Design for Systemadministration

Lars Fischer
CTO, NORDUnet

AaU, februar 2006

Før pause

- Baggrund & mål for design
- Komponenter og principper
- Købe eller udvikle
- Standardisering
- Instrumentering
- Vedligehold
- Automatisering

Efter pause

- Skalerbarhed
- Tuning
- Reedundans
- Virtualisering
- Et lille eksempel
- Håndtering af nedbrud

Design?

- Systemer opstår ikke bare
  - ...de designes
- Systemdesign, systemplanlægning
  - Det er her systemernes skæbne afgøres!
  - Det er her der er brug for erfaring og teknisk kompetence
  - Det er her der er brug for ingeniører!
- Samspil mellem design og organisation
- Design med drift for øje - og det kræver drifts-folk i design-fasen

Mål for design

- Sikkerhed
- Tilgængelighed
- Skalerbarhed
- Cost-of-Ownership
  - Omkostninger ved drift
  - Omkostninger ved vedligehold, opdrageringer, etc.
  - Omkostning ved uddannelse
- Integration med andre systemer

Det totale system

- Bevar overblikket – fokuser på det totale system
  - Der er mange komponenter – standardiser hvor det er muligt
  - Ingen kan overskue 100vis af enkelt-systemer
  - Skab ensartethed
- Brug sund fornuft
  - Beslutninger har langtrækkende konsekvenser
  - Ingen hurtige løsningen, ingen hovsa-beslutninger
- Behov for et komplet, gennemtænkt design
  - For hvert enkelt system
  - For systemernes samspil

Design-komponenter

- Det der skal drives
  - En platform (hardware, OS, middleware, netværk)
  - Applikationer
  - Adgangs-systemer (AAI, grænserflader, web-systemer, ...)
- Værktøjskasse til drifts-støtte
  - Overvågning
  - Instrumentering
  - System adgang (Console management)
  - Bruger styring
  - Resource styring

Platform (I)

- Servere
  - Hvilken slags hardware bruges?
  - Hvor mange slags hardware bruges?
  - Dyre server-systemer el. billig standard-hardware (google-modellen?)
  - Clustering?
- OS
  - Kan vi nøjes med ét?
- Storage-enheder
  - SAN?
  - Lokale RAIDs?

Platform (II)

- Netværk
  - Det kan være mange netværk i et system
  - Hver maskine kan være på mange net
  - Design af netværk skal være tænkt ind fra starten
- Database-systemer
  - Størelse – Oracle el. MySQL?
  - Kan kræve masser af specialist-viden
- Backup-systemer
  - Stor interaktion med alle andre systemer
  - Høj grad af kompleksitet

Applikationer

- Antal
  - Et generelt system til mange slags brug
  - Et specialiseret system med én applikation
- Standard-applikationer
  - Oracle Financials, SAS, SAP, osv
  - Tilhørende værktøjer til overvågning og drift
  - Kræver special-viden, uddannelse, mm
- Lokale applikationer
  - Kræver kendskab til lokale traditioner
  - Kræver adgang til lokale udviklere

Principper

- Hvordan sættes komponenterne sammen?
- Enkelhed
  - Bevar overskueligheden
  - Skab forudsigelighed
- Opdeling af systemer
  - Undgå, at en fejl skaber domino-effekt
  - En funktion, en server
- Fail-over
  - så ting kan gå i stykker uden at brugeren påvirkes
- Gør vedligehold nemt

Beslutninger

- At designe med drift for øje kræver rigtig mange beslutninger
  - ... Og de har langtrækkende virkning
  - Kræver erfaring
  - Kræver indsigt og kompetence i rigtig mange systemer
  - Kræver at vi respekterer mange kompentecer
  - Går nemmere når vi har nogen simple principper at styre efter
- Tag og fasthold beslutninger
  - Det er afgørende at vi tager beslutninger, for eller tager omstændighederne dem for os
  - Det er vigtigt, at vi holder os til de beslutninger vi tager – det er så nemt at lade sig ”inspirere” til noget slemt rod

Standard-systemer el egen udvikling?

- Ofte et helt kritisk valg
  - Hvilken type organisation?
  - Hvilken type medarbejdere?
  - Bredde at applikationer, systemer?
  - Bredde af kundegruppe / brugergruppe?
  - Hvor hurtigt forandre applikationer og systemer sig?
  - Hvor ofte laves ting om i organisationen?
- Gælder alle komponenter i design, både applikationer og drift-støtte
- Hoveregel: hav altid en rigtig god grund til at kode selv, og hav altid et stramt design

Standard systemer

- Gør det nemmere at skifte komponenter ud.
- Gør det nemmere at lave om på, hvordan organisationen arbejder
- Er ofte omkostningstungt
- Er ofte ikke specielt fleksibelt
- Gør det nødvendigt at leve med kompromisser – passer ofte ikke helt til ens design
- Kan være tunge at installere, især One-Size-Fits-All systemerne
- Kan tilpasses, men det kan være mere besværligt end at udvikle selv

Egen udvikling

- Fleksibelt, kan laves præcis til ens behov
- Gør det muligt at implementere præcis det design man ønsker
- Udvikling af systemet bliver nogen gange et mål i sig selv
- Gør det tit meget arbejdeskrævende at skifte systemer ud
- Kræver personale, der kan og vil kode
- Giver alle de problemer med dokumentation og overlevering, som en udviklingsafdeling har

Er der en mellemvej?

- Masser af open source systemer
- Systemer med ”open points”, dvs. systemer der kan tilpasses lokale behov
  - Kræver stadig lokale udvikler kompetence
  - ”Tilpasningerne” har det med at vokse
- For nogen organisationer er det muligt at bruge open source systemer til at finde en gylden middelvej mellem egen udvikling og standard systemer
- Kræver bevidste valg om at bruge denne vej

Standardisering

- Standardisering er en væsentlig del af design med drift for øje
  - Reducerer antallet af forskellige komponenter (fx servere), teknologier, software-systemer, arbejdsmetoder
  - Undgå én medarbejder pr. system eller teknologi syndromet
- Det er prisen værd
  - Ja, det koster penge at holde fast i den samme server
  - Ja, det kan være besværligt altid at implementere med det samme OS
  - Ja, det kan være tungt at bruge de samme support systemer hver gang
  - ... Men når krisen er der, er det det hele værd

Overhold standarderne

- Få alle support-systemerne på plads fra starten
  - Consol adgang
  - Overvågnings systemer
  - Log rotation
  - ...
- Ha´ tjek-lister for systemer som skal i drift
- Stil krav til systemer der kommer ind fra andre
- Et ”udvikler-system” er ikke klar til drift
- Et design er kun noget værd, hvis det bruges
- Det er for sent at gøre noget ved det når krisen er opstået

Instrumentering

- For at kunne finde fejl, skal man vide hvad der afviger fra normalen
  - Trivielt? Måske, men alt for ofte er informationen der ikke
- Normalen skal være beskrevet
  - Løbende registrering
  - Logs
  - Grafer (MRTG kan tegne meget andet end bitrater)
- Instrumentering skal kunne styres
  - Ekstra information i krise-situationer

Hvad skal instrumenteres?

- Alle afvigelser fra normalen
  - Log filer af alle væsentlige begivenheder
  - Med tid, tilstand, osv.
- Alle regelmæssige hændelser
- System-tilstand i faste intervaller
- Relevante performance-tal (tællere for transaktioner, trafik, mm)
- Instrumeterings-niveau skal kunne varieres
  - Hæves i perioder med mistanke om problemer

Indbygget instrumentering

- Instrumentering skal være tænkt ind fra begyndelsen
  - Det er ikke noget der skal sættes på bagefter
- Hovsa-instrumentering ender med at blive logfiler over det det var muligt at logge, ikke det der er væsentligt.
- Instrumentering der er designet ind giver mulighed for at se, hvad der foregår inde i systemerne, og dermed forstå, hvorfor systemerne gør det de gør
- Instrumentering der er designet ind giver mulighed for at følge udvikling i nøgle-variable

Ekstern instrumentering

- Til performance brug
  - Regelmæssig registrering af respons tider
- Til overvågning
  - Regelmæssig regisrering af korrekt opførsel
  - Gør det muligt at observere systemer fra ”bruger synspunkt”
- Skal gøres systematisk
  - Ét system til al ekstern instrumentering
  - En fælles metode, som alle kender og forstå
  - Et kendt sted, hvor data samles og kan ses

Planlagt nedetid

- I gamle dage havde store installationer ”planlagt nedetid” hver nat, eller en gang om ugen
- Det er ikke længere acceptabelt
  - Men behovet findes endnu
- Alle systemer har brug for vedligehold
  - Der skal skiftes diske
  - Der skal flyttes til nye racks
  - Security-patches!
  - Der kommer nye versioner af al software

Design til vedligehold

- Design for at undgå nedetid
  - Brugersigte
- Design for at gøre vedligehold nemmere
  - Fokus på personale-omkostning
- Parametre
  - Hvor meget nedetid er acceptabelt?
  - Hvor lang varsel?
  - Hvor meget natarbejde?
  - Hvor meget kompliceret planlægning kan vi leve med

Måder at lette vedligehold

- Redundante systemer
  - Drift på den ene kopi mens den anden opgraderes
  - Tvungen fail-over
  - Mulighed for fail-back ved fejl i nye versioner
- Modularitet
  - Ensartethed af software og hardware
  - Separer systemer og komponenter
  - Hav kontrol over hver enkelt komponent – pas på defaults
- Omkostninger
  - Hvar er billigst – planlægning, extra hardware, eller ekstra arbejde med vedligehold

Automatisering

- Automatisering er afgørende for al effektiv systemadministration
  - Vi har ikke råd til repetition
  - Vi har laver for mange fejl ved manuelt arbejde
- Systemer skal være designet til at kunne automatiseres
  - Lette at styre fra script sprog
  - Indbyggede muligheder for scripting
  - Systemer og komponenter bygger med tanke på, at de ikke køres at et menneske

Automatiserings teknikker

- Lær script sprog – og brug dem
  - En god sysadm gør aldrig det samme tre gange
  - Behersk flere script sprog (sh, perl, python)
  - Standardiser på de sprog der bruges i organsationen
- Tænk altid i automatisering
  - Adskil det der kræver beslutninger fra det der blot skal gøres
  - Et godt program gør een ting, og gør det godt – tænk på UNIX maner
- Gør automatisering til en del af hverdagen, ikke til et projekt

PAUSE

Skalerbarhed

- (Næsten) alle systemer vokser
- Skalering skal være tænkt ind fra starten
  - Det er OK med simple løsninger, men de skal være kendte
  - Der skal være en plan for, hvordan systemet opgraderes når de valgte løsninger ikke mere er tiltrækkelige
- Det er umuligt at designe mere effektive løsninger midt i et skalerings-sammenbrud
- Flaskehalse skal være kendt
  - Så vi ved, hvor skalerings-problemer kan opstå
  - Så vi kan overvåge og ikke blive overraskede

Tuning

- Flere mål
  - At sikre bedst ydelse for brugere
  - At vride mere ud af hardware
  - At sikre jævn drift
- Valg: mange problemer kan løses billigt med mere hardware
  - ... Men intet hardware kan få dårlig software til at køre godt
- Afgørende at vide, hvad performance problemer skyldes
  - ...og så vælge hvordan vi løser dem

Eksempel: Et mail system

- Et klassisk UNIX mail system
- Størelse af mailboxe
  - Klassisk UNIX mbox har een fil per mailbox, som læses sekventielt
  - Hvad sker der, når brugere har 1000-vis af store mails?
- Login hastighed
  - Et klassisk UNIX system slår brugere ved linær søgning i en password fil
  - Hvad sker der når man har 1000-vis af brugere?
  - Hvad sker der, når brugere tjekker mail hvert minut?

Redudans

- En afgørende parameter i ethvert design med drift for øje
  - Redundans er en voldsom omkostning, men osse en voldsom lettelse i driften
  - Redundans beskytter os i tilfælde af nedbrud
  - Redundans kan lette det daglige arbejde
- Redundans kræver kompentece
  - Kræver omhyggeligt design for at virke efter hensigten
  - Kræver tests
  - Clustere er svære at bygge og administrere
  - Redundans øger kompleksiteten voldsomt

Alt kan gøres redundant

- Strøm
- Storage
- Servere
- Netværk
- Dele af det enkelte system
- Komplette systemer
- Sites (plane crash redundancy?)
- Service-funktioner

Redundant strøm

- PSU
- Nettavle, strøminstallationer
- Kabeltræk – adskilte kabler til racks
- 2 x bystrøm
- UPS
- 2xUPX
- Diesel
- 2 x diesel
- Leverance af ekstra diesel?

Redundante Servere

- Flere separate med samme funktion
  - Eksempel: DNS
  - Serverne er uafhængige, kan være fysisk adskildt
- Flere samlet med samme funktion
  - Eksempel: web frontend servere, cluster regne servere
  - Serverene er uafhænige, men samlet
  - Mulighed for loadbalancing & skalering
  - Mulighed for availability management
- Cluster
  - To eller flere maskiner sammenbygget til een
  - Mange teknikker
  - Typisk kun én maskine på applikations nivea
  - OS support, fx. delt IP nr.

Redundans af servicefunktioner

- Leverandører
  - Hvad når din internet-leverandør går fallit?
  - Hvad med kritiske servere?
- Lagre
  - Hvad når lageret med reservedele brænder ned?
- Kontorbygninger
  - Brand? Det er ikke nok med et redundant EDB-rum i den anden ende af byen
- Personale
  - Funktions-overlap, undgå sårbarhed
  - Når en flyver rammer dit højhus...

Virtualisering

- Stammer fra mainframe perioden
  - Specielt IBM: MVS
  - Giv hver bruger en illusion om at have sin egen maskine til rådighed
- Grund idé: udnyt et stykke hardware som om det var man mange fysiske maskiner
  - Tillad applikationer at køre på sin ”egen” maskine
  - Tillad forskellige OS varianter
  - Tillad eksperimenter

Virtuelle Servere

- To niveauer
  - Host : den fysiske maskine
    - Host OS: kører den fysiske hardware
    - Virtual machine software: skaber virtuelle maskiner
  - Client: den virtuelle server
    - Client OS: normalt OS, der kører oven på den virtuelle hardware
    - Applikation: kører på client OS, ved intet om virtualisering
- Virtual machine software
  - Virtualisering af komplet hardware (inkl. bios, CPU, medier, netkort, netværksadgang, subnets)
  - Client OS installeres som normalt, tror det lever på alm. Hardware
  - Moderne PC hardware er rigeligt kraftig til virtualisering

Fordele ved Virtuelle Servere

- Mindre hardware
  - Mindre hardware at holde ved lige
  - Lavere omkostninger
  - Udnytte den fulde kapacitet i moderne hardware
  - Gør det muligt at have mere avanceret hardware
- Fastholde én applikation, én maskine
  - Undgå påvirkninger mellem applikationer

Ulemper ved virtuelle maskiner

- Endnu et niveau af komplikationer
- En virtuel maskiner ér kun virtuel
  - Der er ikke rådighed over den komplette hardware
  - Der er ikke direkte adgang til hardware
- Det kan blive rigtig spændende når Host OS crasher
- Det er en (afgørende) design valg

Virtual server migration

- Grundlæggende ide: i stedet for at flytte applikationer mellem maskiner, så flyttes komplette virtuelle maskiner mellem host-maskiner
  - En virtuel maskine har typisk langt mindre binding til sin underliggende platform, end en applikation har
  - Gør det muligt at flytte applikationer mellem sites
  - Gør det muligt at flytte applikationer mellem fysiske maskiner for at tage hensyn til load
- Kan endda gøres for kørende, virtuelle maskiner

Virtual server cloning

- Cloning tillader at der skabes en kopi af en virtuel server, ude at der skal installeres igen
- Gør det muligt at lave eksperimenter
  - F.eks. til at afprøve en slagplan for en opgradering eller anden større operation
- Ved omhyggeligt design kan det være muligt at opgradere en clon og derefter svinge driften over, uden nedetid

Virtualisering af storage

- Storage kan virtualiseres på mange niveauer
  - Diske
  - Filsystemer
  - Tape
- SAN er det klassiske eksempel
- Fordel: frigør den enkelte maskine fra at have de kompleksiteter storage giver

Eksempel: console adgang (I)

- System console
  - En system admnistrator har behov for at kunne komme til sine systemer
  - ...osse når der er krise, netværksnedbrud, etc.
  - Sikkerhed når der udføres kritiske opgaver
- Behov:
  - At kunne få adgang til flest muligt (alle) systemer
  - At ha' et system der giver adgang til alle systemer
  - At ha' remote adgang
  - Beskyttelse mod uvedkommende adgang
  - At ha' færrest muligt forudsætninger for adgang
  - Normal systemadgang

Console adgang (II)

- Løsning: Fysisk skærm eller terminal på alle systemer
- Pro:
  - Enkelt
  - Ingen forudsætninger (virker altid)
  - Sikkert
  - Giver normal systemadgang
- Cons:
  - Ufleksibelt, langsomt
  - Besværligt i det daglige arbejde
  - Kræver fysiske adgang
  - Ingen remote arbejde

Console adgang (III)

- Løsning netværks logon (ssh etc)
- Pro
  - Nemt, enkelt, billigt
  - Fleksibelt, hurtigt
  - Kræver ikke fysisk adgang
  - Kan gøres sikkert (evt. Jumphost)
  - Remote arbejde muligt (virker alle steder fra)
- Cons
  - Forudsætter normalt drift (slås ud i krise)
  - Kan have begrænset systemadgang

Console adgang (IV)

- Løsning: Virtuelle desktops (VNC)
- Pro
  - Fleksibelt, billigt
  - Kræver ikke fysisk adgang
  - Kan gøres sikkert
  - Remote arbejde muligt (virker alle steder fra)
  - Giver normal systemadgang
- Cons
  - Kan være komplekst
  - Kan være langsomt og tungt at arbejde med
  - Forudsætter normalt drift (slås ud i krise)

Console adgang (V)

- Løsning: Console servers (tty el. PC)
  - Systemer der samler fysisk adgang til hver enkelt server i ét system
- Pro
  - Kræver ikke fysisk adgang
  - Kan gøres nemt og hurtigt at arbejde med
  - Sikkert
  - Giver normal systemadgang
  - Kan kombineres med logning af consol output
  - Virker (næsten) altid (osse i en krise)
- Cons
  - Kan være komplekst, kræver forberedelse, dyrt
  - Remote arbejde ikke muligt

Console adgang (VI)

- Løsning: Netværskbaseret console servers
  - Virtualisering af console, via console server eller i den enkelte host
- Pro
  - Kræver ikke fysisk adgang
  - Kan gøres nemt og hurtigt at arbejde med
  - Sikkert
  - Giver normal systemadgang
  - Kan kombineres med logning af consol output
  - Virker (næsten) altid (osse i en krise) (især virtualisering i console server)
  - Remote arbejde muligt
- Cons
  - Kan være komplekst, kræver forberedelse, dyrt

Console adgang (VII)

- Text-only console adgang er enkelt
- GUI skaber komplikationer
  - Dyrt at virtualisere
  - Dyrt at kable og installere til konsole servere
  - Tungere at bruge remote
  - Sværere at sikre adgang i krise-situationer
  - Nemmere at logge
- Design-valg: er det muligt for den aktuelle installation at skabe en situation, hvor drift er (reelt) mulig med text-only?
  - Og hvor personalet ved hvordan man gør?

Håndtering af nedbrug

- Det er ved nedbrud, vores design skal stå sin prøve
  - Hvem er ramt
  - Hvordan er de ramt
  - Hvad koster det
- Hvad nu?
  - Har vi den information vi skal bruge?
  - Har vi de værktøjer vi skal bruge
  - Har vi adgang?
  - Ved vi, hvad vi skal gøre
  - Kanvi overskue det?

Design med nedbrud for øje

- Det går galt – uanset hvad vi gør
- Tænk hele tiden over, hvordan vi kommer på ret køl
- Tænk i scenarier – forudse fejl, forudse hvordan det er at stå med dem
  - Hvor lang tid tager det at indlæse ting fra backup?
- Planlæg hvad der skal gøres når fejlen opstår

Tak for denne gang

<lars@nordu.net>