

Multiprogrammering på PDP-11 under RSX11M

belyst med centrale eksempler fra
fjernkontrollsystemet BECOS-30

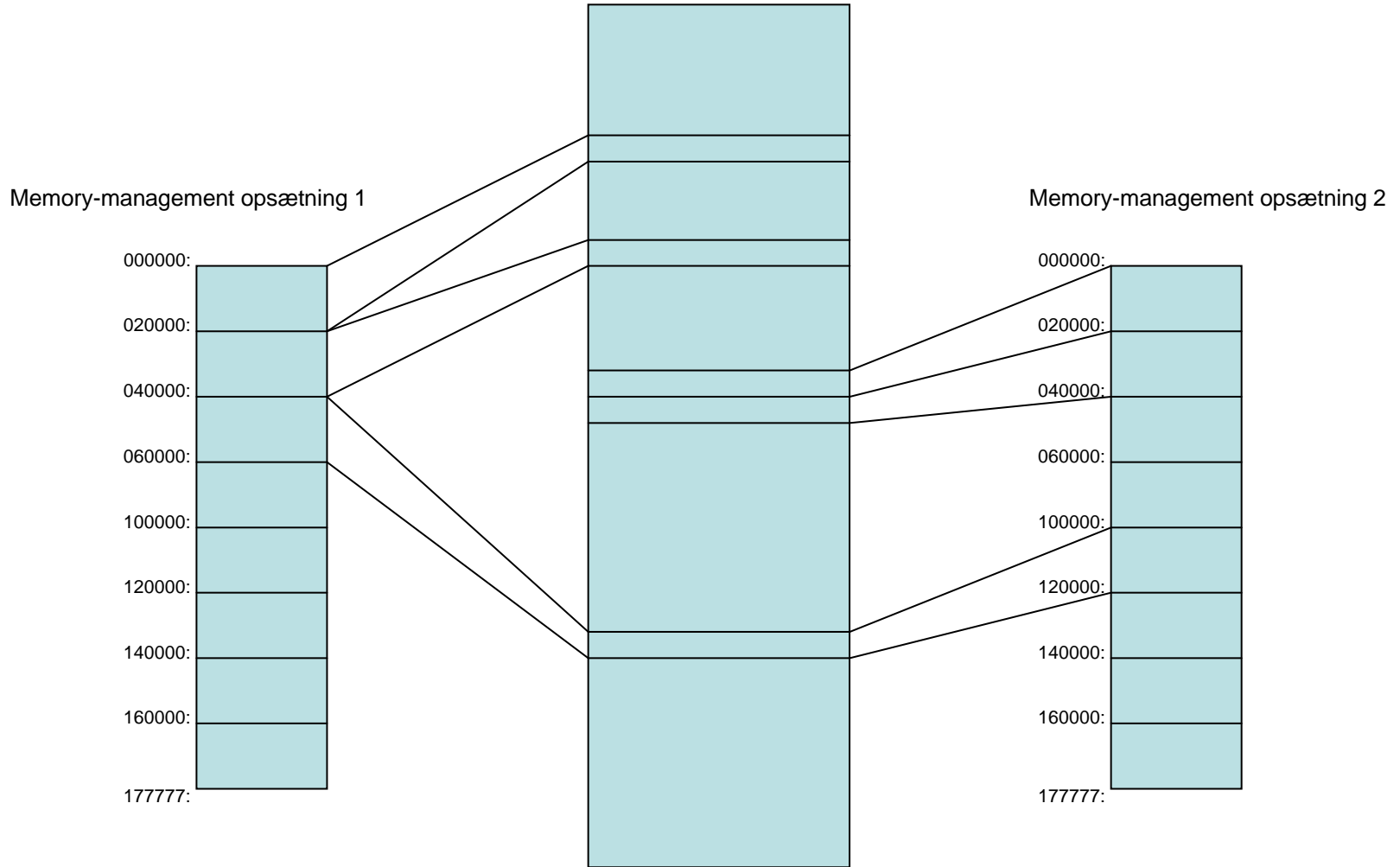
Oversigt

- BECOS-30: Fjernkontrollsystem, videreudvikling/efterfølger af Indactic 33/20 (aka. BECOS-20).
- Semafor-monitor på PDP-11 under operativsystemet RSX11M.
- BECOS-30's dobbeltsystems-funktion, med særlig fokus på, hvordan opstarten af standby-systemet foregår.

PDP-11's arkitektur og lager

- Processor: 16-bit adresserum, max 64kB
- Programmerbar 'Memory management' enhed.
- Opfanger processorens lager-accesser, tilføjer ekstra programmerbare adressebit, før fysisk lager (meget større end 64 kB) tilgås.
- Opdeler 16-bit adresserummet i 8 blokke på hver 8 kB.

CPU-adresser vs. fysisk lager



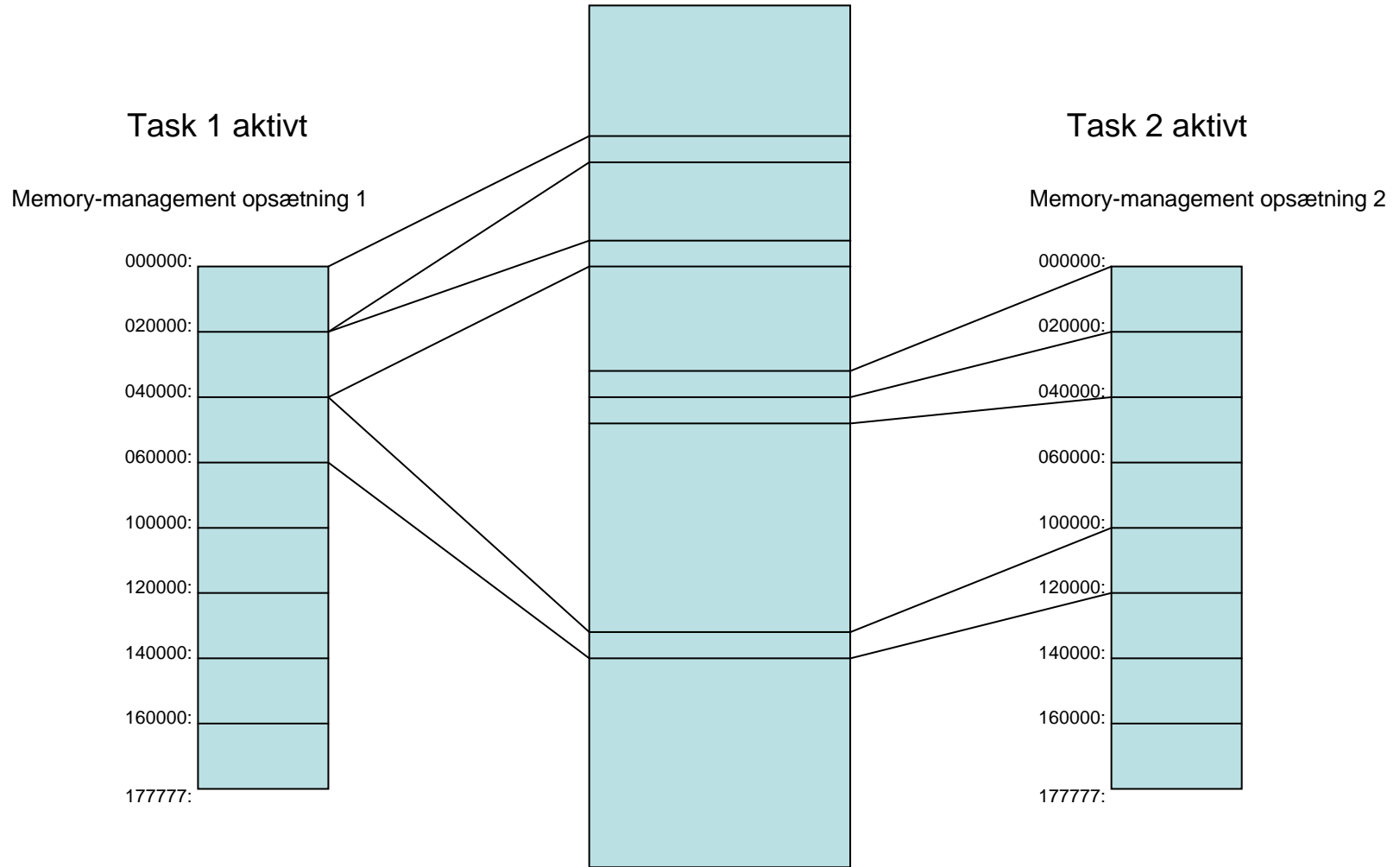
RSX11M – vigtige begreber

- Task: Grundlæggende ‘execution unit’. Ækvivalent med ‘proces’.
- Partition: Navngiven del af fysisk lager.
- Konfigurerbart, hvilke tasks skal køre i hvilke partitioner.
- Muligt at installere data/kode i partition.

RSX memory management

- RSX styrer 'memory management' enheden.
- RSX opsætter enheden i overensstemmelse med hvert enkelt task's krav/behov.
- Kan specificeres på TaskBuild (=link) tidspunkt.
- Mulighed for dynamisk opsætning.

CPU-adresser vs. fysisk lager



AST – Asynchronous System Trap

- Operativsystem-genereret 'interrupt'.
- Altid foranlediget af en af tasket initieret operativsystemsfunktion.
- Eksempler: Afslutning af I/O, udløb af timer, modtagelse af lille data-besked fra andet task.

Behovet for parallelle aktiviteter #1

- Task = forholdsvis kostbar operativsystems-ressource under RSX.
- Derfor ønske om mange parallelle aktiviteter indenfor samme task.

Behovet for parallelle aktiviteter #2

- Men også: Muligheden for at et antal aktiviteter indenfor et task kommunikerer med et antal andre aktiviteter indenfor samme, eller et andet RSX task.

Behovet for parallelle aktiviteter #3

- Så vidt muligt transparent, således at en aktivitet ikke behøver at vide, om en kommunikationspartner befinder sig i samme, eller et andet, RSX task.

Behovet for anvendelse af kendt teknik

- Fjernkontrol er kompliceret, derfor ønske om genbrug af velkendte teknikker.
- Derfor: Implementation af semafor-monitor under operativsystemet RSX.

Løsningen valgt i BECOS-30.

- Semafor-monitoren: 8 Kb fælles lager mellem alle deltagende tasks i en dertil indrettet RSX partition.
- Indeholder monitorkoden, som er implementeret positions-uafhængigt for at tillade de enkelte tasks selv at vælge, hvor de vil have monitoren liggende i adresserummet.
- Kø-pegepinde er relative til starten af monitoren.
- Monitoren indeholder desuden alle semaforer og alle buffere, som sendes frem og tilbage mellem coroutiner'ne.

Lyn-gennemgang af semafor-monitor funktioner

Simpel semafor (S): Ikke-negativt heltal, eller kø af ventende coroutiner.

Kø semafor (Q): Tom, kø af buffere, eller kø af ventende coroutiner.

CRECO	Opret en coroutine.
SIGS	Hvis en coroutine venter, aktiver, ellers tæl 1 op.
WAITS	Hvis > 0 tæl 1 ned og fortsæt, ellers vent.
SIGQ	Hvis en coroutine venter, aktiver med buffer, ellers sæt buffer i kø.
WAITQ	Hvis kø har buffere fortsæt med første buffer, ellers vent.
TWAITQ	Hvis kø har buffere fortsæt med første buffer, ellers fortsæt uden buffer.
PREAST	Nødvendig under RSX, forklaring følger.

Monitor-låsen, kravene

- Skal sikre udelelig adgang når monitor-datastrukturer manipuleres.
- Udeleligheden skal sikres såvel task-internt som mellem forskellige tasks.
- Ønskes implementeret uden RSX-kald, hvis åben (må ikke koste et (flere) RSX-kald at låse monitoren, hvis ingen andre er igang).

Monitor-låsen 1

- Monitor-låsen er en selvmodificerende subrutine på 2 16-bit maskinord, hvoraf det første skifter udseende alt efter om låsen er låst eller åben.
- Låsen er placeret som global variabel i monitoren, i princippet kendt af alle tasks, men kun brugt (læst/ændret) af monitoren selv.

Monitor-låsen 2

- Monitor-låsen i åben tilstand:

```
MONMX ::      MOV  R1 , ( R3 )  
              JMP  ( R4 )
```

- Monitor-låsen i lukket tilstand:

```
MONMX ::      EMT  377  
              JMP  ( R4 )
```

Monitor-låsen: Kaldet

- Kaldet:

```
(mov #monmx, r3)
MOV (PC)+,R1
EMT 377 ;104377
1$: MOV #SPNDD,R4 ;1*256.+45.
JSR R4,(R3)
BPL 1$ ; Igen hvis låst
TST (SP)+ ;; Fik låsen
```

- Monitor-låsen i åben tilstand:

```
MONMX:: MOV R1,(R3)
JMP (R4)
```

- Monitor-låsen i lukket tilstand:

```
MONMX:: EMT 377
JMP (R4)
```

Monitor-låsen: Oplåsningen

- Oplåsningen:

```
(mov #monmx, r3)
MOV (PC)+, (R3)
MOV R1, (R3)
; Udfør RSUM$ på alle tasks,
; der venter
```

- Monitor-låsen i åben tilstand:

```
MONMX:: MOV R1, (R3)
        JMP (R4)
```

- Monitor-låsen i lukket tilstand:

```
MONMX:: EMT 377
        JMP (R4)
```

Monitorens AST-håndtering

- Problemet: AST'en kan have behov for at kalde monitoren (SIGS/TWAITQ/SIGQ).
- Ikke ønskeligt at lukke for AST'er ifb. hvert eneste monitor-kald.
- AST'en kan derfor risikere at afbryde et igangværende monitor-kald, mens tasket har monitoren låst.

Løsningen: PREAST

- Hovedprincip: Tillad tasket at færdiggøre et evt. igangværende monitor kald.
- Fortsæt derefter AST'en, med tilladelse til (visse) monitor kald.

PREAST detaljer 1

- Hvis task ej i monitoren: Returner straks til AST'en.
- Ellers: Afstak RSX AST-kontekst til task-lokalt AST-save areal.
- Returner og tillad monitor at afslutte igangværende kald.

PREAST detaljer 2

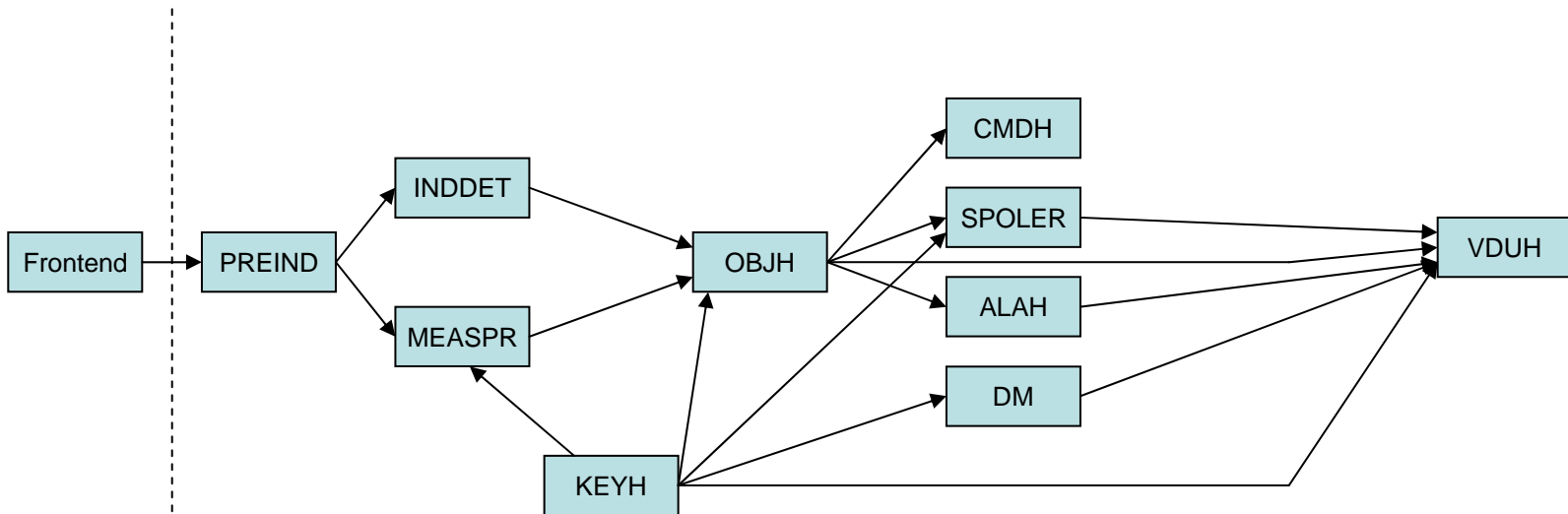
- Ved afsluttende oplåsning af monitoren: Tilbagestack AST-kontekst fra AST-save areal.
- Dog med endelig returadresse = udhop fra monitor, og ikke oprindelig afbrudsadresse.

PREAST detaljer 3

- Returner fra PREAST, til den egentlige AST.
- Tasket er nu udenfor monitoren, AST'en kan derfor udføre ikke-ventende monitorkald (SIGS, TWAITQ, SIGQ).
- AST'ens returhop er nu til monitorens udhop, og ikke oprindelig afbrudsadresse (inde i monitoren).

BECOS-30 systemoversigt

- RSX11M tasks, anvendere af monitoren.



Krav til drifts-pålidelighed

- Oppetid bedre end 99%
- Ingen 'single-points-of-failure'
- Vedligeholdelse/udskiftning af systemkomponenter må ikke indebære tab af drift.

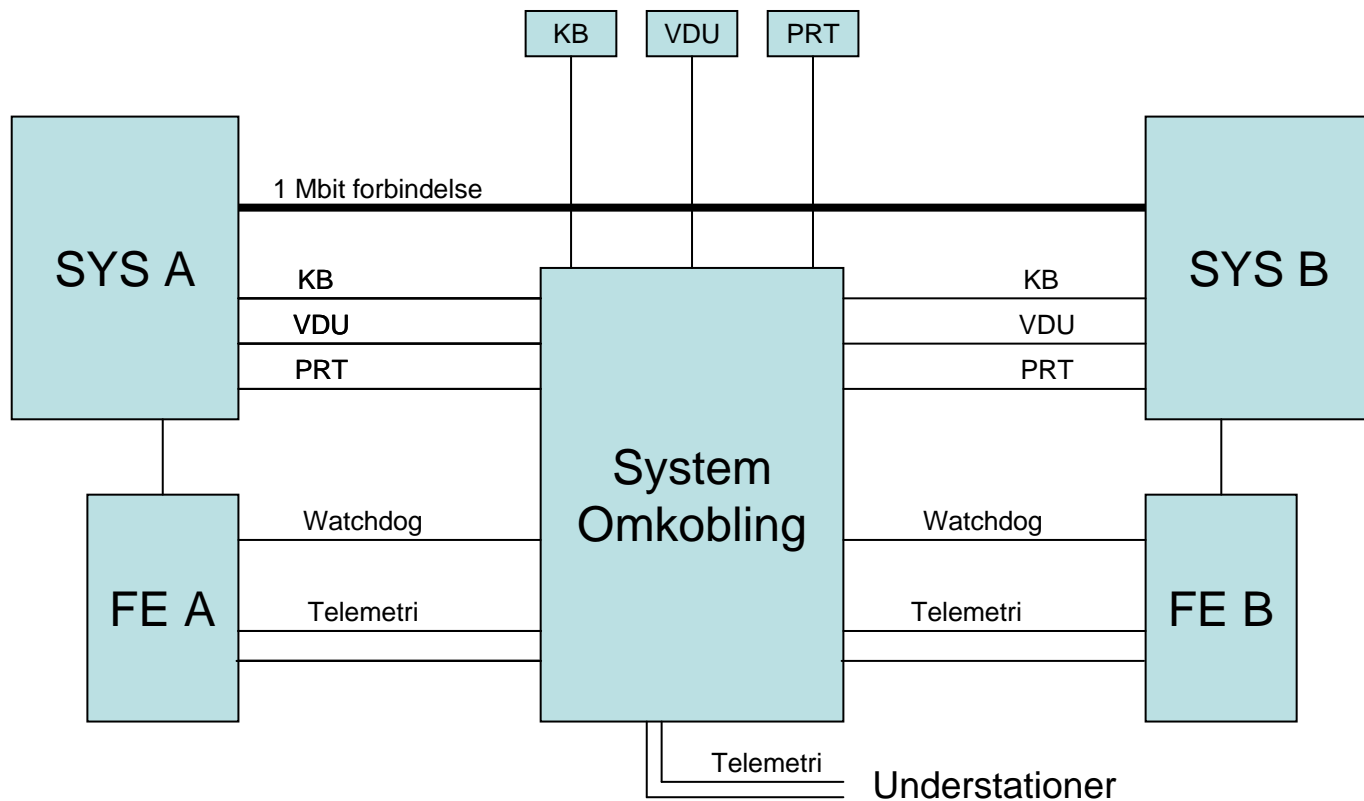
Løsningsvalg: Dobbeltssystem

- Dublering af samtlige centrale systemkomponenter.
- Implementation af 'online + hot-standby' system.

'Hot-standby' - definition

- 'hot-standby': Skal være klar til at overtage systemdrift på få sekunder.
- Kræver: Konsistent, til enhver tid opdateret database på 'hot-standby' systemet.

Dobbeltsystem HW-konfiguration



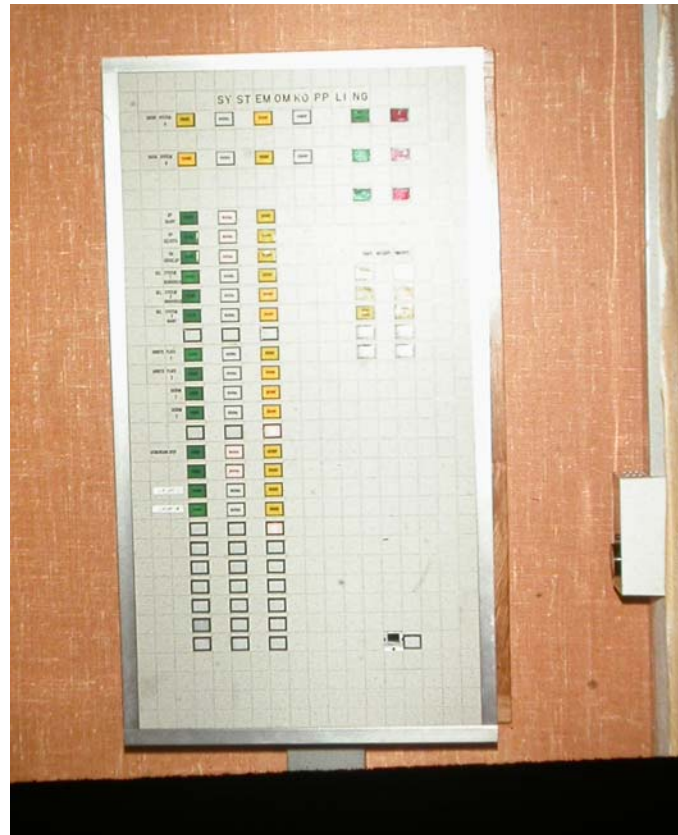
System-omkobling

- Elektromekaniske relæomskiftere,
- Kan omskifte forbindelserne fra ydre enheder (transmissionslinier til understationer, skærme, tastaturer, printere) mellem de 2 maskiner.

Watchdog-funktion

- Modul i Frontend maskinerne.
- Afgiver WD-alarmsignal ved manglende aktivering.
- Udløser automatisk omskiftning af alle enheder til standby-systemet,
- Som derefter overtager driften.

Systemomkobling, betjening



Systemomkobling, relæskab



Jesper Naur, 2010-03-16

Link Handler

- Styrer trafikken gennem den indbyrdes dataforbindelse.
- Muliggør parallelle uafhængige datastrømme hen over forbindelsen.
- 'remote' start af programmer i 'den anden ende'.

Database struktur

- Navngivne tabeller (såkaldte 'array's), består af rækker.
- Hver række består af et antal navngivne attributter (felter), med individuelle egenskaber: Datatype, størrelse, placering i fysisk databasefil, mfl.

Database tilgang

- Standardiserede funktioner til læsning/skrivning.
- Anvender navne på arrays og attributter.
- Lager-residente attributter: Mulighed for direkte adressering.

Kørende dobbeltsystem

- Onlinesystemet håndterer den samlede drift,
- 'Signifikante' hændelser dubleres over 1 Mbit-forbindelsen,
- hvilket holder standby-systemet opdateret og klar til at tage over, hvis online-systemet fejler.

Simple databaseskrivninger

- Dubleres på standby-systemet af databasens standard-skrive-funktion.

Komplekse/direkte skrivninger

- Består af flere sammenhørende skrivninger.
- Og/eller direkte skrivninger i lagerresidente attributter.
- Disse dubleres ikke af databasefunktioner.
- Den operation/transaktion, som indebærer de sammenhørende skrivninger, dubleres og udføres ligeledes på standby-systemet.

Programmet DUPLO

- I online-systemet: Modtag transaktioner, udvalgte typer.
- Dupliker transaktion til DUPLO i standby-systemet.
- Videre-send transaktion til rette modtager i online-systemet.

Programmet DUPLO

- I standby-systemet: Modtag transaktioner fra DUPLO i online-systemet.
- Hvis buffer-ressource til videresendelse mangler: Gem i diskbaseret 'superbuffer'.
- Videresend til rette modtager i standby-systemet.

Opnåelse af konsistent standby-database.

- Initiel kopiering: Af online-systemets database til standby-systemet.
- Flere trin:
- Hovedpart – data, som tilgås via databasefunktioner.
- Stærkt dynamiske, lager-residente, tilgås direkte udenom databasefunktioner.

Hovedparten

- Udføres af dertil indrettet program.
- Herefter udføres alle 'simple' databaseskrivninger som dublerede skrivninger.

Stærkt dynamiske data

- Konsistent snapshot opnås ved at lade online-systemet 'holde pause' under kopieringen.
- Når 'pausen' er overstået, opfatter online-systemet sig som 'hot online'.
- Transaktioner involverende 'stærkt dynamiske data' duplikeres af DUPLO.

Holde pause

- Hver eneste coroutine kan modtage operationen 'hold pause'
- Skal svare 'jeg holder nu pause', når alle igangværende operationer er konsistent afsluttet.
- Kan stole på, at der ikke forekommer nye operationer, der skal behandles, før operationen 'pause forbi' modtages.

Pause Handler

- Sender operationen 'hold pause' til samtlige coroutiner og afventer 'jeg holder nu pause' i den 'rigtige rækkefølge'!
- Den 'rigtige rækkefølge' skal sikre, at ingen coroutine udsættes for rigtige operationer, mens de holder pause.

Den 'rigtige rækkefølge'

- Findes ved at sikre, at en coroutine først bliver tilsendt 'hold pause', når samtlige andre coroutiner, der kan sende operationer til den, har meddelt 'jeg holder nu pause'.

Tilmelding til Pause Handler

- Skal gøres af alle coroutiner under initialiseringen.
- Skal indeholde:
 - Input semafor adresse
 - Liste af semaforer (adresser), hvortil denne coroutine kan sende buffere.

Håndtering af pause-tilmelding

- Pause Handler udfører topologisk sortering for at beregne den 'rigtige rækkefølge'.

Standby-systemstart

- Situation: Online-system kørende som enkeltsystem, håndterer den samlede drift.
- Standby-system bootes.
- Loader monitor
- Starter Link Handler, som etablerer forbindelse til online LH
- Starter DUPLO, som afventer trafik fra online DUPLO
- Starter 'remote'-aktivitet i online-system, som udfører:
 - 1) Initiel databasekopiering af 'forholdsvis konstante' data.
 - 2) Holde pause
 - 3) Initiel databasekopiering af 'stærkt dynamiske data'.
 - 4) Pause forbi, nu 'hot online', DUPLO sender 'signifikante hændelser' til standby-systemet.
- Herefter fortsætter standby-systemet sin opstart, DUPLO opsamler 'signifikante hændelser' fra online-systemet i sine 'superbuffere' indtil standby-systemet er i stand til at behandle dem.

Erfaringer med Pause #1

- Alle coroutiner skal lære tilmelding, samt operationerne 'hold pause' og 'pause forbi'
- Ret omfattende indgreb, kræver indsats af samtlige udviklere (oveni alle deres øvrige opgaver).

Erfaringer med Pause #2

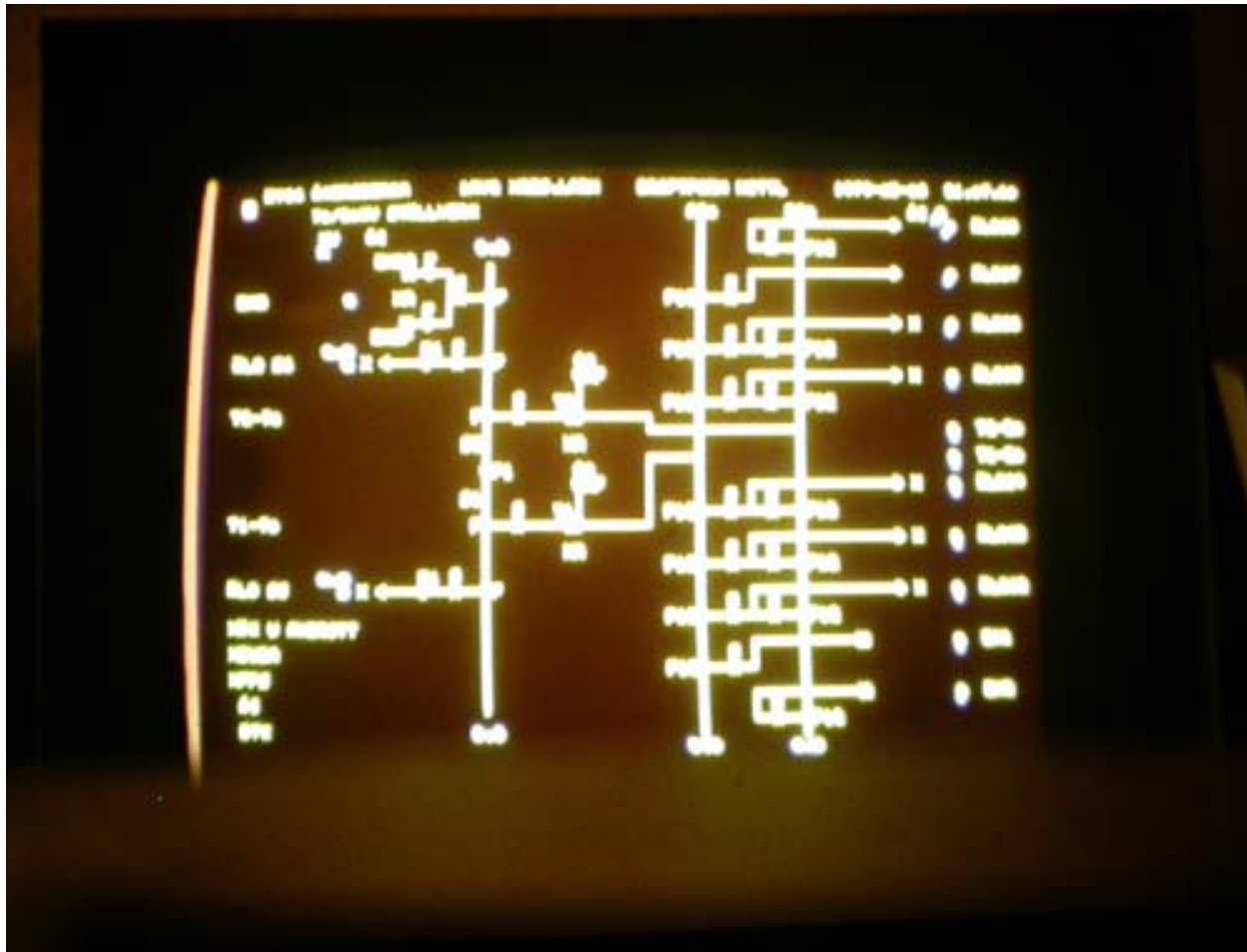
- Topologisk sortering i første omgang umulig pga. cykler.
- Måtte brydes ved bevidst at lyve ved visse coroutiners tilmelding!
- Viden om den 'rigtige rækkefølge' spredt i alle coroutiners initialisering – reelt hensigtsmæssigt?

Billedgalleri



Jesper Naur, 2010-03-16

Billedgalleri



Jesper Naur, 2010-03-16

Billedgalleri



Jesper Naur, 2010-03-16

Billedgalleri



Jesper Naur, 2010-03-16

Har I nogen spørgsmål?

SLUT

Tak fordi I kom.