Test Laufzeitverhalten SmartVis

Inhalt

1. Einleitung

Die Kernaufgabe der Anwendung SmartVis ist die Visualisierung von aufgezeichneten Smart Meter-Messwerten. Um eine Einschätzung zum Laufzeitverhalten der Anwendung auf einem Raspberry Pi 2 zu bekommen, werden Testdaten in größerem Umfang (ca. 2 Mio. Messwerte) generiert und anschließend die Antwortzeiten der SmartVis-Anwendung bei diesen Rahmenbedingungen analysiert.

Aktuell erfolgt die Datenhaltung für SmartVis mit einer MySQL-Datenbank. Im Rahmen der Tests soll auch überprüft und verglichen werden, welches Laufzeitverhalten eine PostgreSQL-Datenbank aufweist.

In diesem Dokument wird die Vorgehensweise zum Test des Laufzeitverhaltens beschrieben.

2. PostgreSQL - Installation und Konfiguration

Der Datenbank-Server PostgreSQL wird auf dem Pi2 mit dem Packet-Manager installiert (siehe Listing 1).

```
root@jrz-smartvis:~# apt-get install postgresql-9.1
. . .
Adding user postgres to group ssl-cert
Building PostgreSQL dictionaries from installed myspell/hunspell packages...
Setting up postgresql-9.1 (9.1.18-0+deb7u1) ...
Creating new cluster (configuration: /etc/postgresql/9.1/main, data:
/var/lib/postgresql/9.1/main)...
Moving configuration file /var/lib/postgresql/9.1/main/postgresql.conf to
/etc/postgresql/9.1/main...
Moving configuration file /var/lib/postgresql/9.1/main/pg_hba.conf to
/etc/postgresql/9.1/main...
Moving configuration file /var/lib/postgresql/9.1/main/pg_ident.conf to
/etc/postgresql/9.1/main...
Configuring postgresql.conf to use port 5432...
```

```
update-alternatives: using /usr/share/postgresql/9.1/man/man1/postmaster.1.gz to provide
/usr/share/man/man1/postmaster.1.gz (postmaster.1.gz) in auto mode
[ ok ] Starting PostgreSQL 9.1 database server: main.
```

Listing 1: Installation PostgreSQL auf Pi 2

Die Client-Zugriffsberechtigungen auf die PostgreSQL-Datenbank ist in der Datei /etc/postgresql/9.1/main/pg_hba.conf zu editieren/konfigurieren (siehe http://www.postgresql.org/docs/9.0/static/auth-pg-hba-conf.html). Die Anmeldung mit dem DB-Benutzer „postgres“ wird ermöglicht, indem die Berechtigungsoption „trust“ gesetzt wird. (Dadurch kann sicher jeder Benutzer auf dem Pi2 mit „postgres“ auf der DB anmelden). Zusätzlich muss mit dem host-Eintrag der ferne Zugriff ermöglicht werden (siehe Listing 2).

```
local   all             postgres                                trust

host    all             all             all             trust
```

Listing 2: pg_hba.conf - auth-Einstellungen

In der Datei /etc/postgresql/9.1/main/postgresql.conf sind Verbindungseinstellungen zu hinterlegen (siehe Listing 3 bzw. http://www.postgresql.org/docs/9.1/static/runtime-config-connection.html)

```
listen_addresses = '*'
. . .
port = 5432
```

Listing 3: postgresql.conf – connection settings

Die Änderungen der Einstellungen werden durch Neustart der Datenbank wirksam (siehe Listing 5).

```
sudo service postgresql restart
```

Listing 4: Neustart PostgreSQL-Server

Im nächsten Schritt wird eine DB-Clientsitzung mit „psql –U postgres“ gestartet und der SmartVis-Benutzer „fhs“ für den DB-Zugriff erstellt. Weiters wird die SmartVis-Datenbank „db_meters“ erstellt und der Benutzerzugriff für „fhs“ eingerichtet (siehe Listing 5).

```
postgres=# CREATE USER fhs WITH PASSWORD 'fhs';
postgres=# ALTER USER fhs CREATEDB;

postgres=# CREATE DATABASE db_meters
postgres=# GRANT ALL ON DATABASE db_meters TO fhs;
```

Listing 5: Einrichten SmartVis-Benutzer „fhs“ und –Datenbank “db_meters“

Damit ist die Installation und Konfiguration der PostgreSQL-Datenbank für SmartVis abgeschlossen.

3. PostgreSQL – Anpassungsbedarf SmartVis

Um die PostgreSQL-Datenbank in der SmartVis-Anwendung verwenden zu können, sind einige Anpassungen durchzuführen:

- Datenbank-Treiber für PostgreSQL in Projekt hinterlegen (Dependency in pom.xml eintragen)
- Erstellung SmartVis-Datenbankobjekte in PostgreSQL. Die Datenbanken MySQL und PostgreSQL weisen (trotz SQL-Standards) in einigen Punkten syntaktische Unterschiede auf. Die SQL-Anweisungen zur Datenbank-Erstellung sind daher zu adaptieren.
- JDBC-Zugriff in den „application.properties“ entsprechend konfigurieren (siehe Listing 6).

```
#DB-Properties for DB1 on PostgreSQL
db1.driver=org.postgresql.Driver
db1.jdbcPrefixAndVendor=jdbc:postgresql
db1.address=jrz-smartvis
db1.port=5432
db1.name=db_meters
db1.username=fhs
```

```
db1.password=fhs
db1.mode = gateway
```

Listing 6: SmartVis – application.properties für JDBC-Zugriff auf PostgreSQL-Datenbank

4. Testdaten auf Pi 2 übertragen und in DB einspielen

Für die Analyse und den Vergleich des Laufzeitverhaltens werden idente Testdaten in der MySQL- und PostgreSQL-Datenbank bereitgestellt. Für die Tests sind besonders die Messwerte in der Tabelle „meter_data“ von Interesse. Es werden REDD-Testdaten vom 1.1.2015 bis 31.12.2015 im 15-Sekunden-Takt unter der Meter-ID 1 erzeugt. Die Tabelle enthält (in jeder DB) 2.101.920 Datensätze.

Testdaten in MySQL einspielen:

```
root@jrz-smartvis:/# mysql -U root -p db_meters
mysql  Ver 14.14 Distrib 5.5.43, for debian-linux-gnu (armv7l) using readline 6.2
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

mysql> load data infile '/etc/smartvis/db_meters.meter_data.meter1.csv' INTO TABLE
meter_data FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n';
Query OK, 2101920 rows affected (6 min 42.58 sec)
Records: 2101920  Deleted: 0  Skipped: 0  Warnings: 0
```

Listing 7: Laden Testdaten in MySQL-Datenbanktabelle „meter_data“

Testdaten in PostgreSQL einspielen:

```
root@jrz-smartvis:/# psql db_meters -U postgres
psql (9.1.18)
Type "help" for help.

db_meters=# COPY meter_data FROM '/etc/smartvis/db_meters.meter_data.meter1.csv' DELIMITER
',' CSV;
COPY 2101920
```

Listing 8: Laden Testdaten in PostgreSQL-Datenbanktabelle „meter_data“

In der SmartVis-Anwendung erfolgen die Datenbank-Zugriffe auf die Tabelle „meter_data“ immer mit Angabe die Datenfelder „meter_id“ und „timestamp“. Um das Laufzeitverhalten für diese Zugriffe zu optimieren, werden (eindeutige) Indizes über diese Datenfelder erstellt (siehe Listing 9 und Listing 10).

```
ALTER TABLE meter_data ADD UNIQUE KEY (`meter_id`, `timestamp`);
```

Listing 9: Index-Erstellung MySQL

Index für effizienteren Zugriff auf meter_data (in PostgreSQL) hinzugefügt:

```
ALTER TABLE meter_data ADD UNIQUE (meter_id, timestamp);
```

Listing 10: Index-Erstellung PostgreSQL

5. Lasttest mit Apache JMeter

Apache JMeter ist ein freies, in Java geschriebenes Werkzeug zum Ausführen von Lasttests in Client/Server-Anwendungen. JMeter ermöglicht es mittels Zusammenstellen eines Testplanes zu spezifizieren, welche Teile der Anwendung durchlaufen werden sollen, um konkrete Ergebnisse über das Antwortzeitverhalten zu bekommen (siehe http://jmeter.apache.org/).

Für die Analyse des SmartVis-Laufzeitverhaltens wurde der Testplan „SmartVis Test Plan 01“ erstellt

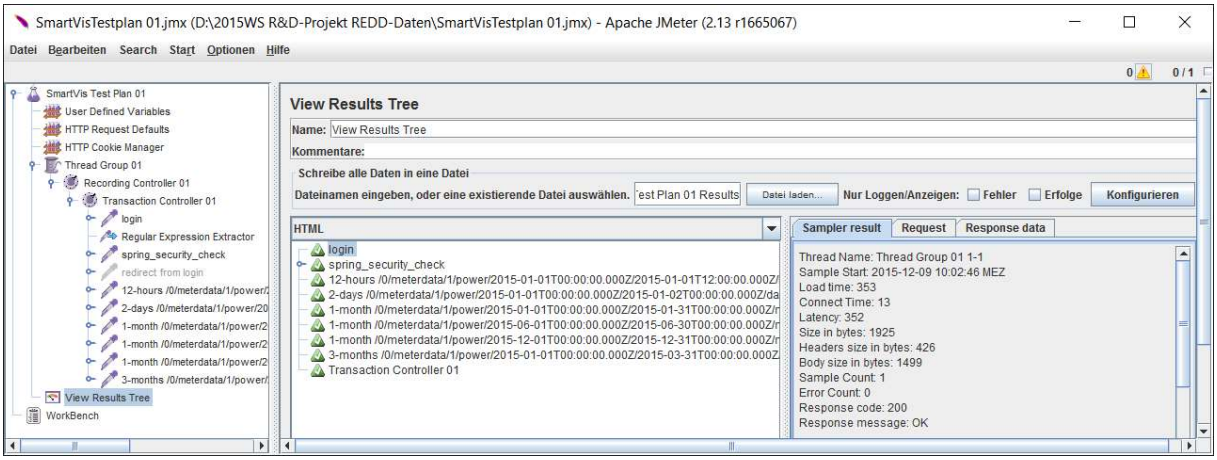

Abbildung 1: JMeter SmartVis Testplan 01

6. Laufzeitverhalten SmartVis: Vergleich MySQL- und PostgreSQL-Datenbank

In MySQL (in Datei /etc/mysql/my.cnf) wird das Attribut query_cache_size=0 (statt 16M) gesetzt, damit Tests nicht durch Cache-Ergebnisse beeinträchtigt werden. In PostgreSQL erfolgt (defaultmäßig) kein Caching der Query-Ergebnisse, die Standardeinstellungen nach der Installation wurden nicht verändert.

	Results MySQL			Results PostgreSQL		
Sample ID (1. Ausführung)	**Sample Start (Exec 1)**	**Load time (ms)**		**Sample Start (Exec 1)**	**Load time (ms)**	**Vergleich zu MySQL**
login	2015-12-08 07:18:38,182	466		2015-12-08 09:00:13,376	378	81,1%
spring_security_check	2015-12-08 07:18:38,667	2393		2015-12-08 09:00:13,758	2583	107,9%
12-hours /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-01T12:00:00.000Z/hour/1	2015-12-08 07:18:41,061	234		2015-12-08 09:00:16,345	445	190,2%
2-days /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-02T00:00:00.000Z/day/1	2015-12-08 07:18:41,298	436		2015-12-08 09:00:16,791	203	46,6%
1-month /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-31T00:00:00.000Z/month/1	2015-12-08 07:18:41,738	9386		2015-12-08 09:00:16,996	5362	57,1%
1-month /0/meterdata/1/power/2015-06-01T00:00:00.000Z/2015-06-30T00:00:00.000Z/month/1	2015-12-08 07:18:51,126	8886		2015-12-08 09:00:22,364	4966	55,9%
1-month /0/meterdata/1/power/2015-12-01T00:00:00.000Z/2015-12-31T00:00:00.000Z/month/1	2015-12-08 07:19:00,015	8038		2015-12-08 09:00:27,333	5095	63,4%
3-months /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-03-31T00:00:00.000Z/month/1	2015-12-08 07:19:08,055	72317		2015-12-08 09:00:32,433	20549	28,4%
Transaction Controller 01	2015-12-08 07:18:38,163	102156		2015-12-08 09:00:13,342	39581	38,7%
Sample ID (2. Ausführung)	**Sample Start (Exec 2)**			**Sample Start (Exec 2)**		
login	2015-12-08 08:19:34,457	602		2015-12-08 09:02:46,221	353	58,6%
spring_security_check	2015-12-08 08:19:35,066	2258		2015-12-08 09:02:46,582	2515	111,4%
12-hours /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-01T12:00:00.000Z/hour/1	2015-12-08 08:19:37,328	232		2015-12-08 09:02:49,101	192	82,8%
2-days /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-02T00:00:00.000Z/day/1	2015-12-08 08:19:37,562	344		2015-12-08 09:02:49,294	200	58,1%
1-month /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-31T00:00:00.000Z/month/1	2015-12-08 08:19:37,908	9380		2015-12-08 09:02:49,496	5018	53,5%
1-month /0/meterdata/1/power/2015-06-01T00:00:00.000Z/2015-06-30T00:00:00.000Z/month/1	2015-12-08 08:19:47,290	9091		2015-12-08 09:02:54,520	5545	61,0%
1-month /0/meterdata/1/power/2015-12-01T00:00:00.000Z/2015-12-31T00:00:00.000Z/month/1	2015-12-08 08:19:56,383	8053		2015-12-08 09:03:00,068	4586	56,9%
3-months /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-03-31T00:00:00.000Z/month/1	2015-12-08 08:20:04,440	72506		2015-12-08 09:03:04,654	15585	21,5%
Transaction Controller 01	2015-12-08 08:19:34,440	102466		2015-12-08 09:02:46,182	33994	33,2%

Tabelle 1: Laufzeitvergleich MySQL – PostgreSQL

7. Laufzeitverhalten SmartVis: MySQL mit Interpolation ohne Zählerwerte

Die Laufzeitergebnisse aus Kapitel 6 wurden mit einem SmartVis-Softwarestand erhoben, bei dem Aggregationsfunktionen der Datenbank (SUM, MAX, AVG, ..) zur Abfrage der SmartMeter-Werte verwendet wurden. Im Gegensatz dazu werden bei den Tests in diesem Kapitel (7) keine DB-Aggregierungen durchgeführt. Stattdessen werden alle Meterdaten des Betrachtungszeitraums in eine Liste in den Hauptspeicher ausgegeben und in einer Schleife über alle zu betrachtenden Intervalle werden dann die (linear interpolierten) Wert ausgegeben.

Sample Start (Exec 3)	Load time (ms)	Sample ID (mit Interpolation ohne Counter)	Response Code
2016-01-14 20:26:21,870	495	login	200
2016-01-14 20:26:22,366	2285	spring_security_check	200
2016-01-14 20:26:24,656	1660	12-hours /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-01T12:00:00.000Z/hour/1	200
2016-01-14 20:26:26,319	3431	2-days /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-02T00:00:00.000Z/day/1	200
2016-01-14 20:26:29,751	177883	1-month /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-31T00:00:00.000Z/month/1	200
2016-01-14 20:29:27,636	164510	1-month /0/meterdata/1/power/2015-06-01T00:00:00.000Z/2015-06-30T00:00:00.000Z/month/1	200
2016-01-14 20:32:12,148	250633	1-month /0/meterdata/1/power/2015-12-01T00:00:00.000Z/2015-12-31T00:00:00.000Z/month/1	200
2016-01-14 20:36:22,785	112164	3-months /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-03-31T00:00:00.000Z/month/1	500
2016-01-14 20:26:21,865	713061	Transaction Controller 01	

Tabelle 2: Laufzeitverhalten SmartVis mit MySQL und Interpolation

Die SmartVis-Anwendung bricht in diesem Test bei der Abfrage des 3-Monats-Intervalls mit folgendem Fehler ab: *"Internal Server Error: {""timestamp"":1452890293344,""status"":500,""error"":""Internal Server Error"",""exception"":""java.lang.OutOfMemoryError"",""message"":""Java heap space"",""path"":""/0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-03-31T00:00:00.000Z/month/1""}"*

Die großen Datenmengen führen zu einem Hauptspeicherüberlauf.

8. Laufzeitverhalten SmartVis: MySQL mit Interpolation mit Zählerwerte

Die Laufzeitergebnisse aus Kapitel 7 werden wiederholt, allerding werden in diesem Szenario Zählerwerte bei der Ermittlung von interpolierten Werten verwendet. In diesem Szenario wird zur Interpolation eines Intervall-Grenzwertes jeweils ein Datensatz vor und ein Datensatz nach der betrachteten Intervallgrenze zur linearen Interpolation gelesen.

Sample Start (Exec 4)	Load time (ms)	Sample ID (mit Interpolation mit Counter)	Response Code
2016-01-17 22:36:41,091	382	login	200
2016-01-17 22:36:41,476	2218	spring_security_check	200
2016-01-17 22:36:43,697	683	12-hours /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-01T12:00:00.000Z/hour/1	200
2016-01-17 22:36:44,382	168	2-days /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-02T00:00:00.000Z/day/1	200
2016-01-17 22:36:44,551	122	1-month /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-01-31T00:00:00.000Z/month/1	200
2016-01-17 22:36:44,674	96	1-month /0/meterdata/1/power/2015-06-01T00:00:00.000Z/2015-06-30T00:00:00.000Z/month/1	200
2016-01-17 22:36:44,770	95	1-month /0/meterdata/1/power/2015-12-01T00:00:00.000Z/2015-12-31T00:00:00.000Z/month/1	200
2016-01-17 22:36:44,865	171	3-months /0/meterdata/1/power/2015-01-01T00:00:00.000Z/2015-03-31T00:00:00.000Z/month/1	200
2016-01-17 22:36:41,086	3935	Transaction Controller 01	200

Tabelle 3: Laufzeitverhalten SmartVis mit MySQL und Interpolation und Counter-Werten

9. Fazit

Die erzielten Antwortzeiten sind bei großen Auswertungszeiträumen trotz der Verwendung von Indizes sehr hoch. Bei den Tests in Kapitel 7 führen die großen Datenmengen sogar zu einem Abbruch der SmartVis-Anwendung.

Aus dem Laufzeittest in Kapitel 6 geht hervor, dass die PostgreSQL-Datenbank in den meisten Fällen deutlich performanter als die MySQL-Datenbank ist.

Im Lauzeittest in Kapitel 8 werden interpolierte Werte nur an den Intervallgrenzen auf Basis von Zählerwerten ermittelt. In diesem Szenario hängt die Laufzeit nicht mehr von der Auswertungsperiode ab, sondern von der Anzahl der darzustellenden (und abzufragenden) Intervallen. Im Test-Setting ist diese Methode deutlich am schnellsten.