Postcode en Woonplaats:

Jaar van eerste inschrijving:

Datum: 7 februari 2025

Naam docent:

3

Er wordt op de machines gebruik gemaakt van een logische klok. De klokken op de machines hoeven niet gelijk te lopen met elkaar op met de globale klok. Wel moet de tijd in elk systeem altijd oplopen.

Voor events a en b binnen proces P geldt dat C(a) < C(b) (de timestamp van a is kleiner dan de timestamp van b), als a volgens de logische klok eerder gebeurde dan b.

Als a het verstuuren van een boodschap m is en b het ontvangen van m, dan geldt dat a eerder gebeurde dan b en dat C(a) < C(b).

Een timestamp geeft de tijd aan waarop een event voorkomt en wordt aan de event gekoppeld

Wanneer gebruik wordt gemaakt van timestamps, kunnen events in de tijd geordend worden. Hierdoor kan ervoor gezorgd worden dat events op verschillende plaatsen binnen een gedistribueerd systeem in de goede volgorde voorkomen, zodat er geen inconsistentie ontstaat.

Een voorbeeld waarbij volgorde belangrijk is: Er is een bank. Van de database van de bank zijn twee kopien K_1 en K_2. Er zijn twee personen: een klant wil 100 euro op zijn rekening zetten en een medewerker wil klanten 1% rente geven. De klant is dichtbij K_1 en de medewerker is dichtbij K_2. Stel dat er geen gebruik wordt gemaakt van (Lamport) timestamps. Wanneer beide personen tegelijk hun actie willen uitvoeren, kan het zijn dat de actie van de klant eerder wordt uitgevoerd dan die van de medewerker op K_1, terwijl dat voor

dat de klant volgens K_1 (1000 + 100) x 1,01 = 1111 euro heeft en 1000 x 1,01 + 100 = 1110 euro volgens K_2. De volgorde van beide acties is niet heel belangrijk, maar om de gegevens consistent te houden, moet de volgorde ~~van~~ van uitvoering bij beide ~~de~~ kopies het zelfde zijn. Hiervoor kan gebruik gemaakt worden van timestamps.

4 Wanneer clients een bestand lezen, willen ze de juiste ~~informatie~~ data lezen, de meest recent geschreven ~~informatie~~ data. Maar als er meerdere replicaties zijn die mogelijk niet altijd bij elkaar zijn, kan het erg veel tijd en moeite kosten om alle replicaties bij elke schrijfoperatie (dat kunnen er veel zijn) bij te werken. Het kan zijn dat clients hierdoor steeds lang moeten wachten en dat is niet wenselijk. Daarom worden verschillende consistency modellen gebruikt. In een situatie waarin veel geschreven wordt en weinig gelezen, bijvoorbeeld, hoeven de replica's niet na elke schrijfoperatie consistent te worden gemaakt.

Een consistency model geeft als het ware aan wat de eisen zijn voor het systeem. Wanneer en hoe replica's bijgewerkt moeten worden verschilt per model.

1 Bij RPC en RMI is meer sprake van distribution transparency dan bij het uitwisselen van IP-pakketten. Wanneer een client bij RPC- of RMI iets gedaan ~~wil~~ wil hebben, hoeft er eenvoudigweg niet veel meer gedaan te worden dan een aanroep. De applicatie die de aanroep doet hoeft niks te weten ~~over~~ van een server die de uiteindelijke operaties uitvoert. Om deze simpliciteit is het gebruik van RPC en RMI zo populair. Bij het gebruik van IP-pakketten moet uitgezocht worden

Postcode en Woonplaats: | Jaar van eerste inschrijving: | Datum: Naam docent:

(1) middleware gedaan worden. Applicatieprogrammeurs laten dergelijke dingen dus liever voor hen doen (door het systeem), dat bespaart werk.

Een RMI tussen twee verschillende machines:
Op de client machine wordt een methode aangeroepen via een interface van het object. De bijbehorende proxy verwerkt de methode-aanroep in een message en zorgt ervoor dat deze naar de server machine gestuurd wordt. Daar zet de skeleton de informatie uit de message om in de juiste methode-aanroep en roept die methode aan bij het object. Het resultaat wordt vervolgens door de skeleton verwerkt in een nieuwe message en naar de proxy gestuurd. De proxy reconstrueert het resultaat uit de message en geeft deze door aan de applicatie.

2 Het aantal referenties naar een entiteit wordt niet altijd goed bijgehouden. Wanneer een proces die entiteit een bericht stuurt om aan te geven dat het ernaar gaat refereren, kan het bericht verloren raken. Ook een bericht om aan te geven dat een proces niet meer naar een entiteit zal refereren, kan verloren gaan. In het geval dat berichten bevestigd moeten worden, kan het ook fout gaan. Wordt dan een bericht ontvangen en verwerkt (teller voor referenties omhoog of omlaag), maar de acknowledgement raakt verloren, dan zal het proces het bericht opnieuw versturen en wordt het nog eens verwerkt (en dus te vaak).

Als de entiteit het aantal referenties naar zichzelf niet goed bijgehouden heeft, kan het zijn dat de entiteit denkt dat er geen referenties meer naar hem

kunnen dat de entiteit denkt dat er wel referenties naar hem zijn, terwijl dat niet zo is. In beide gevallen is er een probleem. Zo kan een entiteit waar nog referenties naar zijn opgeruimd worden en sommige entiteiten blijven onnodig voortbestaan.

Het aantal referenties naar een entiteit moet daarom goed bijgehouden worden. Dit kan als processen idempotente berichten versturen om aan te geven dat ze een referentie willen of juist niet meer. Dat wil zeggen dat de berichten, bijvoorbeeld bij het verloren raken van de acknowledgement, bij een tweede keer niet ~~de~~ de teller weer zullen beïnvloeden. Hiervoor zouden de berichten van een nummer voorzien kunnen worden. Door het verzenden van acknowledgements (werken met timeouts), zorgen we ervoor dat berichten die ~~verloren~~ (mogelijk) verloren zijn geraakt opnieuw verstuurd zullen worden, zodat ze minstens één keer aan zullen komen (en verwerkt zullen worden).

Elk bericht wordt zo precies één keer verwerkt, waardoor het aantal referenties correct bijgehouden kunnen worden.

5 Wanneer het kanaal niet veilig is, kan gebruik worden gemaakt van de sleutel $g^{xy} \bmod n$. Hierbij mag bekend worden gemaakt wat g en n zijn. Alice kiest x en stuurt g, n en $g^x \bmod n$ naar Bob. Bob kiest y en stuurt Alice $g^y \bmod n$. Alice kan de sleutel berekenen: $(g^y \bmod n)^x$. Bob kan de sleutel berekenen: $(g^x \bmod n)^y$. En als ~~gg~~ de getallen goed gekozen zijn ~~Eggee~~ (voldoende groot), is de sleutel zeer moeilijk te raden.

authenticatie: