

**TU-Dortmund Fakultät für Mathematik**

# **Abschlussbericht**

**Studienprojekt-Technomathematik 2015-2016**

Studienprojektgruppe Technomathematik

16. März 2016

Betreuer: Dipl.-Inf. Markus Geveler

# Inhaltsverzeichnis

<b>1 Motivation und Problembeschreibung</b>	<b>3</b>
<b>2 Einleitung</b>	<b>3</b>
<b>3 Hardware</b>	<b>4</b>
3.1 Hardwareauswahl und Motivation . . . . .	4
3.2 Rackdesign . . . . .	4
3.2.1 Anordnung der Rechenknoten . . . . .	4
3.2.2 Strom und Netzwerkanbindung . . . . .	5
3.3 Dashboard . . . . .	6
3.3.1 Einrichtung Webserver . . . . .	6
3.3.2 Sammeln der Messdaten . . . . .	7
<b>4 Software</b>	<b>8</b>
4.1 Löser . . . . .	8
<b>5 Ergebnisse</b>	<b>9</b>

# 1 Motivation und Problembeschreibung

## 2 Einleitung

Im Bereich der hardware-orientierten Numerik wurden bereits in den vergangenen Jahren einige Untersuchungen gemacht. So wurde in den Jahren 2010 und 2011 durch Geveler et al. die Lattice-Boltzmann-Methode zur Lösung von Navier-Stokes- und Flachwassergleichungen auf unterschiedlicher Hardware analysiert. Es stellte sich heraus, dass GPUs Multi-Core CPUs mit einem Speedup von bis zu acht überlegen sind, ohne dabei an Genauigkeit der numerischen Lösung einzubüßen. In „Efficient Finite Element Geometric Multigrid Solvers for Unstructured Grids on GPUs“ wurde bereits 2011 nach dem Paradigma der hardware-orientierten Numerik ein geometrisches Mehrgitterverfahren mit finiter Elemente Assemblierung entwickelt, welches nur aus einer Reihe von Sparse Matrix-Vektor Multiplikationen besteht. Durch diese Implementierung, welche keinen Leistungsverlust nach sich zog, war es möglich, die Parallelität von Rechenarchitekturen noch besser auszunutzen. Besonders durch die Verwendung von GPUs anstatt von Multi-Core CPUs ergab sich ein durchschnittlicher Speedup von 8. Geveler et al. betrachtete 2013 in „Towards a complete FEM-based simulation toolkit on GPUs: Geometric Multigrid solver“ die Lösung partieller Differentialgleichungen mit finiten Elementen und Mehrgitterverfahren auf unstrukturierten Gittern. Es wurde gezeigt, dass sich die Laufzeit der Anwendung erheblich verbessern lässt, sobald GPUs anstatt von Multi-Core CPUs benutzt wurden. Des Weiteren wurde in „Energy efficiency vs. Performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster“ ein Cluster aus 96 ARM Cortex-A9 Dual-Core Prozessoren einer Rechenarchitektur, basierend auf x86-Prozessoren gegenüber gestellt. Dabei wurde die verbrauchte Energie zur Lösung von drei wissenschaftlichen Anwendungen, einem Finite-Elemente Code mit Mehrgitter-Löser, einer strömungsmechanischen Anwendung und einem Code zur Ausbreitung von Schallwellen mit Hilfe der Spektral-Elemente Methode, analysiert. Schließlich wurde gezeigt, dass die verbrauchte Energie zur Lösung erheblich gesenkt werden kann, im Einklang mit einer akzeptablen Erhöhung der Laufzeit. In „Porting FEASTFLOW to the Intel Xeon Phi: Lessons Learned“ zeigte Geveler et al. die Leistungsverbesserung von axpy-Vektor Operationen und Sparse Matrix-Vektor Multiplikationen durch Nutzung des Intel Xeon Phi Koprozessors. Diese Analyse, sowie in „FFF2: Future-proof High Performance Numerical Simulation for CFD with FEASTFLOW (2)“ durchgeführten Untersuchungen, welche sich mit der Entwicklung von numerischen Methoden zur parallelen Lösung von partiellen Differentialgleichungen für realitätsnahe industrielle und wissenschaftliche Probleme befassen, basierten auf der Software Infrastruktur „FEASTFLOW“. Dieses Paket umfasst Software zur numerischen Lösung der Navier-Stokes-Gleichungen in 2D und 3D.

### 3 Hardware

Der zweite wichtige Aspekt beim Aufbau eines unkonventionellen Supercomputers ist der physikalische Aufbau des Computers. Hierbei muss man darauf achten ein Gleichgewicht zwischen den Faktoren: Energieverbrauch, Rechengeschwindigkeit und Kühlung zu finden. Zusätzlich soll unabhängig von der Größe des Clusters eine einfache Lösung für das Monitoring bereitgestellt werden.

#### 3.1 Hardwareauswahl und Motivation

HIER NOCH INHALT EINFÜGEN! (wenn nicht redundant mit der Einleitung)

#### 3.2 Rackdesign

Für ein Jetson-TK1-Cluster mit insulärer Stromversorgung ist das Design des Racks der entschiedene Punkt um auch unter dauerhafter Höchstbelastung eine stabile Funktion der Rechenknoten zu gewährleisten. Durch die Verwendung eines Lithium-Ion-Akkus zur Energiespeicherung und das Risiko eines Kurzschlusses durch Kondensationswasser muss ein geschickt aufgebautes Rechencluster gewährleisten, dass die Temperatur der gesamten Anlage in dem stark restringierten Bereich von 18°C bis 40°C gehalten wird.

##### 3.2.1 Anordnung der Rechenknoten

Hinsichtlich der oben beschriebenen Problematik ist die Anordnung der Knoten das zentrale Instrument mit dem man die Bildung potenzieller Wärmenerste gegen den Platzverbrauch des Clusters abwägen kann.

Wegen zu hoher Energiekosten muss auf den Einsatz einer Wasserkühlung verzichtet werden um eine möglichst hohe Energieeffizienz zu erreichen. Die einfachsten Ansätze zum Aufbau des Supercomputers, welche sich mittels Luftkühlung umsetzen lassen, sind:

- 1) Rechenknoten in horizontalen Schichten anzurufen
- 2) Rechenknoten vertikal anzurufen und nebeneinander aufzustellen

Je nach Aufbau ergeben sich dabei einige Vor- und Nachteile für den Großrechner, welche hier anhand der Wärmebildern gezeigt werden.

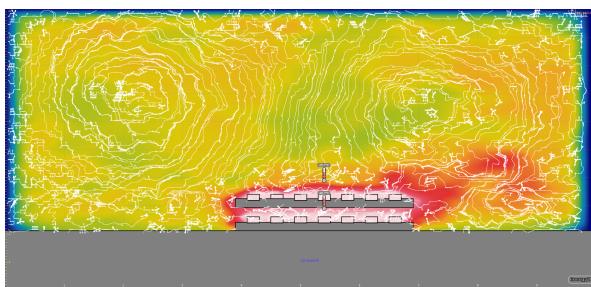


Abbildung 1: Horizontaler Aufbau

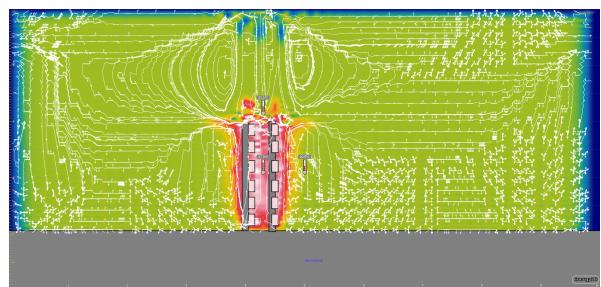


Abbildung 2: Vertikaler Aufbau

Vergleicht man diese Beiden Ansätze, so sieht man, dass im horizontalen Szenario zwischen den Platten, auf denen die Rechenknoten angebracht sind, Wärme-Nester entstehen. Zudem liefert die Analyse des Stromlinien Diagramms, dass besonders in der Mitte der Platten die Wärme nur schlecht abtransportiert werden kann.

Im Gegensatz dazu ist der Wärmeabtransport beim vertikalem Ansatz durch aufsteigende warme Luft besser möglich. Hierbei sollte allerdings bemerkt werden, dass durch die schiefe Lage des Ventilators die Gravitationskraft Unregelmäßigkeiten bei der Rotation verursachen könnte, was schlussendlich zu einer verkürzten Lebensdauer des Ventilators führen könnte.

Um die Vorteile der Beiden Ansätze zu kombinieren und gleichzeitig für Wartbarkeit des Rechenclusters zu erhöhen wählt man einen Ansatz, bei dem die Bords in einer Doppelhelix-Struktur angeordnet werden.

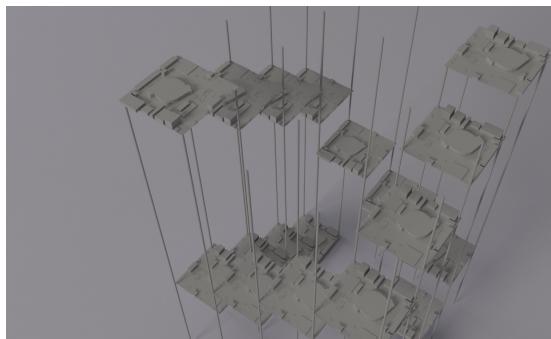


Abbildung 3: Render des Serveraufbaus

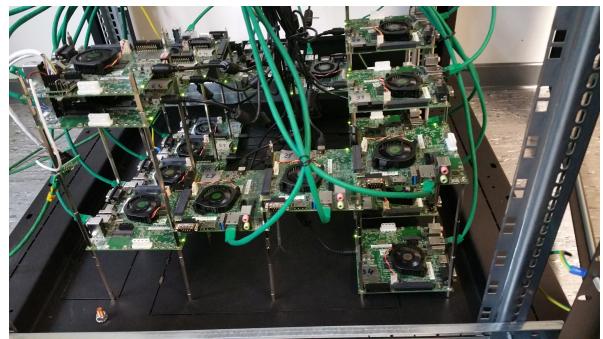


Abbildung 4: Serveraufbau (7. März 2016)

Wie auf diesen Abbildungen zu erkennen ist wird durch die Verwendung der Doppelhelix-Struktur ein größerer Abstand zwischen den Bords für eine verbesserte Kühlung, bei gleichzeitig geringem Platzverbrauch ermöglicht.

### 3.2.2 Strom und Netzwerkanbindung

Über die Anordnung hinaus muss nun die Stromversorgung und die Netzwerkanbindung der Bords geregelt werden. Eine besondere Herausforderung hierbei ist die Anordnung der Netzteile der einzelnen Rechenknoten. Um zu gewährleisten, dass sich durch das Aufheizen der Netzteile keine gefährlichen Wärmestrukturen bilden wird eine spezielle Halterung verwendet. Unter Nutzung moderner 3D-Druck Technologien, wie sind häufig in der 'Maker-Szene' eingesetzt werden, wurde eine Halterung produziert, welche trotz geringem Materialaufwand eine sehr hohe Stabilität und eine gute Luftzufuhr ermöglicht.

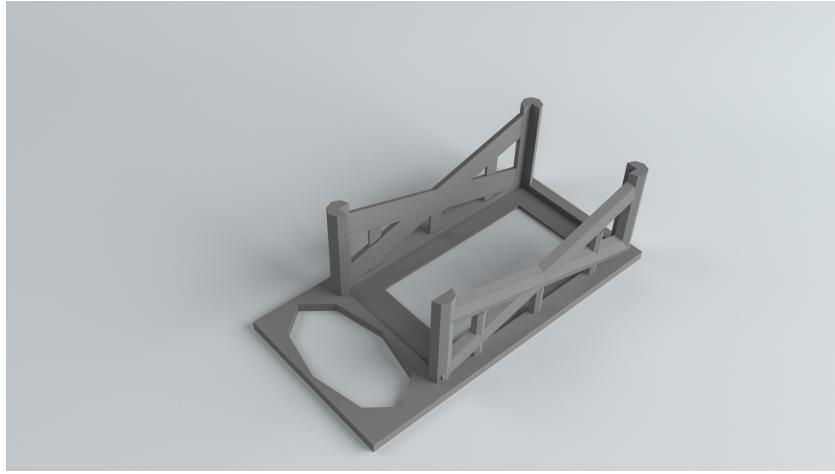


Abbildung 5: Halterung für die Netzteile

Hinsichtlich des Aspektes 'Green-Computing' ist es wichtig ein Druck Filament zu verwenden, welche Bio-Kompatibel ist. Hierfür bietet sich der aus Mais-stärke gewonnene Biokunststoff PLA (Polylactide) an. Polyacide hat einen Schmelzpunkt von 150°C bis 160°C und ist daher ohne weiteres für die Halterung von Netzteilen verwendbar.

### 3.3 Dashboard

Um das Monitoring und die Fernwartung des Clusters möglichst komfortabel zu bewerkstelligen wurde ein Dashboard eingerichtet. Hierfür ist ein RaspberryPi der ersten Generation ausreichend.

#### 3.3.1 Einrichtung Webserver

Um die gemessenen Daten verwalten und darstellen zu können wird auf dem RaspberryPi ein LAMP-Server eingerichtet (Linux-Apache-MySQL-PHP). Die Daten werden folglich in der MySQL Datenbank abgespeichert und über PHP-Skripte ausgelesen und verarbeitet. Die Darstellung der Daten erfolgt mittel HTML, CSS und Javascript. Des Weiteren muss der RaspberryPi so konfiguriert werden, dass erstens das Webinterface von außen einsehbar ist , und vor allem, dass man von außen in die Datenbank schreiben kann.

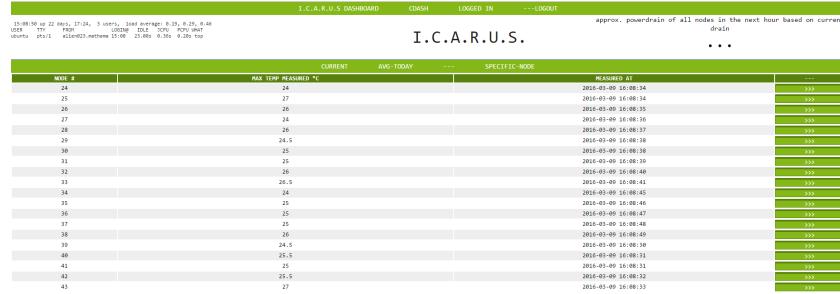


Abbildung 6: Dashboard Frontend

### **3.3.2 Sammeln der Messdaten**

Die Temperaturen können lokal auf den einzelnen Boards ausgelesen werden. Dies wird verwendet um mittels C++ ein Programm laufen zu lassen, welches sich auf den einzelnen Rechenknoten nacheinander einloggt, die Temperatur Daten an verschiedenen Stellen des Jetson-Boards ausliest und anschließend eben jene in die MySQL-Datenbank auf dem Webserver einfügt.

Um die restringierte Hardware des Raspberry Pi's nicht zu überlasten wird das Programm auf dem Gateway-Knoten des Clusters ausgeführt, weshalb der oben Genannte Zugriff von außen des RaspberryPi's wichtig ist.

Um qualitative Aussagen über die Energieeffizienz machen zu können wurde eine PDU angeschafft, mit welcher man den Stromverbrauch der einzelnen Knoten messen können sollte.

— HIER TEXT WARUM DAS MIT DER PDU NICHT GEHT —

— HIER TEXT BZGL NEUER MESSMETHODEN —

## 4 Software

### 4.1 Löser

Das als Löser verwendete Verfahren des Projektes ist das sogenannte BiCGStab-Verfahren (engl: „biconjugate gradient stabilized method“), welches zur Klasse der Krylov-Unterraum-Verfahren zählt. Diese iterative Methode wurde von H.A. van der Horst als Löser für nicht-symmetrische lineare Gleichungssysteme entwickelt und benötigt somit keine zusätzlichen Anforderungen an die Systemmatrix  $A$  wie zum Beispiel beim CG-Verfahren. Sein Name ist davon abgeleitet, dass die im nicht-vorkonditionierten Algorithmus verwendeten Residuen biorthogonal sind, also  $(r_i, \hat{r}_j) = 0 \forall i \neq j$ , und die Suchrichtungen bikonjugiert sind bezüglich der Systemmatrix  $A$ , also  $(Ap_i, \hat{p}_j) = 0 \forall i \neq j$ . Man entschied sich für dieses Verfahren, da das konkurrierende GMRES-Verfahren weniger speichereffizient arbeitet und das BiCGStab-Verfahren leichter zu implementieren ist. Außerdem kann es mit beliebigem Vorkonditionierer und ohne Vorkonditionierer implementiert werden. Der Nachteil des angewendeten Verfahrens ist, dass die Konvergenz nicht allgemein bewiesen ist. So kann es im Falle von einer Systemmatrix mit großen komplexen Eigenwerten zur Stagnation des Algorithmus kommen, ohne dass eine gute Näherungslösung ermittelt wurde.

- 
1.  $r_0 = b - Ax_0$
  2.  $\hat{r}_0 = r_0$
  3.  $\rho_0 = \alpha = \omega_0 = 1$
  4.  $v_0 = p_0 = 0$
  5. for  $i=1,2,\dots$ 
    6.  $\rho_i = (\hat{r}_0, r_{i-1})$
    7.  $\beta = (\rho_i / \rho_{i-1})(\alpha / \omega_{i-1} - 1)$
    8.  $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1}v_{i-1})$
    9.  $y = K^{-1}p_i$
    10.  $v_i = Ay$
    11.  $\alpha = \rho_i / (\hat{r}_0, v_i)$
    12.  $s = r_{i-1} - \alpha v_i$
    13. Wenn  $\|s\|$  klein genug, setze  $x_i = x_{i-1} + \alpha p_i$  und beende
    14.  $z = K^{-1}s$
    15.  $t = Az$
    16.  $\omega_i = (K_1^{-1}t, K_1^{-1}s) / (K_1^{-1}t, K_1^{-1}t)$
    17.  $x_i = x_{i-1} + \alpha y + \omega_i z$   
Wenn  $\|x - x_i\| < TOL$  wird das Verfahren beendet
    18.  $r_i = s - \omega_i t$

---

Der von rechts vorkonditionierte BiCGSTAB-Algorithmus NUMMER startet ausgehend von einem Anfangsvektor  $x_0$ , welcher immer als Nullvektor implementiert wurde. Des Weiteren wurde die maximale Iterationszahl auf 10.000.000.000 gesetzt und die Toleranz mit  $TOL = 10^{-9}$  festgelegt. Der Vorkonditionierer hat dabei die allgemeine Form:  $K = K_1 K_2 \approx A$

## 5 Ergebnisse