Inhaltsverzeichnis

1 Einleitung

Paralleles Rechnen stellt Informatiker, Mathematiker und Elektrotechniker vor viele Herausforderungen. Zum einen muss die entsprechende Computer-Hardware dafür ausgelegt sein, effizient und problemlos parallele Probleme zu lösen. Zum anderen muss eine entsprechende Programmiergrundlage existieren damit eine gegeben Hardware-Plattform auch auf einer hohen, abstrakten Programmierebene angesprochen werden kann[1].
Der rasante Fortschritt der Computer-Hardware ermöglicht es, dass es heutzutage recht einfach und günstig ist, ein paralleles System anzuschaffen. Mit dem Einsatz von Multicore-Prozessoren in Endbenutzer Hardware sind auch die ersten parallelen Ansätze in der Mitte der Gesellschaft angekommen. Mit dem Fortschritt der Prozessortechnik und der damit einhergehenden hohen Taktung, ist es möglich sehr viele Befehle pro Sekunde auszuführen. Demzufolge muss ein Konzept erfunden werden, welches möglichst viele Befehle für den Prozessor erzeugt aber nur wenig Programmier-/Schreibarbeit für den Programmentwickler bedeutet.
Im Sinne dieser Seminararbeit, wird im folgenden ein Programmiermodell zur Behandlung von Parallelen Problemen behandelt. Das Bulk Synchronous Parallel (BSP) Modell, dies steht für Massensynchrone Parallelrechner welches den Ansatz hat, dass ein Vielzahl von Prozessoren ihren eigenen Speicher haben(MIMD-Multiple Instruction Multiple Data)[2]. Im folgenden werden einige Grundprinzipien des BSP erläutert und anhand eines simplen Programmierbeispiels anschaulich am Code erklärt.

2 Programmiermodell

Ein Programmiermodell ist eine Notwendige Abstraktion der Hardware. Es schlägt die Brücke zwischen der eigentlichen Programmiersprache und der zugrundeliegenden ausführenden Hardware[6]. Es ist sinnvoll Abstraktionsebenen zu bilden, man möchte den Programmierer entlasten und die Maschine aber dafür auslasten.

Ein Programmiermodell sollte auch gewissen Ansprüchen genügen, wie in [1] gezeigt gibt es 4 Anforderungen an ein Programmiermodell:

- Übertragbar - Ein Programm sollte ohne eine komplette Umstrukturierung, oder auch ohne neu Design auf eine andere Plattform übertragbar sein.
- Effizient - Modelle sollen die Gesamte Leistung, Möglichkeiten des Parallelensystems ausnutzen.
- Einfachheit - Je Einfacher das Modell gehalten ist, mit seinen Befehlen und Regeln, umso einfacher ist es für einen Programmierer schnell, fehlerfreien und damit auch effizienten Code zu schreiben.
- Berechenbare Laufzeit - Die Laufzeit eines Codeentwurfs vorauszuberechnen, ist essentiell dafür um aus vielen Möglichkeiten, ein Problem zu Implementieren, die beste herauszufinden.

Im folgenden wird das Programmiermodell BSP vorgestellt, welches jede oben genannte Eigenschaft besitzt. Ein Programm welches dem BSP-Modell zugrunde liegt, ist auf jedes BSP geeignete Verteiltesystem portierbar. Es wird die möglichkeit gegeben Rechenlast gleichmäßig über das System zu verteilen. Dem BSPlib Standard liegen nur rund 20 Befehle zu Grunde[1], demzufolge ist es recht überschaulich gehalten und erfüllt den Punkt der Einfachheit. Zuletzt besitzt der BSPlib Standard ein Rechenmodell womit die Performance eines Systems unter bestimmten Umständen bestimmt werden kann.

3 Bulk Synchronous Parallel

3.1 Geschichte

Das Modell BSP wurde von einem Britischen Informatiker namens Leslie Valiant, an der Harvard Universität entwickelt. Über die 1980er Jahre entwickelte er seine Vorstellung über ein Paralleles Programmiermodell was davon ausging, dass eine Kommunikation und Synchronisation in einem Parallelen Rechner nicht gleich gegeben oder sogar trivial ist[3]. Nachdem Valiant seine Arbeit 1990 veröffentlichte arbeitete er mit Bill McColl an der Verfeinerung seiner Ideen und Konzepte. McColl führte Mitte der 1990er Jahre ein Internationales Forscherteam an der Oxford Universität. 1997 veröffentlicht McColl seine Arbeit in einem Standard der heutzutage als Oxford BSPlib Standard bekannt ist[4]. In den folge Jahren gibt es verschiedene Umsetzung dieses Standards und auch einige Erweiterungen. Der Oxford BSPlib Standard, fand seine erste Implementierung in Fortran und wurde später in heutzutage gängigen Programmiersprachen implementiert, wie C,C++ oder Java[5].

Abbildung 1: Leslie Valiant

3.2 Aufbau einer BSP Maschine

Um ein BSP-System zu Implementieren werden zunächst einige Dinge benötigt, zum einem handelt es sich um einen MIMD Ansatz, also benötigt man mehrere Prozessoren die einen dedizierten Speicher haben.[1]
Im BSP wird auch eine Kommunikation zwischen den Prozessoren, oft in einem Netzwerk, benötigt. Wie man dieses Netzwerk gestaltet liegt zunächst in der Hand des Systemerstellers. Um eine Datenübertragung zwischen den Prozessoren zu gewährleisten müssen die Verbindungen über einen Router laufen, der dafür sorgt das die Prozessoren sich untereinander verständigen können. Die Kommunikation ist getrennt von jeglicher Synchronisation. Die Synchronisation ist teil der Barriere welche garantiert, dass der Datenaustausch zwischen den Prozessoren abgeschlossen ist und erst dann die übertragenen Daten für den einzelnen Prozessor zugänglich werden. Demzufolge arbeiten alle Prozessoren erst weiter, wenn der Datenaustausch abgeschlossen ist. Diesen Vorgang nennt man im BSP-Modell "Barrier-Synchronisation".[1]

3.3 BSP Algorithmen

Wie man schon aus dem Aufbau einer BSP Maschine erkennen kann, besteht auch der Algorithmus eines BSP Programmes aus 3 Teilen. Wie in Abbildung 2 gezeigt wird zuerst die Rechenarbeit die jedem Prozessor zugeteilt wurde abgearbeitet, dann beginnen die Prozessoren untereinander Daten zu verschicken und im letzten Schritt werden diese Daten dann lokal für die einzelnen Prozessoren sichtbar. Diesen Ablauf nennt man einen *Superstep*.
Ein BSP-Algorithmus besteht aus einem oder mehreren Supersteps. In einem Superstep werden möglichst viele Aktionen bzw. Befehle versucht unter zu bringen, damit man alle verteilten Prozessoren möglichst gut auslasten kann, daher auch "Bulkßu deutsch "Masse oder auch großes Bündel".

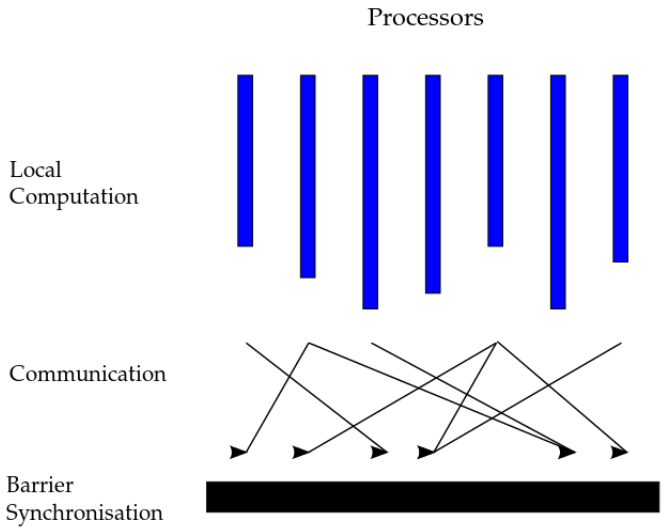

Abbildung 2: Ein Superstep

Im folgenden wird an einem Code Beispiel der Ablauf eines BSP Algorithmus deutlich erklärt. Verwendet wurde dabei eine Bibliothek von Albert-Jan N. Yzelman, welche er in seiner Promotion entwickelte und auch später weiter führte. Die Entwickelte Bibliothek MulticoreBSP für Java und C, ermöglicht es auf modernen Mehrkernprozessoren ein BSP System zu implementieren.
Hier wurde ein Simpler ansatz gewählt um die Vorgehensweise zu verdeutlichen. Es wird in einem Superstep über zwei Prozessoren zunächst die Oberfläche und das Volumen einer gegebenen Kugel berechnet. Nach diesem Superstep wird eine Barrierensynchronisation durchgeführt und der erste Prozessor berechnet dann im zweiten Superstep wie viele dieser Kugeln in einen Kubikmeter Würfel passen.

- Aufbau, Zweck, Einsatz
- Vorteile

4 Netzwerke in BSP

- Erklärung von TCP/IP mit Bezug auf BSP
  - Packing
  - Destination Schedule
  - Bestätigungen(Acknowladgment)
  - Fehler Verträglichkeit(Error recovery)
- Was wird durch BSP wie optimiert.

5 BSP in der Softwareentwicklung

- Datamining(Bagging)
- Neurale Netzwerke und Logic Programming

- Eigene Praktische Aufgabe

6 Fazit

- Vorteile
- Nachteile
- Resümee Nutzen

7 Referenzen

1 R.Correa et al (eds), Models for Parallel and Dirstibuted Computation. Theory, Algorithmic Techniques and Applications S.: 85-115

2 https://de.wikipedia.org/wiki/Massively_Parallel_Processing (Abruf 01.01.2017)

3 Leslie G. Valiant, A bridging model for parallel computation, Communications of the ACM, Volume 33 Issue 8, Aug. 1990

4 http://www.cse.unt.edu/~tarau/teaching/parpro/papers/Bulk%20synchronous%20parallel.pdf (Abruf 01.01.2017)

5 http://www.bsp-worldwide.org/implmnts/oxtool/

6 https://www.math.tu-cottbus.de/~kd/parallel/vorl/vorl/node18.html

8 Abbildungen

- Abbildung 1 http://cacm.acm.org/magazines/2011/6/108643-beauty-and-elegance/fulltext
- Abbildung 2 https://upload.wikimedia.org/wikipedia/en/thumb/e/ee/Bsp.wiki.fig1.svg/768px-Bsp.wiki.fig1.svg.png