DKuug nr. 168 September 2012

NYT

DKUUG - Unix Brugere, system administratorer
Dansk Forum for Åbne Systemer
Mødested for IT-specialister og IT-interesserede.

Indholdsfortegnelse + Arrangementer, foredrag, events - side 3
Hadoop - Google's *big data lagring og analyse* - side 4

DKUUG

Redaktionelle strøtanker

I dette nummer af NYT har vi fået mange artikler om Open Source projekter, fra Hadoop over Android, Node.JS Python til redning af slettede filer med Photorec - og så endda lidt til.

Kenneth Geisshirt har skrevet om Hadoop, Google-software til at gennempløje store mængder data - et produkt eller projekt, som han holdt foredrag om 21.august (billedet på forsiden er fra foredragsaftenen). Som man kan se af artiklen, er der tre applikationer, som bruger Hadoop som base, nemlig Pig, Hive og Hbase til opbygning af parallelliserede databaser og data-mining applikationer, og vi håber at der er medlemmer, som har lyst og mod på at skrive og læse om disse værktøjer. Det er teknologiens frontlinie og at både studerende og forskere kunne have godt af hjernemassage indenfor de felter.

Mobil- eller smart-phones er et andet af de felter, hvor teknologien tager stormskridt fremad i disse år. Det skyldes selvfølgelig ikke mindst radio-teknologien, som har gjort hurtig datatransmission via G3 og G4 muligt. Der har dog vist sig grænser, hvis en storbybefolkning vil kommunikere samtidigt, egentlig ikke noget, der kan undre en generation af garvede system-administratorer, som har oplevet det ene hype efter det andet.

Vores skeptiske redaktør tog fat på et rundspørge til begejstrede iPhone fans. Måske var det en forkert aldersgruppe, han fik fat i, for den ene forklarede, at den lå i skuffen takket være at den var lidt unyttig, den anden erklærede, at iPhone havde dårlig netværksteknologi. Men der er jo heldigvis andre smartphones, og vi har her på redaktionen set adskillige mærker og modeller. Kommende numre vil bringe opdateringer af emnet set ud fra en bruger-administrativ synsvinkel.

Android er som bekendt et alternativ til iPhone. Google har stået for udviklingen af det - bortset fra at kernen er en Linux-kerne. Android er Open Source, og hvis du har lyst til at lave din egen version af Android, er det bare at tage fat - begynd med at læse på side 15, hvor Jacob Nordfalk har beskrevet hvordan man kompilerer Android kernen fra bunden.

Herunder kommer han ind på, hvordan man henter kildetekst til projekter, der styrer versionsopdatering med /Git/, programmet, som blev til da Linus Torvalds syntes at Linux kernen var blevet for stor til almindelig håndtering og efter at Bitkeeper (proprietært system) havde vist sig at give vanskeligheder.

På OSD2012 var der et foredrag om kreative anvendelser af Git - blandt andet til at holde styr på kalenderen og lignende. Ideen er næsten genial. Redaktionen håber, at der vil komme flere artikler, som kommer ind på design og filosofi bag Git - sammenlignet med SVN, CVS og det tidlige SCCS program. I den sammenhæng skal også Microsofts pengeskabs-system, Source-Safe og Vault og et system som *Clear-Case,* som skjuler information om disk og filsystem for brugeren, hvilket gør det uegnet til udvikling af hardware nære netværks-programmer. Det gælder om, at et arkiv- og versions-system ikke vanskeliggør opgaver for ikke-priviligerede medarbejdere.

Når vi får tid - og hvis der er interesse - vil undertegnede invitere til til et bladmøde, hvor vi samler kendere af de forskellige systemer og derefter skriver og redigerer tekster om de grundlæggende ideer og forskelle i versions-styringssystemer. Forhåbentlig kan der så komme en god artikel ud af det.

Prøv at tage et kig på det lille klip fra en rapport om Free Open Source Software (FOSS) i US Department of Defense (DoD). Vi tager en lille læsepause ☺

Det er karakteristisk for alle os, der arbejder med Open Source og programmering, system-administration og maskin-arkitektur, at vi må leve med, at vores fag er næsten totalt uforståeligt for andre end fagfolk. Ikke som i gamle dage. En sadelmager lavede sadler og seletøj, og det vidste man hvordan skulle se ud - eller rettere, teknologien udvikledes så langsomt at man kunne nå at følge med. Ikke desto mindre har forskerne idag fundet ud af, at oldtidens metallurger måske alligevel havde mere styr på, hvordan materialerne skulle bruges. Fx. har det ved restaurering af templerne på Akropolis i Athen vist sig, at oldtidens teknikere og arkitekter havde mere styr på materialernes holdbarhed, end nutidens teknikere. Vi kan stadig lære af fortiden. Vigtig viden kan gå tabt. Vigtige sammenhæng går til stadighed tabt, og det ved vi, der har arbejdet som system-administratorer er et stort problem.

Den viden, som findes i en IT afdeling, bedømmes på en anden måde af en administrativ leder end af en teknisk specialist, og nogen tror fejlagtigt, at enhver position i en virksomhed kan erstattes af en anden arbejdskraft. Allerede Frederick Brooks vidste, hvor forkert denne ledelsesfilosofi er, og han skrev derfor bogen "Den Mytiske Mande-Måned", "The Mythical Man Month", hvor han harcellerer over at man regner med kreativ arbejdskraft som om der skulle graves grøfter. Bogens titel henviser til de gammeldags regneøvelser: "Når én mand er to dage om at grave en grøft, hvor længe er så to mand om at grave den?"

Indenfor IT, systemadministration og programmering, holder den slags regnestykker ikke, og hvad mere er, produkternes kvalitet kan heller ikke bedømmes ud fra hvad de koster.

Det er derfor, at det er så vigtigt at forklare, at Open Source ikke kun er fri software, men også vidensdeling. Alle, der gider, kan få et indblik i, hvordan Linux kernen fungerer - lære programmeringssprog, se hvordan man laver en applikation, forstå begrænsninger og muligheder.

Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense

2.2.1 FOSS Software is Vital to DoD Information Security

The survey identified 44 examples where organizations involved in DoD Security use FOSS software. The FOSS communities contribute to DoD security in two ways. Firstly, it has produced infrastructure software such as OpenBSD with low rates of software failure combined with early and rapid closure of security holes, which makes such systems useful as the security linchpins in broader security strategies. Secondly, the FOSS communities have had a long-term fascination with developing more and more sophisticated applications for identifying and analyzing security holes in networks and computers, resulting in FOSS products such as SARA and Snort that are invaluable to in-depth analyses of security risks.

Klip fra en 2003 rapport om anvendelse af Open Source i US Forsvarsministerium (DoD)

Forsidebillede: Foredrag i Symbion om Hadoop (Jon Bendtsen)

DKUUG-Nyt er medlemsblad for DKUUG, foreningen for Åbne Systemer og Internet

Udgiver:
DKUUG
Fruebjergvej 3
2100 København Ø
Tlf. 39 17 99 44
email: dkuugnyt@dkuug.dk
Redaktion:
Donald Axel (ansvarshavende)
og mange flere
Forsidefoto:
Foredrag om Hadoop 21. august 2012
Design og layout:
DKUUGs PR-gruppe
Annoncer:
pr@dkuug.dk
Tryk:
Digital-trykkeriet i Aarhus
Oplag:
500 eksemplarer

Artikler og inlæg i DKUUG-Nyt er ikke nødvendigvis i overensstemmelse med redaktionens eller DKUUGs bestyrelses synspunkter

Medlem af Dansk Fagpresse
DKUUG-Nyt
ISSN-1395-1440

Vores møder og **foredrag** holdes - med mindre andet utrykkeligt angives - på vores adresse:

DKUUG
SYMBION
Fruebjergvej 3
2100 København Ø

Hvis man kommer lidt før, er der tid til en snak på kontoret. DKUUG bor i en virksomheds"farm", Symbion, hvor der er åbne døre indtil kl.18. Efter den tid har vi på foredragsaftener en vagt ved døren.

INDHOLD:

BREV FRA REDAKTØREN 2
HADOOP VÆRKTØJSKASSE TIL BIG DATA 4
LINUX SOM INTERAKTIV VÆGDEKORATION 10
AAARGH! NÅR DATA OVERSKRIVES 13
OVERSÆTTE ANDROID FRA BUNDEN 15
NODE.JS FOR SYSTEM ADMINISTRATORER 20
IPv6 PRIVATE ADRESSER 21
GLEM HAMMEREN! GØR DET RIGTIGE 22

OSD 2013 lørdag-søndag d. 9-10 marts 2013. Der vil muligvis blive kurser dagen før.

thecamp.dk har lagt en lang række video'er på **video.thecamp.dk**

Foredrag i Symbion:

Torsdag d. 27. september kl. 18: Aflyst: ~~Apache Open Office~~ layout.

Torsdag d. 27. september kl. 18: Quick and dirty debugging for system debugging for system administrators. Bryan Østergaard.

En kort gennemgang af en række kendte og mindre kendte værktøjer til debugging og overvågning af resource forbrug generelt. Foredraget er rettet mod system administratorer, og fokuserer på en række tricks til lynhurtigt at finde og løse problemer, hvis årsag er svær at finde.

Ud over at kigge på standard værktøjer som for eksempel strace og iostat vil der også blive demonstreret en række 'hemmelige' værktøjer som følger med alle Linux systemer og som kan hjælpe dig i situationer som er næsten umulige at løse med mere kendte værktøjer.

Tirsdag d. 2. oktober kl. 19: IPv6: multihoming og mobile-devices - akilleshæl? Dax.

Torsdag d. 25. oktober kl. 19: C++11 highlights. Bryan Østergaard.

C++11 beskrives af mange som *et nyt sprog,* fordi mange af de nye features grundlæggende ændrer på hvordan man skriver gode C++ programmer. Bryan gennemgår kort de vigtigste nye features og giver et indblik i nogle af de problemer der er løst i forhold til den gamle C++ standard.

Kendskab til C++ er ikke et krav, men en smule programmeringserfaring er en forudsætning.

Torsdag d. 1. november kl. 19: C++ templates for begyndere. Bryan Østergaard.

Introduktion til templates for begyndende C++ udviklere. C++ standard biblioteket bygger i høj grad på templates så det er ofte svært at komme uden om brugen af templates i C++ kode.

Men templates kan også være en god abstraktionsform der kan føre til programmer der er nemmere at forstå og vedligeholde.

Der kræves ingen forudgående kendskab til C++ programmering.

Torsdag d. 8. november kl. 19: C++ policy based design. (Bryan)

Policy based design er en af grundpillerne i "modern C++" og bliver ofte brugt sammen med template metaprogramming. Policy based design kan dog sagtens stå alene som et værdifuldt værktøj i C++.

I foredraget gennemgåes hvad der der menes med policy based design og der gives eksempler på hvordan det kan udnyttes til at skrive bedre C++ kode. Kom og få dine kompetencer plejet - hold et foredrag om det, der interesserer dig (og os).

Torsdag d. 15. november kl. 19: C++11 metaprogramming. (Bryan)

Såkaldt template metaprogramming (program kode der bliver evalueret ved compile tid) er en af grundpillerne i hvad der bliver omtalt som "modern C++". Men det er også et område som mange har betragtet som et grimt og uoverskueligt hack. Med C++11 er mange af problemerne ved template metaprogramming løst og der er kommet mange nye muligheder til. Bryan vil gennemgå de store nyheder i forhold til template metaprogramming og vise nogle interessante nye muligheder.

Der er bred support blandt C++ compilere for de features der beskrives i foredraget.

Fortsættes side 23.

Hadoop – værktøjskasse til Big Data

af Kenneth Geisshirt

Baggrund

Vores datamængder vokser eksponentielt. Det er ikke atypisk at have flere terabyte data liggende. Supermarkeder som *Bilka* har enorme data warehouses hvor kundernes transaktioner er gemt. Teleselskaberne er forpligtet til at gemme informationer om telefonsamtaler, og det er ikke svært at forestille hvilke mængder, de må arbejde med.

Data er først interessant når det analyseres og giver ny viden og indsigt. I februar kunne man læse at amerikanske *Target* er i stand til at forudsige om deres kunder er gravide – når de selv opdager det[1]. Big Data og Data Science er begreber, som bruges uformelt om lagring og analyse af disse store datamængder.

Apache Hadoop er en platform at at hjælpe med at lagre og analyse data. Der er tale om et større open source projekt, men mange kommercielle spillere deltager lystigt i projektet ved at tilbyde enten support eller tillægsprodukter.

I denne artikel vil jeg sætte fokus på det rene open source projekt. Emnet er alt for stort til en omfattende gennemgang, så der er mere tale om en introduktion som forhåbentlig kan inspirere til at kaste sig ud i at lege med Hadoop.

I dag er Hadoop meget udbredt, og det anslås at omkring 30 pct. af CPU-tiden hos Amazons cloud-løsning bruges af kunders Hadoop-analyser. Oprindelig stammer Hadoop fra Yahoo!, men styres nu – juridisk set – af Apache Software Foundation. Projektet er stadig ungt, idet det blev påbegyndt i 2009 af Douglas Read Cutting, mens han arbejdede for Yahoo!. I dag arbejder han for Cloudera, som en af de store kommercielle spillere i Big Data og Hadoop-markedet.

Komponenter

Hadoop er ikke bygget som et monolitisk system. Det består af en række byggestene eller komponenter. Det er muligt at skifte enkelte komponenter ud med andre. Men i denne artikel vil jeg se bort fra alternativer og fokusere på de komponenter som følger med. I en Hadoop-klynge vil ikke alle maskiner have alle komponenter kørende, men det er ofte lettere bare at installere det hele. Den nuværende version af Hadoop (version 1.0.x) bygger på at der er en central *master* med en række *slaves*.

En klynge består ofte af hyldevarer, dvs. hardware du kan købe hos enhver leverandør. Du behøver ikke hardware med stor kapacitet. Filosofien i Hadoop er, at det er bedre at købe mange små og billige maskiner end få dyre og kraftige maskiner. På den måde minder filosofien meget om de tidligere tiders filosofi omkring bruges af Linux og hyldevarer i *High Performance Computing*. I dag er det nok sjældent at finde en masse desktop-PC'ere stående på Ikea-reoler og agere supercomputer.

Inden vi kommer for godt i gang med at diskutere de forskellige komponenter, er det en god idé at forstå hvad Hadoop kan. Der er to centrale funktioner i Hadoop. Den første funktion er et distribueret filsystem. Det betyder at data spredes over en lang række maskiner i klyngen – typisk slaverne. Filerne klippes op i blokke, og blokkene placeres på de forskellige server. Som default er en blok på 64 MB men det er muligt at bestemme blokstørrelsen. Eftersom filosofien bag Hadoop er at bruge billige hyldevare-PC'ere, er sandsynlighed for at en slave går ned meget stor. For at sikre sig mod nedbrud, gemmes blokkene mere end én gang – typisk tre gange. Du kan tænke på Hadoops distribuerede filsystem som et RAID hvor harddiskene sidder placeret i forskellige maskiner.

Den anden centrale funktion i Hadoop er at styre afviklingen af parallelliserede programmer. Det vil sige at Hadoop indeholder et køsystem som kan håndtere at en maskine går ned. Sker det, vil Hadoop genstarte jobbet – eller rettere en del af jobbet – og sikre at brugeren får analyseret data.

Der er primært fire komponenter i Hadoop.

- **NameNode** holder styr på filernes metadata. Eftersom filerne er brudt op i blokke, som er fordelt over en række maskiner, er det nødvendigt at have styr på hvor de enkelte blokke er placeret.
- **JobTracker** styrer brugernes job. Der er med andre ord tale om et køsystem. Et job brydes ned i en række mindre opgaver eller *tasks*.
- **TaskTracker** overvåger de enkelte tasks.
- **DataNode** står for at styre de distribuerede blokke.

De to komponenter **NameNode** og **DataNode** implementerer et distribueret filsystem. Med Hadoop følger HDFS (Hadoop Distributed File System) som er en implementering af sådan et filsystem. Det er muligt at implementere andre. Fordelen med HDFS er at det er godt afprøvet, men ulempen er at der ikke er tale om et filsystem, som kan mountes som et alm. filsystem. I en klynge vil master-maskinen være **NameNode** og slaverne vil køre **DataNode**. I en mindre klynge vil master-maskinen også være **DataNode** men har du en stor klynge, bruger masteren al CPU-tid på at servicere de mange **DataNode**s.

Komponenterne **JobTracker** og **TaskTracker** implementerer et fejltolerant køsystem.

Installation

At installerere Hadoop er ikke trivielt. Der er mange ting, som skal passe sammen, og rækkefølgen af kommandoer er ikke underordnet. I denne artikel vil jeg installere Hadoop på en lille klynge bestående af tre maskiner.

Alle tre maskiner kører som udgangspunkt Ubuntu Linux 12.04 (server edition). Grunden til dette er, at det er en distribution som er meget populær blandt cloud-udbydere. Det betyder at der ofte er images klar og du let kan klargøre en Linux-server. Du kan i princippet godt komme i gang med Hadoop ved at bruge en maskine, men det er ikke sjovt at se parallelliserede programmer køre på en maskine.

Min installationsvejledning bygger på en Michael Nolls glimerende opskrift[2].

1 http://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/

2 http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

Hadoop er skrevet i Java, og det er derfor nødvendigt at have et Java-miljø installeret på maskinerne. Oprindelig var Hadoops funktionalitet og stabilitet afhængig af hvilket Java-miljø, som var installeret. Faktisk var det sådan, at kun SUNs (nu Oracles) Java-implementation var god nok. Det anbefales derfor at du bruger Oracle Java. Udfordringen er blot, at Ubuntu Linux ikke har pakker for Oracle Java, da den valgte licens ikke er kompatibel med Ubuntu-licenspolitik. Heldigvis findes der flere løsninger på den udfordringen. Den lette er at benytte en række pakker fra webup8[3]. Det kræver at du på hver af de tre servere udfører følgende kommandoer.

```
add-apt-repository ppa:webupd8team/java
apt-get update
mkdir -p /usr/lib/mozilla/plugins
apt-get install oracle-jdk7-installer
#Kommandoer til installation af Java
```

Du er nu klar til at installere Hadoop. I skrivende stund er Ubuntu Linux håbløs bagud mht. versioner. Der findes en lille gruppe Hadoop-entusiaster, som pakker Hadoop til Ubuntu Linux[4]. Ved at bruge disse pakker får du installeret den nyeste stabile udgave af Hadoop. Det kræver blot tre kommandoer – igen på alle maskiner i din klynge.

```
apt-add-repository ppa:hadoop-ubuntu/stable
apt-get update
apt-get install hadoop pig hive hbase
#Kommandoer til installation af fire hadoop elementer
```

Kommandoerne ovenfor installerer mere end Hadoop. Applikationerne Pig, Hive og Hbase installeres også. Det er tre applikationer, som benytter Hadoop-platformen til at bygge parallelliserede databaser og analyseværktøjer. De tre applikationer er ikke diskuteret i denne artikel.

Med Hadoop installeret er det fristende at tro, at det er nok. Men det er det naturligvis ikke. For at sikre en adskillelse af systembruger fra Hadoop, er det en fordel at oprette en bruger specielt til Hadoop. Du kan gøre det med kommandoen.

```
adduser –ingroup hadoop hduser
#Oprettele af bruger til Hadoop
```

Brugeren `hduser` skal oprettes på alle maskiner i klyngen. Endvidere skal du tilføje `JAVA_HOME` til `.bashrc` og Hadoops placering til `PATH`. Det kan se ud som følger:

```
export HADOOP_HOME=/usr/lib/hadoop
export JAVA_HOME=/opt/java/64/jre1.7.0_05/
export PATH=$PATH:$HADOOP_HOME/bin:
#Environment opsætning
```

Hadoop-klyngen bruger netværket til at koordinere både de distribuerede filsystem og jobbene, er det nødvendigt at alle maskiner kan nås fra de andre maskiner. Det er vigtigt at maskinerne har fornuftige navne og ikke blot hedder `localhost.localdomain`. Du kan med fordel styre det fra en lokal nameserver eller rette filer som `/etc/hosts` til på passende vis. Endvidere er det vigtigt at brugeren `hduser` kan logge ind med SSH vha. nøgler (uden passphrase). En Hadoop-klynge vil typisk være placeret bag et firmaets firewall, og det er nok heller ikke urimeligt at antage at slaverne er forbundne gennem et privat netværk som kun de deler. Du kan med andre ord godt tage sikkerheden en smule mere afslappet end du normalt vil gøre.

Det distribuerede filsystem skal have plads til at gemme blokkene. Det sker på det lokale filsystem på alle maskinerne. Her kommer `hduser`-brugeren til sin ret, idet du kan sikre at kun denne bruger har mulighed for at få adgang til blokkene. Følgende kommandoer skal udføres på alle maskiner (master og slaver) for at gøre det lokale filsystem klar til Hadoop (har du andre ønsker om placering, kan du frit vælge det).

```
mkdir -p /app/hadoop/tmp
chown hduser.hadoop /app/hadoop/tmp

#Kommandoer for Hadoop katalog/directory
#udføres på alle maskiners filsystemer
```

Det er nu tid til at konfigurere Hadoop. I kataloget `/etc/hadoop/conf` finder du de vigtigste konfigurationsfiler. De er kort beskrevet nedenfor.

- `masters` angiver hvilke maskiner som er masteren i klyngen, mens `slaves` angiver hvilke slaver, der er i klyngen
- `core-site.xml` sætter op hvor det distribuerede filsystem skal placere blokke

3 http://www.webupd8.org/2012/01/install-oracle-java-jdk-7-in-ubuntu-via.html

4 https://launchpad.net/~hadoop-ubuntu/+archive/stable

```
<property>
  <name>hadoop.tmp.dir</name>
      <value>/app/hadoop/tmp</value>
</property>
```
Indhold af filen `core-site.xml`

- I `hdfs-site.xml` kan du angive hvor mange kopier, du ønsker blokkene skal replikeres i

```
<property>
  <name>dfs.replication</name>
  <value>3</value>
</property>
```
Indhold af filen `hdfs-site.xml`

- Med `mapred-site.xml` sætter op hvilken maskine, som styrer jobbene.

Endvidere skal du i `hadoop-env.sh` angive hvor Java-miljøet er placeret. Du skal bruge den samme værdi, som du sætte `JAVA_HOME` til tidligere.

Så er vi ved at være klar til at starte vores Hadoop-klynge op. Det er vigtig at være logget ind som `hduser` på masteren. Først skal du dog formatere det distribuerede filsystem. Det gør du med følgende kommando:

```
hadoop namenode -format
#Kommando til at formatere det distribuerede filsystem
```

Når filsystemet er formateret, er det tid til at starte klyngen op. De to følgende kommandoer starter alle processer op – både på masteren og slaverne.Når filsystemet er formateret, er det tid til at starte klyngen op. De to følgende kommandoer starter alle processer op – både på masteren og slaverne.

```
start-dfs.sh
start-mapred.sh
#kommandoer til opstart af klyngen
```

Der findes naturligvis kommandoer til at lukke klyngen ned, men det overlades til læseren at finde ud af hvad de hedder[5].

5 Det kan nævnes at de er placeret i samme katalog som kommandoerne til at starte klyngen.

Map/Reduce

Som det allerede bør fremgå tydeligt, er Hadoop en platform til udvikling af parallelle programmer. Det er ikke en generel platform, men understøtter et særlig paradigme i parallelprogrammering. Dette paradigme omtales ofte som Map/Reduce. Det kendes fra mange funktionsprogrammeringssprog (Haskell, Lisp, ML) og i nyere tid er det primært Google, som har gjort det kendt[6].

Tricket i Map/Reduce-paradigme er at identificere hvilke beregninger, som kan udføres uafhængig af hinanden. Denne fase kaldes for Map-fasen, mens samlingen af delresultater til et samlet resultat omtales som Reduce-fasen. Det er Map-fasen, som kan parallelliseres.

Et meget primitivt – og delvist trivielt – eksempel på Map/Reduce-teknikken er at en sum over en række tal kan deles op i en række mindre summer. Resultaterne af de mindre summer kan til slut lægges sammen og det endelige resultat udregnes. En mere matematisk formulering er

$$\sum_{k=1}^{N} x_k = \sum_{j=0}^{n-1} \sum_{i=1}^{N/n} x_{j+n \cdot i}$$

I en Hadoop-klynge er data distribueret udover en række maskiner. Opgaven for Hadoop er at bryde et program ned i mindre stykker og sende programmet hen til data. Derved vil netværkskommunikationen holdes på et minimum – husk at det kun er delresultaterne som skal sendes over netværket.

Lad mig komme med et ikke-trivielt eksempel. I USA registreres alle flyvninger[7] af Bureau of Transportion Statistics. Data er offentlig tilgængelige, og der er tale om ca. 5-600.000 flyvninger hver måned. Der registeres omkring 110 værdier for hver flyvning. Datasættet for første halvår af 2012 giver omkring 3.000.000 flyvninger og en fil på ca. 1,3 GB. I dette eksempel ønsker vi en oversigt over hvor mange adgange, der er fra en given lufthavn. I SQL vil oversigten kunne frembringes af følgende kommando[8]:

```
SELECT Origin, COUNT(*) FROM OnTime GROUP BY Origin;
-- SQL kommando til optælling af afgange (fra lufthavn)
```

6 http://research.google.com/archive/bigtable.html
7 http://www.transtats.bts.gov/Fields.asp?Table_ID=236
8 Det antages at hver registeret værdi er en attribut i en SQL-tabel.

Test-data består en af række CSV-filer – en for hver måned. Det er ikke nok at du har test-data liggende på din lokale harddisk hvis du skal bruge filen i et Hadoop-program. Lad os antage at vi har samlet de seks første måneder i én stor fil på 1,3 GB. Du kan kopiere filen til Hadoop med kommandoen:

```
hadoop dfs -copyFromLocal On_Time_Performance_2012_H1.csv /
#Kommando til at flytte test-data til Hadoop filsystemet
```

Hadoop vil begynde at distribuere filen ud til maskinerne i klyngen. Det kan let tage en rum tid, men du kan følge med i log-filerne på en af slaverne og se hvor meget arbejde, der udføres.

Et Map/Reduce-program i Hadoop-platformen er et Java-program. Programmet består af to underklasser, nemlig **`map`** og **`reduce`**. Map-fasen foregår ved at programmer læser en linje ad gangen fra den datablok, som er på den pågældende maskine. Linjen analyseres og output udsendes. Output er parret (lufthavn, 1). Findes Los Angeles' internationale lufthavn (LAX), tre gange i datasættet, vil der være tre gange parret (LAX, 1) i output.

Dette output modtages i Reduce-fasen hvor der sker en sammenlægning af alle 1-tallerne. På den måde findes det endelige resultat udregnes.

Nedenfor finder du eksemplet implementeret som et Map/Reduce-job i Hadoop. Programmeringssproget er Java.

I Exemplet finder Hadoop programmet antallet af rettidige afgange fra Los Angeles International Airport, kodenavn LAX

```
package dk.dkuug;

import java.io.*;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.filecache.DistributedCache;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

public class OnTime {
   public static class Map extends MapReduceBase implements Mapper<LongWritable, Text,
Text, IntWritable> {
     private final static IntWritable one = new IntWritable(1);

     public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException {
       String line = value.toString();
       String[] strArr = line.split(","); // CSV-fil
       Text airport = new Text(strArr[14]);         // nul-baseret
       output.collect(airport, one);
     }
   }

   public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable,
Text, IntWritable> {
     public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
IntWritable> output, Reporter reporter) throws IOException {
       int sum = 0;
       while (values.hasNext()) {
         sum += values.next().get();
       }
       output.collect(key, new IntWritable(sum));
     }
   }
  public static void main(String[] args) throws Exception {
     JobConf conf = new JobConf(OnTime.class);
     conf.setJobName("ontime");

     conf.setOutputKeyClass(Text.class);
     conf.setOutputValueClass(IntWritable.class);

     conf.setMapperClass(Map.class);
     conf.setCombinerClass(Reduce.class);
     conf.setReducerClass(Reduce.class);

     conf.setInputFormat(TextInputFormat.class);
     conf.setOutputFormat(TextOutputFormat.class);

     FileInputFormat.setInputPaths(conf, new Path(args[0]));
     FileOutputFormat.setOutputPath(conf, new Path(args[1]));

     JobClient.runJob(conf);
   }
}
```

#Et Map/Reduce job som Java source

For at bruge ovenstående Map/Reduce-job skal programmet oversættes. Når der er tale om et lille program som dette, kan det let klares med to enkle kommandoer.

```
javac -classpath /usr/lib/hadoop/hadoop-core-1.0.2.jar -d ontime_classes OnTime.java
jar -cvf ontime.jar ontime_classes
```

Text 6:

Du kan nu analysere data-sættet med følgende tre kommandoer.

```
hadoop dfs -rmr /output
hadoop jar ontime.jar dk.dkuug.OnTime /On_Time_Performance_2012_H1.csv /output
hadoop dfs -cat /output/part-00000
```

Text 7:

Den første kommando sletter evt. output fra tidligere kørsler. Det er ikke muligt at ændre en fil, som er placeret i HDFS. Du er derfor nødt til at slette den først og så skrive den nye version. Den anden kommando udfører selve Map/Reduce-jobbet. Køretiden afhænger naturligvis af hvor mange slaver, du har i din klynge. Den sidste kommandoer viser dig indholdet af output-filerne.

Konklusion

Det er ikke trivielt at sætte en Hadoop-klynge op. Det kræver kendskab til Java, Linux og netværk – og sikkert en del andet. Men når klyngen kører, har du mulighed for at skrive software til en – potentiel – stor parallel supercomputer.

Som paradigme er Map/Reduce interessant for software-udviklere. Det giver en let adgang til at parallellisere software til dataanalyse.

Der findes en lang række af applikationer ovenpå Hadoop, f.eks. et data warehouse under navnet Hbase, et dataflow-sprog med navn Pig og database med SQL-lignende syntaks med navnet Hive. Det spændende med disse applikationer er at de på mange måde rykker ved den hellige gral i programmering: automatisk parallellisering til distribuerede systemer.

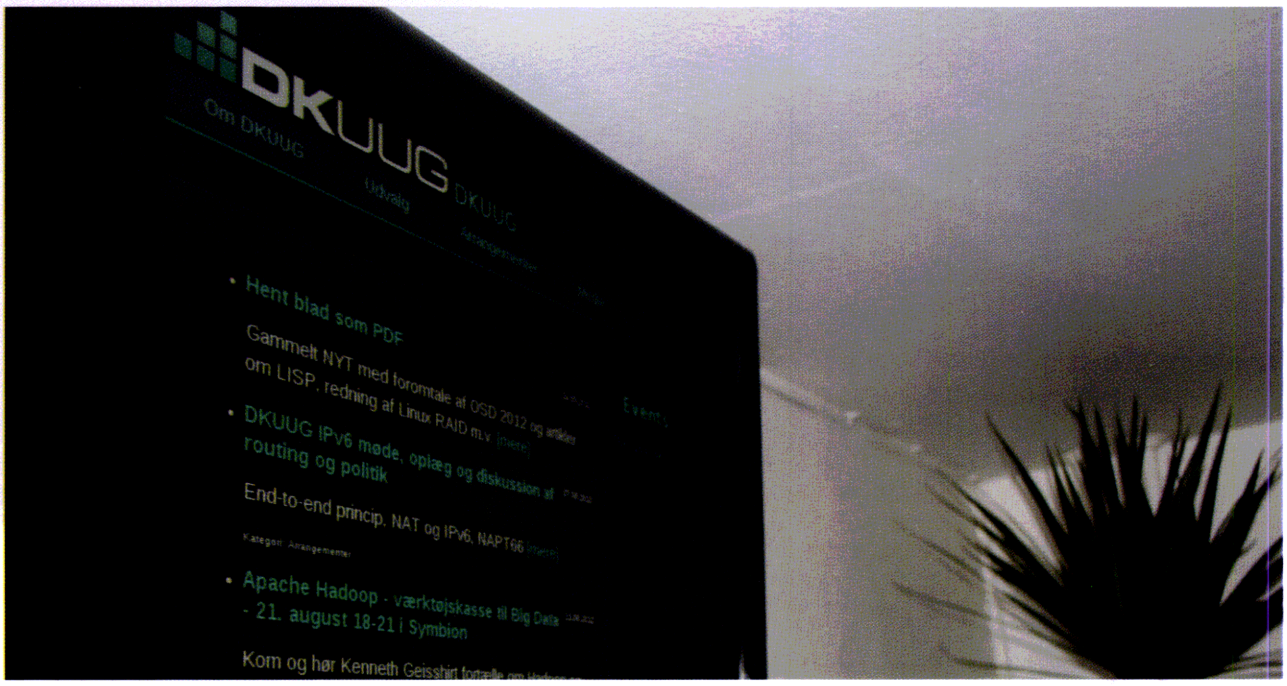
Om forfatteren:

Kenneth Geisshirt er kemiker af uddannelse og er en stærk fortaler for fri software. Hen brugte julen 1992 til at installere SLS Linux, og GNU/Linux har været hans favorit operativsystem lige siden. For tiden er han konsulent vedrørende videnskabelig beregninger og Linux clusters. Han bor i København med sin kone og deres to børn. Man kan kontakte Kenneth Geisshirt via http://kenneth.geisshirt.dk/.

Litteratur

- ***Hadoop: The Definitive Guide, 2nd edition*. T. White. O'Reilly Media, 2010.**
- ***Hbase: The Definitive Guide*. L. George. O'Reilly Media, 2011.**
- ***Programming Pig*. A. Gates. O'Reilly Media, 2011.**
- ***Data-Intensive Text Processing with MapReduce.* J. Lin & C. Dyer. Morgan&Claypool, 2010.**

Linux som Interaktiv vægdekoration



Af Mikkel Mikjær Christensen

Umiddelbart skulle man ikke tro det, men der er faktisk ikke særlig meget materiale tilgængeligt der beskriver hvordan man får en simpel Linux distribution som f.eks. Debian til at boote direkte op i en browser som viser en forudbestemt hjemmeside i fuld skærm.

Mit konkrete behov var at vise resultater fra over overvågningsserver på én central skærm på kontoret så alle kan se med, men der er mange flere muligheder end det. Det kunne også være en skærm i receptionen der byder gæster velkommen, eller fortæller hvilke mødelokaler der er ledige.

Umiddelbart er opgaven nogenlunde overskuelig:

1. Find en gammel fladskærm med fornuftige dimensioner.
2. Find en gammel computer der kan trække en browser.
3. Installer Debian på maskinen og bor fladskærmen op på væggen.
4. Installer X og en browser på Debian maskinen.
5. Få X og Browseren til at starte efter boot.
6. Få browseren til at gå ind på den side vi ønsker,
7. … og fylde hele skærmen, med siden vel og mærke, ikke toolbars og andet gøgl.

Punkt 1-4 var nemme, TV'et i frokoststuen blev skiftet ud da TV Signalet gik over til digitalt, den gamle skærm havde en HDMI Indgang, af computer valgte vi en gammel Shuttle PC med en 1200Mhz AMD Processor og 640MB RAM, jeg fandt en borremaskine kyndig person der blev bestukket med pizza til at hænge skærmen op på væggen og så kørte vi ellers en standard Debian netinstall uden X ind på maskinen.

Herfra begyndte det at blive sjovt, hvis man stiller spørgsmålet på nettet:"Hvordan får jeg min maskine til at logge ind uden password" så får man en lektion i sikkerhed, og en formaning om at det må man ikke. Når man efterfølgende forklarer at man udmærket har styr på sikkerhed og at opsætningen er til en installation uden mus og keyboard indrømmer de fleste at de faktisk ikke ved hvordan man gør det, så det har jeg selv måttet finde ud af.

Jeg valgte derfor at starte med det nemme, troede jeg, det viste sig nemlig at hverken Firefox eller Crome havde nogen umiddelbar mulighed for "kiosk-mode". Min oprindelige tanke var, at det var en selvfølgelighed, for jeg kunne da umuligt være den første med det behov; samtidig kunne jeg konstatere, at Firefox brugte alt for mange resourcer. Så jeg var nødsaget til at se mig om efter et alternativ.

Sådan lidt i delvis desperation og delvis stædighed gav jeg mig til at undersøge hvor kompliceret det kunne være og interface med en eksisterende HTML renderingsmotor, og efter lidt søgen fandt jeg nogle forskellige eksempler som jeg hurtig fik stykket sammen til en fungerende browser. 15 minutter og 15 linier (inklusiv et par linieskift for at gøre koden overskuelig) senere havde jeg en simpel webbrowser der lige præcis opfyldte mine krav:

```
#!/usr/bin/env python

import gtk, webkit

window = gtk.Window()
window.set_position(gtk.WIN_POS_CENTER)
window.set_size_request(1152,864)

browser = webkit.WebView()
browser.open("http://www.eksempel.dk");
window.add(browser);

window.show_all()

gtk.main()
```

Jeg er ikke den stærkeste selv til Python, men det var da nogenlunde til at gå til. Det mest forvirende var at maximize() funktionen ikke virkede, jeg må indrømme at jeg ikke testede det men jeg formoder kraftigt at det havde noget at gøre med at jeg kørte en rå X, dvs. uden Window manager.

For at kunne køre mit script skal jeg bruge en X og webkit med Python bindings, på Debian er det nemt klaret, og fordi min hardware ikke er helt ny behøver jeg faktisk ikke sætte X op selv:

```
# apt-get install x-window-system
```

Herefter starter jeg med med

```
# startx
```

Nu skifter hele skærmen til X og jeg bliver præsenteret for en xterm i det ene hjørne af skærmen, og i den skriver jeg:

```
# ./browser.py
```

Hvorefter hele skærmen nu først bliver hvid, hvorefter dkuug.dk toner frem: (billedtekst: min hjemmelavede browser i en virtualbox installeret til formålet)

Min hjemmelavede browser i en virtualbox installeret til formålet

Næste og sidste skridt er at få skidtet til at starte selv, ingen gider til at sætte keyboard og mus til maskinen omme bag reolen bare på grund af et strømsvigt, så jeg starter med at tage mig af automatisk login, først opretter jeg en bruger og så intallerer jeg debian pakken mingetty:

```
# useradd -m dummy
# apt-get install mingetty
```

Næste skridt kan godt være lidt farligt, og når jeg siger farligt så mener jeg at du kan risikere at skulle have fat i en livecd hvis du laver fejl i den her fil, det er nemlig /etc/inittab du skal have fat, den bestemmer overordnet set hvilke processer og terminaler der skal startes når maskinen booter. Find linien:

```
1:2345:respawn:/sbin/getty 38400 tty1
```

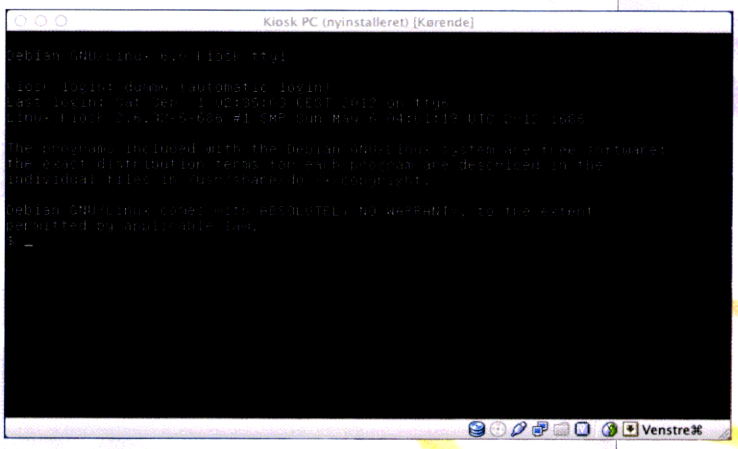
og ret den til:

```
1:2345:respawn:/sbin/mingetty --autologin
dummy tty1
```

og genstart maskinen:

```
# reboot
```

Når maskinen kommer op igen ser consolen sådan her ud: (igen er der tale om en virtualbox)



For at starte X tilføjer jeg følgende i bunden af /home/dummy/.bashrc:

```
if [ $(tty) == /dev/tty1 ];
        then
                echo "Starting X...."
                /usr/bin/xinit
fi
```

Det er sådan set ikke nødvendigt med andet end xinit, men på den her måde har vi mulighed for at logge ind som "dummy" brugeren fra en anden konsol hvis vi på et tidspunkt får lyst til det.

Opret en fil kaldet /home/dummy/.xinitrc med flg. Indhold:

```
#!/bin/bash

setterm -blank 0 -powersave off -powerdown 0
xset s off
xset s 0
xset -dpms

/home/dummy/browser.py
```

Det er egentlig også ret simpelt, setterm og xset kommandoerne forhindrer Linux i at gå i dvale eller på anden vis slukke skærmen, og sidste linie starter dit script, og nu ser endevæggen af vores kontor således ud: (Jeg må hellere få sat vores oprindelige overvågningsside på plads inden på mandag)

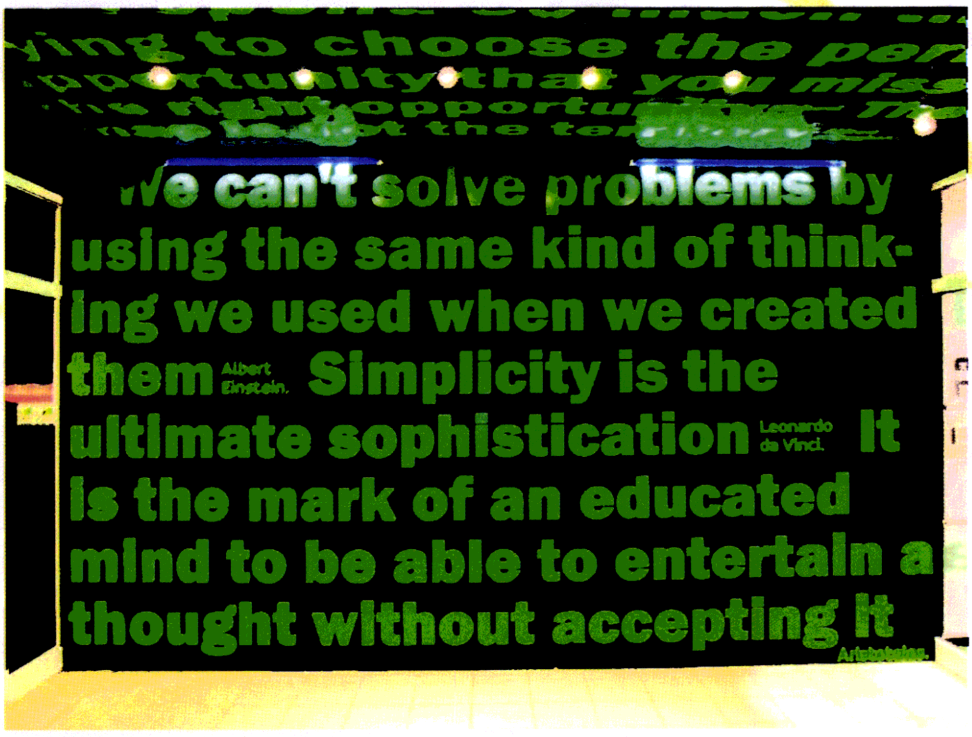
Hvad så nu? Ja maskinen har snart kørt uafbrudt i 3 måneder, i gamle dage havde vi en Windows XP maskine til samme formål, den crashede ofte dagligt men i hvert fald hver uge, og den var generelt meget langsommere til at skifte indhold.

Efter vi kørte den over på Linux har vi fået frigjort resourcer nok til at den kan spille netradio for os på kontoret samtidig med, og vi har sågar koblet den sammen med vores fastnet telefoner så den kan slukke musikken når der bliver snakket i telefon.

Det eneste jeg har været ude og købe til det her er et langt HDMI kabel (49kr) og et vægbeslag (149kr) til fladskærmen. Jeg håber i har fået blod på tanden til at lege lidt videre selv, jeg syntes i hvert fald det her er et genialt eksempel på hvad man kan få ud af gammel hardware hvis man gider bruge lidt tid på det.

Om forfatteren:

Mikkel Mikjær Christensen arbejder til dagligt med opsætning, optimering og sikring af professionelle hostingløsninger baseret på Opensource igennem sit eget firma Mikjaer Aps (https://www.mikjaer.com), Mikkel blogger til dagligt på sin egen blog http://www.specialhosting.dk hvor han kommer med gode råd og guides relateret til sit daglige arbejde med Linux og andre Opensource systemer.



Fra indgangen til Symbion: Enkelhed er den ultimative sofistikerethed (Leonardo da Vinci)

AAARRRGGGHHH! - Når data overskrives

Af jon.bendtsen@jonix.dk

Jon Bendtsen har arbejdet med mange former for IT siden 97 og kan hyres som freelance IT konsulent. Jon Bendtsen er kandidat i datalogi fra DIKU, har bred viden og erfaring inden for Open Source og systemadministration. I fritiden er Jon Bendtsen nok mest kendt som en aktiv ildsjæl i DKUUG og Open Source Days.

Egentlig var det slet ikke meningen at jeg skulle skrive denne artikel nu. Men når man nu alligevel skal lege arkæolog for at redde vigtige data så kan man jo lige så godt skrive artiklen. Specielt også fordi emnet er egentlig noget som jeg havde planlagt at skrive om.

0) Før ulykken

I DKUUG optager vi en del foredrag, men det har knebet lidt med at få udgivet videoerne. Det skyldes forskellige ting, bla. at vi er for få "tordenskjoldssoldater" men også at vi længe bare ikke har fundet det rigtige værktøj til at håndtere filerne fra vores kamera, specielt også fordi det er ret store data der skal overføres – (HD) videoredigering kræver en ret kraftig computer, og data skal helst ligge lokalt på hurtige diske.

En af mine potentielle ny kunder ville have noget videreudvikling af et open source video content management system, og under dette faldt jeg over et open source online flash baseret videoredigerings værktøj. Opgaven for kunden blev aldrig til noget, sådan er det jo nogle gange, men jeg mente at open source værktøjerne kunne bruges i DKUUG. Jeg gik i gang med at eksperimentere fordi jeg vil egentlig gerne have en online redigeringsløsning, som let kan crowdsources således at I alle sammen kan hjælpe.

Desværre så virkede editoren ikke super godt, det gav hakker frem og tilbage lige når man klippede en video over. Jeg fandt dog på en work-around: man kan bare klippe i en textbox, der var lagt oven på alle videoerne, og det løste problemet med hakkerne. Når man laver denne online "redigering" - egentlig bare en indikation af foredrags start, pause, genoptag og slut, så er video kvaliteten ikke særlig vigtig. Det vigtige er lyden og at der trods alt er et billede, så video delen er meget kraftigt reduceret.

Når man er færdig med redigeringen kan dette online flash værktøj gemme en XML fil på serveren, så den er jo lige til at parse. Og det var under udviklingen af scripts til at parse denne XML fil og klippe i de rå videofiler fra kameraet at en fejl i FFmpeg[1] gjorde at 4 ud af 5 rå videofiler blev overskrevet. **ØV!**

FFmpeg manualen svarer ikke helt til opførslen

I FFmpeg manualen står der:

> '-passlogfile *prefix* (*global*)'
> Set two-pass log file name prefix to *prefix*, the default file name prefix is "ffmpeg2pass". The complete file name will be 'PREFIX-N.log', where N is a number specific to the output stream

men det viste sig at når man bruger -vcodec libx264 så er -passlogfile ikke et prefix, men det er navnet på den fil som FFmpeg bruger som passlogfil, og i mine scripts så brugte jeg navnet på input filen, altså de rå videofiler direkte fra kameraet, som prefix navn på passlogfilen, fordi jeg godt ville se i den bagefter.

Hint: lad være med at bruge filnavnet på vigtige data som eneste pre-/suffix, brug også noget andet text, i dette tilfælde ville $filename.passlogfile eller passlogfile.$filename have reddet data.

1) Stands ulykken

ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c ctrl+c

Behandling af 5x 2GB store HD videofiler tager altid lang tid, så jeg kørte scriptet i en screen og gik ud i køkkenet for at lave og spise aftensmad.

Hint: lad være med at gå fra dine nyligt udviklede scripts før du har sikret dig at de kører ordentligt, havde jeg gjort det, så ville fejlen have 1, måske 2 destrueret kildefilder i stedet for 4.

I stedet for så opdagede jeg først skaden da jeg kom tilbage fra frokost, og der var 4 ud af 5 filer destrueret. Da jeg opdagede skaden så fik jeg dog ret hurtigt afbrudt mit script med **ctrl+c**

2) Giv datareddende førstehjælp

Da jeg blev klar over at de rå data var destrueret og tabt, så tænkte jeg straks på at det var da godt at det var den nyeste video optagelse jeg sad og arbejdede på, og at kameraet næppe var blevet brugt igen her få dage efter optagelserne. Desværre havde jeg slettet optagelserne fra kameraet for at gøre det klar til næste brug, men jeg tænkte at det måske var muligt at lave en undelete i det dos filsystem som kameraet bruger, eller måske redde data på andre måder, så jeg tog straks ud i DKUUG for at sikre mig kameraet.

Hint: sørg for at **ingen** harddiske bliver overskrevet. Set i bakspejlet så kunne jeg måske have kørt photorec på linux harddisken foruden den på kameraet.

Da jeg ankom til DKUUG så brugte jeg dd til at tage en bit-for-bit kopi af hele kameraets harddisk, således at jeg kunne arbejde på denne kopi uden at risikere at harddiskens indhold blev beskadiget.

3) Alarmer dine venner så de kan fortælle dig om photorec

Der findes mange forskellige data rescue værktøjer, så spørg dine venner hvad de har brugt, både med og uden success. Desuden har dine udenforstående venner den fordel at de ikke selv er stresset over datatabet og de må derfor forventes at holde

1 https://ffmpeg.org/

hovedet koldere og derfor kan foreslå dig at tage en kopi af harddisken der skal reddes data således at dit rescue forsøg ikke før skabet yderligere ødelæggelser.

I mit tilfælde havde jeg selv taget en kopi af harddisken, men det var dog Bryan Østergaard, formanden for Open Source Days ApS og SSLUG der foreslog mig at bruge værktøjet photorec til at forsøge at redde mine data.

Photorec

På trods af navnet **Photorec**[1] så kan det finde mange forskellige slags filer, bla. også MPEG filer som jeg har brug for. Det er et Open Source program og findes til stort set alle platforme. Selvom at det er tekst baseret, bruger det nogle menuer til at stille dig nogle spørgsmål om kildedisken, bla. opbygning, filsystem, du kan endda justere hvilke filer den leder efter. Derefter er det bare at starte søgningen og vente for at se hvad den finder. Det er ret lige til så jeg vil ikke gå nærmere ind i detaljerne, den øvelse overlader jeg til læserne.På debian er programmet photorec en del af pakken testdisk. Jeg brugte følgende kommando linie for at starte photorec:

```
PhotoRec 6.11, Data Recovery Utility, April 2009
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org

Disk ../sdb.dd - 40 GB / 37 GiB (RO)
     Partition          Start            End        Size in sectors
 1 P FAT32 LBA       0   1  1    4863  30 63   78125985 [CANON_HDD]


36 files saved in . directory.
Recovery completed.
m2ts: 24 recovered
jpg: 6 recovered
sqlite: 2 recovered
tx?: 2 recovered
apple: 1 recovered
gz: 1 recovered
```

`/usr/sbin/photorec /l /d rescued_filer ../sdb1.dd`

Hvis du ikke har plads til at tage en kopi af harddisken, så kan man godt køre photorec direkte på harddisken, photorecs manual side lover at de kun læser fra disken. Men jeg er belært af erfaringen med FFmpeg's manual der også lovede at passlogfilen hed noget mere, så jeg vil hellere være på den helt sikre side og kun arbejde på en kopi af disken når jeg forsøger at redde data.

Resultatet og kvaliteten af de reddede data

Først reddede jeg mine data og derefter så kørte jeg photorec en gang til for at tage screenshots osv. til denne artikel. Desuden så gav det mig den mulighed at jeg kunne sammenligne resultatet imellem de 2 rescue forsøg, og her er jeg desværre lidt skuffet, idet at ingen af de reddede filer har en identisk størrelse imellem de 2 rescue forsøg. I et 3. forsøg har jeg sporet forskellen til at skyldes forskellige valg i menu systemet undervejs.

Som I måske har bemærket tidligere i denne artikel, så fik jeg kun beskadiget 4 ud af 5 filer, så jeg burde vel kunne finde den 5. fil iblandt de reddede filer? Ja og nej. Indholdet er der, men også her er der forskel på filernes størrelse. Forskellen er at den reddede fil er ca. 70 MB (ud af ca. 2GB) mindre end den gode fil jeg stadig har tilbage. Dette skyldes måske at kameraet gemmer mere i .MTS filen end kun lyd og billede.

I det store hele så har **photorec** dog reddet langt det fleste data, for hvis jeg afspiller de reddede filer i VLC så kan jeg godt se Kenneth Geisshirt holde et foredrag om Apache Hadoop optaget tirsdag den 21. august 2012. Det ser endda stort set normalt ud. Der er et lille problem i overgangen imellem nogle af de reddede videofiler, det ser ud til at der mangler nogle få sekunder. Ærgerligt, men ikke katastrofalt. Så det ser ud til at jeg en eller anden dag kan få udgivet en video med Kenneth Geisshirts foredrag, men det bliver forsinket

Konklusion: Photorec kan godt anbefales!

4) Imens du venter på at photorec bliver færdig, så tænk, erkend og ret dårlige vaner der leder til datatab

Det tog ca. 2. timer at køre photorec på min 38GB kopi af kameraets harddisk, og imens jeg ventede så brugte jeg tiden til at tænke over hvad der ledte op til destruktionen af data samt hvad jeg kan gøre for at formindske risikoen for at det sker igen.

1. Lad være med at køre dine testscripts som root
2. Lad være med at køre dine testscripts med den bruger der ejer filerne
3. Lav vigtige filer read-only
4. Tag backup af vigtige filer før du kører dine testscripts
5. Lad være med at slette filerne på kameraet før du skal bruge pladsen
6. Brug relativ path til input filer og ingen path til temporære filer, eller omvendt.

Ved (blad)redaktionens afslutning er videoen produceret og sendt gennemsyn hos foredragsholderen inden den endelige offentliggørelse.

1 http://www.cgsecurity.org/wiki/PhotoRec

Oversætte Android fra bunden

Af Jacob Nordfalk

Kildekoden til Android ligger jo offentligt tilgængelig, under navnet AOSP (Android Open Source Project).

Som Androidudvikler og -underviser finder jeg ofte mangler i dokumentationen, og dette - og generel nysgerrighed - gjorde at jeg besluttede mig for at prøve at hente kildekoden ned oversætte min egen udgave.

Det gav et lille nostalgisk stik, for det er over 15 år siden jeg droppede at oversætte min egen kerne til den Linux jeg bruger i mit daglige arbejde.

Opsætning

Processen er rigtig fint beskrevet på http://source.android.com , men jeg gengiver her essensen. Den klart nemmeste måde er at bruge 64-bit Ubuntu Linux 11.04. Andre Linuxer eller Mac er mulige, med mere besvær, mens Windows er umulig.

Herunder er processen beskrevet for Ubuntu Linux 11.04

Først skal byggemiljøet sættes op og en række standardpakker installeres (http://source.android.com/source/initializing.html). For at gøre det nemmere at kopiere ind har jeg ikke sat $-tegn i venstre side:

```
sudo apt-get install python-software-properties
sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
sudo apt-get update
sudo apt-get install sun-java6-jdk
sudo apt-get install git-core gnupg flex bison gperf build-essential \
 zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs \
 x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev \
 libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown \
 libxml2-utils
# Kommandoer som tilføjer repository for utilities til kompilering,
# incl. Java6.
```

Nu kan vi hente kildekoden.

Hente kildekoden

Kildekoden til Android ligger i versionskontrolsystemet git og fylder p.t. (september 2012), sammen med forskellige støttebiblioteker o.a. omkring 15 GB.

Først skal kommandoen 'repo' hentes og derefter initialiseres med hvilken version af Android man ønsker at arbejde med:

```
mkdir ~/bin
PATH=~/bin:$PATH
curl https://dl-
ssl.google.com/dl/googlesource/\
git-repo/repo > ~/bin/repo
chmod a+x ~/bin/repo

# Opret mappen til kildekoden
mkdir android-src
cd android-src
repo init -u \
https://android.googlesource.com\
/platform/manifest
# kommandoer til opsætning af download
```

Dette giver "master"-branchen, dvs den allernyeste udgave, men man får også information om de andre forskellige 'tags':

```
* [new tag] android-2.3.7_r1 -> android-2.3.7_r1
* [new tag] android-4.0.1_r1 -> android-4.0.1_r1
* [new tag] android-4.1.1_r4 -> android-4.1.1_r4
```

Vil man lege med den "officielle Android 4.1" kan man tage `android-4.1.1_r4`-branchen:

```
repo init -b android-4.1.1_r4
```

Eller, for nyeste Android 2.3.7:

```
repo init -b android-2.3.7_r1
```

Senere kan man hoppe tilbage til 'master' med:

```
repo init -b master
```

Bemærk at der ikke er nogle tags for Android 3. Google har begrundet det med at de ville undgå at udviklere begyndte at kigge i (og basere sig på!) kode der ikke var moden og har derfor fjernet tags'ene. Kildekoden er der dog hvis man leder på finere niveau (changeset-niveau i stedet for tag-niveau).

Når du har valgt version og er klar til at hente kildekoden skriver du:

```
repo sync
```

repo sørger derefter for at hente kildekoden fra alle git-depoterne i den rigtige version. Det kan godt tage nogle timer, da der i alt er tale om ca. 15 GB. Er der netværksfejl undervejs kan man kalde 'repo sync' igen for at fortsætte overførslen.

Når alt er klart (f.eks. næste dag :-), så skal der endelig bygges:

```
. build/envsetup.sh
lunch full-eng
make -j5
```

Den første kommando sætter en række miljøvariabler. Næste konfigurerer at vi ønsker at oversætte til emulatoren (og ikke til pandaboard eller en af Googles nexus-telefoner) med alle udviklerpakker.

Til sidst starter vi bygningen med 'make' med 5 parralelle processer (passer godt til min 2-kernemaskine).

Bidrage til koden

Der er god mulighed for at bidrage til kildekoden. Først bør man melde sig ind i diskussionsgruppen http://groups.google.com/group/android-contrib og læse lidt hvad folk skriver. Listen er modereret og til tider bliver indlæg afvist med begrundelsen at de ikke hører til i gruppen, men generelt er udviklerne hjælpsomme og hurtige til at svare så længe det gælder tekniske problemer.

Mere principielle spørgsmål undlader de ofte at svare på, givetvis fordi det er en offentlig postliste og de svar de giver kunne opfattes som forpligtende løfter.

Introduktion til git

git er et såkaldt distribueret versionskontrolsystem, hvilket betyder at man kan arbejde uden nødvendigvis at have noget centralt repository.

Ligesom i det mere velkendte versionskontrolsystem svn (subversion) har man en lokal kopi så man ikke behøver kontakte en central server for at se hvilke ændringer man har foretaget.

Den største forskel er at man i git først committer ændringerne til det lokale repository og kan vente til senere med at skubbe dem over til en centrale server.

Her er en kort sammenfatning af de vigtigste kommandoer.

`git status`

Denne kommando fortæller hvilke filer man har ændret i og hvilke evt ekstra filer der ligger og flyder rundt. Svarer til 'svn status', men er mere forklarende og fortæller f.eks at...

`git reset HEAD <filnavn>`

bruges til at nulstille sine ændringer (dvs svarer til 'svn revert'). Filnavn kan være punktum (.) for at nulstille alle ændringer i den aktuelle mappe.

`git diff`

Viser ændringerne man har foretaget. Kan også bringes til at vise differens mellem to tidligere ændringer. Svarer til 'svn diff'.

`git add <filnavn>`

Tilføjer ændringerne i nogle filer/mapper til næste commit (kaldet 'indexet'). Folk der plejer at bruge svn kunne fristes til at man opbygger en liste af filer *før* man committer, og det er næsten rigtigt, men det er altså *ændringerne* til disse filer man registrerer. Vil man registrere alle ændringerne i den aktuelle mappe (.) skriver man 'git add .'

Man kan se hvilke ændringer man har lagt i kø i indexet med 'git status' og denne kommando viser også hvad man skal gøre hvis man vil droppe ændringerne i en fil.

`git commit`

Bruges når man er klar til at comitte sine ændringer. Man skal skrive en meddelelse der beskriver ændringerne. Derefter committes til det *lokale* repository, det vil sige det virker som om ændringen er lagt ind og man har 'ren røv at trutte i', men ændringen ligger altså stadig kun på den lokale maskine.

`git push`

Sender/skubber alle ændringer (commits) man har foretaget på det lokale repository til et andet repository, typisk tilbage til der hvor man hentede filerne i første omgang. Denne kommando bruges ikke i Android-sammenhænge men beskrives her for fuldstændighedens skyld.

Dermed kan subversionbrugere altså tænke på `svn commit` som delt op i 3 processer, nemlig git add, git commit og git push.

Andre sjove git-kommandoer

`git log`

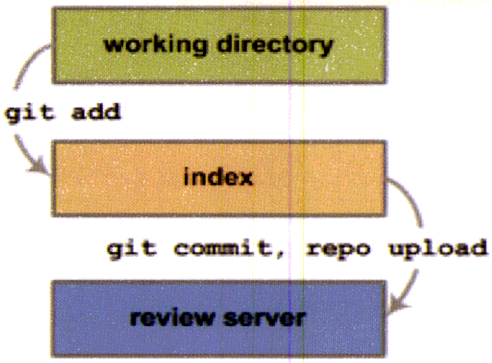
Giver en log over ændringer der er foretaget. Det er sjovt at kigge i hvordan udviklingen er foregået.

`git log --graph -p .`

Variant hvor man får man en graf ude til venste der viser forgreninger og sammensmeltninger, man ser diff'er over hvad der er er ændret og man ser kun hvad der er sket i den aktuelle mappe og undermapper

Lave rettelser

Android er et meget stort projekt og det er derfor fordelt ud på en række forskellige mindre git-repositorier (pt. er der 228). For at hjælpe med styringen bruger man kommandoen 'repo', der koordinerer de forskellige git-depoter, sådan at versionerne på dem stemmer overens.



Man kan naturligvis ikke direkte skrive i Androids kildekode. I stedet sendes/skubbes rettelseforslagene til gerrit, som er en ændrigsforslagsserver hvor dine rettelsesforslag kan blive vurderet og diskuteret.

Før man laver en rettelse

Før man går i gang bør man se http://source.android.com/source/submit-patches.html igennem. Først skal man lave en forgrening (branch) af koden

```
repo start NAVN
```

Navnet er bare hvad du selv vil kalde din forgrening og er i sig selv ligegyldigt. Det er kun dig der ser det.

Derefter laver man sine ændringer og når man er færdig skriver man

```
git add <filnavne>
```

```
git commit
```

og skriver en kort beskrivelse af din rettelse. Denne kommer i ændringsloggen og er derfor vigtig.

Sende rettelserne

Rettelsesforslagssystemet hedder Gerrit. I Gerrit kan man i fuld offentlighed følge med i de forslag til patches som kommer ind og hvordan de bliver behandlet:

https://android-review.googlesource.com/#/q/status:open,n,z

Kildekoden til Android blev, sammen med Gerrit, oprindeligt huset sammen med Linux-kernen på kernel.org. Den blev desværre hacket og kompromitteret af ukendte gerningsmænd medio 2011.

Koden fandtes selvfølgelig et utal af andre steder, men hovedtræet, historikken, rettelsesforslag og hele godkendelsessystemet var utilgængelige i et halvt års tid indtil det hele blev genetableret, på Googles egne servere.

Generelt er Googles politik omkring udvikling at ting bliver offentliggjort "når de er klar". Det vil sige at man ikke på forhånd kan vide hvornår f.eks. en ny udgave af Android bliver frigivet eller præcist hvad den indeholder.

Hvad kan man bidrage med?

Som udvikler kan man *altid* bidrage med at opfriske dokumentationen.

Der er nemlig masser at tage fat på, som f.eks. følgende udtræk viser:

http://code.google.com/p/android/issues/list?can=2&q=Documentation&sort=stars&colspec=ID+Type+Status+Owner+Summary+Stars&cells=tiles

Dokumentationen er delt op javadoc og anden dokumentation.

Javadoc ligger jo som specielle kommentarer i kildekoden, så vil du for eksempel rette i http://developer.android.com/reference/android/app/Activity.html skal du åbne
frameworks/base/core/java/android/app/Activity.java
og rette her.

Resten af dokumentationen, incl. de forskellige guides, ligger andetsteds. Vil du rette i f.eks. http://developer.android.com/guide/topics/fundamentals/fragments.html

skal du åbne
frameworks/base/docs/html/guide/topics/fundamentals/fragments.jd
og rette her.

Git - fast version control - logo

Eksempel - javadoc

I dokumentationen står der at Application-objektet bliver initialiseret 'før noget andet', men det passer ikke helt, for content providers bliver faktisk initialiseret før Application-objektet, hvilket ikke kunne ses af dokumentationen (se http://code.google.com/p/android/issues/detail?id=8727).

Først gik jeg ind i frameworks/base/ hvor kildekoden til alle de klasser man normalt bruger (såsom Activity, Intent etc.) er. Så lavede jeg en `'repo start min_rettelse'.` Jeg rettede derefter i core/java/android/app/Application.java så dokumentationen var korrekt. Derefter kaldte jeg

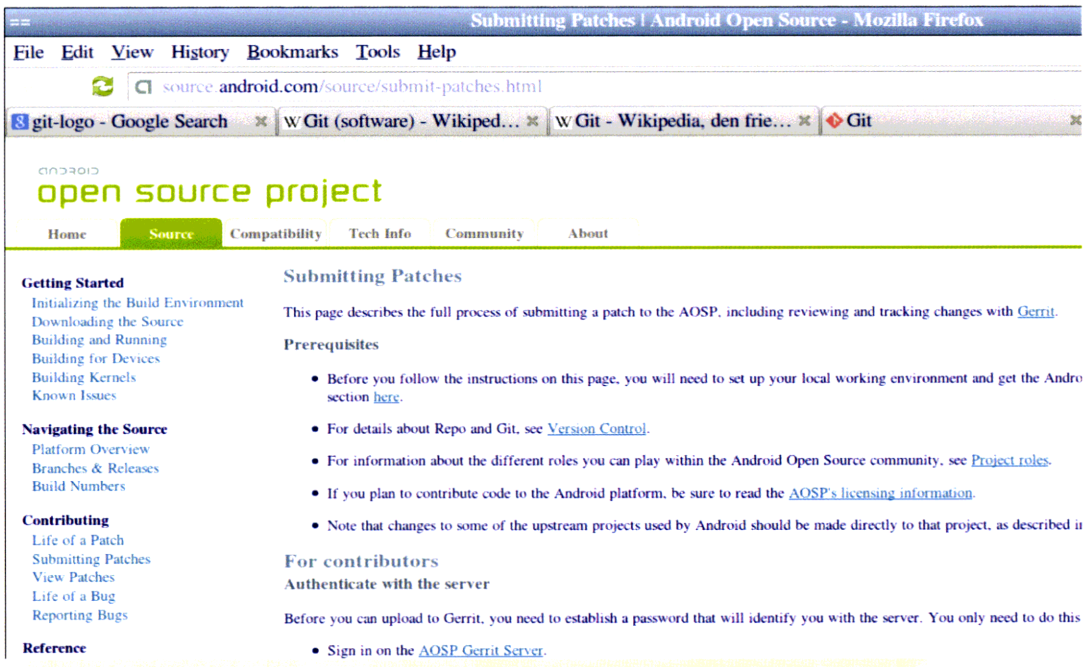
```
repo diff
```

for at tjekke at alt så rigtigt ud. Det gjorde det:

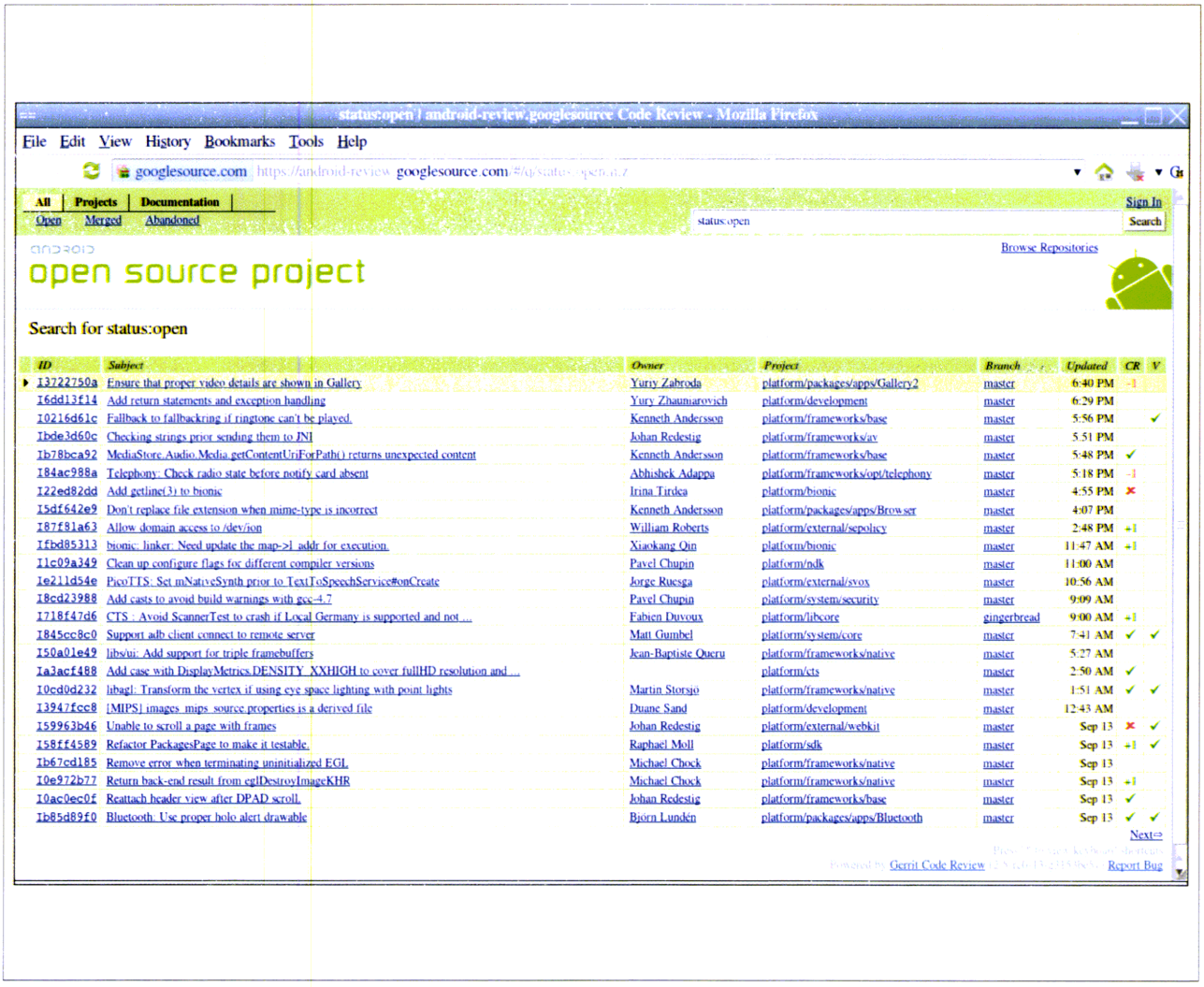
```
project frameworks/base/
diff --git a/core/java/android/app/Application.java
b/core/java/android/app/Application.java
index dd9ea26..3f30790 100644
--- a/core/java/android/app/Application.java
+++ b/core/java/android/app/Application.java
@@ -64,8 +64,11 @@ public class Application extends ContextWrapper implements
ComponentCallbacks2 {
     }

     /**
-     * Called when the application is starting, before any other application
-     * objects have been created.  Implementations should be as quick as
+     * Called when the application is starting, before any activity, service,
+     * or receiver objects have been created.
+     * Note that content providers are created before the application object
+     * (see http://code.google.com/p/android/issues/detail?id=8727).
+     * Implementations should be as quick as
      * possible (for example using lazy initialization of state) since the time
      * spent in this function directly impacts the performance of starting the
      * first activity, service, or receiver in a process.
```

Derefter fulgte jeg resten af vejledningen i http://source.android.com/source/submit-patches.html , hvorefter min rettelse blev optaget (som https://android-review.googlesource.com/#/c/30921/).



Jacob Nordfalk er cand.scient. i fysik og arbejder nu som lektor på Center for Videreuddannelse på Ingeniørhøjskolen i København og som konsulent og underviser for Lund&Bendsen A/S. P.t. afholder han kurser i Androidprogrammering, men tidligere har han afholdt kurser i objektorienteret programmering, web- og serverprogrammering, videregående programmering, objektorienterede metoder og i Linux som arbejdsstation og som server. Han bruger Ubuntu Linux til dagligt og har skrevet tre bøger om Javaprogrammering, der ligger på http://javabog.dk .



Bog om Git

En bog om *Git* på engelsk, skrevet af Scott Chacon, udgivet af Apress, kan læses på websitet http://git-scm.com/book. Alt indholdet er under Creative Commons Attribution Non Commercial Share Alike 3.0 license. Print versioner af bogen kan købes via Amazon.com. Herfra citeres (egen oversættelse):

A Short History of Git

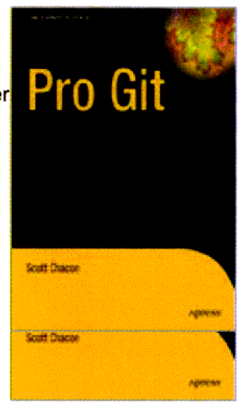
Som så mange andre store ting i livet begyndte Git med en smule kreativ destruktion og rasende skænderier. Linux-kernen er et open source software projekt af en anseelig størrelse. Gennem den første tid af Linux kernens levetid, nemlig fra 1991-2002, blev ændringer og tilføjelser sendt rundt som patches og arkiv-filer (tar/zip). I 2002 begyndte Linus Torvalds at bruge et proprietært distribueret versions-kontrol system (DVCS) ved navn **Bitkeeper**.

I 2005 blev forholdet mellem kerne-udviklerne og det kommercielle firma, som udviklede Bitkeeper, anstrengt, og programmets "anvendelse uden vederlag-status" blev trukket tilbage. Det fik Linux udviklerne - særligt Linus Torvalds, ophavsmand til Linux - til at udvikle sit eget program, baseret på nogle af de erfaringer, der var gjort mens de brugte Bitkeeper.

Nogle af målene med det nye system var som følger:

- Hurtighed
- Simpelt design
- God support for non-linear udvikling (tusinder af parallelle forgreninger)
- Fuldt distribueret
- I stand til at håndtere store projekter som fx. Linux kernen effektivt (hurtigt trods store mængder data)

Siden fremkomsten i 2005 har Git udviklet sig og er modnet til et let anvendeligt system, som stadig har disse grundlæggende kvaliteter. Det er utroligt hurtigt, det er meget effektivt for store projekter, og har et utroligt branching system for ikke-lineær udvikling. (Se Kap. 3 i bogen).



***

Det kan tilføjes, at Linux-kernen i dag har omkring 50.000 filer og fylder ca. 500MB som kildetekst.

Node.JS for system administratorer

af David Askirk

Node.JS er et stadig nyt framework fra 2009. Det er bygget oven på Google V8 motor, som er den JavaScript motor som også er en del af Chrome browseren. Det er lavet af Ryan Dahl, og stærkt støttet af Joyent. Der er en del større virksomheder som benytter sig af Node.JS her iblandt Microsoft, der bruger det på deres Azure, og Yahoo. Joyent bruger det selvfølgelig også på deres no.de hosting. Det er endda muligt at deploye Node.JS applikationer til sådan noget som Heroku.

Node.JS er event baseret, det vil sige at i stedet for at køre det multitrådet fungerer det ved en masse små events, såsom pakke modtaget eller lignende.

I denne artikel vil der blive beskrevet hvordan Node.JS kan bruges i system administrations arbejdet til at automatisere nogle arbejdsgange.

Inden vi lige hurtigt gennemgår installationsmuligheder kommer her et meget hurtigt eksempel, der viser Node.JS styrke.

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://127.0.0.1:1337/');
```

Dette eksempel er en meget meget lille http server, men skrevet på 6 linjer! Se det er fantastisk, men er også det Node.JS er lavet til.

Installation

Men hvordan installerer man Node.JS? Der er flere muligheder, hvis man ikke vil bruge den nyeste stable udgave, kan det installeres med din yndlings pakkemanager til linux, som en binary til windows og en pakke til Mac OS X.

Men hvis man gerne vil køre nyeste stable udgave, og det er der ikke nogen grund til at lade være med, skal man hente den fra deres side, eller compile det selv. Der kommer ny udgave ca hver 14. dag. Man henter en source tarball ned, og laver den klassiske `./configure && make` og derefter en `make install` som root, så er man igang. Samtidig får man også installeret npm, som er Node Package Manager.

Node Package Manager indeholder en stor mængde software, blandt andet mulighed for at snakke med en serialport, eller styre en Lego Mindstorms Robot. Man kan sige `"npm search <søgord>"` og installerer med `"npm install <pakke navn>"`

Der findes mange gode guides til at bruge nodejs. O'Reilly har en udmærket bog: http://shop.oreilly.com/product/0636920015956.do som er en god guide til at komme igang med Node.JS.

Hvad kan Node.JS bruges til?

Men hvad kan Node.JS så bruges til? Mange ting, der er allerede bindings til mange C-libraries fx. SDL og libpcap.

Da Node.JS bygger på en javascript motor, bliver det hele skrevet i JavaScript, man kan også bruge Coffeescript.

Men hvor passer nodejs så ind i en system administrators hverdag?

På mit arbejde kører der flere forskellige nodejs services som hver især opfylder et behov, jeg vil beskrive nogle af dem.

Den første lille service er en lille service der holder øje med antallet af online brugere og viser dem på et display. Den anden er en service der henter apache status og viser nogle grafer over webnoderne og den sidste er en service der gør det nemt at styre start og stop af webnoder.

Den første service består af et display med 4 syvsegmenter. Denne henter så antallet af online brugere fra vores analyseværktøj og viser tallet så man kan følge med i bruger antallet. Dette giver et godt billede hvis der sker en teknisk fejl, da antallet af brugere så vil falde drastisk. Denne benytter sig af Node.JS på grund af to ting. Den er skrevet sammen med vores frontend udvikler som er en haj til javascript, samt det at lave http request er meget meget nemt i Node.JS.

Den anden service henter status fra apache servere som har http://httpd.apache.org/docs/2.0/mod/mod_status.html aktiveret. Denne kan så tælle antallet af apache processer der har travlt, og derved kan vi få et billede af hvilke webnoder i systemet der har meget at lave. De bliver så plottet ved at skrive data'en til en fil, og bede gnuplot, via Node.JS's subprocess modul, om at plotte de sidste linjer, så det passer med man altid ser data 3 timer bagud. Vi ved hvor grænserne er sat i henhold til serveren og kan på den måde sige hvilke webnoder der har meget at lave, og sætte ind der.

Den sidste service er en meget simpel wrapper omkring vores control scripts til at styre vores webnoder. Ved hjælp af disse scripts er det så muligt at starte eller lukke webnoder efter behov. Dette har gjort at vi har kunne automatistere antallet af webnoder så det matcher det forventede antal brugere, samt evt. peak antal inden for forskellige tidsrum af døgnet. Alt dette gøres via simple http kald, så ved hjælp af Node.JS har vi fået en mulighed for at styre vores serverfarm på en simpel og ensartet måde.

Men hvorfor så bruge Node.JS og ikke skrive det hele i perl. Jeg har fundet ud af gennem mit arbejde at fordelen ved at bruge Node.JS er at alle kan udvide de services der bliver lavet, eller ihvertfald forstå koden, dette giver en form for gennemsigtighed der er meget rar for alle parter. Da Node.JS også er et kraftigt værktøj til at lave små webservices i gør det det også nemmere for mindre tekniske folk at følge med i den daglige drift, samt give dem mulighed for at påvirke driften hvis det er nødvendigt. JavaScript er samtidig et sprog som har mange ligheder med tankegangen i funktionelle sprog, og dette gør det også spændende set med de mere akademiske briller.

Så prøv at leg med Node.JS og se hvor nemt det er at skrive din næste lille systemadministrator service i Node.JS.

David har arbejdet med utroligt mange forskellige slags sprog, men er gladest for dem der lugter mest af *lisp*.

Til dagligt arbejder David som systemadministrator hos EduLab ApS, skaberne af matematikfessor.dk

IPv6 Private adresser

Kan man spore dit IPv6 nummer?

af Donald Axel

I 1995, da Internet Engineering Task Force (IETF) var ifærd med at planlægge IPv6, forestillede man sig ikke, at der kunne være behov for at kunne kommunikere anonymt over Internettet. Ved Murens Fald betød midlertidige forbindelser at nyhedsstrømmen fra Østblok-landene blev spredt både i Øst og Vest, og denne nyhedsstrøm var med til at skabe håb om at de kommunistiske styrer ville lade Jerntæppet falde.

Senere har flere tilfælde med tracking og eavesdropping gjort det klart, at der er scenarier, hvor der er legitime grunde til at klienter ikke ønsker at kunne blive identificeret. Anonyme IPv6 adresser kan sammenlignes med et hemmeligt telefonnr. eller "privat nummer" i tlf.displayets nummer-visning.

Med IPv4 var der yderligere et sikkerheds-aspekt ved computere bag en NAT router (NAPT44). NAT for IPv6 er imidlertid deprecieret, fordi det bryder princippet om stateless end-to-end konnektivitet [rfc4966, Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status].

Et af forslagene går på at et system (en klient/server) skal kunne have to adresser, en permanent og en midlertidig (rfc4941 Privacy Extensions for Stateless Address Autoconfiguration in IPv6). Men selvsamme RFC understreger, at en privat, midlertidig adresse ikke hindrer tracking (eller aflytning, spoofing m.v.). Det er ineffektivt overfor sofistikerede former for trafik-analyse. Hvis man vil opnå større sikkerhed, må man bruge Onion Routing, d.v.s. en proxy, som skal være sikret mod inspektion.

Men til gengæld har IPv6 andre metoder til at oprette lokale netværk med adresser, som ikke routes globalt. Denne og andre extensions til IPv6 specifikationerne ser for forfatteren af denne artikel ud til at være under stadig debat - IPv6 er en skydeskive i bevægelse, som det hedder i en kritik af rfc4941 (www.breakingpointsystems.com/resources/blog/6-surprising-facts-about-ipv6/)

I de kommende numre af NYT vil vi fortsætte analyse og erfaringer med IPv6 og sammenligne med kendte teknikker fra IPv4. Det undrer nok ikke erfarne system-administratorer, at der er masser af problemer at tage fat på!

Donald Axel er redaktør på dkuug-NYT og har været aktiv i foreningen siden 2004, det seneste år som redaktør af bladet. Axel har været systemadministrator på RUC og har tidligere arbejdet med programmering af både systemprogrammel, netværk og som kursusinstruktør og meget andet.

Jon Bendtsen er kandidat fra DIKU

Glem hammeren, gør det rigtige

Af jon.bendtsen@jonix.dk

Som regel er pakkesystemet i Debian fantastisk, men nogle gange så støder man ind i dets rigide regler. Fx. Så stod jeg i slutningen af bearbejdningen af videoen optaget ved Kenneth Geisshirts foredrag om Apache Hadoop d. 21 august 2012, med en dependency der pegede bagud på en ældre udgave af et library.

Versionsuoverensstemmelser.

Jeg ville gerne have nyeste udgave af programmet MP4Box der er en del af projektet GPAC[1], for Debian multimedia projektet[2] tilbyder kun 0.4.6~svn20101218-0.0 og det manglede noget funktionalitet. Den nyeste udgave af GPAC er på nuværende tidspunkt version 0.5.0, og GPAC projektet tilbyder endda en Debian pakke så det burde bare lige være til at installere.

Problemet jeg rendte ind i var at at gpac_0.5.0_amd64.deb kræver libdirectfb-1.2-0 og jeg havde altså libdirectfb-1.2-9. Når der er dependency issues, så ved man aldrig om det vil kunne lade sig gøre at køre programmet eller ej, men ofte er biblioteker bagud kompatible, så når GPAC manglede version 1.2.0 og jeg havde 1.2.9, så forventede jeg at det ville virke efter en installation.

Så længe det kun er nogle få pakker **og** disse pakker ikke er essentielle systempakker, så har jeg ingen problemer med at tvinge installationen igennem med **dpkg --force-depends-version** og GPAC blev da også installeret, og virkede endda.

Apt-get brokker sig hele tiden

Problemet med denne tvang, er at hver evig eneste gang jeg skal installere eller upgradere et nyt program, så kommer apt-get hver evig eneste gang og brokker sig over den manglende dependency på libdirectfb-1.2-0, og yderligere så er dens eneste forslag **apt-get -f install** hvilket afinstallerer GPAC fordi apt-get ikke kan finde libdirectfb-1.2-0.

Okay, det er relativt nemt at installere GPAC på ny efter hver upgrade, men man bliver altså træt af det, og heldigvis så kan man da løse problemet permanent.

Ret i dependencies

Det er heldigvis ret nemt at rette denne type fejl permanent. I løbet af 5 trin har du en ny .deb pakke der virker:

1. `mkdir tmpfixingdir`
2. `dpkg-deb -x\ gpac_0.5.0_amd64.deb\ tmpfixingdir/`
3. `dpkg-deb --control\ gpac_0.5.0_amd64.deb\ tmpfixingdir/DEBIAN`
4. `vi\ tmpfixingdir/DEBIAN/control`

```
jonbendtsen@media:~$ sudo dpkg -i libgpac-dev_0.5.0_amd64.deb gpac_0.5.0_amd64.deb
(Reading database ... 45235 files and directories currently installed.)
Preparing to replace libgpac-dev 0.5.0 (using libgpac-dev_0.5.0_amd64.deb) ...
Unpacking replacement libgpac-dev ...
Selecting previously deselected package gpac.
Unpacking gpac (from gpac_0.5.0_amd64.deb) ...
dpkg: dependency problems prevent configuration of gpac:
 gpac depends on libdirectfb-1.2-0; however:
  Package libdirectfb-1.2-0 is not installed.
dpkg: error processing gpac (--install):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of libgpac-dev:
 libgpac-dev depends on gpac; however:
  Package gpac is not configured yet.
dpkg: error processing libgpac-dev (--install):
 dependency problems - leaving unconfigured
Processing triggers for man-db ...
Errors were encountered while processing:
 gpac
 libgpac-dev
```

1 http://gpac.wp.mines-telecom.fr/
2 http://deb-multimedia.org/

Debian bruger heldigvis tekst filer til konfiguration, så det eneste du nu skal gøre er at finde linien der starter med ordet: "**Depends:**" og så skal du finde segmentet hvor der står **libdirectfb-1.2-0** og erstatte **0** med **9** så der kommer til at stå **libdirectfb-1.2-9**. Derefter gemmer du og afslutter din editor.

```
5. dpkg -b tmpfixingdir/\
   fixed.gpac_0.5.0_amd64.deb
```

Og denne nye .deb pakke installere helt uden problemer af nogen art:

Så glem hammeren, og gør det rigtigt, så tingene passer sammen som de skal.


```
jonbendtsen@media:~$ sudo dpkg -i fixed.gpac_0.5.0_amd64.deb libgpac-dev_0.5.0_amd64.deb
(Reading database ... 45291 files and directories currently installed.)
Preparing to replace gpac 0.5.0 (using fixed.gpac_0.5.0_amd64.deb) ...
Unpacking replacement gpac ...
Preparing to replace libgpac-dev 0.5.0 (using libgpac-dev_0.5.0_amd64.deb) ...
Unpacking replacement libgpac-dev ...
Setting up gpac (0.5.0) ...
Processing triggers for man-db ...
Setting up libgpac-dev (0.5.0) ...
```

(Fortsat fra side 3:)

Arrangementer, foredrag i Symbion (DKuugs hjemsted) - Fruebjergvej 3, 2100 København Ø, ved Ryparken Station eller Emdrup Station

Torsdag d. 22. november kl.19: Reverse engineering talk. (Bryan Østergaard)

Et foredrag for de mere tekniske blandt nørderne, eller for nysgerrige folk, der gerne vil have et kort indblik i nogle teknikker til at kigge inde i binære programmer.

Foredraget giver en introduktion til emner som dynamic binary instrumentation, dynamisk patching af programmer og brugen af emulatorer til reverse engineering. Der vil også blive gennemgået nogle praktiske eksempler på hvad disse teknikker kan bruges til for eksempel i forbindelse med IT sikkerhed.

De forskellige teknikker bliver præsenteret på en måde der ikke kræver forudgående kendskab til reverse engineering eller programmering.

Tirsdag d. 18. december kl.19: Hvordan brydes kryptografi?

Et sjovt foredrag der giver et basalt indblik i nogle af de metoder der bruges til at bryde krypteret data. Der bliver vist praktiske måder at bryde mange simple krypteringsalgoritmer men der vil også blive gennemgået nogle svagheder ved moderne algorithmer som for eksempel RSA.

Kristen Nielsen har lovet at finde sted i Aarhus og dato, så Bryan Østergaard kan holde C++ foredrag.

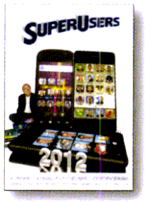
Kom og få dine kompetencer plejet - hold et foredrag om det, der interesserer dig (og os).

Vores kontor i Symbion er i orden nu; vi nu kan have gruppe-arbejde og afholde små kurser og workshops, både dag og aften. Skriv til pr@dkuug.dk eller til bestyrelsen i DKUUG, bestyr@dkuug.dk, og hør om lokalet er ledigt den dag du vil arrangere et møde. Der er hurtig internetforbindelse, både wired og wireless.

DKUUG

SUPERUSERS

Bestil kurser og katalog på
T: 48 28 07 06
W: superusers.dk
E: super@superusers.dk



Android kurser i Udvikling & Administration

Udvikling

LX-901 Android Programmering Grundkursus

Varighed: 2 dage
Pris: Kr. 8.900,- ekskl. moms. Inkl. Android Tablet.

Forudsætninger:
Kendskab til objektorienteret programmering.

Beskrivelse:
Du lærer at udvikle apps til Android og får et godt overblik over udviklingsmiljø og Google Play (tidl. Android Market) regelsæt. Du får udleveret en Android tablet som en del af kursusmaterialet.

Indhold bl.a.:
- Udviklingsmiljø (Eclipse, Android SDK, Java)
- Hvor og hvordan skriver man kode
- Opbygning af brugergrænseflade
- Activities/Intents
- Test af apps i eget device (udleveret tablet)
- Hvordan kommer man i Google Play (tidl. Android Market)?

JUL	AUG	SEP	OKT	NOV	DEC	JAN	FEB	MAR	APR	MAJ	JUN
12-13	16-17	13-14	11-12	8-9	13-14	10-11	7-8	7-8	2-3	7-8	6-7

LX-902 Android Programmering Workshop

Varighed: 2 dage
Pris: Kr. 7.400,- ekskl. moms.

Forudsætninger:
LX-901 Android Programmering Grundkursus.

Beskrivelse:
Vi designer/udvikler app, som kan gemme/ hente data lokalt, både i filer samt i SQL-database. Videre til netværksforbindelser og kommunikation med web-services.

Applikationen indeholder bl.a.:
- Professionel brugergrænseflade.
- Datahåndtering i apps: Filer, databaser og netværk.
- Web-services fra app: Data fra og til web-services.

JUL	AUG	SEP	OKT	NOV	DEC	JAN	FEB	MAR	APR	MAJ	JUN
12-13	13-14		15-16		17-18		11-12		4-5		10-11

LX-905 Android Programmering Videregående

Varighed: 2 dage
Pris: Kr. 7.400,- ekskl. moms.

Forudsætninger:
LX-902 Android Programmering Workshop.

Beskrivelse:
Vi arbejder med apps som anvender de mere avancerede Android-funktionaliteter (adgang til services og hardware).

Indhold bl.a.:
- Håndtering af kontakter og kalender fra app
- GPS og maps integreret i app
- Tilgang til foto/video-kamera og lyd
- Diverse sensorer og interface til disse
- Brug af notifications og meget mere

JUL	AUG	SEP	OKT	NOV	DEC	JAN	FEB	MAR	APR	MAJ	JUN
2-3•30-31		27-28		26-27			21-22		22-23		17-18

Support & Management

LX-920 Android Support

Varighed: 2 dage
Pris: Kr. 8.900,- ekskl. moms. Inkl. Android Tablet.

Forudsætninger: Generelt IT-kendskab
Beskrivelse: For personer som skal udføre first level support (konfiguration/fejlfinding) for virksomhedens medarbejdere på Android-enheder. Vi gennemgår aktivering, opdatering, synkronisering og konfiguration af Android smart-phones/tablets. Hvilke fejl opstår typisk, og hvordan afhjælper og forebygger man problemerne?

Indhold bl.a.:
- Aktivering, opdatering, backup, synkronisering og konfiguration af:
  - Wi-Fi & VPN samt Notification håndtering
  - Lokaltjenester, Netværk og Bluetooth
  - Google services og Tredjeparts-produkter
  - E-mail konti, kontakter, kalendere ...

JUL	AUG	SEP	OKT	NOV	DEC	JAN	FEB	MAR	APR	MAJ	JUN
	30-31		25-26		20-21	21-22		14-15		21-22	

LX-921 Android Management

Varighed: 2 dage
Pris: Kr. 7.400,- ekskl. moms.

Forudsætninger: LX-920 Android Support

Beskrivelse: For personer som skal udføre management (central styring) af Andorid smartphones / tablets fra server-side i virksomheden: Hvorledes integrerer man Android-enheder i sit øvrige enterprise-netværk? Hvilke teknologier og værktøjer findes, når man fra centralt hold skal holde styr på sine Android-enheder i virksomheden..

Indhold bl.a.:
- Exchange Active Sync (EAS)
- Central konfiguration af:
  - E-mail konti samt Kalendere og kontakter
  - Wi-Fi, VPN og alternativer samt Remote wipe (sletning)
  - Restrictions for den enkelte enhed og meget mere
- Google og 3. parts-services og -værktøjer

JUL	AUG	SEP	OKT	NOV	DEC	JAN	FEB	MAR	APR	MAJ	JUN
9-10		10-11		8-9				25-26		30-31	

Karlebogaard: Karlebovej 91 • 3400 Hillerød **&** **Kampehøjgaard:** Krajbjergvej 3, Vorre • 8541 Skødstrup
Tlf.: 48 28 07 06 • **Mail:** super@superusers.dk • **Web:** www.superusers.dk