Architectuur en Netwerken (Informatica)

29 juni 2004

De Architectuur vragen.

1. Een belangrijk mechanisme om een computer sneller te maken is het gebruiken van een *pipeline*. Geef een duidelijke uitleg van de opbouw *en* werking er van, en leg daarbij ook uit wat de te behalen winst is.

   Leg tot slot uit wat het grootste probleem bij een pipeline is. Geef ook een mogelijke (deel)oplossing.

2. Leg aan de hand van o.a. de begrippen *exceptions*, *memory management unit* en *kernel-/user-* mode duidelijk en concreet uit hoe een Operating System robust gemaakt kan worden in de zin dat voorkomen kan worden dat de verschillende processen elkaar verminken.

3. Een belangrijk onderdeel van een zogenaamd *demand paged virtual* geheugen is de strategie waarmee pagina's vervangen worden.

   Geef een duidelijke uitleg van twee mogelijke methoden. Geef daarbij ook aan onder welke omstandigheden het *niet* goed werkt.

De Netwerk vragen.

1. Geef de fundamentele grens aan van de hoeveelheid informatie die over een communicatie kanaal gestuurd kan worden, zoals aangegeven door Shannon. Geef ook een duidelijke intuïtieve verklaring van deze formule.

2. Geef een duidelijke uitleg van de opbouw en werking van een *router* in (of beter tussen) netwerken.

   Laat met name zien hoe het lagenmodel gebruikt wordt, alsmede de header-informatie uit de verschillende soorten informatiepakketten. En wat is er te zeggen over de fysieke lagen van de beide netwerken?

3. *Quality of Service* is een steeds belangrijker wordend begrip in de netwerkwereld. Geef duidelijk aan wat hier mee bedoeld wordt en hoe een bepaalde QoS in netwerken realistisch gerealiseerd kan worden.

   Gebruik multimedia (video, audio) diensten en de daarbij gebruikte protocollen als een voorbeeld om je verhaal toe te lichten.

Naam: Thomas ter Laat
Adres: Aquamarijnstraat 419
Postcode en
Woonplaats: 9743 PK Groningen
Studentnummer: 1339862
Studierichting: Informatica & Wiskunde
Jaar van eerste inschrijving: 2002

Tentamen: Architectuur & Netwerken
Datum: 29-06-2004
Naam docent: Achtersop

Architectuur

1) Bij een CPU processor zonder pipeline bestaat het uitvoeren van een ISA-instructie uit een aantal stappen: fetch (haal de instructie op), decode (bepaal wat er gedaan moet worden), execute (laat de ALU de instructie verwerken), write (schrijf het resultaat terug naar de registers). Deze stappen zijn per instructie van elkaar afhankelijk. Het idee van een pipeline is dat alle stappen tegelijk werken, maar op verschillende instructies. Terwijl de ene wordt uitgevoerd wordt de volgende al gedecodeerd en de daaropvolgende opgehaald, enz. De instructies duren nog steeds even lang om uit te voeren (time to completion) maar er hoeft niet meer met de volgende instructie gewacht te worden tot de vorige klaar is. Dit levert een grote snelheidswinst, omdat per tijdseenheid meer instructies gestart kunnen worden.

Er treden hierbij wel een aantal problemen op. Ten eerste gaat het niet goed als een instructie afhankelijk is van het resultaat van een voorgaande instructie die nog niet is voltooid. Dit probleem kan worden opgelost door per register bij te houden of een instructie ~~van plan is~~ in de pipeline van plan is ernaar te schrijven. Als conflicten optreden wordt de nieuwe instructie pas toegelaten tot de pipeline als de oude voltooid is.

Een tweede probleem wordt gevormd door branches, met name conditionele. Er is dan niet bekend wat de volgende instructie gaat worden tot de branch-instructie is voltooid, dus kan de processor in de tussentijd niets anders doen. Dit kan worden verholpen met branch prediction: voorspel waar de branch heen zal gaan en ga die instructies alvast uitvoeren. Hiervoor zijn wel tijdelijke scratch-registers nodig, zodat de waarden in de echte registers niet worden veranderd in het geval dat de voorspelling het mis had.

2 In een Operating System is een deel van het geheugen gereserveerd voor het OS zelf. Andere delen zijn gereserveerd voor de verschillende programma's/processen. Het OS mag bij al het beschikbare geheugen; de processen mogen alleen bij hun eigen deel. Hiertoe kan de processor in twee verschillende modi werken: kernel mode en user mode. In kernel mode zijn alle instructies en al het geheugen beschikbaar; deze mode wordt voornamelijk door het OS gebruikt. In user mode zijn bepaalde low-level instructies, die programma's niet mogen gebruiken, niet beschikbaar, evenals het deel van het geheugen dat door het OS en andere programma's is gereserveerd. Dit wordt gerealiseerd vanuit de Memory Management Unit. Deze zorgt ervoor dat de programma's niet bij andermans geheugen kunnen, door alle geheugentoegang in user mode via de MMU te laten lopen. Wordt geprobeerd toegang te krijgen tot een verboden geheugengebied, dan licht de MMU middels een exception het OS in. Het OS kan dan besluiten het proces af te breken met een foutmelding zoals "Access violation" of "Segmentation fault". Zo kan het dus niet voorkomen dat processen elkaars of geheugen lezen of veranderen.

3 Bij paged virtual memory is de adresruimte opgedeeld in een aantal pagina's van gelijke grootte. Sommige van deze pagina's bevinden zich in het geheugen; de rest staat op de harde schijf. Als een pagina wordt opgevraagd die niet in het geheugen aanwezig is, moet deze daarheen worden gekopieerd. Dit gaat ten koste van een andere pagina in het geheugen, die naar de harde schijf wordt teruggeschreven. Om te bepalen welke pagina wordt vervangen zijn verschillende methoden beschikbaar.
Eén manier is om bij elke pagina in het geheugen bij te houden wanneer deze voor het laatst is gebruikt. De pagina die het langst niet is opgevraagd wordt dan verwijderd. Deze methode gaat echter de mist in bij de volgende situatie: het geheugen kan 8 pagina's bevatten, er een programma voert een lus uit die 9 pagina's in beslag neemt. Dan is telkens de volgende pagina die nodig is net teruggeschreven,

Naam: Thomas ter Laat
Adres: Aquamarijnstraat 419
Postcode en
Woonplaats: 9743 PK Groningen
Studentnummer: 1338862
Studierichting: Informatica & Wiskunde
Jaar van eerste inschrijving: 2002

Tentamen: Architectuur & Netwerken
Datum: 29-06-2004
Naam docent: Achterop

3 (vervolg) Een tweede mogelijkheid is om bij elke pagina een teller bij te houden. Deze wordt op 0 gezet als de pagina in het geheugen wordt geladen, en elke keer als een pagina vervangen wordt, worden alle tellers met 1 opgehoogd. De pagina met de hoogste teller heeft de meeste vervangingen meegemaakt en wordt als volgende verwijderd. Dit algoritme verwijdert dus altijd de pagina die het langst in het geheugen zit. Bij het voorgaande voorbeeld gaat ook deze methode echter mis, resulterend in een vervanging voor elke nieuwe pagina.

Netwerken

1 De precieze formule weet ik niet, maar deze moet ongeveer van deze vorm zijn:

$$d = 2f \left(1 - \frac{1}{\frac{S}{N}+1}\right)$$

Hierin is:
d de maximale hoeveelheid data (bits/s)
f de maximale frequentie die het kanaal kan overbrengen (Hz)
$\frac{S}{N}$ de signaal/ruisverhouding in het kanaal (tussen 0 en ∞)
Bij een perfect kanaal is $\frac{S}{N} = \infty$ en wordt de factor $(1 - \frac{1}{\frac{S}{N}+1})$ dus 1. Dan is de maximale hoeveelheid data gelijk aan de Nyquist frequentie $2f$. Bij een slecht kanaal ($\frac{S}{N} \approx 0$) is de factor $(1 - \frac{1}{\frac{S}{N}+1})$ bijna 0, dus kan er niet veel data per tijdseenheid worden overgebracht. Naarmate $\frac{S}{N}$ toeneemt, neemt ook de maximale bitrate toe. Als f hoger wordt kan ook meer data worden overgebracht; het verband tussen d en f is lineair.

2 Een router heeft interfaces met meerdere netwerken. Hij opereert voornamelijk in de netwerklaag, hoewel de onderliggende lagen hierbij natuurlijk onmisbaar zijn. Als een interface een frame binnenkrijgt in de datalinklaag wordt in de header hiervan gekeken of het pakket bestemd is voor de router. Zo ja, dan wordt het pakket doorgegeven aan de netwerklaag.

De netwerklaag bekijkt de headers van het pakket. Hierin staat het netwerkadres van de eindbestemming. (In tegenstelling tot het frame, dat het MAC-adres van de router bevatte.) In de routeringstabellen wordt opgezocht op welke interface het pakket verzonden moet worden om het op de plaats van bestemming te krijgen. Het wordt weer ingekapseld in een frame, met als bestemmingsadres het MAC-adres van de volgende router (of, als er een rechtstreekse verbinding mee is, de eindbestemming) en het frame wordt op de juiste interface verstuurd. De fysieke laag en de datalinklaag van beide netwerken hoeven niet compatibel te zijn, en zijn compleet van elkaar gescheiden; de netwerklaag moet (multiprotocol routers uitgezonderd) wel op bron-, tussenliggend en bestemmingsnetwerk hetzelfde zijn.

3 Quality of Service is het garanderen van bepaalde kwaliteiten van een verbinding. Dit kan gaan om vertraging, bandbreedte of jitter (fluctuaties in vertraging van verschillende pakketten). Bij een connectie-georiënteerd netwerk is het goed te doen om dit soort QoS-garanties te geven: bij het opzetten van de verbinding wordt de benodigde bandbreedte, geheugenruimte en rekenkracht gereserveerd, zodat deze altijd beschikbaar is zolang de verbinding in stand blijft. Bij connectieloze netwerken zoals IP is dit veel lastiger, omdat in de routers niets over een connectie bekend is en niet alle pakketten dezelfde route hoeven te nemen. Om dit te verhelpen zijn een aantal protocollen ontwikkeld die het mogelijk maken, mits de routers het ondersteunen, om ook over deze netwerken verbindingen met een bepaalde QoS op te zetten. Eén van de mogelijkheden is hierbij om streams (vaak live audio en/of video) samen te voegen zolang ze dezelfde route volgen. Zo hoeft een uitzending maar één keer te worden verstuurd door de bron en wordt deze gesplitst in de routers waar dat nodig is. Dit bespaart bandbreedte en andere resources en helpt zo mee een hogere QoS te kunnen garanderen.

7 1/2