Prüfungsbericht zur Diplomprüfung im Nebenfach Informatik zum Fach Physik
(Betriebssysteme und Netzwerke / Cluster Computing)
Prüfer: Ludwig
Note: 1,0
Datum: 13.2.2009 (Freitag der 13-te ....)
Dauer: 30 Min

F: Also fangen wir mal mit den Betriebssystemen an. Was können Sie denn zur Auflösung von Verklemmungen sagen?
(Nachdem ich gerade einmal 1-2 sec. überlegt hatte, wie ich denn anfangen solle)
Oder sagen Sie zuerst mal was zu Verklemmungen allgemein!

A: Voraussetzung für Existenz eben Wechselseitiger Ausschluss, kein Resourcenentzug und gleichzeitiger Zugriff auf min 2 Resourcen. Prozess A hat Resource 1 und wartet auf 2, Prozess B hat Resource 2 und wartet auf 1. Allgemein ist hinreichende Bedingung eben geschlossene Kette von Prozessen, die jeweils auf Resource von jeweils nächstem warten.

F: Und was macht man jetzt dagegen?

A: In modernen Betriebssystemen eigentlich gar nichts, man hofft das es nicht auftritt. Was man z.B. ganz einfach und wenig intelligent machen kann ist eine Liste der Resourcen die alle Prozesse maximal brauchen. Jetzt startet man keine Prozesse wenn die maximale Menge der benötigten Gesamtresourcen größer als die verfügbare Menge ist. Nur ist dies schrecklich ineffizient und verschwenderich.

F: Und wie geht es etwas intelligenter?

A: Das Verfahren aus der Vorlesung beschrieben. Man geht nur in sichere Zustände über. Zustand sicher falls man Stück für Stück Prozesse durcharbeiten lassen kann.

F: Wo wird denn sowas benutzt.

A: Naja heute kaum noch. Kann man für Langfristiges Scheduling benutzen. Also ehemals bei Stapelverarbeitung. Heute dann eben evtl. für große Cluster.

F: Genau, beim Supercomputing hat man ja z.T. noch Stapelverarbeitung. Und für welche Resourcen?

A: Naja Prozessoren z.B.. Evtl noch für Speicher aber nur bei Shared Memory, normalerweise hat ja jeder Knoten eigenen Speicher. Dann eben noch für Massenspeicher könnte ich mir vorstellen

F: OK, das reicht erstmal zu dem Thema. Was können Sie denn zu Semaphoren sagen?

A: Benutzt für wechselseitigen Ausschluss, 2 Typen (binär, zählend), hat 3 Methoden, erzeugen, signal, wait, wohl eher 4 Methoden (k.A. warum Stallings 3 schreibt), Vernichten muss man eigentlich auch noch kennen. Größer Null signalisiert eben verfügbare Resource. Bei Verringerung unter 0 wird der Prozess suspendiert und wieder aktiviert wenn Resource freigegeben wird.

F: Was unterscheided den Semaphoransatz jetzt von wechselseitigem Ausschluss im Prozess selbst?

A: Der Ausschluss kann auch in Prozess realisiert werden. Algorithmus von Decker und von Peterson erwähnt. Diese allerdings erstmal für 2 Prozesse. Kann man zwar erweitern (z.B. indem man sukzessive 2 gegen 2 Entscheide macht), wird dadurch aber nur ineffizienter. Semaphor eben algemeines Konstrukt das nur einmal implementiert wird und dann als Black Box genutzt. Am besten mit unterstützung durch Hardware implementiert. Bei x86 z.B. durch XCHG.

F: Das war jetzt alles richtig aber was gibts noch für Unterschiede?

A: *(Nach etwas nachdenken)* Betriebssystem kann den Prozess direkt suspendieren. Das ist besser als aktives warten. Wobei der Prozess ja auch nen sleep Befehl absetzen könnte.

F: Aber dann würde er ja niemals aufwachen...

A: Er könnte ja ein Timeout setzen. Das Betriebssystem kann es aber schon besser regeln

F: Hm, das stimmt eigentlich. Kommen wir mal zu Netzwerken.
Wie funktioniert denn die Flusskontrolle bei TCP?

A: Flusskontrolle betrifft erstmal nur Sender und Empfänger, geht davon aus das Netzwerk kann das alles zustellen. Sonst wäre es Überlastkontrolle. Basiert auf Sliding Window Algorithmus. Habe den eben erklärt. Kam kurz zu Verwirrung weil ich erst den Algorithmus algemein ohne TCP erklären wollte. Da gab es eben kein Last Written und Last Read.

F: Und wie groß sind die Fenster jetzt?

A: Naja müssen die Prozesse festlegen, so viel wie sie eben Puffer spendieren wollen.

F: Was ist da sinnvoll damit ein Netzwerk ausgelaster werden kann?

A: Muss größer gleich Verzögerungs-Bandbreite Produkt sein. Das eben kurz erklärt.

F: Was wenn Netzwerk star ausgelastet?

A: Dann nicht so wichtig.

F: Wenn das Advertise Window jetzt mal 0 war, wie fängt die Kommunikation wieder an?

A: Sender schickt in regelmäßigen Abständen trotzdem 1-Byte Pakete

F: Und dann?

A: Naja wenn der Empfänger Platz hat schickt er ja dann ein ACK mit neuem Advertise Window

F: Genau, wie groß kann so ein Fenster sein?

A: Also im TCP Header ist das Feld 16 Bit. Man kann aber TCP Optionen setzen die das Feld skallieren. Ich weis jetzt nicht direkt ob es sonst auch schon skalliert wird. Wohl so im Bereich zwischen 1 und 16 bytes im Fall der Fälle.

F: Weis ich jetzt auch nicht genau. Wäre aber gut möglich. Mit den 65k kann ja heute niemand mehr was anfangen. *(Habe es gerade nachgeschlagen. Wird normal nicht skalliert)*
Kommen wir mal zu den Clustern. Was können Sie denn zur Reduce Operation sagen?
(Reduce scheint mir hier sehr wichtig zu sein. Ist glaub auch mit das einzige was man sinnvoll fragen kann. Bei der Scheinprüfung zu Cluster Computing hat er sich hier jedenfalls auch ne Weile aufgehalten)

A: Eben reduce erklärt. Auch wie man das programmieren würde. Kann man gut über Baumstruktur realisieren. Vielleicht nicht unbedingt Binärbaum

F: Warum, wenn ich da so an das Schachfeld und das Reiskorn denke, das geht doch schnell?

A: Hm Anzahl der Stufen geht auf jeden Fall mit dem Logarithmus. Aber wenn man 4 statt 2 Verknüpfungen nimmt ist das ja schon ein Faktor 2 in der Zeitkonstanten.

F: Da haben Sie recht. Welche Operationen kann man nehmen?

A: Alles was kommutativ und assoziativ ist.

F: Beispiel?

A: Plus, Mal, Max, Min, Xor...

F: Welche logischen?

A: Zumindest AND, OR, XOR, bei NAND bin ich mir nicht ganz sicher

F: Ich mir auch nicht müsste man sich mal überlegen.

A: Eigentlich kann NAND nicht gehen. mit NAND hat man ja NOT und mit NOT, AND, OR hat man alles. *(Bin mir jetzt bei der Argumentation nicht sicher. Aussage stimmt aber:*
$\{ (1 \text{ NAND } 0) \text{ NAND } 1 = 0 \neq 1 = (1 \text{ NAND } 1) \text{ NAND } 0 \}$

F: Wenn Sie jetzt so eine MPI Bibliothek implementieren. Wie würden Sie vorgehen?

A: Naja zuerst muss man kommunizieren können. Also Unterste Schicht Send/Recv. Darauf aufbauend kann man dann noch etwas Kontrolle, z.B. für den Programmstart überhaupt. Ist jetzt etwas fragwürdig ob das nach Send/Recv kommt, andererseits muss man hier ja Senden/Empfangen. Darauf aufbauend dann höhere Funktionen wie z.B. Reduce.

F: Was gibts sonst noch für höhere Funktionen?

A: Scatter, Gather

F: Bei Send und Recv zuerst Synchron oder erst Asynchron?

A: Erst Asynchron

F: Und wie dann Synchron?

A: Naja das asynchrone benutzen und warten.

F: Das Reduce Synchron oder Asynchron ?
A: Würde ich beides implementieren, die Anwendung kann dann wählen
F: Und wie benutzen sie Send/Recv?
A: So synchron wie nötig, so asynchron wie möglich.
F: Wie ist das jetzt beim Reduce mit mathematischen Funktionen. Was muss man da beachten.
A: Das die Floating Point Darstellung nicht exakt ist. Und damit nicht ganz assoziativ/kommutativ.
F: Können Sie das beheben?
A: Die Reihenfolge des Reduce festlegen. Z.B. nach Rank.
F: Spielt hier Synchron/Asynchron eine Rolle?
A: Schon, je asynchroner destso schneller destso mehr Probleme mit der Reihenfolge. Wobei man alles in einem Array speichern kann (asynchron) und danach bei fester Reihenfolge reduziert.
F: Wie ist das bei OpenMP ?
A: Gleiches Problem.
F: Wie machen Sie eine Scatter funktion?
A: Auch sowas wie Baumstruktur. Außer die Hardware unterstützt Multicast. Dann damit.
F: Wo bekommen Sie den Baum eigentlich her?
A: Den rechne ich einmal aus. Mit MST oder so. Wenn ein Knoten ausfällt eben nochmal.
F: Wie realisieren Sie jetzt die Send Funktion selbst?
A: ????
F: Naja auf Betriebssystemebene, z.B. UNIX?
A: Über Sockets..... Also erstmal bis ich was lauffähiges habe. Dann am besten direkt. Sockets sind viel Software Overhead für Supercomputing.
(Folgte kurze Diskussion über Ethernet und Myrinet)

Die Prüfungsathmosphäre bei Ludwig ist extrem (fast schon übertrieben) locker. Er erzählt auch manchmal gerne noch selbst etwas so zum Abschluss eines Themas. Immer so ca 1 min. Kommt dann aber schnell wieder zurück. Zur Benotung hier kann ich nicht viel sagen. Ich habe aber in Erinnerung dass er bei der Betriebssysteme und Netzwerke Klaususr recht hart benotet hat. Muss allerdings hiermit nichts zu tun haben. Er hat damals auch extra darauf hingewiesen. Mir persönlich schien der B.u.N. Teil wesentlich schwerer. Auch braucht man hier recht viel Faktenwissen und sollte es direkt anwenden können. Cluster Computing beschränkt sich fast ausschließlich auf Programmieren. Wer in C wirklich fit ist hat hier nichts zu befürchten. Zur Vorbereitung habe ich den Stallings und Peterson/Davies gelesen. Sind beide recht dick und evtl. etwas overkill. Dafür muss man dann aber nicht jedes Detail behalten. Dann am Ende nochmal seine Folien durchgeblättert. Die Cluster Computing Folien waren damit auch alles für den Cluster Computing Teil. Ein echter Nachteil an den Büchern ist, dass Informatik Bücher sehr schnell überholt sind, was mich vor allen bei den Betriebssystemen störte. Andererseits richtet Ludwig seine Vorlesung quasi 1:1 nach den Büchern. Vorbereitungszeit waren ca 10 Tage. Die allerdings auch recht ausgiebig, schon alleine weil es ne Weile dauert 2 mal 800 Seiten zu lesen...

Viel Erfolg!