NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

Løsning på eksamen i TDT4190 Distribuerte systemer
Fredag 26. mai 2006, 1500-1900

Det ønskes korte og konsise svar på hver av oppgavene. Det vesentlige er å kunne dokumentere forståelse, beherske prinsipper og se sammenhenger - ikke å kunne gjengi en mengde detaljer.

Der det synes å mangle noen opplysninger, må det angis hvilke antagelser som synes å være naturlige. Merk at viktige begreper er angitt på både norsk og engelsk.

Oppgave 1 – Distribuert synkronisering (Distributed synchronization) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver synkronisering i distribuerte systemer står overfor i forhold til synkronisering i sentraliserte systemer

SVAR:

Generelt

Unngå avhengighet av enkeltnoder / enkeltmeldinger
Unngå / begrens flaskehalser i systemet (noder / lenker)
Begrens meldingsmengder / prosesseringsforsinkelser

b) Beskriv og vurder ringalgoritmen for distribuert primas utvelgelse (distributed process election)

SVAR:

Beskrivelse

```
Start Valg-rutine:
    1a) Send Valg-melding m/ Egen identifikator
```

```
Får Valg-melding:
    2a) Med høyere identifikator og Egen ikke inkludert / Egen inkludert:
            Send videre m/ Gitt identifikator
    2b) Med lavere identifikator og Egen ikke inkludert:
            Send videre m/ Egen identifikator
    2c) Med lavere identifikator og Egen inkludert:
            Stopp Valg-melding
    2d) Med samme identifikator og Egen inkludert:
            Send Sjef-melding m/ Egen identifikator
```

```
Får Sjef-melding:
    3a) Med høyere identifikator: Send videre m/ Gitt identifikator
    3b) Med samme identifikator: Avslutt rutine
```

Vurdering

Det totale antall meldinger kan bli høyt (-> 3N)
Den totale forsinkelse kan bli lang (-> 3N)
Hver node må kjenne adressen til alle de andre i ringen
Ingen virkelig god algoritme, men denne er mye brukt

Oppgave 2 – Distribuerte filsystemer (Distributed file systems) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver filsystemer for distribuerte omgivelser står overfor i forhold til filsystemer for sentraliserte omgivelser

SVAR:

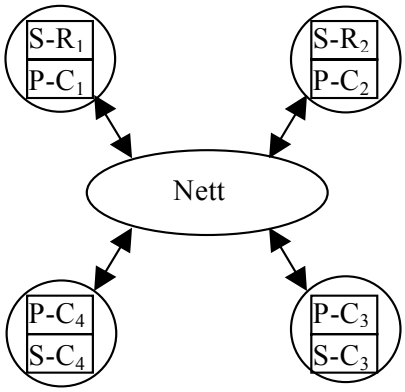
Generelt

Ulike filer kan være plassert på forskjellige noder, under ulike aksessregimer (autonomi), på ulik form (modell) og med permanente kopier (replikater) og/eller temporære kopier (cacher) på andre noder igjen.

b) Beskriv og vurder en algoritme for oppdatering av cachete (cached) og repliserte (replicated) filkopier (file copies)

SVAR:

Beskrivelse



Her betraktes en bestemt fil i et distribuert filsystem med fire noder. Filen eksisterer i to replikater på nodene N_1 og N_2 – henholdsvis S-R_1 og S-R_2. På et gitt tidspunkt forekommer det også diskcachete versjoner av filen på nodene N_3 og N_4 – henholdsvis S-C_3 og S-C_4, samt vanlig cachete versjoner i hovedlagerne på hver av nodene N_1, N_2, N_3 og N_4 – henholdsvis P-C_1, P-C_2, P-C_3 og P-C_4.

* Lokal caching – dvs. forhold mellom P-C og S-R samt mellom P-C og S-C:
Bruk tilbakeskriving – altså ikke umiddelbar oppdatering av S-versjon ved oppdatering av P-versjon

* Global caching – dvs. forhold mellom en P-C ved en S-C og en P-C ved en S-R:
Bruk gjennomskriving – altså umiddelbar oppdatering av S-R cache ved oppdatering av S-C cache

P-C_3 := X => (vanligvis med invalidering mer enn endring)

Umiddelbart:
P-C_2 & P-C_1 := P-C_3 (uansett); P-C_4 := --- / P-C_3 (avhengig av invalidering ! / endring ?)
Hvis invalidering av P-C_4 => Også invalidering av S-C_4

Senere:
S-R_2 := P-C_2 & S-R_1 := P-C_1 & S-C_3 := P-C_3 (uansett)
Hvis ikke invalidering av P-C_4 => Også S-C_4 := P-C_4

P-C_2 := X => (vanligvis med endring mer enn invalidering)

Umiddelbart:
P-C_1 := P-C_2 (uansett); P-C_4 & P-C_3 := P-C_2 / --- (avhengig av endring ! / invalidering ?)
Hvis invalidering av P-C_4 & P-C_3 => Også invalidering av S-C_4 & S-C_3

Senere:
S-R_1 := P-C_1 & S-R_2 := P-C_2 (uansett);
Hvis ikke invalidering av P-C_4 & P-C_3 => Også S-C_4 := P-C_4 & S-C_3 := P-C_3

Vurdering

En ganske effektiv algoritme – og i henhold til standard aksjons-semantikk

Oppgave 3 – Distribuerte databasesystemer (Distributed database systems) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver databasesystemer for distribuerte omgivelser står overfor i forhold til databasesystemer for sentraliserte omgivelser

SVAR:

Generelt

Dataene kan bli oppdelt i mindre fragmenter som så blir fordelt på ulike noder

Fragmentene kan igjen forefinnes i flere kopier som tilsvarende repliseres på ulike noder

b) Beskriv og vurder algoritmen for 2-fase-låsing (2-phase-locking)

SVAR:

Beskrivelse

2PL

* Enhver transaksjon kreves å sette en lås på ethvert dataelement før det skal aksesseres / oppdateres, og å fjerne låsen igjen når dataelementet ikke lenger skal aksesseres / oppdateres

* Systemet forventes kun å tillate kompatible låser mellom ulike transaksjoner - dvs. at låser som ikke kan tillates samtidig på et gitt dataelement, faktisk ikke aksepteres på vegne av ulike transaksjoner

* En transaksjon kan ikke sette en ny lås etter at en av dens eksisterende låser er fjernet – dvs. alle dens låser settes før noen av dem fjernes

Vurdering

En funksjonelt sett god algoritme – men samtidig en effektivitetsmessig krevende algoritme. Den tillater kun serialiserbare utførelser, men den tillater mindre parallellitet enn hva full serialiserbarhet tilsier.

Oppgave 4 – Distribuert pålitelighet (Distributed reliability) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver pålitelighet i distribuerte systemer står overfor i forhold til pålitelighet i sentraliserte systemer

SVAR:

Generelt

Noder kan feile uavhengig av hverandre, og meldinger mellom noder kan forsvinne, dubliseres, ødelegges, omrokkeres, avlyttes og/eller manipuleres

Et system må likevel kunne brukes selv om noen noder er tilgjengelige mens andre noder er utilgjengelige på et vilkårlig tidspunkt, og et system bør også kunne brukes som om det hadde bestått av en eneste node

b) Beskriv og vurder en algoritme for sikring av atomiskhet (atomicity) ved meldingsutveksling (message exchange)

SVAR:

Beskrivelse

”2PC”

* Avsender sender meldingen til alle tiltenkte mottakere - med beskjed om å bekrefte mottaket og samtidig avvente agering på mottaket

* Mottakere som ikke bekrefter å ha fått meldingen, vil få den tilsendt et visst antall ganger til

* Hvis absolutt alle mottakere bekrefter mottaket, vil alle få beskjed om å agere på mottaket. Hvis minst en mottaker ikke bekrefter mottaket, vil alle få beskjed om å kansellere mottaket.

Vurdering

En funksjonelt sett god algoritme – men samtidig en effektivitetsmessig krevende algoritme. Den sikrer atomiskhet, men den krever fire meldingsrunder.

Oppgave 5 – Distribuerte navnetjenester (Distributed name services) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver navnetjenester i distribuerte systemer står overfor i forhold til navnetjenester i sentraliserte systemer

SVAR:

Generelt

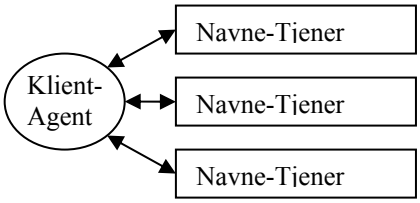
Navnerommet kan være fysisk fordelt på ulike navnetjenere – og det på en helt annen måte enn den måten navnerommet logisk er oppdelt på

Således vil både replikasjon av adresseinformasjon og caching av søkeresultater på ulike noder kunne variere. Likeledes vil navigasjon mot navn og binding av navn ved opprulling av flernivånavn kunne variere.

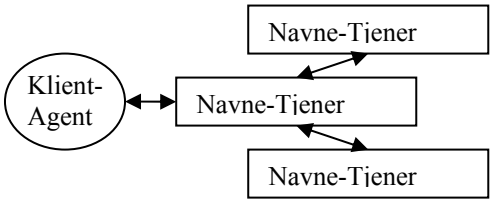
b) Beskriv og vurder en algoritme for opprulling (resolution) av flernivånavn (multilevel names) i hierarkiske navnerom (hierarchical name spaces)

SVAR:

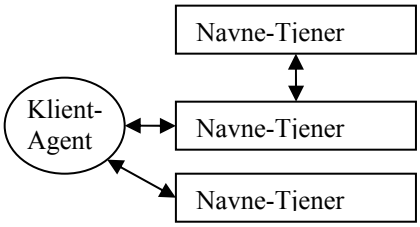
Beskrivelse



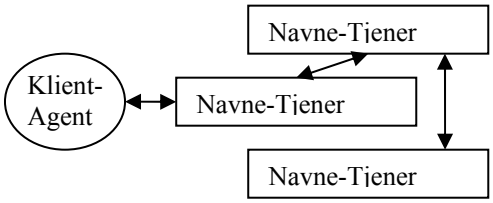
Mange - Iterativt



En - Iterativt



Mange - Rekursivt



En - Rekursivt

4 ulike varianter – hvorav en er nok

Mange – Iterativt

Klienten avklarer ett og ett delnavn v.hj.a. hver sin første-tjener

En – Iterativt

Klienten kontakter en første-tjener som så avklarer ett og ett delnavn v.hj.a. hver sin andre-tjener

En – Rekursivt

Klienten kontakter en første-tjener som avklarer første delnavn før den kontakter en andre-tjener som avklarer andre delnavn før den kontakter en tredje-tjener som avklarer tredje delnavn osv.

Mange – Rekursivt

Kombinasjon av Mange-Iterativt og En-Rekursivt

Vurdering

4 ulike varaianter – hvorav en er nok

Mange – Iterativt

-: Klienten blir flaskehals
+: Tillater caching på klienten
-: Tillater ikke sentralisert autorisering
+: Ingen første-tjener blir ”single-failure” utsatt

En – Iterativt

+: Klienten blir ikke flaskehals
-: Tillater ikke caching på klienten
-: Første-tjeneren blir flaskehals
+: Tillater caching på første-tjeneren
+: Tillater sentralisert autorisering
-: Første-tjeneren blir ”single-failure” utsatt

En – Rekursivt

+: Klienten blir ikke flaskehals
-: Tillater ikke caching på klienten
+: Første-tjeneren blir ikke flaskehals
-: Tillater ikke caching på første-tjeneren
+: Tillater sentralisert autorisering
-: Første-tjeneren blir ”single-failure” utsatt

Mange – Rekursivt

-: Klienten kan bli flaskehals
+: Tillater delvis caching på klienten
-: Tillater ikke sentralisert autorisering
+: Er mest fleksibel av alle opplegg

Oppgave 6 – Distribuert delt lager (Distributed shared memory) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver delt lager i distribuerte systemer står overfor i forhold til delt lager i sentraliserte systemer

SVAR:

Generelt

Et distribuert delt lager tilsier en simulering av felles lager i en omgivelse uten felles lager. Dette gir både funksjonalitetsspørsmål (hvordan bør det gjøres for at det skal bli mest likt et felles lager) og effektivitetsspørsmål (hvordan bør det gjøres for at skal bli raskest nok).

b) Beskriv og vurder algoritmen for lesing og skriving av dataelementer i IVY/MIRAGE

SVAR:

Beskrivelse

Sidebasert (dvs. OS-simulering) – Sekvensiell konsistens (som lagringskrav)

* Måte:
Globale sider er spredt på lokale noder – Ingen eier en gitt side permanent
En gitt side kan være skitten (W-tilstand) eller ren (R-tilstand)
En skitten side vil finnes i en kopi – En ren side kan finnes i flere kopier
En ønsket side kan være på tilsvarende node eller ikke

* Resultat:
6 ulike tilfeller for leseaksess & 6 ulike tilfeller for skriveaksess
Eier-lokalisering:
Benytt sentral forvalter / Forfølg sannsynlig eier
Kopi-lokalisering:
Forvalter evt. eier(e) holder kopiliste(r) / Forespørrer benytter kringkasting
Flere aktive aktører gjør spontane invalideringer mulig:
Behov for nye sideutbyttingsalgoritmer

Vurdering

En god algoritme – som dekker mange relevante aspekter for aksessering og oppdatering i et distribuert delt lager

Oppgave 7 – Distribuerte multimediasystemer (Distributed multimedia systems) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver multimediasystemer for distribuerte omgivelser står overfor i forhold til multimediasystemer for sentraliserte omgivelser

SVAR:

Generelt

Svært korte tur-retur tider kreves for interaktive MM-applikasjoner
Store, skiftende ressurskrav som ikke kan monopolisere parallelt kjørende applikasjoner

b) Beskriv og vurder bøttebufferalgoritmen (token bucket algorithm)

SVAR:

Beskrivelse

* En mediakilde tøyles ved at det innsettes en spesiell buffer mellom den og en medieavtaker * Bufferen inkluderer en generator som kontinuerlig fyller en bøtte med buffertoken * Bøttebufferen vil nedskalere uten helt å fjerne variasjonene i mediastrømmen

Vurdering

En ganske effektiv algoritme – ved at den implementerer en lineært bundet ankomststrøm

Oppgave 8 – Distribuert sikkerhet (Distributed security) – 12.5 %

a) Angi hvilke ekstra utfordringer og oppgaver sikkerhet i distribuerte systemer står overfor i forhold til sikkerhet i sentraliserte systemer

SVAR:

Generelt

Forespørsler og resultater kan gå via meldinger over ett / flere nett hvor de kan lett kan utsettes for ulike former for manipulasjon (aksess og endring i ulike former)

Kontroll av ulike aspekter blir vanskeligere på tvers av noder (identitet og rettigheter generelt sett)

b) Beskriv og vurder Needham-Schroeder algoritmen for meldingsautentisering (message authentication)

SVAR:

Beskrivelse

Autentisering (av aktør A og B seg imellom via en tjener S) 1) A->S: A, B, N_A 2) S->A: $\{N_A, B, K_{AB}, \{K_{AB}, A\}K_B\}K_A$ 3) A->B: $\{K_{AB}, A\}K_B$ 4) B->A: $\{N_B\}K_{AB}$ 5) A->B: $\{N_B\text{-}1\}K_{AB}$

Vurdering

En funksjonelt sett ganske riktig algoritme – kun manglende et tidsmerke i melding nr. 3