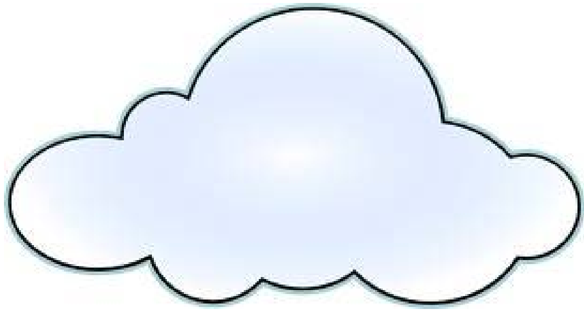
MemTest: A Novel Benchmark for In-memory Database

Qiangqiang Kang, Cheqing Jin, Zhao Zhang, Aoying Zhou

Institute for Data Science and Engineering,
East China Normal University, Shanghai, China

Outline

- Motivation & Introduction
- Metrics
- Data Model
- Workload
- Experiment
- Conclusions



Motivation

- The price of memory continues to decrease, making it possible to deploy a computer system with huge memory size
- Several commercial and open source providers have devoted huge resources to develop IMDBs (In-Memory Databases)
- The appearance of more and more IMDB products brings a need to devise a benchmark to test and evaluate them fairly and objectively

Database Benchmark

- The development of benchmark has shown great success in the past 30 years
  - For relational DBMS (RDBMS)
    - Wisconsin Benchmark, CH-Benchmark, Set query Benchmark
    - TPC-X series: TPC-C, TPC-E, TPC-H, TPC-DS, etc.
  - For object-oriented DBMS
    - OO7 and Bucky, etc
  - For XML data
    - XMARK and EXRT, etc
  - For big data applications
    - YCSB, YCSB++ and BigBench, etc

Some Properties in IMDBs

- Data Compression
  - In IMDBs, data is often stored in compressed form.
- Minimal Memory Space
  - Each IMDB has a request for the minimal memory space.
- CPU
  - Most conventional systems attempt to minimize disk access
  - IMDBs have overlooked I/O cost and focus more on the processing cost of CPU
- Cache
  - Cache is employed to reduce the data transfer between memory and CPU.

Components in MemTest

- Metrics
  - Compression ratio, response time, minimal memory space, CPU usage and cache miss.
- Data model
  - Based on an inter-bank transaction scenario
  - A star schema
    - one big table (>200 columns)
    - five small tables (<30 columns)
  - A new data generator
    - uniform and skew distribution
    - Scalable
- Workload
  - Covering business questions
  - OLAP & OLTP
    - 12 queries and 2 transactions

Comparison with existing works

Characteristic	SSB	TPC-C	TPC-H	TPC-DS	CH-Benchmark	MemTest
OLTP	×	√	×	×	√	√
OLAP	√	×	√	√	√	√
Multi-user model	×	√	√	√	√	√
Compression Ratio	×	×	×	×	×	√
Minimal Memory Space	×	×	×	×	×	√
CPU Usage	×	×	×	×	×	√
Cache Miss	×	×	×	×	×	√

Outline

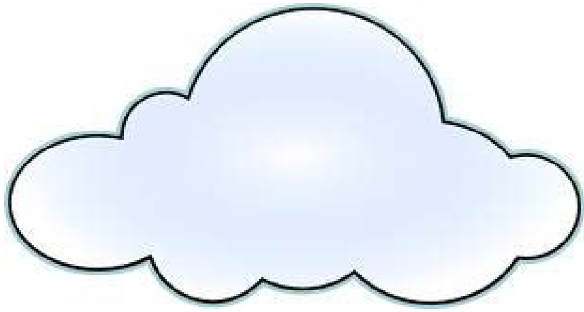
- Introduction & Motivation
- Metrics
- Data Model
- Workload
- Experiment
- Conclusions

Metrics

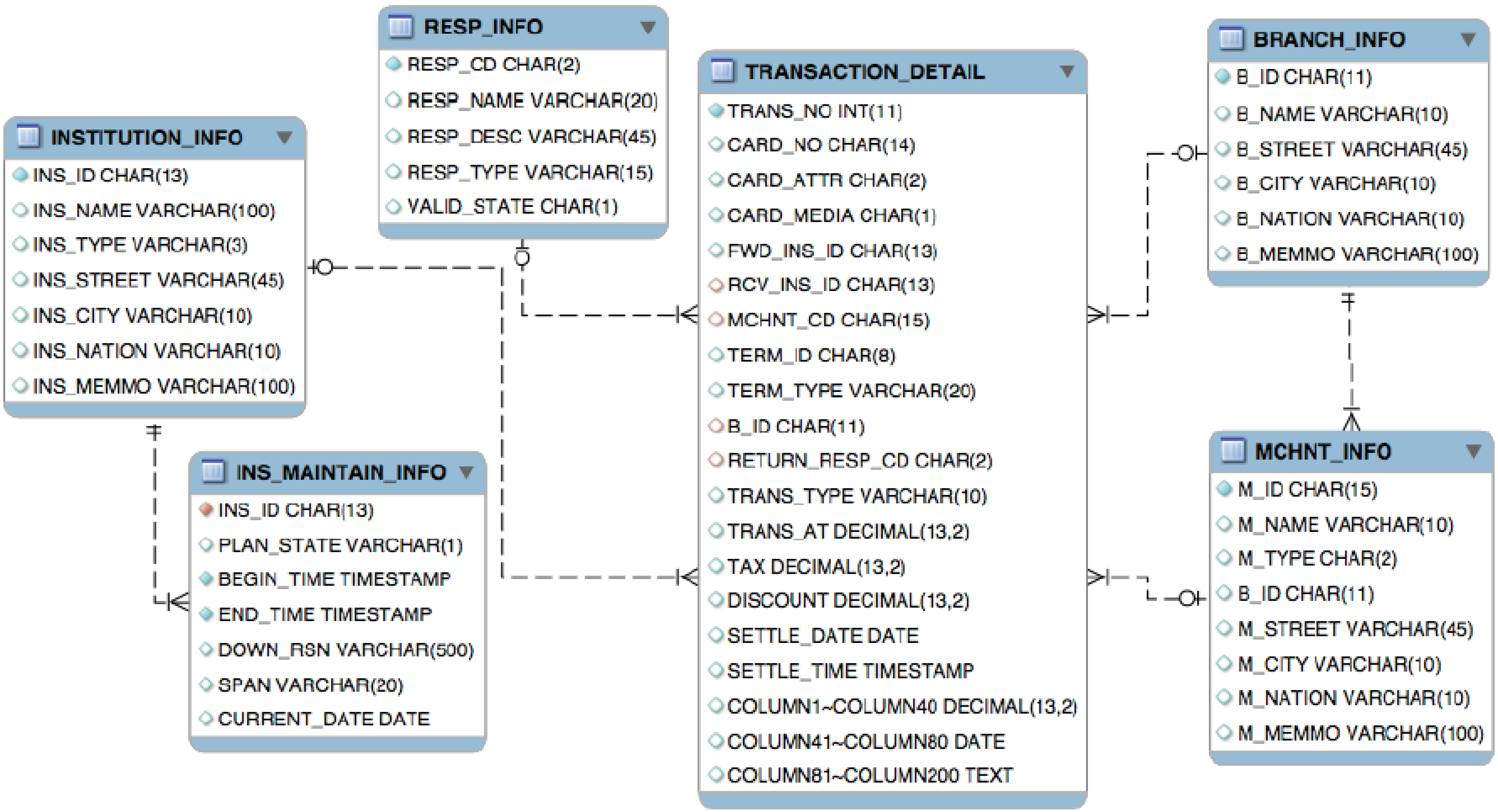
- Basic Performance Measure
  - Response time
    - The total time of 12 queries and two transactions
- CPU Measures
  - CPU Usage
  - Cache Miss
- Memory Measures
  - Compression Ratio
    - S_{Mem} : the data size in memory
    - S_{Disk} : the data size in disk
  - Minimal Memory Space
    - Except the data size, it needs to consider the memory allocation to execute the workload

Outline

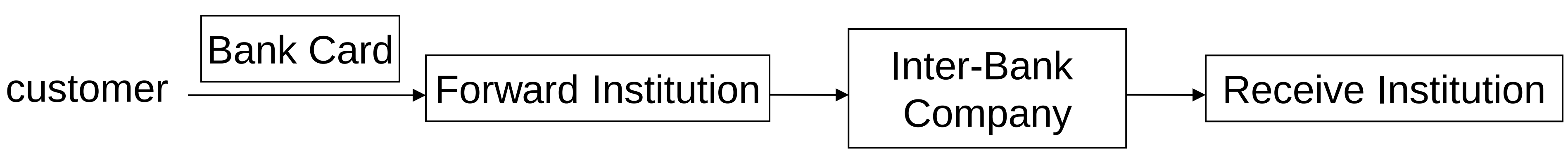
- Introduction & Motivation
- Metrics
- Data Model
- Workload
- Experiment
- Conclusions



Schema



RESP_INFO
RESP_CD CHAR(2)
RESP_NAME VARCHAR(20)
RESP_DESC VARCHAR(45)
RESP_TYPE VARCHAR(15)
VALID_STATE CHAR(1)
INSTITUTION_INFO
INS_ID CHAR(13)
INS_NAME VARCHAR(100)
INS_TYPE VARCHAR(3)
INS_STREET VARCHAR(45)
INS_CITY VARCHAR(10)
INS_NATION VARCHAR(10)
INS_MEMMO VARCHAR(100)
TRANSACTION_DETAIL
TRANS_NO INT(11)
CARD_NO CHAR(14)
CARD_ATTR CHAR(2)
CARD_MEDIA CHAR(1)
FWD_INS_ID CHAR(13)
RCV_INS_ID CHAR(13)
MCHNT_CD CHAR(15)
TERM_ID CHAR(8)
TERM_TYPE VARCHAR(20)
B_ID CHAR(11)
RETURN_RESP_CD CHAR(2)
TRANS_TYPE VARCHAR(10)
TRANS_AT DECIMAL(13,2)
TAX DECIMAL(13,2)
DISCOUNT DECIMAL(13,2)
SETTLE_DATE DATE
SETTLE_TIME TIMESTAMP
COLUMN1~COLUMN40 DECIMAL(13,2)
COLUMN41~COLUMN80 DATE
COLUMN81~COLUMN200 TEXT
BRANCH_INFO
B_ID CHAR(11)
B_NAME VARCHAR(10)
B_STREET VARCHAR(45)
B_CITY VARCHAR(10)
B_NATION VARCHAR(10)
B_MEMMO VARCHAR(100)
INS_MAINTAIN_INFO
INS_ID CHAR(13)
PLAN_STATE VARCHAR(1)
BEGIN_TIME TIMESTAMP
END_TIME TIMESTAMP
DOWN_RSN VARCHAR(500)
SPAN VARCHAR(20)
CURRENT_DATE DATE
MCHNT_INFO
M_ID CHAR(15)
M_NAME VARCHAR(10)
M_TYPE CHAR(2)
B_ID CHAR(11)
M_STREET VARCHAR(45)
M_CITY VARCHAR(10)
M_NATION VARCHAR(10)
M_MEMMO VARCHAR(100)



Bank Card
customer
Forward Institution
Inter-Bank Company
Receive Institution

Table Structure

- TRANSACTION_DETAIL
  - details the transaction information, including the receive institution, the forward institution, the card
- RESP_INFO
  - uses RESP_CD to judge whether the transaction is successful and have many types of response codes
- BRANCH_INFO
  - stores the information of branches, including its name, nation, city, street and so on
- MCHNT_INFO
  - stores the information of merchants, including name, type of a merchant, address and so on
- INSTITUTION_INFO
  - records the information of institutions.
- INS_MAINTAIN_INFO
  - stores the abnormal information of institutions

Data Generation and Distribution

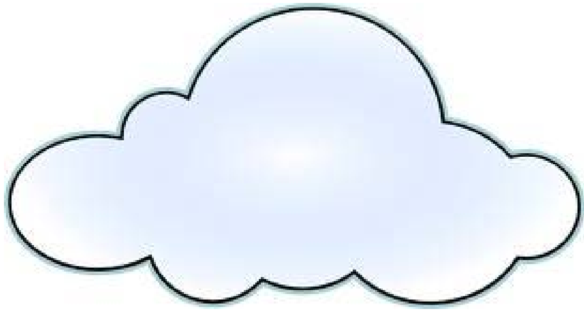
- The following table lists the detailed distribution of data after analyzing a realistic applications for one day.

	Ins	Mchnt	Branch		Resp_Cd		Trans Freq
Big Cities	60%	70%	50%	Suc	90%	Big Ins	80%
Small Cities	40%	30%	50%	Fail	10%	Small Ins	20%

- Generate data which conforms uniform distribution and skew distribution
  - Uniform distribution: use RND function
  - Skew distribution
    - Given a set {r1, r2, r3, r4, r5}, assume the skewed rate of r1 is 0.7
    - Generate a Random Number n between 1 and 10 by using RND function
    - If n<=7, r1 will be generated, else one of the set {r2, r3, r4, r5} will be generated randomly

Outline

- Introduction & Motivation
- Metrics
- Data Model
- Workload
- Experiment
- Conclusions



Business Cases (1)

- **There are four query groups and two transactions.**
- **Query Group 1: Institutions' Transaction Statistics and Analysis**
  - Query 1.1 computes the total amount, average tax and average discount of each institution for each day
  - Query 1.2 computes the transaction frequency of each institution for each day
  - Query 1.3 keeps count of the successful rate and failure rate of each institution for each day according to the response code.
- **Query Group 2: Transaction Quality Analysis**
  - Query 2.1 finds the total number and amount of transactions for each day according to different response codes
  - Query 2.2 keeps count of the total number of failure transactions according to the failure response code
  - Query 2.3 computes the total number of institutions, terms and merchants.

Business Cases (2)

- Query Group 3: Transaction Compliance Analysis
  - Query 3.1 computes low-amount transaction number and average transaction amount according to the term, merchant and branch
  - Query 3.2 records the sign-in transactions and return the top 10 merchants and terms in the branch
  - Query 3.3 return the high-rate failure transactions for each day

- Query Group 4: Institutions’ Abnormal Statistics and Analysis
  - Query 4.1 divides the institutions into three classes (e.g. high-incident, middle-incident, low-incident) according to their abnormal frequency in the history
  - Query 4.2 computes the number of cards, terms, branches and merchants, etc for each abnormal institution in the transaction
  - Query 4.3 computes the total amount, average tax and average discount of transactions during the institutions incident

Business Cases (3)

- Transaction 1: Generating New Transactions
  - insert data into table TRANSACTION_DETAIL every day
- Transaction 2: Capturing the Most Abnormal Institutions
  - find the most abnormal institution

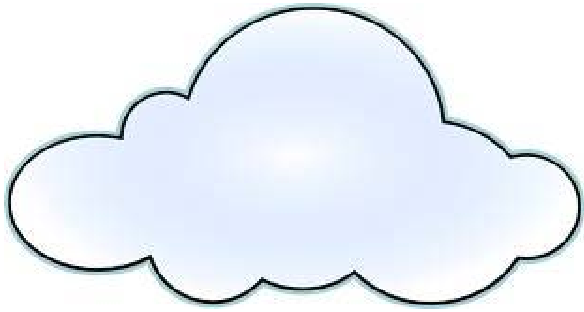
Technical Details

- In the workload, all queries have aggregations, 90% of queries have join operators and a transaction includes update operations.
- A query may have some arguments, like SETTLE_DATE, VALID_STATE and RESP_TYPE. All the arguments can be randomly replaced by a query generator from a predefined dictionary.
- We list the specification of query 2.3 as an example.

```
SELECT RESP_CD, RESP_NAME, INS_NAME, TERM_ID, TERM_TYPE
    , M_NAME, COUNT(*) TRANS_NUM
FROM TRANSACTION_DETAIL T,RESP_INFO R,INSTITUTION_INFO
    I ,MCHNT_INFO M
WHERE T.RETURN_RESP_CD=R.RESP_CD AND T.RCV_INS_ID=I.
    INS_ID AND T.MCHNT_CD=M.M_ID AND RESP_TYPE='DELTA1'
    AND VALID_STATE='DELTA2' AND SETTLE_DATE BETWEEN '
    DELTA3' AND 'DELTA4'
GROUP BY RESP_CD,RESP_NAME,INS_NAME,TERM_ID,TERM_TYPE,
    M_NAME
ORDER BY TRANS_NUM DESC;
```

Outline

- Introduction & Motivation
- Metrics
- Data Model
- Workload
- Experiments
- Conclusions

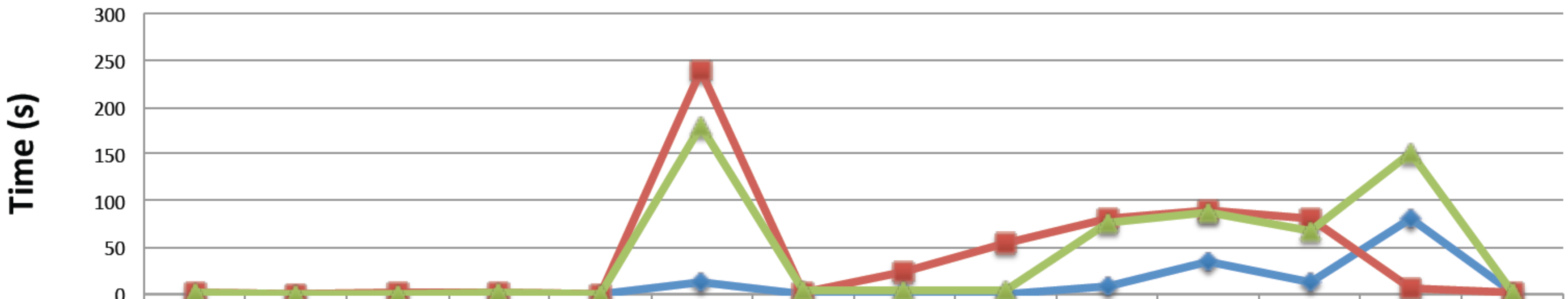


Experiments

- Experimental Configuration
  - Server Under Test
    - 0.5 TB memory and 8 CPUs
    - L1 Cache (private): 32KB
    - L2 Cache (private): 256KB
    - L3 Cache (shared within a CPU): 30MB
  - Three IMDBs: DBMS-X, DBMS-Y, and MonetDB
- By default, the memory size that three IMDBs can use is 90% of total memory and all queries are generated previously and keep unchanged during the benchmark run

Testing for Response Time (1)

- Execution time of Q1.1, Q1.2, Q1.3, Q2.1, Q2.2, Q3.1 and T2 are low in three IMDBs since their operators are relatively simple.



	Q 1.1	Q 1.2	Q 1.3	Q 2.1	Q 2.2	Q 2.3	Q 3.1	Q 3.2	Q 3.3	Q 4.1	Q 4.2	Q 4.3	T1	T2
DBMS-X	0.428	0.068	0.103	0.329	0.036	13.801	0.427	2.102	0.327	7.881	34.836	13.412	81.53	1.069
DBMS-Y	1.659	0.653	0.908	0.968	0.335	238.429	1.985	24.533	54.851	81.441	90.062	81.373	5.234	2.213
Monetdb	2.43	0.508	0.146	1.179	0.255	178.426	4.341	3.16	4.748	76.746	87.785	67.109	151.2	1.5

Fig. 2: Testing for Query and Transaction (SF=2)

Testing for Response Time (2)

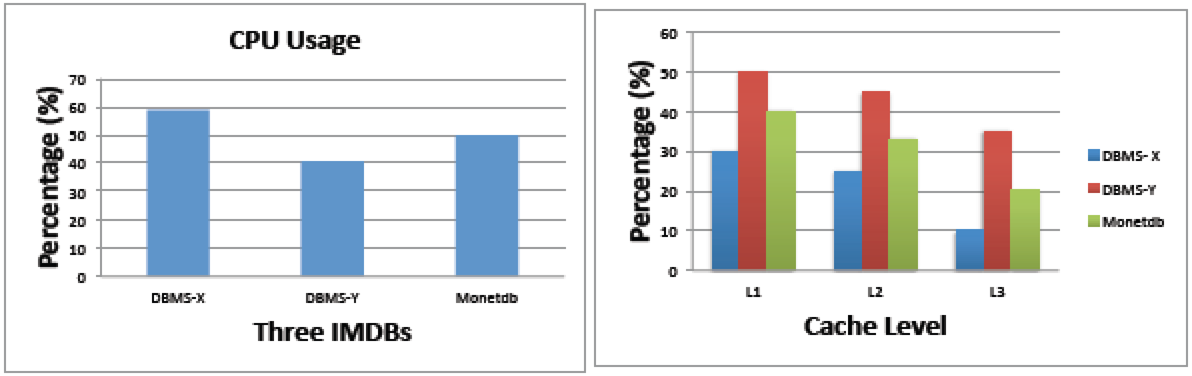
- We also test the performance of IMDBs when multiple users execute the whole workload.
- In this test, we implement a parallel tool to simulate the operator of multiple users.
- We can see that the results of three IMDBs are similar and DBMS-X has a slightly better scalability than other two IMDBs..

Table 3: Testing in the multi-user model

IMDB \ Users	2	4	6	...
DBMS-X	558.12s	1412.51s	$\geqslant$ 1500s	...
DBMS-Y	824s	$\geqslant$ 1800s	–	...
Monetdb	613s	$\geqslant$ 1500s	–	...

Testing for CPU Measures

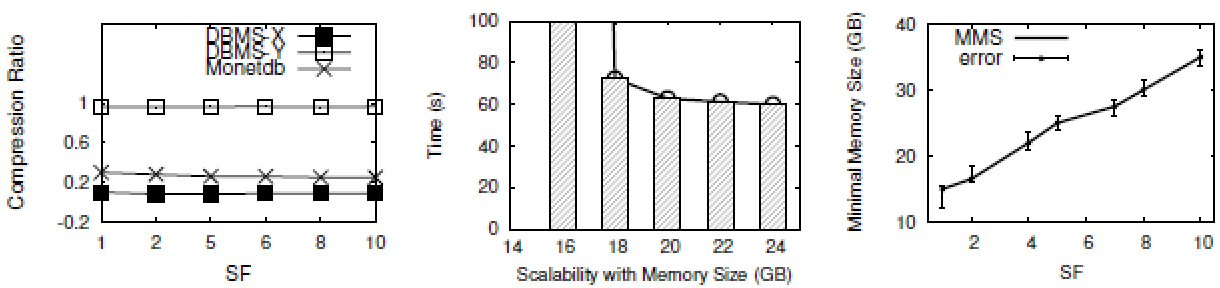
- We can see that three IMDBs have a relatively high CPU usage (> 40%) and DBMS-X and Monetdb behave better than DBMS-Y. This phenomenon can be explained based on the physical organization.
- If the data is organized based on the column-store style, it can load more data tuples that are referenced



(a) CPU Usage (%)   (b) Cache Miss (%)

Testing for Memory Measures

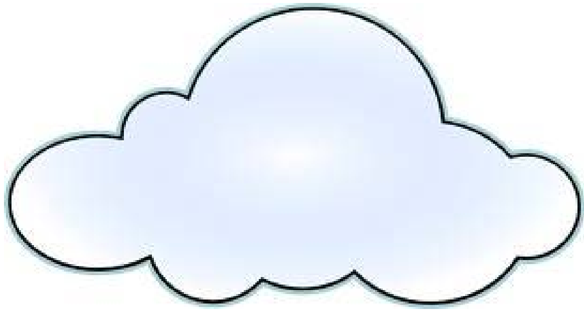
- It is observed that DBMSX and monetdb have better compression ratios, approximately 0.1 and 0.3. DBMS-Y almost has no compression. That is because in the column-store system a field stores the data sets which have the same type. (The first figure in the left)
- Testing for *MMS* is an iterative plan and we need to change the memory size continuously. (Two figures in the right)



(a) Three IMDBs (b) DBMS-X (SF=2) (c) DBMS-X

Outline

- Introduction & Motivation
- Metrics
- Data Model
- Workload
- Experiment
- Conclusions



Conclusion

- MemTest takes special consideration of main characteristics of IMDBs.
  - Novel metrics are especially designed for testing and evaluating IMDBs.
  - A schema based on inter-bank transaction applications are provided and the workload is devised to cover OLAP and OLTP operations.
  - Finally, experiments are conducted to verify the effectiveness and efficiency of our benchmark.
- For future work, we may devise more complex queries and conduct the experiments on more IMDBs.

Thank You

Questions ?