NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

Løsning på kontinuasjonseksamen i TDT4190 / SIF8042 Distribuerte systemer
August 2005, 0900-1300

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 – Distribuerte systemer generelt (Distributed systems in general) – 12.5 %

a) Angi kort forskjellene mellom multiprosessorer (multi CPUs) og multimaskiner (multi computers)

SVAR:

- Multiprosessorer har felles lager og felles klokke
  Multimaskiner mangler felles lager og felles klokke
- Multiprosessorer er lette å programmere men vanskelige å skalere
  Multimaskiner er vanskelige å programmere men lette å skalere

b) Beskriv hvilke utfordringer det gir å designe og implementere distribuerte systemer

SVAR:

- Implementasjon av transparens – dvs. å kunne skjule distribusjon for brukeren
  - lokalisering, aksessering, fragmentering, replisering, skalering, migrering, parallellitet, feil

- Implementasjon av konsistens – dvs. å kunne maskere distribusjon innen systemer
  - systemtid, ressurstilstand, infoverdier

Oppgave 2 – Modeller og standarder (Models and standards) – 12.5 %

a) Angi kort hva vi bruker mellomvare (middleware) til

SVAR:

En konstruerer et programvare lag (med et generelt API) mellom globale applikasjoner og lokale ressurser slik at en gitt applikasjon kan nå en vilkårlig lokal ressurs, og en gitt ressurs kan nås av en vilkårlig applikasjon – hvor mellomvare laget står for ruting av forespørslene til riktig node og oversetting av forespørslene til riktig språk

b) Beskriv hovedprinsippene i OSFs DCE (Distributed Computing Environment)

SVAR:

Mellomvare-arkitektur – RPC-type

DCE-tjenester
- Fundamentale tjenester
- Delingstjenester

Fundamentale tjenester
- Tråder: POSIX 1003.4a Ptråder
- RPC: En/flere bindingsmåter per tjeneste – En/flere grensesnitt per tjener
- Tid: Fysisk klokke, UTC-basert
- Navn: Hierarkisk, lokal og global – Interoperabel, X.500 og DNS
- Sikkerhet: Autentisering, Autorisering, Verifisering, Kryptering

Delingstjenester
- Sekundærlager filer: POSIX 1003.1 Filer – Disk basert
- Primærlager filer: BOOTP- og TFTP-basert
- Forvaltning: Høy tilgjengelighet via høy replisering

Oppgave 3 – Kommunikasjon og synkronisering (Communication and synchronization) – 12.5 %

a) Angi kort hva vi trenger henholdsvis logiske klokker (logical clocks) og vektorklokker (vector clocks) til

SVAR:

Logiske klokker og vektorklokker brukes til å implementere et felles tidsbegrep

Logisk klokker holder oversikt over hvor mange hendelser på alle nodene samlet som er kjent på en gitt node

Vektorklokker holder oversikt over hvor mange hendelser i hver enkelt node som er kjent på en gitt node

Vektorklokker – ut over logiske klokker – skiller mellom hendelser på ulike noder

b) Beskriv en algoritme for distribuert gjensidig utelukkelse (distributed mutual exclusion)

SVAR:

Av de tre nedenfor viste varianter (enkel / avansert / typisk) holder det med en -
helst den typiske eller den avanserte, helst ikke den enkle

Sentralisert algoritme (enkel)

Eget Inn-ønske: 1a) Send Inn-melding til Global Koordinator 1b) Vent på OK-melding fra Global Koordinator

Eget Ut-ønske: 2a) Send Ut-melding til Global Koordiator

Ulike Inn-meldinger: 3a) Ingen inne: Send OK 3b) Noen inne: Sett i Kø

Ulike Ut-meldinger: 4a) Noen i Kø: Fjern fra Kø og Send OK

Distribuert algoritme (avansert)

Eget Inn-ønske: 1a) Send N-1 * Inn-meldinger m/ Lokalt tidsmerke 1b) Vent på N-1 * OK-meldinger

Andres Inn-meldinger: 2a) Ikke ventende: Send OK 2b) Ventende, høyere tidsmerke: Send OK 2c) Ventende, lavere tidsmerke: Sett i Kø (Like tidsmerker: Identifikator avgjør)

Eget Ut-ønske: 3a) Send $\leq$ N-1 * OK ut fra Kø

Ringalgoritme (typisk)

Eget Inn-ønske: 1a) Vent på OK-melding

OK-melding: 2a) Ikke ventende: Send OK-melding Videre 2b) Ventende: Behold OK-melding

Eget Ut-ønske: 3a) Send OK-melding Videre

Oppgave 4 – Distribuerte filsystemer (Distributed file systems) – 12.5 %

a) Angi kort hva vi bruker henholdsvis caching (caching) og replisering (replication) til

SVAR:

Caching og replisering innfører kopier av et dataelement nærmere bruksstedene enn det dataelementet i seg selv tilsier

Caching tilbyr en forholdsvis temporær kopi
Replisering tilbyr en mer permanent kopi

Disse kopiene – permanent i større grad enn temporær – tilsier også økt sikkerhet i tillegg til økt nærhet

b) Beskriv hovedprinsippene i CMUs AFS (Andrew File System)

CMU AFS	
Globalt filsystem:	Felles
Cache type:	Hele filer
Konsistens:	Underretting
Oppdat. Semantikk:	Fil
Replikat type:	Kun RO
Orientering:	Feiltoleranse
Implementasjon:	Bruker + Kjerne

Oppgave 5 – Distribuerte databasesystemer (Distributed database systems) – 12.5 %

a) Angi kort hva vi trenger 2-fase-låsing (2 phase lock) til

SVAR:

2PL brukes til å forhindre gale resultater i.f.m. at flere applikasjoner aksesserer og/eller endrer data samtidig

Inkonsistente uthentinger og tapte oppdateringer er typiske eksempler på dette

b) Beskriv hvordan 2-fase-bekrefting (2 phase commit) virker

SVAR:

2PC innfører en todelt avslutning: 1.fase m/stemming & 2. fase m/utføring –
etter alle-har-veto prinsippet koplet til enten feil eller autonomi

2PC sikrer autonomi og effektivitet med lite overhead for normaltilfeller uten mye feil

Oppgave 6 – Distribuert pålitelighet (Distributed reliability) – 12.5 %

a) Angi kort hvilke utfordringer replisering (replication) og / eller kommunikasjonsfeil (communication errors) gir i distribuerte databasesystemer (distributed database systems)

SVAR:

Uten replisering – Uten kommunikasjonsfeil:
- Ingen store utfordringer

Uten replisering – Med kommunikasjonsfeil:
- 3PC-terminering av transaksjoner kan gi blokkering i forsøk på å oppnå atomiskhet

Med replisering – Uten kommunikasjonsfeil:
- 2PL-låsing for transaksjoner må få utvidelser for å sikre kollisjon av tilhørende låser

Med replisering – Med kommunikasjonsfeil:
- MajKonsensus- / VirtPartisjon-algoritmer må til for å utpeke partisjon for oppdatering
- 2PL-låsing for transaksjoner må få utvidelser for å sikre kollisjon av tilhørende låser
- 3PC-terminering av transaksjoner kan gi blokkering i forsøk på å oppnå atomiskhet

b) Beskriv en algoritme for distribuert oppdatering (distributed updating) hvor både replisering (replication) benyttes og kommunikasjonsfeil (communication errors) forekommer

SVAR:

Av de tre nedenfor viste varianter (1 enkel / 2 avanserte) holder det med en,
og til og med den enkle – selv om den ikke er en komplett løsning

- *Tilgjengelige kopier (enkel – kun serialiserbarhet per node)*

Aksessering underveis: Les og R-lås en tilgjengelig kopi ! Skriv og W-lås alle tilgjengelige kopier !

Validering til slutt: Er alle tidligere tilgjengelige kopier fortsatt tilgjengelige ? Er alle tidligere utilgjengelig kopier fortsatt utilgjengelige ?

- *Majoritets konsensus (avansert – også serialiserbarhet mellom noder)*

Uthenting: Les og R-lås en majoritet av tilgjengelig kopier Returner den kopi med høyeste versjonsnummer

Oppdatering: Skriv og W-lås en majoritet av tilgjengelig kopier Inkluder i kopi et høyere versjonsnummer

- *Virtuelle partisjoner (avansert – også serialiserbarhet mellom noder)*

Kontroller etter hver tilstandsendring: Sikre tilgang til en majoritet av lesbare kopier i løpende tilstand Sikre tilgang til en majoritet av skrivbare kopier i løpende tilstand

Aksesser mellom tilstandsendringer: Les og R-lås en kopi i løpende tilstand Skriv og W-lås alle kopier i løpende tilstand

Oppgave 7 – Distribuerte navnetjenester (Distributed name services) – 12.5 %

a) Angi kort hvordan vi kan utnytte caching (caching) og replisering (replication) i slike tjenester i forhold til hvordan de må benyttes i distribuerte filsystemer (distributed file systems) generelt

SVAR:

I navnetjenester tilbyr caching og replisering kopier av pekerinformasjon
I filsystemer generelt tilbyr caching og replisering kopier av verdiinformasjon

Om pekerinformasjon er gyldig eller ikke oppdages ved å forfølge pekerne
Om verdiinformasjon er gyldig eller ikke avdekkes kun ved en sjekk mot originalen

Navnetjenester trenger således i mye mindre grad enn filsystemer generelt kontinuerlig å kontrollere gyldigheten av sin informasjon

b) Beskriv hovedprinsippene i GNS (Global Name System)

SVAR:

- *GNS*

Tillater sammenslåing og restrukturering
Tillater fleksibel attributtstruktur

Oppgave 8 – Distribuert delt lager (Distributed shared memory) – 12.5 %

a) Angi kort forskjellene mellom gradsavballansert konsistens (degrees of consistency) og tidsavballansert konsistens (times of consistency)

SVAR:

Gradsavballansering:

- Kopling av konsistens til anvendelser ved å avgjøre i hvilken grad DSM må være konsistent

  Prosessor konsistens:
  Alle prosessorer ser lokale operasjoner i samme rekkefølge

  Kausal konsistens:
  Alle prosessorer ser lokale og koplete, globale operasjoner i samme rekkefølge

  Sekvensiell konsistens:
  Alle prosessorer ser lokale og globale operasjoner i samme rekkefølge

  Altså gradvis tyngre krav: PK => KK => SK

Tidsavballansering:

- Kopling av konsistens til synkronisering ved avgjøre til hvilke tider DSM må være konsistent

  Båndleggingskonsistens:
  Fellesdata gjøres konsistente før kritiske regioner

  Frigjøringskonsistens:
  Fellesdata gjøres konsistente etter kritiske regioner

  Svak konsistens:
  Fellesdata gjøres konsistente både før og etter kritiske regioner

  Altså ikke helt gradvis tyngre krav: BK => SK & FK => SK

b) Beskriv hovedprinsippene i LINDA og ORCA

SVAR:

LINDA:

- Kjøretidssystem simulering av delt lager (dvs. postbasert)
- Objektbasert overføring (altså ikke variabelbasert)
- Enkel objektmodell (PROLOG-lignende)

ORCA:

- Kjøretidssystem simulering av delt lager (dvs. postbasert)
- Objektbasert overføring (altså ikke variabelbasert)
- Full objektmodell (JAVA-lignende)