen überhaupt ‚dynamische gelinkte' Programme. Aufgrund der speziellen Architektur der Blue Gene/Q existieren aber keine Implementationen dieser Tools auf JUQUEEN. Daher wurde ein Versuch mit STAT (Stack Trace Analysis Tool) [11] unternommen, das Spuren während der Ausführung einer parallelen Anwendung sammelt und so die Sequenz der Routine-Aufrufe aufzeigt. Leider ist die Menge an Information bei einer so großen und komplexen Anwendung wie OpenFOAM viel zu unübersichtlich, um als Startpunkt für eine Analyse geeignet zu sein. Die gesammelten Daten stehen aber weiter zur Verfügung, um zu einem späteren Zeitpunkt, eine Detail-Analyse zu ermöglichen.

Im nächsten Schritt wurde der ‚dynamisch gelinkte' Löser *rhoPimpleFoam* auf JUROPA mit den dort verfügbaren Tools HPCToolkit und TAU analysiert:

HPCToolkit

Das HPCToolkit [10] gibt einen allgemeinen Überblick über das Laufzeitverhalten der untersuchten Anwendung. In Tabelle 2 werden die verschiedenen Anteile an der Laufzeit bei gegebenem Datensatz aber variabler Prozessorzahl gezeigt (starke Skalierung).

Anteil an der Ausführungszeit [%]	Anzahl Prozesse		
	512	1280	2048
MPI Aufrufe	13.6	26.3	49.6
Berechnungen	82.7	68.9	44.1
Rest	3.7	4.8	6.3

Tabelle 2: Ergebnis der Performance-Analyse von *rhoPimpleFoam* mit dem HPCToolkit

Wie man leicht erkennt, spielt der Anteil der MPI Aufrufe, d.h. der Anteil der Kommunikation zwischen den Prozessen, bei 512 Prozessen noch eine relativ kleine Rolle, steigt aber deutlich an und ist für 2048 Prozesse schon der größte Anteil. Mögliche Ursache könnte z.B. ein ungünstiges Kommunikationsschema oder eine ungleichmäßige Verteilung von Kommunikation sein. Detailinformationen dazu kann man mit Hilfe des Tools TAU erhalten.

TAU

Es wurde mit Hilfe von TAU [9] Daten für eine Rechnung mit 512 Prozessen gesammelt und diese anschließend mit Vampir [8] visualisiert. Einige interessante Beobachtungen zum Verhalten des Lösers *rhoPimpleFoam* sind im Folgenden dargestellt. Die Abbildungen 2, 3 und 4 zeigen die Kommunikations-Matrix für verschiedene Metriken. Dabei sind die Prozesse der x- und y-Achse zugeordnet und die Farbe zeigt die Intensität bzgl. der betrachteten Metrik; maximale Werte sind rot, minimale Werte blau dargestellt.

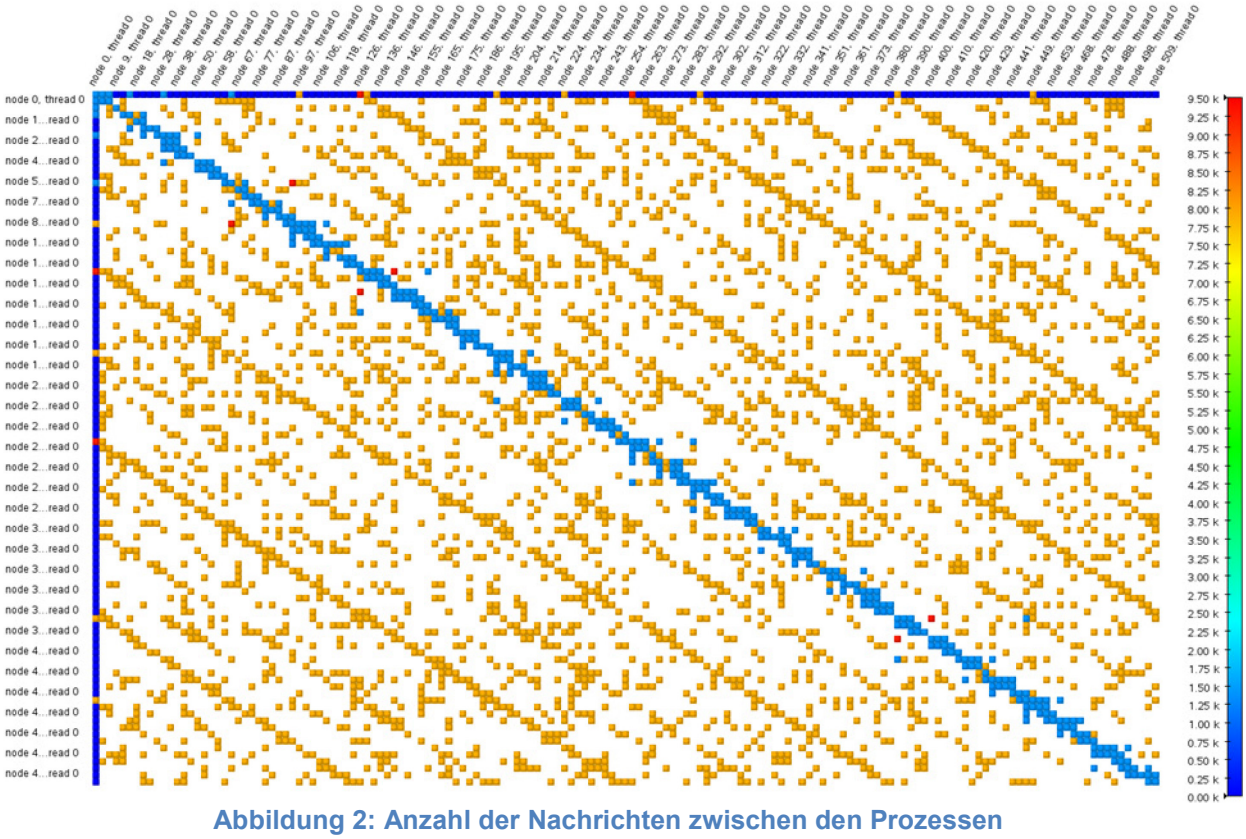


Abbildung 2: Anzahl der Nachrichten zwischen den Prozessen

Abbildung 2 zeigt, dass die Anzahl der verschickten/ empfangenen MPI-Nachrichten nicht gleichmäßig verteilt ist. Prozesse auf der Hauptdiagonale kommunizieren eher wenig, Prozesse, die ober- und unterhalb der Hauptdiagonalen dargestellt sind, tauschen Nachrichten sehr viel häufiger aus.

In Abbildung 3 stellt die Farbe die durchschnittliche Dauer der Kommunikation dar. Wie in der vorangegangenen Abbildung, sieht man, dass Prozesse ober- und unterhalb der Hauptdiagonale sehr viel mehr Zeit für Kommunikation benötigen. Man sieht weiterhin, dass die Zeit für das Verschicken/Empfangen von Nachrichten bei den ersten Prozessen linear ansteigt. Die bestätigt den ersten Eindruck eines Lastungleichgewichts.

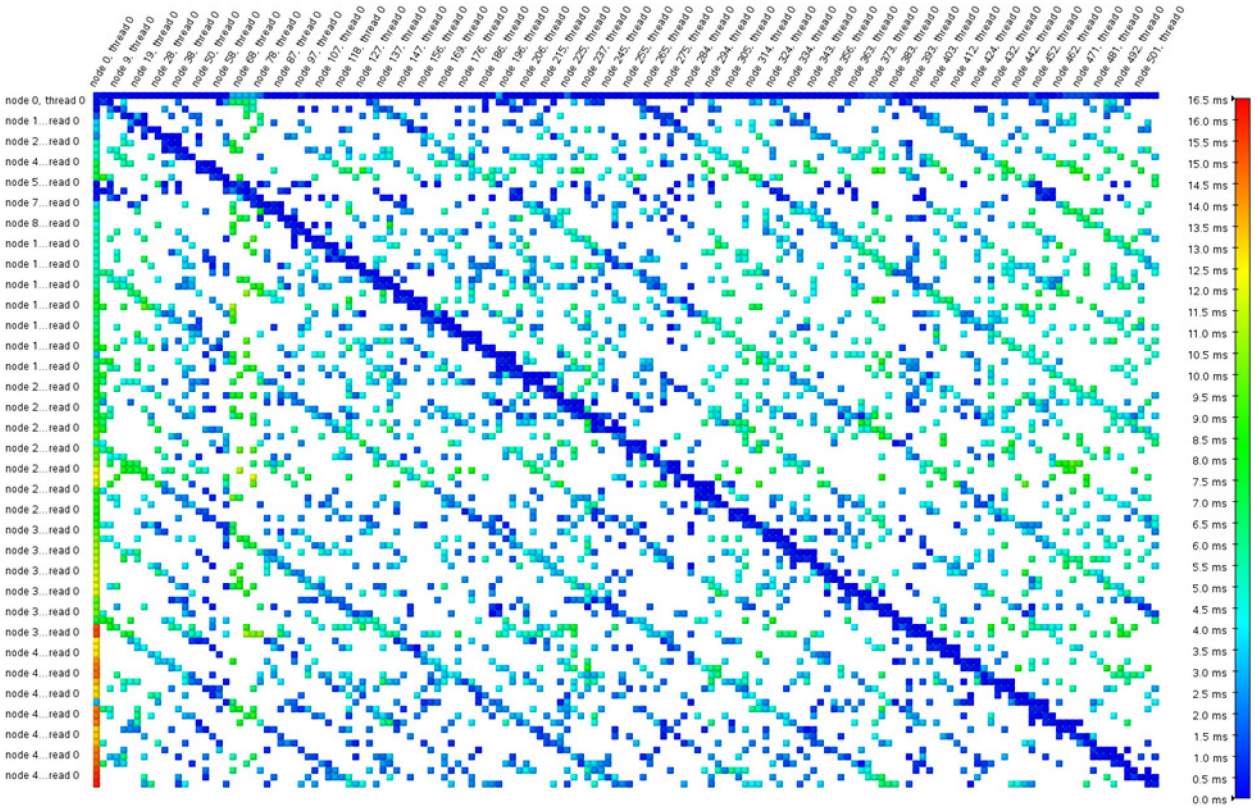


Abbildung 3: Durchschnittliche Kommunikationsdauer

In Abbildung 4 sind „ähnliche“ Prozesse, .d.h. Prozesse mit vergleichbarem Verhalten bzgl. Kommunikation, Berechnungen etc., zusammengefasst dargestellt. Der akkumulierte Zeitaufwand für die Funktionsgruppe „Berechnung“ wird grün, die Zeit für „Kommunikation“ rot gezeigt. Wieder sind erhebliche Unterschiede zwischen verschiedenen Prozess-Gruppen und damit Ungleichgewichtung der Arbeit erkennbar.

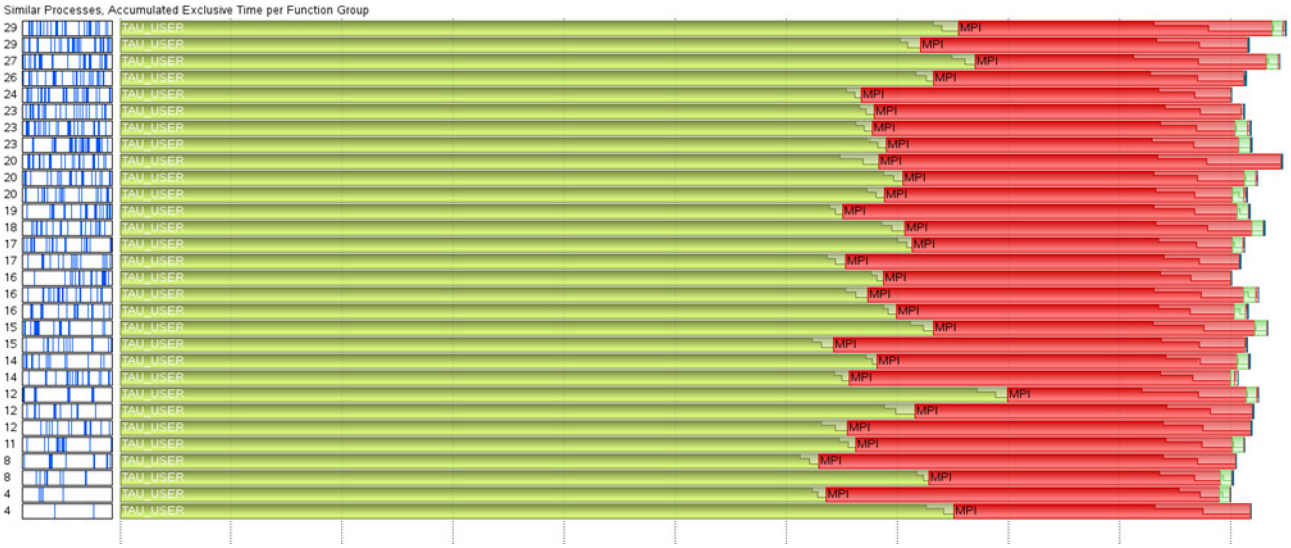


Abbildung 4: Akkumulierte Ausführungszeit pro Funktionsgruppe für „ähnliche“ Prozesse

Die Messungen mit 512 Prozessoren konnten auf JUROPA für größere Prozessorzahlen nicht wiederholt werden, da das Sammeln aller zur Laufzeit erzeugten Daten mehr Hauptspeicher benötigt als auf den JUROPA Login-Knoten verfügbar war. Bei Bedarf können diese Untersuchungen auf den Knoten mit viel Hauptspeicher, die mit JURECA verfügbar werden, fortgeführt werden. Es ist aber zu vermuten, dass die beobachteten Ungleichgewichte bei größeren Prozessorzahlen weiter anwachsen und die schlechte Skalierung der Anwendung erklären.

Implementierung eines ‚statisch gelinkten' OpenFOAM auf Blue Gene/Q

Vorangegangene Versuche, eine ‚statisch gelinkte' OpenFOAM-Installation zu erstellen, waren wegen Problemen mit inkonsistenten Laufzeit-Auswahltabellen nicht erfolgreich, so wurden benötigte Elemente wie `polyPatch` bei der Ausführung der Löser nicht gefunden.

In Zusammenarbeit mit dem Beratungsdienstleister OpenCFD Ltd. konnten diese Fehler schließlich behoben werden. Dies war nur möglich, da das JSC über den Beratungsvertrag der Firma Siemens Zugriff auf das Entwickler-Knowhow von OpenCFD Ltd. erhielt.

Die Beseitigung der Inkonsistenzen erforderte etliche Eingriffe sowohl in den Quelltext, als auch in den ‚Make'-Prozess des jeweiligen Lösers. Abhängigkeiten in den dynamischen Bibliotheken mussten manuell aufgelöst werden, eine zeitintensive und nicht automatisierbare Modifikation, die für jeden Löser einzeln durchgeführt werden muss. Daher wurden zunächst nur zwei Löser, nämlich *rhoPimpleFoam* und *XiFoam,* auf ‚statisches Linken' umgestellt. Anschließend wurde die bisher nicht mögliche Performance-Analyse mittels Scalasca in Angriff genommen.

Performance-Analyse von *rhoPimpleFoam* mit Scalasca

Die Performance-Analyse mittels Scalasca [7] auf JUQUEEN wurde für die schon mehrfach untersuchte Kombination Löser *rhoPimpleFoam* und Modell *BC4* von Siemens mit einer Gebietszerlegung für 512 Prozesse durchgeführt.

Die Compiler-basierte Instrumentierung des Lösers hat sich dabei als möglich aber nicht empfehlenswert herausgestellt. Der Grund dafür liegt darin, dass bei GCC-basierter

Instrumentierung jeder Prozess am Anfang die komplette Symbol-Tabelle einliest, die bei einer komplexen Anwendung wie einem OpenFOAM-Löser sehr groß sein kann. Entsprechend kann diese Phase dann sehr viel länger dauern als die eigentliche Rechnung des Lösers, hier 2 Stunden verglichen mit einigen Minuten.

Daher wurde für die weitere Performance-Analyse die manuelle Instrumentierung durch den Nutzer gewählt. Diese wurde durch entsprechende Anpassungen im Quelltext der Datei *rhoPimpleFoam.C* erreicht und das resultierende Profil führte zu folgenden Schlussfolgerungen:

- Die meiste Rechenzeit wurde von den beiden Lösern *UEqn* und *pEqn* benötigt; dabei entfällt ca. 10% der kompletten Ausführungszeit auf MPI-Kommunikation.
- Bei beiden Lösern wird sehr viel mehr Zeit mit *MPI_Waitall* verbracht als mit allen anderen MPI-Routinen.
- Die am häufigsten aufgerufenen MPI-Routinen sind *MPI_Irecv* und *MPI_Isend.*
- Die am häufigsten gestartete Anwendungsroutine ist

  Foam::eqOp<double>::operator()(double&, double const&) const

- Die Anwendung *rhoPimpleFoam* hat eine ungleichmäßige Lastverteilung: Die eine Hälfte der Prozesse hat eine zu hohe Arbeitslast, während die andere Hälfte nicht ausgelastet ist. Abbildung 5 zeigt dieses Lastungleichgewicht im Scalasca-Profil mittels der CUBE Visualisierung.

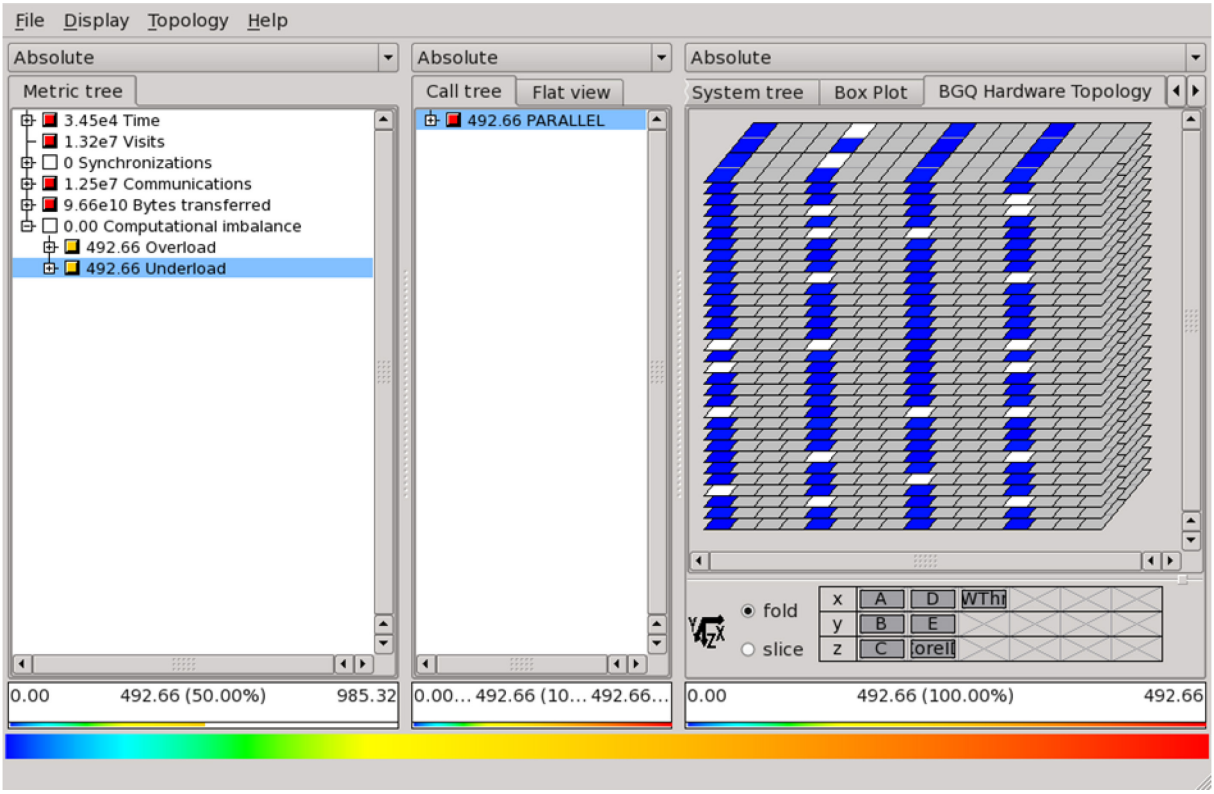


Abbildung 5: Visualisierung des Scalasca-Profils von *rhoPimpleFoam* mittels CUBE

Eine weitere Analyse des Scalasca-Profils zeigte ineffiziente Kommunikationsmuster: So wurden z.B. ‚Late Sender' (Abbildung 6) und ‚Late Receiver'-Konstellationen gefunden.

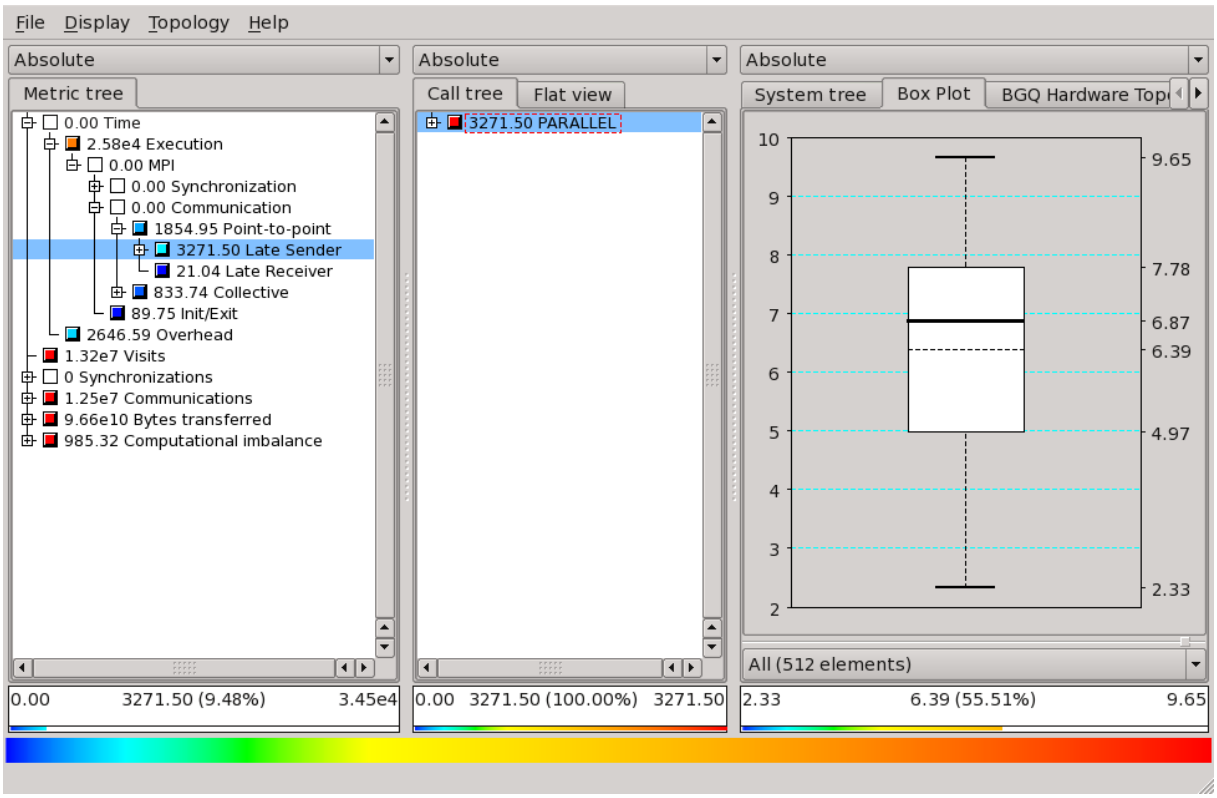


Abbildung 6: CUBE-Visualisierung des Kommunikationsmusters von *rhoPimpleFoam*

Untersuchungen zum IO-Verhalten von OpenFOAM auf GPFS-Dateisystemen

Das IO-Verhalten von OpenFOAM – die Anzahl der Dateien steigt linear mit der Anzahl eingesetzter Prozesse – stellt große Anforderungen an jedes Dateisystem. Diese Einschätzung wurde durch Berichte aus anderen Supercomputer-Zentren bestätigt. Insbesondere das von IBM entwickelte Dateisystem GPFS [12] ist dort als problematisch bewertet worden. Es ist allerdings bisher ungeklärt, ob dies tatsächlich an Einschränkungen im GPFS liegt, oder ob vielmehr auf diesen Rechnern ungewöhnlich große Berechnungen, d.h. Rechnungen mit sehr vielen Prozessen, durchgeführt wurden.
Entsprechend gab es Diskussionen mit Siemens, um Strategien zu Überwindung der IO-Begrenzung zu entwickeln. Bisher konnten diese Ansätze allerdings nicht umgesetzt werden, da dazu das Knowhow von OpenFOAM-Entwicklern, z.B. von einer Consulting-Firma, benötigt wird.

Zur Vorbereitung weiterer Benchmarks speziell zum IO-Verhalten wurden schon bekannte Probleme, wie der Abbruch / das Einfrieren von OpenFOAM-Rechnungen, reproduziert und die Auswirkung verschiedener Partitionierungs-Methoden untersucht.
Um belastbare Ergebnisse zu erhalten, werden größere modell-spezifische Datensätze benötigt. Das JSC führt im Rahmen verschiedener Projekte Arbeiten zum IO-Verhalten unterschiedlichster Anwendungen durch. Sobald größere Modelle verfügbar sein werden, kann OpenFOAM im Zuge dieser Akitivitäten ebenfalls untersucht werden.

Nutzung von GPUs zur Beschleunigung von OpenFOAM-Rechnungen

Das Software-Paket OpenFOAM-Extend Version 3.1, eine von der „OpenFOAM-Extend"-Community entwickelte Variante von OpenFOAM, unterstützt die Nutzung von GPUs zur Beschleunigung der numerisch aufwändigen und damit zeitintensiven Gleichungslöser. Sie baut dabei auf „Cuda For FOAM Link" (cufflink) [13] sowie den linearen Gleichungslösern der CUSP-Bibliothek [14] und parallelen Algorithmen aus Thrust [15] auf.
Um auf Nachfrage von Siemens die Möglichkeiten der GPU-Beschleunigung für OpenFOAM zu testen, wurden OpenFOAM-Extend und die benötigten Bibliotheken testweise auf dem Cluster JUDGE des JSC installiert. Auf den Knoten von JUDGE stehen zusätzlich zu jeweils zwei Multi-Core Prozessoren jeweils zwei NVIDIA Tesla (Fermi) GPUs zur Verfügung.
Die Funktionalität der GPU-basierten Löser wurde erfolgreich überprüft und ein erstes Benchmarking durchgeführt. Für aussagekräftige weitere Untersuchungen wären hinreichend große realistische Modelle erforderlich gewesen, denn erst bei großen Modellen lohnt sich das Auslagern von Teilen der Rechnungen auf GPUs. Das JSC hat ein originäres Interesse an der Verfügbarkeit leistungsfähiger Löser auch auf Hardware-Beschleunigern wie GPUs. Daher sind diese Ergebnisse eine gute Ausgangsbasis für weitere Arbeiten außerhalb dieses Projekts.

Arbeitspaket 3.1 – Codeoptimierung

Angesichts der schon weitgehend aufgebrauchten Personalressourcen, den uneinheitlichen Ergebnisse der Performance-Analyse und den schwierigen Rahmenbedingungen für eine Codeoptimierung von OpenFOAM wurde entschieden, die verbliebene Zeit für die Installation und Bereitstellung von OpenFOAM auf JURECA zu verwenden.

Arbeitspaket 3.2 – Simulationsunterstützung

Neben den Aktivitäten zur Implementierung und Performance-Analyse von OpenFOAM auf JUQUEEN, unterstützte das JSC auch die laufenden Rechnungen von Siemens auf dem HPC-Cluster JUROPA. Verschiedene Versionen von OpenFOAM wurden entsprechend den Wünschen von Siemens installiert und es wurden Hilfestellungen bei diversen Problemen gegeben: Insbesondere die Handhabung großer Datenmengen (vor allem die linear mit der Prozessorzahl wachsende Anzahl der Dateien), die Netzerzeugung für komplexe Modelle, sowie die Gebietszerlegung für große Prozessorzahlen waren problematisch.

Anfang Juli 2015 wurde der Linux-Cluster JURECA als Nachfolger von JUROPA den Nutzern zur Verfügung gestellt. Mit diesem neuen System ist eine sehr viel leistungsfähigere Ressource sowohl hinsichtlich Rechenleistung (48 virtuelle Cores je Rechenknoten gegenüber nur 8 auf JUROPA) als auch Hauptspeicher (bis zu 512 GB pro Rechenknoten gegenüber 24 GB) verfügbar. In der zweiten Ausbaustufe (Herbst 2015) wurden sogar Knoten mit bis zu 1 TB Hauptspeicher für speicherintensive Schritte, wie z.B. die Gebietszerlegung in OpenFOAM, verfügbar. JURECA wurde sofort nach Inbetriebnahme von Siemens für Produktions-Rechnungen mit OpenFOAM im Rahmen des CEC-Projekts und anderen Forschungsvorhaben eingesetzt.
Dafür wurden verschiedene Versionen von OpenFOAM bzw. der Variante OpenFOAM-Extend auf JURECA implementiert:

- OpenFOAM 2.0.1
- OpenFOAM 2.2.2

- OpenFOAM 2.3.1
- OpenFOAM 2.4.0
- OpenFOAM 3.0.0
- OpenFOAM-Extend 3.1
- OpenFOAM-Extend 3.2

Mit OpenFOAM-Extend 3.2 steht auf JURECA auch eine Installation zur Verfügung, die Rechenarbeit, wie das Lösen linearer Gleichungssysteme, auf GPUs auslagern kann. Diese Möglichkeit besteht seit dem Start von JURECA auch erstmals auf einem der großen HPC-Cluster des JSC, da auf 75 Rechenknoten zusätzlich zu den CPUs auch jeweils zwei GPUs vorhanden sind.

Zusammenfassung und Ausblick

Bei der Portierung von OpenFOAM auf die Blue Gene/Q-Architektur traten unvorhergesehene Schwierigkeiten auf. Eine Implementation basierend auf den GCC-Compilern war letztlich erfolgreich, während nicht alle Fehler, die bei Nutzung der XL-Compiler auftraten, behoben werden konnten. Die OpenFOAM-Installationen mit GCC-Compilern stehen seit dem Abschluss dieses Arbeitspakets auf JUQUEEN allen Nutzern zur Verfügung und werden weiter gepflegt. Parallel dazu wurden verschiedene Versionen von OpenFOAM auf JUROPA zur Verfügung gestellt, die Siemens im Rahmen verschiedener Projekte einsetzte.

Die geringe Einzel-Prozessor-Geschwindigkeit von OpenFOAM auf JUQUEEN ist angesichts der langsamen Taktung der CPUs nicht überraschend. Eine Verbesserung dieser Situation dank einer mit dem XL Compiler erstellten OpenFOAM-Portierung ließ sich wegen der bestehenden Fehler mit dem XL Compiler nicht erreichen. Daher werden Rechnungen auf JUQEEN erst bei deutlich größeren Prozessorzahlen attraktiv. Wie die Benchmark-Rechnungen zeigen, müssen die zu berechnenden Modelle dazu aber erheblich feiner und damit größer werden.

Die Performance-Analyse von OpenFOAM ergab, dass verschiedene Faktoren für die schlechte Performance von OpenFOAM auf JUQUEEN verantwortlich sind:

- Lastungleichgewichte zwischen den Prozessen bzgl. der Rechenarbeit und der Kommunikation

- ein Kommunikationsschema mit Problemen bei der Überlagerung von Kommunikation und Berechnung

- ein komplexer, objekt-orientierter Quelltext, der eine Optimierung und Vektorisierung durch den Compiler erschwert bzw. sogar verhindert

- ein IO-Konzept, die Anzahl der Dateien wächst linear mit der Anzahl der Prozesse, das selbst leistungsfähige Dateisysteme vor Probleme stellt

Außerdem konnten alle Analysen und Messungen nur mit verhältnismäßig kleinen Modellen durchgeführt werden, da die Modellentwicklung nicht mit der erhofften Geschwindigkeit zu größeren Datensätzen führte. Dies schränkte die Aussagefähigkeit der Analysen ein, da größere Modelle zu anderen Auswirkungen der oben genannten Probleme führen oder sogar ganz neue limitierende Effekte offenbaren können.

Mit der absehbaren Ablösung von JUROPA durch JURECA wurde klar, dass die langsam größer werdenden Modelle auch auf diesem neuen HPC-System erfolgreich berechnet werden können. Damit stand eine Alternative zu einer optimierten Installation auf JUQUEEN zur Verfügung und die verbliebenen Ressourcen im Rahmen des Projekts wurden in die Bereitstellung von OpenFOAM auf JURECA investiert.

Da die geschilderten Limitierungen der JUQUEEN-Portierung zusammen mit dem Fehlen größerer Modelle, JUROPA zu einem attraktiven System für die OpenFOAM-Simulation von Siemens machten, verschob sich der Fokus der Arbeiten auf die Bereitstellung aller benötigten OpenFOAM-Versionen auf JUROPA und dem Nachfolge-System JURECA. Aufgrund der deutlich größeren Leistungsfähigkeit von JURECA, sowohl hinsichtlich der Einzel-Prozessor-Geschwindigkeit, der Anzahl der Cores als auch des Hauptspeichers pro Knoten, werden seit Ende 2015 größere Modelle der Ringbrennkammer erfolgreich auf JURECA gerechnet.

II.2 Wichtigste Positionen des zahlenmäßigen Nachweises

Bezogen auf die gesamten Kosten des Vorhabens beläuft sich der Förderanteil des BMWi auf 49.8%; weitere 49.8% der Kosten wurden im Rahmen einer Industriezuwendung von Siemens Energy übernommen.

II.3 Notwendigkeit und Angemessenheit der geleisteten Arbeiten

Die Notwendigkeit und Angemessenheit der geleisteten Arbeit wird durch die dargestellten Ergebnisse und den Vergleich mit der ursprünglichen Aufgabenstellung deutlich.

II.4 Verwertungsplan und Nachhaltigkeit

Wie bereits im Abschnitt *I.2 Stand der Technik* beschrieben, ist das Softwarepaket OpenFOAM für eine große Anzahl von HPC-Nutzern aus verschiedenen wissenschaftlichen Gebieten von Interesse. Die auf Blue Gene/Q portierte OpenFOAM Version ist unmittelbar den nationalen sowie europäischen HPC Nutzern am JSC zur Verfügung gestellt worden und wird über die Restlaufzeit der JUQUEEN weiter gepflegt. Die Portierung verschiedener Versionen von OpenFOAM auf das *General purpose* HPC-System JURECA stellt eine attraktive Alternative zur Installation auf JUQUEEN dar, die von verschiedenen universitären aber auch industriellen Benutzergruppen genutzt wird. Modelle, die nicht komplette Baugruppen wie z.B. Turbinen in hoher Auflösung darstellen, sind auf JURECA Dank der Vielzahl der verfügbaren Rechen-Cores, der Größe des Hauptspeichers pro Knoten sowie den einsetzbaren Hardware-Beschleunigern schnell und effizient berechenbar. Noch größere Modelle, die erst am Anfang ihres industriellen Einsatzes stehen, können auf dem massiv-parallelen System JUQUEEN gerechnet werden. Mit der Pflege und Weiterentwicklung von OpenFOAM auf beiden Plattformen stehen für alle Ansprüche passende Simulationslösungen zur Verfügung.

Ferner können die Software OpenFOAM sowie die im Rahmen dieses Projektes gesammelten Erfahrungen für zukünftige Kooperationen mit dem Industriepartner Siemens AG aber auch mit anderen (Industrie-)Partnern des JSC genutzt werden

II.5 Erfolgte oder geplante Veröffentlichungen des Ergebnisses

1. Beck, C. ; Grotendorst, J. ; Körfgen, B., „HPC for Climate-friendly Power Generation“, Innovatives Supercomputing in Deutschland 12(1), 70-71 (2014)

2. Webseite zum Projekt „CEC - Development of new combustion technologies for climate-friendly power generation“

http://www.fz-juelich.de/ias/jsc/EN/Research/Projects/_projects/cec.html

3. Dieser Schlussbericht entspricht der Veröffentlichung zum Ergebnis des Vorhabens bei der Technischen Informationsbibliothek Hannover (TIB).

Literatur

[1] OpenFOAM, © 2011-2016 The OpenFOAM Foundation, http://www.openfoam.org

[2] Blue Gene/Q, IBM Corp., http://www-03.ibm.com/systems/technicalcomputing/solutions/bluegene

[3] The MPI Forum, MPI: A Message-Passing Interface Standard. Version 2.2. September 4, 2009, www.mpi-forum.org

[4] Victorian Life Sciences Computation Initiative (VLSCI) 2011, https://vlsci.org.au/software/openfoam-0

[5] Free Software Foundation, Inc., GCC, the GNU Compiler Collection, http://gcc.gnu.org

[6] IBM XL compilers, IBM Corp. © 1991, 2007, http://www-01.ibm.com/software/awdtools/xlcpp

[7] Performance-Analyse Tool Scalasca , http://www.scalasca.org/

[8] Grafisches Performance-Analyse Tool Vampir, https://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/vampir

[9] Tuning und Analyse Tool TAU, http://www.cs.uoregon.edu/research/tau/home.php

[10] Integrierte Tool-Suite HPCToolkit, http://hpctoolkit.org/

[11] Stack Trace Analyse Tool (STAT), http://www.paradyn.org/STAT/STAT.html

[12] General Parallel File System (GPFS), IBM Corp., http://www-03.ibm.com/systems/storage/spectrum/scale/index.html

[13] „Cuda for FOAM Link“ (cufflink), https://github.com/chegdan/cufflink-library

[14] CUSP-Bibliothek, Nvidia Corp., https://developer.nvidia.com/cusp

[15] Thrust-Bibliothek, Nvidia Corp., http://docs.nvidia.com/cuda/thrust/