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 $O(n \log n)$ und im parallelen Falle $O(\log n)$ Zeit und $O(n \log n)$ Aufwand. Die naive Matrix-Vektor-Multiplikation hat eine Komplexität von $O(n^2)$.

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