Parallel Data Generation for Performance Analysis of Large, Complex RDBMS

Tilmann Rabl and Meikel Poess
Presented by Mohammad Sadoghi

Agenda

- **Motivation**
- Data generation for DBMS benchmarking
- Classification of data dependencies
- Generation of data dependencies
- Conclusions

Motivation

- Testing performance of today's data management systems is becoming increasingly difficult:

1. Data growth rate
2. System complexity
3. Data complexity

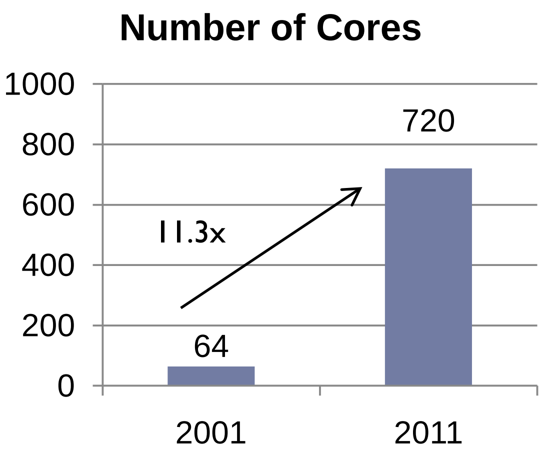
Data Growth Rate

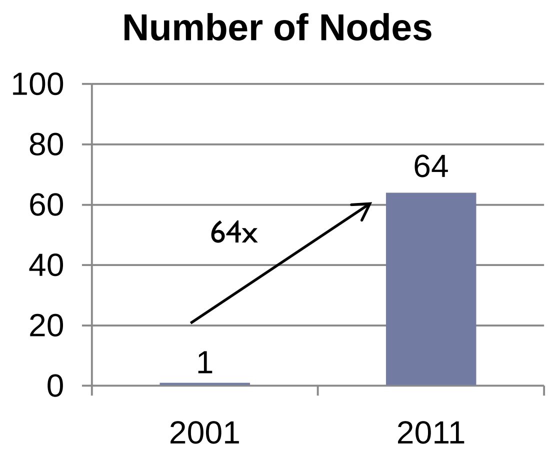
- Amount of data kept in today's systems is growing exponentially:
  - Companies retain more data for a longer period of time
    - For legal purposes
    - For accounting purposes
    - To gain more insight into their business
  - Social media sites collect personal information at a rapid pace *
    - Facebook data 2007 15 TBytes
    - Facebook data 2010 700 TBytes
  - It is all possible, because hardware is cheap and powerful
    - Hard drives, CPUs, etc.

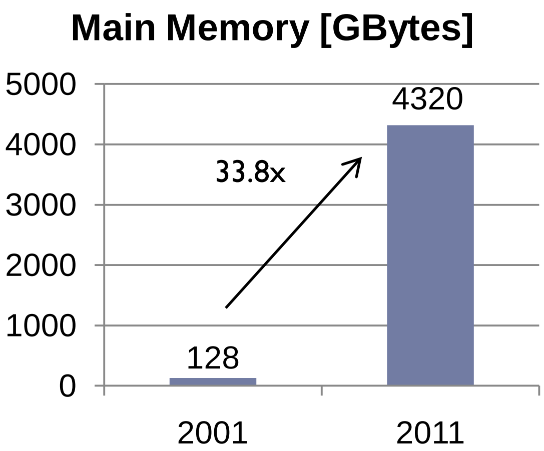
*Thusoo et al. *Hive - a petabyte scale data warehouse using Hadoop*. ICDE 2010: 996-1005

System Complexity

- Dramatic increase in hardware used in TPC-H benchmarks between 2001 and 2011:







Data Complexity

- Systems capture more sophisticated data
  - Number of tables
  - Number of columns
  - Data dependencies
- For performance reasons systems store data with dependencies:
  - Foremost seen in de-normalized data warehouse schemas,
  - But also in OLTP systems

Data Generation Requirements for DBMS Benchmarking

1. Generate Petabytes of data

2. Generate data in parallel
   - Across hundreds of physical nodes
   - Across multiple CPU/cores

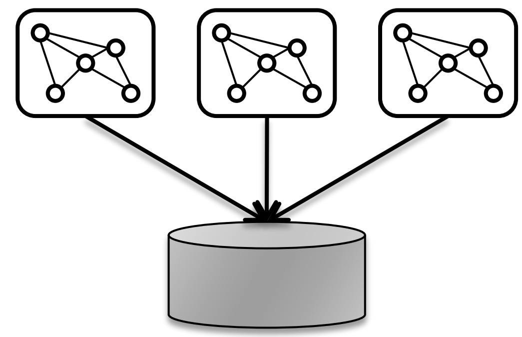
3. Able to generate complex data deterministically
   - Various interdependencies
   - Repeatable generation

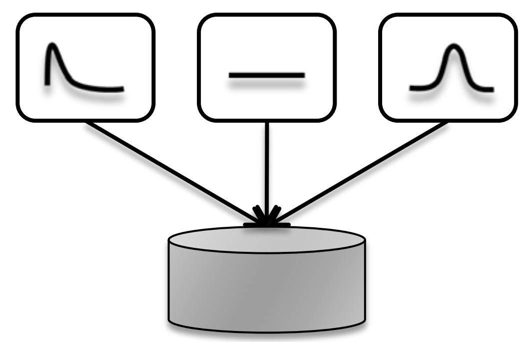
Agenda

- Motivation
- **Data generation for DBMS benchmarking**
- Classification of data dependencies
- Generation of data dependencies
- Conclusions

Methods of Data Generation

- Application specific
  - Implementation overhead
  - Limited adaptability
  - Fast outdated
- Client simulation
  - Graph based
  - Very accurate (complex dependencies)
  - Slow
  - Limited repeatability
- Statistical distributions
  - Based on probability
  - Fast
  - Repeatable
  - Based on random numbers





Random Number Generation

- *Pseudo* random numbers
  - Fast
  - Repeatable
- Linear random number generation
  - High quality random numbers
  - rng(n) = lrng(lrng(…(lrng(seed))…))
- Parallel random number generation
  - Fast random numbers
  - Random hash *
  - rng(n) = prng(seed+n)

```
x := 3935559000370003845 * i
   + 2691343689449507681 (mod 2^64)
x := x xor ( x right-shift 21)
x := x xor ( x left-shift 37)
x := x xor ( x right-shift 4)
x := 4768777513237032717 * x (mod 2^64)
x := x xor ( x left-shift 20)
x := x xor ( x right-shift 41)
x := x xor ( x left-shift 5)
Return x
```


* Press et al. *Numerical Recipes – The Art of Scientific Computing*. 2007. Cambridge University Press.

Deterministic Data Generation

- Exploits determinism in random number generation
  - Seed determines random sequence
  - Every value can be re-calculated
- Generic data generator
  - Parallel Data Generation Framework (PDGF)
  - XML specification defines schema

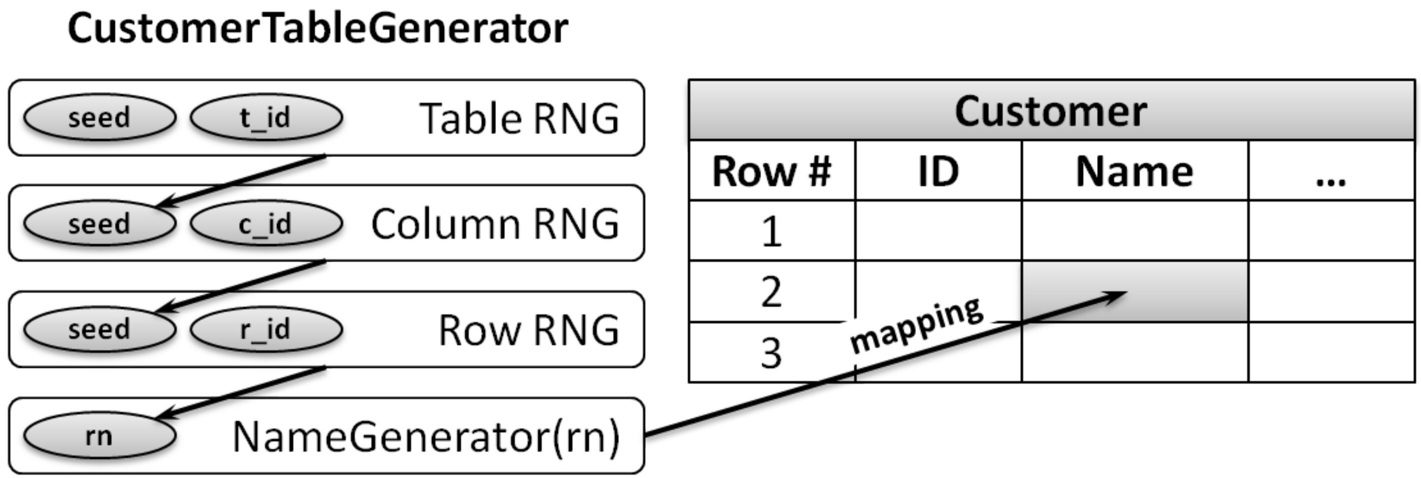
```
<schema name="warehouse">
 <scaleFactor name="custscale">5000</scaleFactor>
 <seed>1234567890</seed>
 <rng name="PdgfDefaultRandom" />
 <tables>
  <table name="Customer">
   <size>custscale</size>
   <fields>
    <field name="ID">
    <type>java.sql.Types.BIGINT</type>
    <generator name="IdGenerator" />
   </field> [...]
```

Data Generators in PDGF

- Data generators are functions
  - Domain: random values
  - Codomain: data domain
  - Same random number results in same value

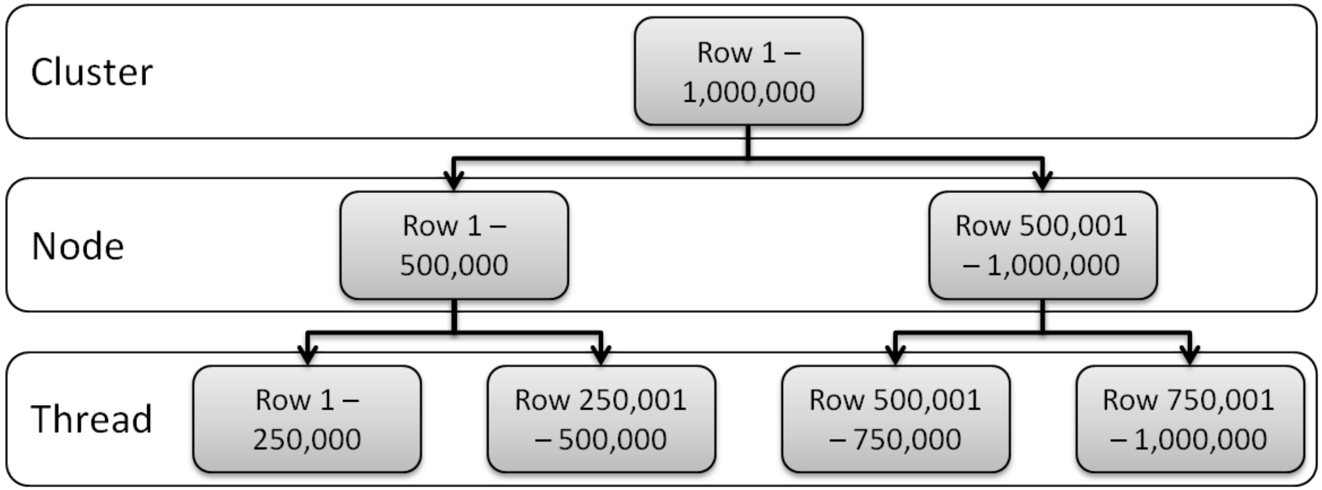
- Examples
  - Dictionary
    - Random number % row count
  - Number
    - Random number % range + offset
  - If multiple random numbers required
    - Random number is seed

Seeding Strategy



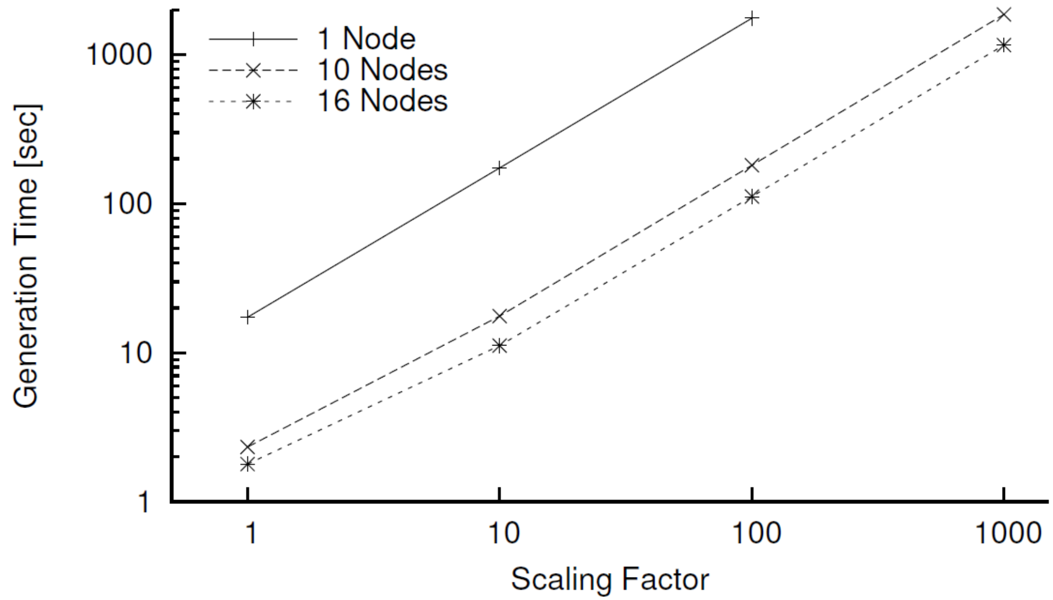
- Hierarchical seeding strategy
  - Schema → Table → Column → Row → Generator
  - Uses deterministic seeds
  - Guarantees that n-th random number determines n-th value
  - Even for large schemas all seeds can be cached
- Repeatable, deterministic generation

Parallel Data Generation



- Each field can be computed independently
- Allows for a static scheduling approach
- Supports horizontal partitioning of tables
- Results in linear speedup

TPC-H Generation Speed

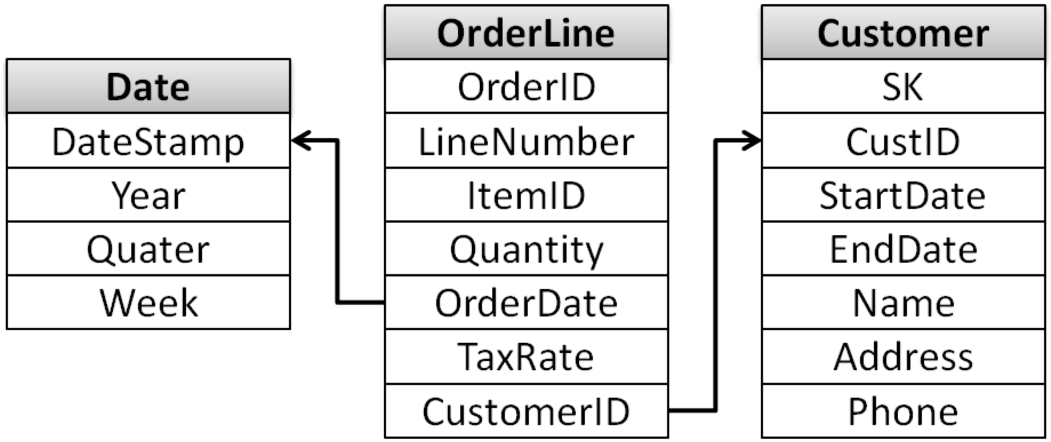


- 16 node HPC cluster
  - Each with 2 QuadCore, 2 HDDs, RAID 0
  - Total of 32 processors, 128 cores, 256 threads, 32 HDDs
- TPC-H data set
  - 1 GB, 10 GB, 100 GB, 1TB – 1, 10, 16 nodes
- Linear speedup, linear scale-out
- Fast, parallel data generation on modern hardware

Agenda

- Motivation
- Data generation for DBMS enchmarking
- **Classification of data dependencies**
- Generation of data dependencies
- Conclusions

Ongoing Example



- Represents a data warehouse scenario
- Simplification of TPC-H / star schema
  - De-normalized dimensions
- Can grow to enormous sizes
  - E.g. largest TPC-H result: 30,000 GBytes of raw data
- Multiple data dependencies

Intra Row Dependency

Date			
DateStamp	**Year**	**Quarter**	**Week**
2011-03-30	2011	201101	2011W13
2011-03-31	2011	201101	2011W13
2011-04-01	2011	201102	2011W13

- Dependency between fields of a single row
- Common for different representations of the same data
- Other Examples:
  - VAT → zip code of purchase
  - City and state → zip code
- Functional dependency: *{DateStamp}* → *{Year,Quarter,Week}*

Intra Table Dependency

OrderLine				
OrderID	**LineNumber**	**ItemID**	**Quantity**	**...**
123	1	43	3	...
123	2	66	50	...
123	3	75	1	...

- Dependency between fields of different rows
- Simple example: surrogate key
- De-normalized fact table
  - Merge of orders and lineitems (e.g. TPC-C, TPC-H)
  - Multiple lineitems per order (between min and max)

Intra Table Dependency II

Customer					
SK	**CustID**	**StartDate**	**EndDate**	**Name**	**...**
1	234	03-01-09	01-02-10	Smith	...
2	123	04-04-04	NULL	Wilson	...
3	234	01-02-10	NULL	Smith	...

- Time related intra table dependency
- History keeping dimension
  - Stores the evolution of a dimension
  - Incrementing surrogate key
  - Multiple entries per CustID
  - Monotonic increasing StartDate per CustID
  - Matching EndDate and StartDate for successive entries per CustID

Intra Table Dependency III

Customer					
SK	**CustID**	**…**	**Name**	**Address**	**Telephone**
1	1	…	Smith	543 Arch	555-23
2	1	…	Smith	543 Arch	555-44
3	1	…	Smith	67 Second	555-23
4	1	…	Smith	67 Second	555-44
5	2	…	Wilson	2 Second	555-67

- Intra table dependency from multi-valued dependency (MVD)
- Usually poor schema design
  - Possibly intended by benchmark designer
- Multiple addresses and phone numbers per customer
- MVDs: {*CustID*} →→{*Address*} and {*CustID*} →→{*Telephone*}

Inter Table Dependency

Daily Quantity		
OrderDate	**CustID**	**Quantity**
2011-03-31	1	350
2011-03-31	2	150
2011-04-01	1	15

OrderLine			
...	**Quantity**	**OrderDate**	**CustomerID**
...	70	2011-03-31	2
...	65	2011-03-31	2
...	15	2011-03-31	2

- Dependency between fields of different tables
- Most common: referential integrity
  - Foreign key must exist
- Redundant data
- Additional data structures: materialized views
  - Aggregation of daily orders per customer

Agenda

- Motivation
- Data generation for DBMS benchmarking
- Classification of data dependencies
- **Generation of data dependencies**
- Conclusions

Intra Row Dependency Generation

- Intra row dependency
  - Affect only a single row
- Solution I
  - Recalculate values
- Solution II
  - Cache single row
  - Faster

Date			
DateStamp	**Year**	**Quarter**	**Week**
2011-03-30	2011	201101	2011W13
2011-03-31	2011	201101	2011W13
2011-04-01	2011	201102	2011W13

Intra Table Dependency Generation

- Surrogate key
  - Use row number

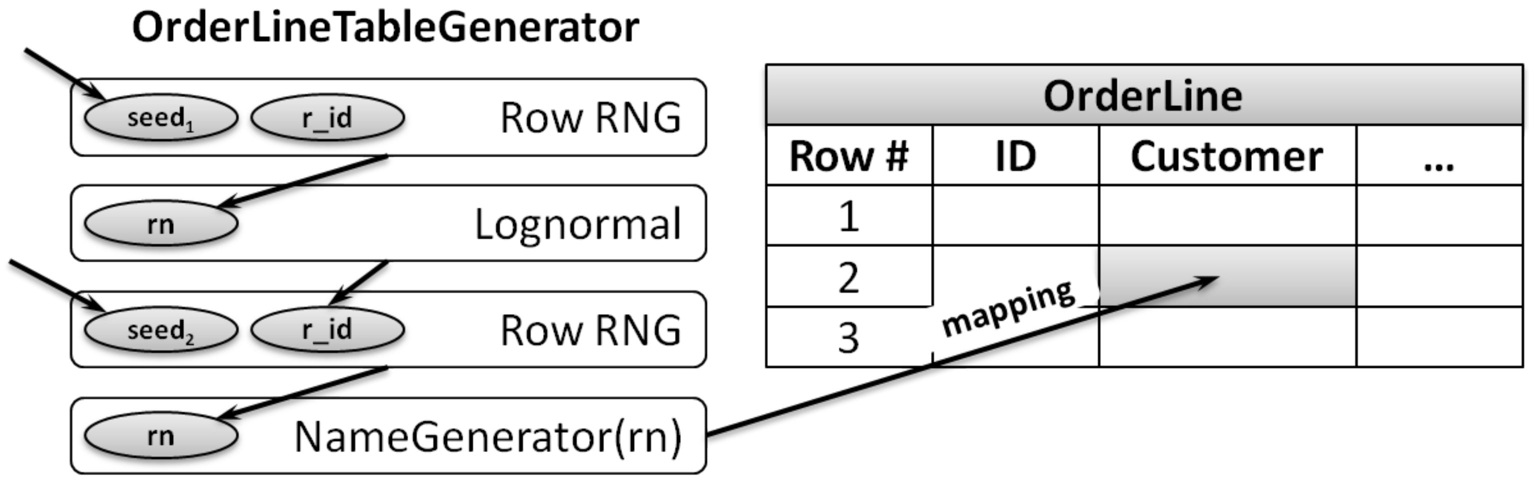
Customer					
SK	**CustID**	**StartDate**	**EndDate**	**Name**	**...**
1	234	03-01-09	01-02-10	Smith	...
2	123	04-04-04	NULL	Wilson	...
3	234	01-02-10	NULL	Smith	...

- Sorted data / time related dependency
  - Serial generation
  - Future work

Customer					
SK	**CustID**	**...**	**Name**	**Address**	**Telephone**
1	1	...	Smith	543 Arch	555-23
2	1	...	Smith	543 Arch	555-44
3	1	...	Smith	67 Second	555-23
4	1	...	Smith	67 Second	555-44
5	2	...	Wilson	2 Second	555-67

- Multi valued dependency
  - Generate multiple values at once

Inter Table Dependency Generation



- Reference Generation
  - Schema → Table → Column → Row → Row → Generator
  - Randomly pick a referenced row
  - Recalculate referenced value
  - Supports various distributions
- Aggregation
  - Recalculate multiple values

Agenda

- Motivation
- Data generation for DBMS benchmarking
- Classification of data dependencies
- Generation of data dependencies
- **Conclusions**

Conclusions

- Requirements of modern benchmark data generation
  - Large data, large systems, complex data
- Dependencies in relational data
  - Intra row, intra table, inter table
- Generic data generation
  - Parallel Data Generation Framework
  - Fast, parallel generation
  - Support for intra row and inter table dependencies
  - Some support for intra table dependencies
  - Currently evaluated by the TPC
- Future Work
  - Further dependencies
  - Implement additional intra table dependencies

Thank You!

- Questions?