Urs-Jakob Rüetschi, Xilab

Disk Usage Accounting

A Simple Solution for Linux Systems

Xilab bietet gegen Entgelt Speicherplatz im Internet an. Das Entgelt des Kunden richtet sich nach dem tatsächlich belegten Speicherplatz auf den Xilab-Servern. Die Belegung muss daher für jeden Kunden regelmässig erhoben, periodisch konsolidiert und in Rechnung gestellt werden.

Funktionsweise

Die Disk Usage jedes Kunden wird täglich erhoben und in einem Logfile abgelegt. Die Speicherbelegung wird dabei auf ganze MegaBytes gerundet. Bei der Rechnungstellung wird der durchschnittliche Speicherbedarf pro Monat ausgewiesen und verrechnet. Der Durchschnitt wird mit der exakten Anzahl Tage (28, 29, 30 oder 31) des Monates berechnet.

Die Erhebung des belegten Speicherplatzes erfolgt automatisch durch den Server. Der Durchschnittswert pro Monat wird durch ein Skript vorgenommen, welches manuell gestartet wird. Sein Output wird per Copy/Paste in die Rechnung (oder in ein Excel Sheet) eingefügt.

Messung der Disk Usage

Idealerweise wird ein File System mit Quota-Support verwendet, weil dieses die Buchführung der durch einen User und/oder eine Gruppe belegten Speicherplatzes automatisch vornimmt. Steht kein solches File System zur Verfügung, kann das Standard Unix Tool **du**(1) verwendet werden, welches von einem oder mehreren Startpunkten aus rekursiv durch den Verzeichnisbaum geht und den belegten Disk Space aufsummiert.

In beiden Fällen ist der tägliche Wert zusammen mit dem Datum in ein Logfile zu schreiben. Bei der Verwendung von **du**(1) ist zu bedenken, dass das eine "teure" Operation ist und entsprechend viel Zeit beanspruchen kann.

Implementation

Das Shell Skript `log-disk-usage.sh` wird jede Nacht und für jedes Kundenverzeichnis */pfad/kunde* durch den **cron**-Dienst aufgerufen wird. Das Kundenverzeichnis wird dem Skript als Parameter übergeben. Die gefundene Disk Usage wird zusammen mit einer Datumsangabe an die Datei */pfad/kunde*`.du` angehängt.

Skriptaufruf: `/root/bin/log-disk-usage.sh` */pfad/kunde*
Speicherort des Logfiles: */pfad/kunde*`.du`
Zeilenformat im Logfile: *yyyy mm dd Mbytes*

Die */dirname*`.du`-Dateien wachsen beständig und werden (gegenwärtig) nicht rotiert.

Vor der Rechnungstellung werden die `.du`-Dateien mit dem Skript `eval-du-log.awk` konsolidiert, d.h., die täglichen Werte zu einem Wert pro Monat aggregiert.

Skriptaufruf: `/root/bin/eval-du-log.awk` [`free=`N] [`cost=`K] */pfad/kunde*`.du`

Dabei ist `free=`N die Anzahl MBytes, die im Grundpreis eingeschlossen sind (default: $N = 0$) und `cost=`K der Preis pro MByte (default: $K = 1$). Der Output ist eine Zeile pro Monat im Format "*yyyy Monat Mbytes Cost*," wobei *Mbytes* auf drei Stellen nach dem Komma und *Cost* auf zwei Stellen ausgegeben wird. Beispiel:

```
2007 Dezember 58.800 58.80
2008 Januar 120.000 120.00
2008 Februar 570.167 570.17
```

Dieser Output kann per Copy/Paste in eine Tabelle oder direkt in das Rechnungsdokument befördert werden.

Das Logging-Skript

Das Logging ist als einfaches Shell-Skript `log-disk-usage.sh` realisiert:

```
#!/bin/sh

# Usage: log-disk-usage.sh <dir>
# To be called regularly by cron.

target="$1" || exit 127
log="${target%/}.du"
today=‘date ’+%Y %m %d’‘ || exit 1
set -- ‘du -s "$target"‘ && blocks=$1 || exit 1
echo $today $(((blocks+512)/1024)) >> "$log"
```

Auf der letzten Zeile wird die Anzahl Disk Blocks (Linux: 1024 Bytes) von **du**(1) auf MBytes gerundet. Hier ein Beispiel des generierten Logfiles:

```
⋮
2008 01 28 68
2008 01 29 73
2008 01 30 72
2008 01 31 72
2008 02 01 123
2008 02 02 125
2008 02 03 125
2008 02 04 144
⋮
```

Das Konsolidierungs-Skript

Die Konsolidierung zur durchschnittlichen monatlichen Belegung und die Bestimmung des Preises dafür erledigt das AWK-Skript `eval-du-log.awk`:

```
#!/usr/bin/awk -f

# Evaluate a <year> <month> <mday> <xbytes> disk usage log.
# Usage: awk -f eval-du-log [free=N] [cost=K] <logfile>

BEGIN { if (!cost) cost = 1
  mon["01"] = "Januar"; mon["02"] = "Februar"; mon["03"] = "Maerz"
  mon["04"] = "April"; mon["05"] = "Mai"; mon["06"] = "Juni"
  mon["07"] = "Juli"; mon["08"] = "August"; mon["09"] = "September"
  mon[10] = "Oktober"; mon[11] = "November"; mon[12] = "Dezember"
  print "#year month load cost"
}

{ mbytes = $4
  year = $1; month = $2
  tot[year,month] += mbytes
  cnt[year,month] += 1
}

END { for (s in tot) A[++n] = s
  isort(A, n) # sort by year then month
  for (i = 1; i <= n; i++) {
    split(A[i], a, SUBSEP)
    y = a[1]; m = a[2]
    d = tot[y,m]/cnt[y,m]
    c = (d - free) * cost
    printf "%s %s %.03f %.02f\n", y, mon[m], d, c
  }
}

# Insertion sort of A[1..n] (from AWK man page)
function isort(A, n,   i, j, hold)
{
  for (i = 2; i <= n; i++)
  {
    hold = A[j=i]
    while (A[j-1] > hold)
    { j--; A[j+1] = A[j] }
    A[j] = hold
  }
  # sentinel A[0] = "" will be created if needed
}
```

Im BEGIN-Block wird der Default-Wert für die *cost*-Variable explizit gesetzt (weil nicht 0) und ein Array aufgebaut, der von Monatsnummern (mit führender Null!) nach Monatsnamen übersetzt.

Der nächste Block wird für jede Zeile des Inputs ausgeführt. Er aktualisiert Totalwert und Anzahl in zwei assoziativen Arrays, die mit Jahr und Monat (mit führender Null!) indiziert werden.

Der END-Block kopiert den assoziativen Array *tot* in den Integer-indizierten Array *A*, welcher dann sortiert und schliesslich ausgegeben wird. Die Sortier-Routine wurde der Manual Page zu **mawk** entnommen.

Hinweise

Wenn in einem Logfile Einträge fehlen weil das Logging-Skript nicht ausgeführt wurde (z.B. weil der Rechner nicht lief), dann wird bloss über die vorhandenen Einträge gemittelt, was immer noch zu einem gültigen Resultat führt.

Hier vorgestellt wurde eine einfache Lösung für ein einfaches Problem. Sollte das Problem grösser werden (mehr Kunden!), muss diese Abrechnungslösung überdacht und angepasst werden.

Created 2008-02-26 by Urs-Jakob Rüetschi