

Gedächtnisprotokoll

Modul: 01727 Parallele Programmierung und Grid Computing

Prüfer: Herr Prof. Dr. Keller

Datum: 6. April 2011

- Wozu parallele Programmierung, wenn heutige PCs bereits sehr leistungsstark sind?
 - Größere Probleme lösen in annehmbarer Zeit
 - Probleme schneller lösen
 - Probleme mit höherer Genauigkeit lösen
 - Entwicklung am Hardware-Markt: Multicore-Prozessoren, dadurch praktisch überall Parallelrechner (und Leistungsreserven) vorhanden

- Warum Multicore-Prozessoren und nicht Taktratenerhöhung?
 - Kühlung von Prozessoren immer schwieriger -> da mehr Abwärme produziert wird
 - Verkleinerung der Chipelemente aufwendig und teuer (obwohl sie höhere Effizienz bringen und weniger Abwärme erzeugen)

- Was hat Amdahl gesagt?
 - Speedup durch sequentiellen Anteil des Programms stark begrenzt (s. Kurstext)

- Wo ist der Pferdefuß?
 - Annahme: Konstante Problemgröße

- Jetzt ein Beispiel zu paralleler Programmierung;
Wie kann man Matrix-Vektor-Multiplikation mittels MPI für 2 Prozessoren parallelisieren?
 - Aufteilung der Matrix zeilenweise auf Prozessoren
 - Aufteilung des Vektors
 - Berechnung von Teilergebnissen
 - Austausch der Vektorelemente
 - Berechnung der Endergebnisse (liegen verteilt auf den Prozessoren)

- Multiplikation von Matrizen mittels MPI erklären (Cannon's Algorithmus).
 - Initiale Ausrichtungsphase -> Prozessoren müssen „kompatible“ Elemente erhalten, die multipliziert werden können -> erklärt
 - Austausch von Elementen beider Matrizen während Kommunikationsrunden

- Parallelisierung des Single Source Shortest Path Problems erklären
 - Dünner Graph -> Adjazenzliste als Eingabe -> Aufteilen auf Prozessoren
 - Distanzvektor ebenfalls aufteilen
 - Jeder Prozessor verwaltet eine Priority Queue
 - Knoten aus Priority Queue mit kleinster Distanz zum Startknoten entnehmen
 - Distanz der Nachbarknoten aktualisieren -> ggf. Kommunikation zu anderen Prozessen nötig
 - Problem: Wenn Distanz eines Knoten aktualisiert wird, der aus Priority Queue entnommen wurde, dann muss dieser wieder der Priority Queue hinzugefügt werden -> spekulativer Parallelismus, abhängig vom Graph

- Wie hängt Grid Computing mit paralleler Programmierung zusammen?
 - Parallele Programme i. A. nicht für Grids geeignet, da Bandbreite des Internets begrenzt ist, sowie über eine hohe Latenz verfügt -> Kommunikation wird teurer
 - Besser: viele sequentielle Programme ausführen lassen
 - Zusammenschluss von Ressourcen über das Internet
 - Keine Interaktion zwischen Benutzer und Job
 - Eingabedaten, Ausgabedaten werden in Dateien abgelegt

- Was ist ein Scheduler?
 - Verteilt Tasks auf Ressourcen entsprechend der Scheduler-Strategie

- ETC-Matrix erklären
 - Zeilen entsprechen Tasks
 - Spalten entsprechen Maschinen
 - Elemente entsprechen voraussichtliche Laufzeiten eines Tasks auf einer Maschine
 - Werden für Scheduler-Strategien verwendet

- Eigenschaften von ETC-Matrizen
 - Maschinen- / Task-Heterogenität
 - Konsistenz

- Eine Scheduler-Strategie erklären
 - Habe Min-Min-Heuristik erklärt, da diese ein sehr gutes Ergebnis bei geringer Berechnungszeit aufweist

- Was ist Condor?
 - Softwarepaket zum Aufbau / Betrieb von Clustern
 - Cluster -> Zusammenschluss einzelner Computersysteme zwecks verschiedener Zielstellungen (z. B. Laufzeit paralleler Programme minimieren - HPC)

- Beispiel für Cluster und Grid nennen.
 - Cluster: Jugene – ca. 75.000 Prozessoren
 - Grid: BOINC – ca. 500.000 PCs -> Ziel: möglichst viele ungenutzte Ressourcen nutzen

Prüfungsfragen Kurs 1727 - Parallel Programming + Grid Computing

Prüfer: Prof. Dr. J. Keller

Datum: 03.02.2011

Note: 1,7

Welche Programmiermodelle für Parallel-Rechner gibt es?

Für was braucht man Kommunikatoren bei MPI?

Wie lautet das Gesetz von Amdahl? Was ist seine Annahme?

Wie unterscheidet sich die Behauptung von Gustafson?

Wie wird eine Matrix in einem MPI-System mit einem Vektor multipliziert?

Wie funktioniert Cannons-Algorithmus zur Matrizen Multiplikation?

Wie unterscheiden sich Grid-Systeme vom Parallelen Programmieren?

Was ist Condor?

Was steht in einer Job-Beschreibungsdatei von Condor?

Was ist eine ETC-Matrix?

Was bedeutet konsistente ETC-Matrix?

Was macht die ETC-Scheduler-Strategie?

Nennen sie eine Scheduler Strategie die eine ETC-Matrix benötigt und erklären Sie sie.

Gedächtnisprotokoll - Fachprüfung Kurs 1727 „Parallel Programming“ - Bachelor Informatik

Prüfer: Prof. Dr. J. Keller

Datum: 07.04.08

Note: sehr gut

Die Prüfung hat im Anschluß an mein Vortrag zum Thema meiner Abschlußarbeit stattgefunden, insofern hat sich die Begrüßungsrunde erübrigt. Wir kamen dann gleich zu den Fragen:

Wieso will man überhaupt parallelisieren, wo heute selbst PC enorm leistungsfähig sind?

Um Probleme schneller und genauer berechnen zu können / größere Instanzen der Probleme berechnen zu können. Hier habe ich auch das Buzzword „grand challenge problems“ eingeworfen ;-)

Wie lautet das Gesetz von Amdahl?

Erklärt.

Wieso lohnt es sich trotzdem zu parallelisieren?

Amdahl's Annahme ist zwar theoretisch richtig, praktisch aber nicht immer relevant (Gustafson). O-Ton von Prof. Keller: Amdahl wollte doch Mainframes verkaufen ;-)

Welche zwei Paradigmen gibt es?

Shared Memory / Message Passing.

Nun scheint Message Passing ein einfacher Ansatz zu sein. Welche Probleme können dort auftreten?

Blockierende Aufrufe / Deadlocks.

Können Sie ein einfaches Beispiel für die Entstehung eines Deadlocks bei Message Passing konstruieren?

Prozess A will eine Nachricht an B senden, B will die Nachricht von A aktiv anfordern und beide kommunizieren über gleichen Puffer.

Bei Shared-Memory-Systemen: Wie wird dort der Zugriff auf gemeinsame Daten geregelt?

Vorhandene Mechanismen aufgezählt: Spinlocks, Mutexe / Semaphoren, Monitore, Barrieren.

Wie kann die parallele Berechnung der Mandelbrot-Sets effizient gestaltet werden (Shared Memory)?

Dynamische Taskzuweisung mit Loadbalancing und Work Sets.

Sollen dann einzelne Pixel an die Prozessoren verteilt werden?

Es muss eine optimale Größe der Teilaufgaben gefunden werden. Bei zu kleinen Arbeitspaketen hoher Kommunikationsaufwand, bei zu großen niedrige Granularität der Zuweisungen.

Wie kann die Multiplikation einer Matrix mit einem Vektor auf einem 2-CPU Multiprozessor ausgeführt werden?

Matrix vertikal halbieren, Vektor horizontal halbieren, Matrixhälften mit Vektorhälften multiplizieren, Ergebnisse addieren.

Wie funktioniert der Cannon-Algorithmus?

Erklärt: Ein Datapoint jeder Matrix pro CPU ($n \times n$ Torus), initiales Alignment, $n-1$ Shifts, Akkumulation der Ergebnisse in jedem Node.

Wie kann Quicksort auf einer Hypercube implementiert werden?

Rekursives Halbieren der unsortierten Menge, Versenden der Hälfte $<$ Pivot an die Nachbarnodes in der gleichen Dimension.

Wie funktioniert die näherungsweise Berechnung der Pi-Zahl mit Hilfe der Monte-Carlo Methode?

Erklärt: Menge von zufällig gewählten Datapoints in ein Quadrat (Seitenlänge 2) mit Kreis (Radius 1) setzen. Verhältnis der Punkte innerhalb des Kreises zu Gesamtpunkte beträgt $\text{Pi} / 4$.

Dann war der offizielle Teil der Prüfung zu Ende. Da es meine letzte Prüfung in dem BoCS-Studiengang war, hat mir Prof. Keller bei Mitteilung der Note gleich zum erfolgreichen Abschluß des Studiums gratuliert und gefragt, ob ich jetzt den Master mache ;-). Die Fragen sind natürlich nicht so wie oben gestellt worden, Prof. Keller leitet die Fragen meistens mit einem praktischen Beispiel ein.

Alles in allem, eine recht entspannte Prüfungsatmosphäre, bezüglich der Fragen gab es keine Überraschungen. Mit der Benotung bin ich absolut zufrieden und kann Prof. Keller als Prüfer uneingeschränkt empfehlen.

Prüfungsprotokoll Diplom, Kurs 1727 – Paralleles Programmieren

Prüfer: Prof. Dr. Keller

Datum: 12.12.2007

Dauer: ca. 30 min

Fragen:

1. Amdahls Gesetz: Ansatz ($t_s = const.$) und Folgen ($S_{p \rightarrow \infty} = \frac{1}{f}$).
2. Arten der Parallelisierung (gemeint waren Shared Memory und Message Passing Computer Systeme). Unterschied erklärt, jeweils Vor- und Nachteile aufgezählt.
3. Frage nach Posix Threads und wie sie sich von Prozessen unterscheiden (gemeinsamer Heap).
4. Frage, wie man eine Matrix mit einem Vektor auf einem Shared Memory System mit zwei Prozessoren multiplizieren würde.
5. Im Unterschied dazu Mandelbrot Set: Menge vorstellen, das Problem der unterschiedlichen Laufzeiten erwähnen und die Begriffe Work Pool + Load Balancing erklären.
6. Strategien zur Parallelisierung (vor allem Divide and Conquer).
7. Frage nach einem Sortieralgorithmus, der gut auf Shared Memory Systemen zu realisieren ist (Mergesort: als Grund wurde von ihm das Pivotelement genannt). Zusätzlich Bucketsort und Bitonisches Sortieren erklärt. Wichtig war ihm, dass man die untere Schranke $n \log(n)$ kennt.
8. Frage nach der Monte Carlo Methode: Erklären, wie man die Zahl π berechnet.

Keine Garantie für Vollständigkeit.

Prof. Keller schafft eine angenehme Atmosphäre. Jedes neue Themengebiet wird mit ein paar Worten eingeleitet so dass man sich schon einmal den Stoff zurechtlegen kann.

Man kann dann über längere Strecken frei über das Thema reden bevor man wieder eingefangenen und zum nächsten Themengebiet geführt wird. Auch das gelingt ihm sehr elegant, meine Note für den Prof: 1.0.

Prüfungsprotokoll Diplom, Kurs 1727 – Paralleles Programmieren

Prüfer: Prof. Dr. Keller
Beisitzer: Udo Hönig
Datum: 07.03.2007
Dauer: ca. 20 min
Note: 1,0

1. Amdahls Gesetz und Gustafsons Gesetz – welchen Ansatz verfolgen die Autoren jeweils, was ist der Unterschied?
2. Mandelbrot Set – wie kann das parallelisiert werden, welches Problem tritt dabei auf?
3. Begriff Work Pool, Load Balancing
4. Bitonisches Sortieren – wie funktioniert das?
5. Gegeben sind zwei Prozessoren und es soll damit eine Matrix mit einem Spaltenvektor multipliziert werden – wie kann das durchgeführt werden?
Ich habe mich dafür entschieden die Matrix in zwei Blockmatrizen aufzuteilen (Spalte 1 bis $n/2$ und Spalte $n/2$ bis n), die jeweils mit der Hälfte des Spaltenvektors multipliziert werden. Anschließend werden dann die Ergebnisse jeweils aufsummiert und so ergibt sich das Ergebnis. Wurde akzeptiert (puh!). Prof. Keller meinte allerdings, er hätte die beiden Prozessoren jeweils erst eine Hälfte der Ergebnismatrix mit je einer Hälfte des Spaltenvektors berechnen lassen. Dann tauschen die beiden Prozessoren ihre Hälften des Spaltenvektors aus und berechnen so die zweite Hälfte der Ergebnismatrix.

Wie andere Prüflinge kann ich nur bestätigen, daß Prof. Keller ein sehr angenehmer Prüfer ist. Jede Frage wird mit einigen Worten eingeleitet, das gibt dem Prüfling Zeit sich auf das entsprechende Thema einzustellen.

Es ist sicherlich der Note dienlich von sich aus nicht nur knapp die Frage zu beantworten sondern gleich sich darum gruppierende Begriffe zu bringen und kurz zu erläutern. Ich denke, Prof. Keller sieht dann recht schnell, ob man sich mit dem Stoff befasst hat.

Auf eine ausführlich Beantwortung der Fragen hier im Protokoll verzichte ich, mithilfe des Buches ist eine erschöpfende ;-) Vorbereitung absolut möglich.

Bitte denkt daran, nach Euren Prüfungen ein Gedächtnisprotokoll anzufertigen!

Gedächtnisprotokoll

Prüfung: Bachelor Fachprüfung „Parallel Programming – 1727“
Prüfer: Prof. Keller
Datum: 09.03.06
Ort: Videoprüfung am STZ an der TU-München
Dauer: ca. 25-30min.

1. Warum benötigt man parallele Systeme ?

Ständig wachsende Problemgröße erfordert höhere Rechenleistungen um in brauchbarer Zeit Ergebnisse zu erhalten. Einige Anwendungen genannt (Wettervorhersage, Klimaforschung, usw.)

2. Welchen Grund gibt es nun für die Wettervorhersage, die ja nun schon ein paralleles System einsetzt, ein noch schnelleres System anzuschaffen ?

Wenn eine höhere Genauigkeit der Vorhersage in der selben Berechnungszeit gefordert wird.

3. Amdahls Gesetz, wie lautet es und was sagt es aus ?

Erklärt inkl. Formel.

4. Wozu dann Parallelisierung wenn laut Amdahl der max. erreichbare Speedup sowieso mit $1/f$ begrenzt ist ?

Argumentation von Gustafson erklärt, Formel aufgeschrieben und mit Amdahl's Ansatz verglichen.

5. Embarassingly Parallel – wie ist eine Parallelisierung bei der Berechnung des Mandelbrot Set realisierbar:

Erklärt, wie die Parallelisierung in der Kurs-EA gelöst wurde. Hinweis auf Problem des Load Balancing, da ja die Berechnung der einzelnen Pixel durchaus unterschiedlich lang dauern kann.

6. Warum werden bei der Mandelbrot-Set Aufgabe die Pixel zeilenweise an die Slaves verteilt und von diesen berechnet und nicht pixelweise?

Um den Anteil der Kommunikationszeit an der Gesamtausführungszeit gering zu halten. Das häufige Versenden kleiner Datenmengen ist wegen des Kommunikationsoverheads (startup time bzw. Message latency) wesentlich „teurer“ als das Versenden weniger großer Datenpakete.

7. Matrix-Vektor Multiplikation. Wie müssen die Daten verteilt werden, wenn 2 Prozessoren zur Verfügung stehen (Prof. Keller hat Matrizen kurz skizziert)?

Prozessoren P1 und P2 berechnen zuerst jeweils eine Hälfte der Ergebnismatrix mit je einer Hälfte des Spaltenvektors B. Danach tauschen die beiden Prozessoren ihre Hälfte des Spaltenvektors B untereinander aus und berechnen damit die 2. Hälfte der Ergebnismatrix. Eine Lösung, bei der beide Prozessoren gleichzeitig mit dem gesamten Spaltenvektor B arbeiten war Prof. Keller noch nicht effizient genug.

8. Was ist das besondere an Cannon's Algorithmus zur Matrizenmultiplikation?

- Implementierung auf einem Mesh
- Initiale Alignment-Phase (damit wird sichergestellt, dass bei den nachfolgenden Iterationen immer nur „korrespondierende“ Elemente aus den beiden Matrizen in einem Rechenknoten zusammentreffen -> anhand der Vorgangsweise beim systolischen Array erklärt).

Prof. Keller ist, wie aus anderen Prüfungsprotokollen schon bekannt, auf jeden Fall zu empfehlen. Wichtig ist ihm offensichtlich das Grundverständnis und die Zusammenhänge. Wenn er merkt das man zu einem Thema fit ist, geht er gleich weiter zur nächsten Frage. Die hier aufgeführte Fragenliste ist nicht unbedingt vollständig sondern repräsentiert einfach jene Fragen, die mir in Erinnerung geblieben sind. Es wurden keine Komplexitäten od. Sortieralgorithmen gefragt. Die angeführten Antworten sind nur Kurzfassungen und sollten in der Prüfung natürlich ausführlicher sein.

Prüfungsprotokoll 1727 Parallel Programming (Bachelor)

Okt. 2003

Prüfer Prof. Keller

Note 1.7

☞ Freundl., ruhige Atmosphäre. Prof. Keller überlegt Fragen sehr genau und stellt diese präzise.

☞ Wozu Parallel Rechner? Schneller, genauer, größere Probleme, mehr Speicher

☞ Amdahls? Law. Formel, erklären daß Speedup limitiert ist. Fester Anteil $f=f(n)$

☞ Programmiermodelle: SPMH, MPMD

☞ Architekture aufzählen; Shared Memory, Multicomputer, Distributed Shared Memory

☞ Vorteile shared Memory. Datendurchsatz, Semaphores, Muster erwähnt.

☞ Def. embarrassing? parallel

☞ Beispiele: Bildverarbeitung, Monte Carlo,.. aufzählen

☞ Partitioning and Divide and Conquer erklären

☞ Beispiele: Mergesort, Quicksort nennen

☞ Zusammenhang mit Trees. Für Parallelrechner auch t-näre Bäume

☞ Wie zählt man pivots? Je nach Vorsortierung, oder zufällig $n/auch$ okay.

☞ Matrizenmultiplikation: Optimale Anordnung im Speicher (Cammon)

$$O(n^3)$$

☞ Strassen Algorithmus erklärt (ich hatte den Begriff eingeworfen)

Zeitkomplexität $O(n^{\log 7})$ hingeschrieben.

☞..

☞..

Die Prüfung geht für den Bachelor nicht sehr in die Tiefe eher in die Breite. Man sollte sich überall etwas auskennen, muß aber keine Komplexitäten vorrechnen. Die Benotung ist okay, ich kam nicht sofort auf Common (hat zx gefragt) und ein paar andere Kleinigkeiten. Wahrscheinlich erhält man eine bessere Note, wenn man noch eine Frage von sich aus etwas erschöpfender erzählt.