T. Rabl, M. Frank, H. Mousselly Sergieh, and H. Kosch
17.09.2010
Second TPC Technology Conference on Performance Evaluation & Benchmarking

A Data Generator for Cloud-Scale Benchmarking
Data sizes grow beyond petabyte barrier
  - Cloud computing to the rescue
  - Need for cloud-scale benchmarking
    - Need for realistic, cloud-scale data sets

Problems:
  - Generating petabytes
  - Storing petabytes
  - Transporting petabytes

Solution:
  - Parallel, on-site generation
- 3 tables with primary and foreign keys
- Non-uniform distributions (lognormal)
- Replicated data

- How to generate consistent data in parallel?
3 Classes of Generators

- No references
  - Only uncorrelated data
  - Simple statistical references
- Scanned references
  - Read data to generate reference
  - Generate tuple pairs
- Computed references
  - Data is generated deterministically
  - Compute data to generate reference

Parallel Data Generation Framework
Deterministic Data Generation

- Parallel pseudo random number generator
  - Deterministic
  - Fast skip ahead
- Generation of values as a function
  - Deterministic
- Seeds + row id + generator allows recalculation
  - *Every value can be computed independently*
Data Generation Example

- Seeding strategy
- All seeds can be cached
- Generation of value in row n with n-th random number
- Reference generation
- Lognormal indexes row_id of user
- Equal seeds and lognormal generator for user_id
PDGF – Architecture

- Java based
- Plug-in concept
  - Generators
  - Distributions
  - RNGs
  - Output
  - Scheduler
- TPC-H plug-in
Configuration

- XML file
- Reflects SQL schema
  - Tables
  - Fields
- Seed
- Size
- Scale factor
- Output

```
<project name="simpleUserSeminar">
  ...
<table name="seminar_user">
  <size>201754</size>
  <fields>
    <field name="user_id">
      <type>java.sql.Types.INTEGER</type>
      <reference>
        <referencedField>user_id</referencedField>
        <referencedTable>user</referencedTable>
      </reference>
      <generator name="DefaultReferenceGenerator">
        <distribution name="LogNormal">
          <mu>7.60021</mu><sigma>1.40058</sigma>
        </distribution>
      </generator>
    </field>
    <field name="degree_program">
      ...
    </field>
  </fields>
</table>
```

A Data Generator for Cloud-Scale Benchmarking - Tilmann Rabl
Parallel Data Generation

- Every value can be computed independently
- No communication
- Workload distribution
  - Configuration for every node
  - Automatic distribution

<?xml version="1.0" encoding="UTF-8"?>
<nodeConfig>
  <nodeNumber>1</nodeNumber>
  <nodeCount>2</nodeCount>
  <workers>2</workers>
</nodeConfig>
Evaluation

- 16 node HPC cluster
  - 2 Intel Xeon QuadCore processors
  - 16GB RAM
  - 2 x 74GB HDD, RAID 0
- SetQuery data set
  - 1 table “Bench”, 21 columns
  - 12 Random numbers, 8 strings
- TPC-H data set
  - 8 tables, 61 columns
  - Data types: integer, char, varchar, decimal, date

A Data Generator for Cloud-Scale Benchmarking - Tilmann Rabl
- **SetQuery data set**
- **100 GB data set (SF 460)**
- **1 to 16 nodes**
TPC-H data set
- Data sizes: 1 GB, 10 GB, 100 GB
- Single node (8 cores)
Conclusion

- Parallel data generation framework
- Linear speed up
- Fast generation
  - Large data sets
  - Realistic data
- Independent generation of tables / columns / values
- Easy configuration and extension
Future Work

- More implementation
  - generators, distributions
  - Benchmarks
  - Graphical user interface
  - Scheduler
- Query generator
  - Consistent inserts, updates, deletes
  - Precomputed query results
  - Time series
That's the problem with randomness: you can never be sure.

Are you sure that's random?

Nine nine nine nine nine nine.

Tour of Accounting: Over here we have our random number generator.
SetQuery data set
  - Data sizes: 220 MB, 2.2 GB, 10 GB, 22 GB, 100 GB
  - Single node
TPC-H Generation Speed

- TPC-H data set
  - Data sizes: 1 GB, 10 GB, 100 GB, 1TB
  - 1, 10, 16 nodes
Real Data
Time Series in Workloads

Daily Website Accesses 24 October 08 – 11 June 09