Kurs: Parallele Algorithmen (01824)
Datum: Oktober 2011
Prüfer: Prof. Verbeek
Dauer: 30 Minuten
Art: Leistungsnachweis (unbenotet, bestanden)

Professor Verbeek ist ein angenehmer Prüfer, der mehr an einem Gespräch als an einem strengen Frage-Antwort-Spiel interessiert ist. Die Prüfung begann ohne viel Smalltalk und dauerte ziemlich genau 30 Minuten. Es hat dabei sehr geholfen, die Prüfungsprotokolle zu kennen. Jede mir gestellte Frage kannte ich hieraus eigentlich bereits im Voraus und ging entsprechend locker durch die Prüfung. Die (doch teils recht schweren) Einsendeaufgaben waren kein Bestandteil der Prüfung, auch wenn es sich für mich sehr gelohnt hat, diese durchzuarbeiten. Einziges Manko ist, dass die Fragen manchmal ein wenig undeutlich formuliert sind. Allerdings gibt Herr Verbeek auf (qualifizierte) Nachfrage ohne weiteres zusätzliche Hinweise worauf er denn hinaus will. Da ich gelegentlich auch Stoff von außerhalb des Kurses als Beispiel eingebracht habe, ging Herr Verbeek bei Fragen auch auf diesen ein. Dessen sollte man sich bewusst sein und entsprechend nur sattelfestes Wissen verwenden. Lieblingsthemen von Herrn Verbeek sind eindeutig die Fouriertransformation und im Groben auch deren Anwendung. Bei den Fragen zu KE1 ist auf präzise (!) Antworten zu achten. Man könnte zu dieser Einheit meinen, sie wäre in der Prüfung am wichtigsten, da in den meisten Protokollen überproportional viele Fragen hierzu zu finden sind, allerdings sind die Antworten in diesem Kapitel schlichtweg sehr viel kürzer abzugeben, weshalb sie hier in Relation mehr Raum einnehmen. Dessen sollte man sich auch beim Lesen anderer Protokolle gewahr sein. Ebenfalls kommt KE6 in dieser Prüfung häufig zur Sprache. Die wenigsten Dinge muss man dabei schriftlich fixieren oder exakt bspw. gemäß Definition wiedergeben. Es kommt lediglich darauf an, die „Intention“ der Algorithmen nachvollziehen zu können und insbesondere die Komplexitätsschranken von Algorithmen auf Grund ihrer Funktionsweise zu erläutern. (Und nicht nur stur die Formel aufzusagen.) Auch insgesamt sollte man versuchen, wenn gebraucht, immer die richtigen Fachwörter zu nutzen und nicht einen umgangssprachlichen Begriff, da Herr Verbeek sonst korrigiert.

→ Wenn man Algorithmen entwickelt, muss man diese ja auch irgendwie implementieren. Was haben wir denn da gemacht?
(P)RAM. Aber auch andere Modelle: Prozessornetzwerke, BSP-Modell.

→ Wie funktioniert denn die RAM.
Erklärung RAM nach Skript.

→ Und was ist dann die PRAM?
Erklärung der parallelen Anwendung nach Skript.

→ Aber wenn da nur eine Anweisung gegeben werden kann (Stichwort: SIMD), machen dann alle Prozessoren immer das selbe?
Nein, indirekte Adressierung nach Prozessorlaufnummer. Darüber kann indiziertes Adressieren simuliert werden.

→ Was für Typen PRAM gibt es denn da?
Typen EREW, CREW, CRCW (common, arbitrary, priority) erklärt. Ebenfalls erwähnt, dass man jeden denkbaren Typen selbst festlegen könnten.

→ Könnte man sagen, dass es da verschiedene Mächtigkeiten gibt?
EREW < CREW < common-CRCW < arbitrary-CRCW < priority-CRCW.
Dazu erklärt warum das so ist in Bezug auf mögliche Anwendung von Algorithmen.
Hab mich hier ein bisschen verhaspelt, weil ich erklärt habe, dass die Prozessoren der EREW beispielsweise nur mittel einer Verbindung gelesen werden könnten. Das ist natürlich Unsinn betreffs der perfekten Vernetzung aller Prozessoren. Dann habe ich noch gesagt, dass der Schreibversuch verschiedener Werte in der common-CRCW bspw. eine Ausnahme erzeugen könnte. Dazu meinte Herr Verbeek, dass das eine Zusatzannahme stelle und in der Regel bspw. ein technischer Fehler als Standard unterstellt würde.

→ Wozu soll denn sowas gut sein „common-CRCW". Wenn die alle nur den gleichen Wert schreiben können. Wo braucht man sowas denn?
Genaue (!) Erklärung der Funktionalität des O(1) Algorithmus für das Bilden des Maximums in KE2.

→ Kommen wir zu einem anderen Thema: Wie schnell kann man denn sortieren?
Argumentation von lexikographischem Sortieren vs. Suchen durch Vergleiche. Omega(n^2) Zeitschranke intuitiv erläutert und Zeitschranke formal hergeleitet.

→ Wenn Sie eine Menge von jeweils unterschiedlichen Binärzahlen haben, was gebrauchen Sie da am besten?
Sortieren durch Vergleiche, da lexikographisches Sortieren nur lohnt, wenn Elemente häufig doppelt vorkommen. (War mir hier aber nicht ganz sicher, ob nicht doch bucket sort gemeint war, Herr Verbeek ist nicht mehr genauer darauf eingegangen.)

→ Was sind denn Beispielalgorithmen zum Sortieren in O(nlogn) Zeit?
Ich habe Merge Sort und Heap Sort erwähnt. Dann noch Quick Sort, wobei mir eingefallen ist, dass dieser O(n^2) hat, was ich angefügt habe. Herr Verbeek bejate das und lies mich Quick Sort trotzdem erklären. Dann umriss er ein Verfahren, mit dem das Pivot-Element so gesucht wird, dass auch Quick Sort in O(nlogn) Zeit durchgeführt werden kann.

→ Wie lässt sich Merge Sort denn parallelisieren? Und wie funktioniert das sequentiell?
Detaillierte Erläuterung aller Verfahren gemäß KE6.

→ Welche Algorithmen aus dem Skript glauben Sie denn sind denn besonders wichtig?
(Wer die Prüfungsprotokolle gelesen hat sagt hier sofort:) Fouriertransformation.

→ Wo wird das denn verwendet?
Z.B. Handyverschlüsselung, JPEG-Kompression, MP3, Signalverarbeitung.

→ Was ist denn die Fouriertransformation?
Ausführlich formal erklärt. Dazu auch gleich den FFT-Algorithmus aus KE5.

→ Kann man diese auf alle Ringe anwenden?
Nein, Inverse von n und von der n-ten primitiven Einheitswurzel muss existieren. Dazu muss auch die n-te primitive Einheitswurzel existieren.

→ Und was kann man denn mit ihr machen? Was bedeutet sie?
Hier war ich mir nicht so sicher und verwies darauf, dass ich dem Konzept im Kurs zum ersten mal begegnet bin. Ich hab gesagt, dass sie gewissermaßen eine Harmonisierung darstellt, also eine Transformation, in der für den Computer „teure" Operationen durch billigere getauscht werden. Damit gab sich Herr Verbeek zufrieden.

Diplom Hauptprüfung
(Gedächtnisprotokoll)

Fachprüfung: Theoretische Informatik
Prüfungsgebiet: 1824 Parallele Algorithmen
Prüfer: Prof. Verbeek
Datum: 18.02.2008
Dauer: 30 Minuten

Fragen:

- *Was ist eine PRAM*
  n Prozessoren, globaler Speicher mit unendlich viel Registern, SIMD Maschine, synchroner Takt für alle Prozessoren
- *CRCW PRAM: welche Arten gibt es*
  common, arbitrary, priority mit Erklärungen
- *Sind Programme die auf einer common CRCW PRAM laufen auch auf den anderen CRCW PRAM Typen lauffähig*
  Ja, alle anderen haben ein Auswahlverfahren für parallelen Schreibzugriff
- *Wieviele Prozessoren kann eine PRAM haben*
  Unendlich viele
- *Wie führen die Prozessoren das Programm aus*
  Sie können parallel den gleichen Befehl abarbeiten, aber jeder mit unterschiedlichen Daten -> SIMD
- *Worin unterscheiden sich die einzelnen Prozessoren*
  Jeder kennt seine eigene Adresse
- *Wie wird der Befehl* $x_i = y_{i+100}$ *ausgeführt*
  Indirekte Adressierung
- *Pascal bietet normal nicht die Möglichkeit zu parallelisieren. Wie konnten wir das doch machen*
  Mit dem pardo Befehl
- *Wie schnell kann man parallel das Minimum finden*
  Der schnellste Algorithmus schafft das in O(1) Zeit
- *Wie geht das*
  n * n parallele Vergleiche der Eingabeelemente, Parallele Initialisierung einer Variablen m für jeden Index der Eingabe, paralleles Schreiben von 0 auf die m_i entspricht einem logischem

UND, paralleler Vergleich der m_i auf 1 wenn ja Ausgabe des x_i es könnte ja mehrere Minima geben wenn die Eingabefolge gleiche Elemente hat.

- *Welche Arten von Sortieren haben wir kennengelernt*
  Das lexikographische Sortieren und mittels Sort und Merge
- *Wie funktioniert lexikographisches Sortieren*
  Initialisierung einer Schlange für jedes Zeichen des zur Verfügung stehenden Alphabets [0,m) , bei k Wörtern der Länge L, L Durchläufe mit O(k) Zeit. Also in Summe O(n) Zeit mit O(m+n) Aufwand.
- *Wie lange brauchen wir um sequentiellen Fall mind. zum Sortieren*
  Bei Eingabelänge n ist der Lösungsraum n!. Mit jedem Vergleich halbiert sich der Lösungsraum -> nach spätestens log(n!) Vergleichen ist die richtige Lösung gefunden. Mittels der Stirlingformel für n! kann man die Lösung auf O(n logn) abschätzen.
- *Wie funktioniert Sort & Merge*
  Rekursives Teilen der Eingabefolge in 2 gleich große Teilfolgen und anschließend „reissverschlußartiges“ Zusammenmischen der beiden sortierten Teilfolgen. Geht sequentiell in O(n logn). Parallelisierung der rekursiven Sortierung bringt die Zeit runter auf O(n). Diese Zeit wird durch den sequentiellen Merge bestimmt.
- *Geht Merge schneller*
  Ja, mit Hilfe des rank Verfahrens.
- *Wie funktioniert rank. Wie schnell ist das*
  rank(x : A) sucht den Index in der sortierten Folge A hinter welchem x eingereiht werden würde. Da rank auf einer sortierten Folge arbeitet, braucht man nur das Element mit Index n/2 mit x zu vergleichen. Abhängig vom Ergebnis sucht man in der unteren oder oberen Hälfte der Folge rekursiv weiter. -> logn Vergleiche notwendig. -> Zeit O(logn). Rank(B:A) ist parallel in O(logn) Zeit und O(m logn) Aufwand möglich. D.h. mittels rank schafft man den Merge in O(logn) Zeit und das Sortieren daher in $O(\log^2 n)$ Zeit.
- *Was ist die Fourier Transformation*

Das Matrix Vektor Produkt der nxn Fourier Matrix mit einem Vektor. Die Elemente der Fourier Matrix sind Potenzen einer primitiven n-ten Einheitswurzel.

- *Wie schnell können wir die machen*
  Mittels FFT. Teilung der Fourier Matrix in Fourier Matrizen halber Größe und Parallelisierung der Matrix Vektor Multiplikationen. Zeit O(logn), Aufwand O (n logn).
- *Wie lange brauchen wir im sequentiellen Fall*
  Gleich dem Aufwand im parallelen Fall -> O (n logn).

Prof. Verbeek ist ein sehr freundlicher Prüfer und die Prüfung verlief in einer angenehmen Atmosphäre. Prof. Verbeek legt scheinbar nicht zu viel Wert auf Wiedergabe von Definitionen wie im Kurstext. Er legt aber großen Wert darauf die Algorithmen und die Herleitung der Komplexitäten genau verstanden zu haben. Z.B. steht im Kurs zwar ein kleines Programm wie das Minimum mit dem O(1) Algorithmus zu finden ist, aber nicht wie man die gefundenen Minima wieder aus der PRAM herausbekommt. Das geht wohl nur über parallelen Vergleich der m_i auf 0 oder 1.

Ich wünsche allen viel Erfolg bei der Prüfung

Diplom-Hauptprüfung

(Gedächtnisprotokoll)

Fachprüfung:	Theoretische Informatik
Prüfungsgebiet:	1824 Parallele Algorithmen (04/01)
Prüfer:	Prof. Verbeek
Datum:	23. November 2005
Dauer:	30 Minuten

Zur Auflockerung der Prüfungssituation wurde von Frau Rempel (Sekretärin) ein Tee angeboten. Ohne Einleitung begann Prof. Verbeek sofort mit der ersten Frage.

- Wie arbeitet eine PRAM?
  Erklärung der PRAM, wie im Kurstext beschrieben.
- Wie wird ein Befehl ausgeführt, z. B. $x_i := y_{i+100}$?
  Hier hatte ich einen großen Hänger. Keine Erklärung war zufriedenstellend bis das Stichwort "indirekte Adressierung" fiel.
- Wie kann man das Minimum parallel finden?
  Genaue Beschreibung des Algorithmus. Statt Minimum habe ich einfach Maximum beschrieben. Dazu gab es keine Rückfrage.
- In welcher Zeit und mit welchem Aufwand geht das?
  Herleitung von $O(1)$ und $O(n^2)$. Die beiden anderen Algorithmen aus der Beschleunigungstechnik waren nicht gefragt.
- Was heißt Sortieren?
  Lexikalisches Sortieren und Sortieren durch Vergleichen erläutern.
- Wie kommt man zur unteren Grenze beim Sortieren?
  Herleitung wie im Kurstext mit Stirlingscher Formel.
- Wie funktioniert das lexikalische Sortieren?
  Bucket-Sort beschreiben
- Wie ist die Zeit und der Aufwand beim parallelen Sortieren?
  Herleitung wie im Kurstext.
- Jetzt zum parallelen Sortieren durch Vergleichen, wie geht das?
  Divide-and-conqueror-Verfahren. Merge zweier sortierter Folgen.
- Das Sortieren ist ja banal. Wie geht das Merging schnell?
  Zunächst Erläuterung des rank-Verfahrens.
- Wie funktioniert nun das parallele Merging?
  Genauer Beschreibung des Algorithmus.
- Welche Zeit und welchen Aufwand benötigt das parallele Merging?
  $O(\log \log n)$ Zeit und $O(n)$ Aufwand.

Für die Vorbereitung eignet sich auch sehr gut ein Fragenkatalog, den Ihr unter http://www.stud.fernuni-hagen.de/q3004317/katalog3/katalog3.htm finden könnt.

Die Prüfung verlief in einer angenehmen und entspannten Atmosphäre. Allerdings legte Herr Professor Verbeek großen Wert auf die genaue Beschreibung der Algorithmen und die Herleitung der Zeit und des Aufwandes. Mit der "indirekten Adressierung" haben wir sehr viel Zeit verloren. So wurde in den 30 Minuten nur ein kleiner Teil des Stoffes behandelt.

Diplomprüfung: 1824 „Parallele Algorithmen“

Prüfer	: Prof. Dr. Verbeek
Beisitzer	: Dr. Osterloh
Datum	: 03. März 2004
Dauer	: 25 Minuten
Note	: 1,3

Vorbereitung

Zur Prüfungsvorbereitung habe ich mir eine Zusammenfassung aller bekannten Prüfungsfragen bei Prof. Verbeek erstellt. Das Inhaltsverzeichnis ist dazu als Fragenkatalog gestaltet. Daneben habe ich mir im Internet verschiedenste Ausarbeitungen der Fourier-Transformation (damit hatte ich im gesamten Studium nichts zu tun) zusammengesucht. Wie sich herausstellen sollte, war das ein Volltreffer.

Prüfungsverlauf

Ohne weitere Einleitung kam Prof. Verbeek gleich zur ersten Frage. Der Verlauf der Prüfung hat sich ungefähr wie folgt abgespielt:

Wie werden im Kurs die parallelen Algorithmen realisiert?
Erklärung der PRAM.

Wie wird denn den Prozessoren mitgeteilt, auf welche Daten sie zugreifen sollen, wenn alle Prozessoren einer SIMD Maschine deselben Befehl bearbeiten?
Als Beispiel nannte er 100 Fensterputzer, die in einem Hochhaus gleichzeitig die Fenster putzen sollen.

Ich hatte bei meiner PRAM Definition die Adressierung, hier insbesondere die indirekte und indizierte vergessen gehabt.

Welche Komplexitätsmaße gibt es?
Zeit = Anzahl aller Takte und Aufwand = Anzahl der Prozessoren multipliziert mit den Takten.

Wenn mehrere Prozesoren gleichzeitg auf eine Speicherstelle schreiben wollen (lesen ist ja unkritisch), welche Modelle gibt es dafür?
Die common, arbitrary und priority CRCW, mit genauer Unterscheidung, was das bedeutet.

Was passiert, wenn bei der common CRCW Prozessoren mit unterschiedlichen Inhalten gleichzeitig schreiben wollen?
Dann wird ein Fehler generiert und das Programm bricht ab.

Welchen schnellen Algorithmus gibt es zur Maximumbildung und auf welcher Maschine läuft er mit welcher Komplexität?
Ich habe den Algorithmus aus dem Kurs erläutert, dass dafür eine common CRCW erforderlich ist und der Algorithmus mit O(1) Zeit und O(n^2) Aufwand läuft.

Kann man den Algorithmus auch auf einer arbitrary CRCW laufen lassen?
Ich zögerte und sagte erst, dass bei den x_i, die nicht maximal sind, eventuell falsche Werte geschrieben würden und dass das entsprechend verhindert werden müsste.

Da erklärte Herr Verbeek, dass diese „Sperre“ ja schon in dem Algorithmus enthalten ist, sonst würde die common CRCW „explodieren“. Jedes Programm, was auf einer common CRCW läuft, kann auch auf einer arbitrary CRCW laufen und auch auf einer priority.

Kommen wir zu einem anderen Thema. Was ist die Fourier-Transformation?
Er fügte noch an, dass er die diskrete meine, da diese im Kurs diskutiert werden. Es gäbe auch noch die kontinuierliche, aber die wolle er nicht hören.

Ich habe ihm grob die Funktion der Diskreten Fourier Transformation (DFT) als eine Matrix-Vektor-Multiplikation $b = F \cdot a$ erklärt und dass die Fast Fourier Transformation (FFT) die rekursive Aufteilung der DFT in zwei DFT mit halber Größe darstelle. Dazu habe ich die n-te Einheitswurzel mit ω als primitiver n-ter Einheitswurzel erklärt und anhand der komplexen Zahlen $\mathbb{C}$ in der komplexen Ebene die Zusammenhänge geometrisch erläutert ($\omega_{\frac{n}{2}} = -1$ und $\omega_n^2 = \omega_{\frac{n}{2}}$).

Herr Verbeek warf ein, dass es Anwendungen gibt, wo ein unendlicher Körper oder Ring aufgrund der Rundungsproblematik nicht gut geeignet sei. Da seien endliche Körper für die Einheitswurzeln besser. Ich erwiderte daraufhin, dass für die Anschauung $\mathbb{C}$ mit dem Einheitskreis sowie den Winkelfunktionen sehr gut geeignet sei, zumal dann auch sofort die Zusammenhänge mit sin und cos erkennbar sin.

Weiter habe ich die Fouriermatrix erläutert ($Fij = \omega^{i \cdot j}$) und dann die Besonderheit der FFT aufgrund der ω Eigenschaften erklärt mit der Aufteilung in gerade und ungerade Zeilen. Die Formel für die Zeilenkoeffizienten der geraden Zeilen $b_{ij} = \sum_{k=0}^{\frac{n}{2}-1} \omega^{2mk}(a_k + a_{\frac{n}{2}+k}) \quad \forall m < \frac{n}{2}$ habe ich notiert und erklärt, dass die ungeraden ähnlich berechnet würden. Dadurch bräuchten bei der FFT praktisch nur noch die Koeffizienten ausgewertet werden.

Welche Komplexität hat die FFT und wie sähe diese bei der naiven Matrix-Vektor-Multiplikation aus?
Die FFT hat sequenziell $\mathrm{O}(n \log n)$ und im parallelen Falle $\mathrm{O}(\log n)$ Zeit und $\mathrm{O}(n \log n)$ Aufwand. Die naive Matrix-Vektor-Multiplikation hat eine Komplexität von $\mathrm{O}(n^2)$.

Herr Verbeek gab noch viele Erklärungen zur FFT, zumal ich ihm gesagt habe, dass ich von der FFT keine Ahnung hatte, bis ich mich in diesem Kurs damit beschäftigen musste. Er erklärte, dass in jedem Handy zur Berechnung der Verschlüsselung Chips enthalten sind, die für ein festes n FFT Berechnungen parallel durchführen. Außerdem würde die FFT im JPEG Bildkompressionsverfahren sowie in der Kryptographie in fast allen Anwendungen vorkommen. Irgenwann in diesen Erklärungen meinte Herr Verbeek dann, dass die Zeit um sei.

Ich kann Prof. Verbeek und diesen Kurs für eine Prüfung uneingeschränkt empfehlen. Es ist sehr freundlich und gibt sehr gute Hilfestellungen, die einem schnell dahin bringen, wie er die Frage beantwortet haben möchte.

Der Vorteil ist, dass Prof. Verbeek i.d.R. nur die Gebiete abfragt, die auch schon in den Prüfungsprotokollen enthalten sind.

Ein Schwerpunkt ist allerdings die Komplexität, die sehr gut sitzen müssen.

Ich wünsche allen viel Erfolg bei den Prüfungen.

Thomas Schwarze

Gedächtnisprotokoll zur Diplomprüfung Theoretische Informatik Teil 2

Kurs: 1824 Parallele Algorithmen

Prüfer: Prof. Dr. Verbeek

Datum: 22.10.01 Dauer: ca. 30 min.

Prüfungsfragen:

- Woraus besteht eine RAM?
(Eingabe-, Ausgabeband, unendlich viele Register, Befehlszähler, Programm)

- Wie wird ein solches Programm dargestellt?
(in PASCAL - ähnlicher Notation)

- PASCAL bietet ja keine Möglichkeit zu parallelisieren. Wie macht man das hier?
("pardo" - Anweisung erläutert)

- Wie läuft das in der Maschine ab?

(Erläutert, wie **man** von einer RAM zur PRAM kommt; Register werden zu eigenständigen Prozessoren, gleichzeitige Ausführung eines Befehls in vielen Prozessoren, SIMD - Modell mit synchronem Zeittakt)

- Bei Parallelität können gewisse Konflikte auftreten; wie werden diese hier behandelt?
(3 Arten von CRCW - PRAMs erläutert)

- Wie würden Sie diese Maschinenmodelle leistungsmäßig ordnen?
(Hier hatte ich nicht die geringste Ahnung; mit etlichen Irrungen + Wirrungen meinerseits und Unterstützung von Herrn Verbeek ergab sich schließlich die folgende Rangliste:
l. priority CRCW . 2. arbitrary CRCW 3. common CRCW 4. CREW)

- Wie ergeben sich bei einer PRAM Laufzeit und Aufwand?
(Laufzeit: Maximale Anzahl von Zeittakten, die die beteiligten Prozessoren zur Lösung des Problems benötigen Aufwand: Anzahl insgesamt ausgeführter Operationen)

- Welche schnellen Sortier verfahren gibt es?
(Mergesort, Heapsort, Quicksort)

- Wie funktioniert Mergesort?

(DAC - Strategie erläutert)

- Wie kann man beim Sortieren eine gegebene Folge in zwei Teilfolgen aufteilen?
(bei Mergesort in zwei gleichgroße .Teile, bei Quicksort unterschiedliche Größe je nach Lage des Schlüsselelementes)

- Wie kann man beim Merge - Schritt vorgehen?
(Hier den in KE 6 beschriebenen Werdegang erläutert: normales Merging nach dem Reißverschlussverfahren, paralleles Merging, paralleles Ranking, Beschleunigungstechnik.. Irgendwann mittendrin war's Herr Verbeek anscheinend zufrieden, er stellte die nächste Frage.)

- Auf welche Werte für Zeit und Aufwand kommt man dabei schließlich?
(Zeit: 0(log n log log n) bzw. 0(log n) mit Pipelining Aufwand: 0(n log n))

- Wie kommt die untere Grenze beim Aufwand zustande?
(n! Permutationen, Darstellung als Entscheidungsbaum mit n! Blättern und minimaler Höhe von log n, Stirling'sehe Formel)

Kleiner Tipp am Rande: Hier lohnt es sich, im Kurs "Datenstrukturen" nachzulesen; dort wird das Ganze ausführlicher hergeleitet.

- Können Sie sich vorstellen, welcher der im Kurs behandelten parallelen Algorithmen in der Praxis besonders häufig zur Anwendung kommt?
(Konnte ich nicht; nach kurzem Grübeln ging ich dann nach dem Motto "Dreimal dürfen Sie raten" vor.

1. Versuch: "List ranking?" Falsch!

2. Versuch: "Präfixsummen?" Wieder daneben!
Herr Verbeek gab Hilfestellung:

"Dieser arithmetische Algorithmus wird häufig zumindest teilweise auf Chips fest verdrahtet. In jedem Handy ist beispielsweise so ein Ding!"

Das half mir auch nicht weiter, ich bin kein Handymane. Mit dem Stichwort "arithmetisch"kam mir dann aber doch noch die Erleuchtung.

3. Versuch: "FFT?" Bingo!)

- Was versteht man unter FFT?
(Fast Fourier'Transform; Fouriertransformation eines Spaltenvektors, Polynomauswertung an den Stellen der n-ten Einheitswurzeln, Matrixstruktur erläutert)

- Wie ist hier die Zeitkomplexität im sequentiellen Fall?

(Matrix-Vektor-Produkt mindestens 0(n))

- Und im parallelen Fall?

(0(log n); Aufteilung in zwei Teilprobleme halber Größe erläutert)

Auch hier ein Tipp: Lest das Ganze noch mal im Kurs "Mathe III" nach; hier ist der ganze Komplex n-te Einheitswurzeln, FFT, Konvolution etc. wesentlich ausführlicher und zusammenhängender dargestellt und auch verständlicher wie ich finde.

Zum Prüfer:

Prof. Verbeek erwies sich auch diesmal wieder als angenehmer Prüfer, der jederzeit Hilfestellung gibt, wenn es mal hapert, so dass man, wenn auch mit Hängen und Würgen, doch noch zum Ziel kommt (PRAM-Rangliste). Wenn solche Fragen kommen, die nicht unmittelbar im Kurs beantwortet werden, bekomme ich Probleme, zumal ich berufsmäßig (Organisationsprogrammierer) in Sachen Theoretischer Informatik nichts beisteuern kann. Dafür schien er Verständnis zu haben, seine Benotung empfand ich als großzügig.

Gedächtnisprotokoll zur mündl. Diplomprüfung

Kurs: 01824 - Parallele Algorithmen

Prüfer: Prof. Dr. Verbeek
Datum: 27.8.2001
Dauer: 25 min

- **Welche Maschinenmodelle wurden im Kurs verwendet ?**
  RAM, PRAM definiert

- **Wie funktioniert eine PRAM ?**
  Wichtig: die Prozessoren können nur durch die ihnen bekannte eigene Adresse i auf unterschiedliche Operanden zugreifen (Adressierung über i)

- **Wie werden gleichzeitige Zugriffe mehrerer Prozessoren auf die gleiche Speicherzelle behandelt ?**
  EREW, CREW, common / arbitrary / priority CRCW

- **Wozu benötigt man die verschiedenen PRAM-Typen ?**
  Es gibt für das gleiche Problem unterschiedliche Algorithmen mit unterschiedlichen Laufzeiten.
  Z.B. Maximumsuche in O(1) Zeit auf einer common CRCW PRAM.

- **Welche Komplexitätsmaße gibt es ?**
  Laufzeit, Aufwand, Speicherplatz, Anzahl Prozessoren;
  Unterschied zwischen Laufzeit und Aufwand.

- **Kann Sortieren parallelisiert werden ?**
  parallelen Mergesort in der einfachsten Variante mittels ranking erläutert.
  Zeit und Aufwand anhand des Algorithmus hergeleitet.
  Verbesserung mittels schneller Suche erläutert.
  Weitere Details des Algorithmus (weitere Aufteilung der beiden zu mischenden Mengen) wurden nicht verlangt.

- **Kann man die Matrixmultiplikation beschleunigen ?**
  Strassen-Algorithmus mit Herleitung der Komplexität.

- **Wie schnell kann man Matrizen invertieren ?**
  Rückführung auf Matrixmultiplikation, daher gleiche Komplexitätsordnung.

- **Stichwort NP, P: sind alle berechenbaren Funktionen parallelisierbar ?**
  Parallelisierbarkeit definiert, NC-P Problem und P-Vollständigkeit erläutert.
  Warum wird zur P-Vollständigkeit logarithmische Bandreduzierbarkeit verlangt ?

Die Prüfung verlief ruhig und entspannt.
Die Problematik NC, P, NP scheint ein Lieblingsthema von Prof. Verbeek zu sein. Es empfiehlt sich, das entsprechende Kapitel aus "Einf. Theo. Inf. B" zu wiederholen.

Viel Erfolg und denke bitte daran, ebenfalls ein Protokoll zu schreiben.

Prüfungsprotokoll 1824

Parallele Algorithmen

Fachprüfung: Theoretische Informatik (Teil 2)
Prüfer: Prof. Verbeek
Datum: 8.1.2001
Dauer: ca. 20 Min.
Note: 1.0

Herr Prof. Verbeek begann die Prüfung mit der Frage nach Komplexitätsmaßen (Zeit, Aufwand) und Kostenmaßen (uniform/logarithmisch) bei parallelen Algorithmen. Darauf aufbauend die Fragen nach

- P-vollständigkeit, NP-vollständigkeit. Was hat log-Bandreduzierbarkeit mit Parallelisierbarkeit zu tun?[1] Was heißt eigentlich "parallelisierbar"?
- PRAM-Modell, verschiedenen Typen (EREW, CRCW), welche Möglichkeiten gibt es, konkurrierende Schreibzugriffe zu verarbeiten?
- Welcher Algorithmus läuft so nur auf CRCW ? Maximumsbildung in O(1). Erläutern Sie den Algorithmus.
- Multiplikation
  - Matrixmultiplikation nach Strassen, kurz beschreiben, wie kommt man auf die Komplexität, geht es noch besser?
  - Karatsuba-Multiplikation
  - Geht es noch besser? Schönhage-Strassen. Worauf basiert dieses Verfahren (ganz grob): FFT.
  - Schnelle Fouriertransformation kurz erläutern.
- Sortieren. Welches Verfahren kann man parallelisieren? Mergesort. Ganz grob den Algorithmus erläutern. Kann man Quicksort parallelisieren? Ja, sicher, auf ähnliche Art und Weise.

Damit war die Prüfung auch schon mit einem "Ich glaube, wir können jetzt Schluß machen." beendet. Im großen und ganzen eine rechte angenehme Prüfung. Prof. Verbeek legte zwischen den einzelnen Fragen immer mal wieder eine Denkpause ein und schaute aus dem Fenster, was irgendwie die Zeit ins "Unendliche" zog.

[1] Steht wohl nicht im Kurs: Algorithmen auf log-Band sind wohl beweisbar parallelisierbar ?!

Eric Hambuch, 10.1.2001