OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

# Build Your Own OctopusDB: Blinktopus Edition

Ali Hashaam, Ali Memon, Guzel Mussilova, Pavlo Shevchenko

Scientific Project: Databases for Multi-Dimensional Data, Genomics and Modern Hardware

July 10, 2017

# Table of Contents

Motivation/Problem Statement

Background

Conceptual Idea and Implementation

Evaluation Setup and Results

Related Work

Conclusion and Future Work

Demonstration

# Motivation

1. Companies need to pick only specialized DBMSs, each tailored to their specific use-case.

# Motivation

1. Companies need to pick only specialized DBMSs, each tailored to their specific use-case.
   ⇒ Need for *one size fits all system* (e.g. HTAP)

# Motivation

1. Companies need to pick only specialized DBMSs, each tailored to their specific use-case.
   $\Rightarrow$ Need for *one size fits all system* (e.g. HTAP)
2. Support OLAP queries for analysis over real-time data (i.e., freshness).

# Motivation

1. Companies need to pick only specialized DBMSs, each tailored to their specific use-case.
   ⇒ Need for *one size fits all system* (e.g. HTAP)
2. Support OLAP queries for analysis over real-time data (i.e., freshness).
   ⇒ Explore the techniques related to more interactive queries (e.g. *Approximate Query Processing*)

# Background

1. OctopusDB

# Background

1. **OctopusDB**
    - uses logs as a primary storage;

# Background

1. **OctopusDB**
    - uses logs as a primary storage;
    - mimicks several types of systems (OLAP, OLTP, etc.) by representing them as *Storage Views*.

# Background

1. **OctopusDB**
    - uses logs as a primary storage;
    - mimicks several types of systems (OLAP, OLTP, etc.) by representing them as *Storage Views*.
2. **BlinkDB**
    - successfully integrates AQP techniques into its architecture.

# Conceptual Idea and Implementation

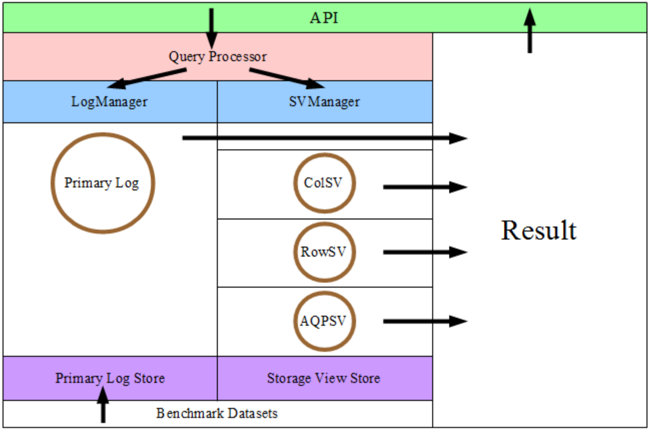


**Figure 1:** OctopusDB Architecture.

# Conceptual Idea and Implementation

**Which synopses to pick?**

[1] https://datasketches.github.io

# Conceptual Idea and Implementation

**Which synopses to pick?**

- Equi-depth histograms
  - suitable for range queries;
  - simple to implement and interpret.

[1] https://datasketches.github.io

# Conceptual Idea and Implementation

**Which synopses to pick?**

- Equi-depth histograms
  - suitable for range queries;
  - simple to implement and interpret.
- Sketches
  - DISTINCT COUNT queries;
  - *HyperLogLog*;
  - *DataSketches* library by *Yahoo!* [1]

[1] https://datasketches.github.io

# Evaluation Setup

# Evaluation Setup

## Machine

- CentOS Linux 7.1.1503
- Java SDK 8u131-b11-linux-x64
- 2 Intel(r) Xeon (TM) E5-2630 v3s CPU @ 3.2GHz processors (8 cores each) and 1024 GiB memory

# Evaluation Setup

## Machine

- CentOS Linux 7.1.1503
- Java SDK 8u131-b11-linux-x64
- 2 Intel(r) Xeon (TM) E5-2630 v3s CPU @ 3.2GHz processors (8 cores each) and 1024 GiB memory

## Benchmark Datasets

- TPC-H datasets (Orders and Lineitems)

# Evaluation Setup

## Machine

- CentOS Linux 7.1.1503
- Java SDK 8u131-b11-linux-x64
- 2 Intel(r) Xeon (TM) E5-2630 v3s CPU @ 3.2GHz processors (8 cores each) and 1024 GiB memory

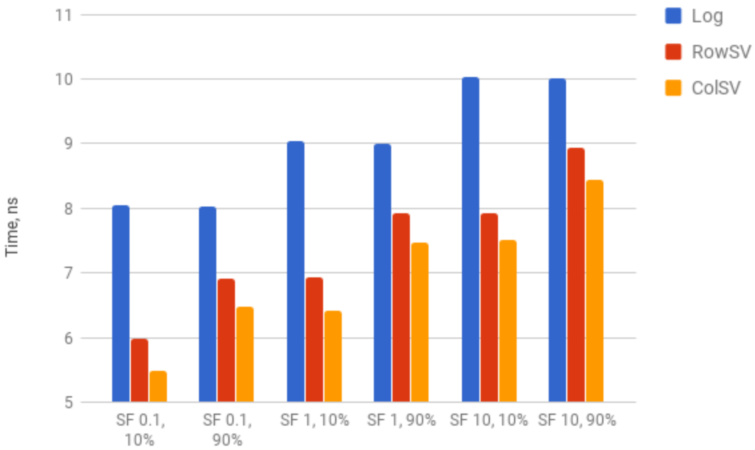
## Benchmark Datasets

- TPC-H datasets (Orders and Lineitems)

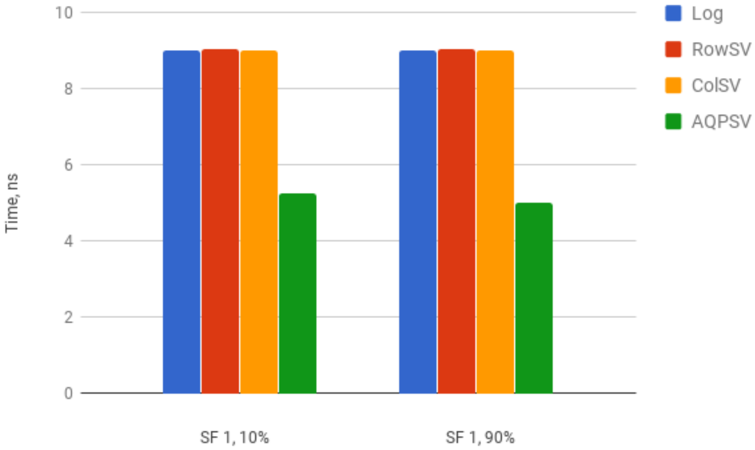
## Experiments

1. Average response time for a range query on the Orders table with various scaling factors and predicate selectivity.
2. Average response time for a count-range query on the Orders table. Comparison with an equi-depth histogram.
3. Average response time for a count distinct query on the Orders table. Comparison with a HLL sketch.

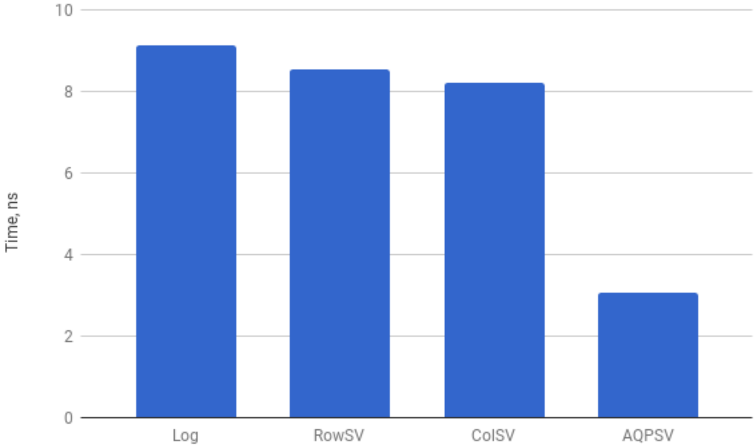
# Results. Experiment 1



# Results. Experiment 2

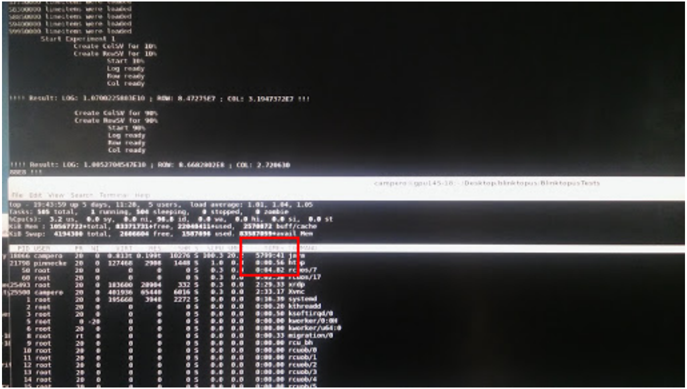


# Results. Experiment 3



# Challenges

# Challenges

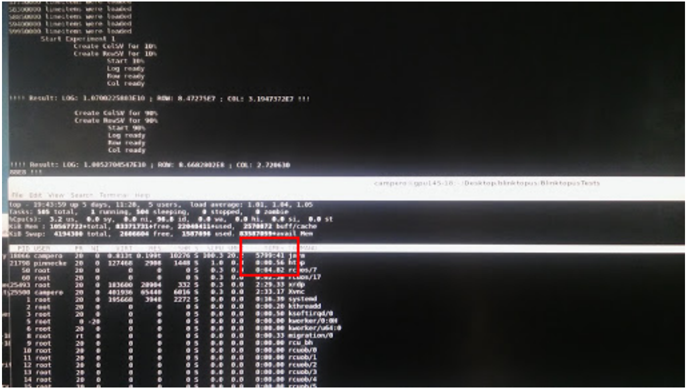


```
58300000 timeitems were loaded
58850000 timeitems were loaded
59400000 timeitems were loaded
59950000 timeitems were loaded
        Start Experiment 1
                Create ColSV for 10%
                Create RowSV for 10%
                        Start 10%
                        Log ready
                        Row ready
                        Col ready

!!!! Result: LOG: 1.0700225803E10 ; ROW: 8.47275E7 : COL: 3.1947372E7 !!!

                Create ColSV for 90%
                Create RowSV for 90%
                        Start 90%
                        Log ready
                        Row ready
                        Col ready

!!!! Result: LOG: 1.0052704547E10 ; ROW: 8.6682002E8 ; COL: 2.720630
8008 !!!
```

campero@gpu145-18:~/Desktop/blinktopus/BlinktopusTests
File Edit View Search Terminal Help

```
top - 19:43:59 up 5 days, 11:28,  5 users,  load average: 1.01, 1.04, 1.05
Tasks: 505 total,   1 running, 504 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.2 us,  0.0 sy,  0.0 ni, 96.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 10567722+total, 83371731+free, 22040411+used,  2570872 buff/cache
KiB Swap:  4194300 total,  2606604 free,  1587096 used. 83587099+avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18866 campero   20   0 0.813t 0.190t  10276 S 100.3 20.2   5799:41 java
21798 pinnecke  20   0  127468   2908   1448 S   1.0  0.0   0:00.56 htop
   50 root      20   0       0      0      0 S   0.3  0.0   0:04.82 rcuos/7
   60 root      20   0       0      0      0 S   0.3  0.0   0:02.38 rcuos/17
25493 root      20   0  183600  20904    332 S   0.3  0.0   2:29.33 xrdp
25500 campero   20   0  401936  65440   6016 S   0.3  0.0   2:33.17 Xvnc
    1 root      20   0  195660   3948   2272 S   0.0  0.0   0:30.39 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.20 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.50 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/u64:0
    8 root      rt   0       0      0      0 S   0.0  0.0   0:00.33 migration/0
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/2
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/3
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/4
   15 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/5
```

# Challenges



```
58300000 lineitems were loaded
58850000 lineitems were loaded
59400000 lineitems were loaded
59950000 lineitems were loaded
        Start Experiment 1
                Create ColSV for 10%
                Create RowSV for 10%
                        Start 10%
                        Log ready
                        Row ready
                        Col ready

!!!! Result: LOG: 1.0700225803E10 ; ROW: 8.47275E7 : COL: 3.1947372E7 !!!

                Create ColSV for 90%
                Create RowSV for 90%
                        Start 90%
                        Log ready
                        Row ready
                        Col ready

!!!! Result: LOG: 1.0052704547E10 ; ROW: 8.6682002E8 ; COL: 2.720630
8008 !!!
```

campero@gpu145-18: ~/Desktop/blinktopus/BlinktopusTests

```
top - 19:43:59 up 5 days, 11:28,  5 users,  load average: 1.01, 1.04, 1.05
Tasks: 505 total,   1 running, 504 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.2 us,  0.0 sy,  0.0 ni, 96.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 10567722+total, 83371731+free, 22040411+used,  2570072 buff/cache
KiB Swap:  4194300 total,  2606604 free,  1587096 used. 83587059+avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18066 campero   20   0  0.013t 0.199t  10276 S 100.3 20.2   5799:41 java
21750 pinnecke  20   0  127468   2908   1448 S   1.0  0.0   0:00.56 htop
   50 root      20   0       0      0      0 S   0.3  0.0   0:04.02 rcuos/7
   60 root      20   0       0      0      0 S   0.3  0.0   0:02.38 rcuos/17
25493 root      20   0  183600  20004    332 S   0.3  0.0   2:29.33 xrdp
25500 campero   20   0  401936  65440   6016 S   0.3  0.0   2:33.17 Xvnc
    1 root      20   0  195660   3948   2272 S   0.0  0.0   0:16.39 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.20 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.50 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/u64:0
    8 root      rt   0       0      0      0 S   0.0  0.0   0:00.33 migration/0
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/2
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/3
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/4
```

5788,41 min = 96 hours = 4 days

# Related Work

# Related Work

1. **Apache Samza**
    - logs as a primary storage;
    - replicates logs on multiple nodes.

# Related Work

1. **Apache Samza**
   - logs as a primary storage;
   - replicates logs on multiple nodes.
2. **Rodent Store**
   - represents data in the various physical layouts;
   - provides DBAs a high-level interface to specify the data physical representation by means of storage algebra.

# Related Work

1. **Apache Samza**
   - logs as a primary storage;
   - replicates logs on multiple nodes.
2. **Rodent Store**
   - represents data in the various physical layouts;
   - provides DBAs a high-level interface to specify the data physical representation by means of storage algebra.
3. **Snappy Data**
   - AQP Support;
   - uses numerous types of synopses (samples, sketches);
   - user defines the level of accuracy and the number of column sets to approximate the results.

# Conclusion

# Conclusion

- Systems with adaptive layouts can be effectively combined with AQP techniques.

# Conclusion

- Systems with adaptive layouts can be effectively combined with AQP techniques.
- OLAP queries can benefit from AQP techniques.

# Conclusion

- Systems with adaptive layouts can be effectively combined with AQP techniques.
- OLAP queries can benefit from AQP techniques.
- Non-optimized central log as a primary storage is quite prohibitive.

# Future Work

- optimize centralized log (e.g. log replication, garbage collection);

# Future Work

- optimize centralized log (e.g. log replication, garbage collection);
- evaluate the efficiency of the concurrency control scheme of OctopusDB;

# Future Work

- optimize centralized log (e.g. log replication, garbage collection);
- evaluate the efficiency of the concurrency control scheme of OctopusDB;
- evaluate the memory footprint of histograms and sketches;

# Future Work

- optimize centralized log (e.g. log replication, garbage collection);
- evaluate the efficiency of the concurrency control scheme of OctopusDB;
- evaluate the memory footprint of histograms and sketches;
- extend Blinktopus architecture to support transactional model;

# Future Work

- optimize centralized log (e.g. log replication, garbage collection);
- evaluate the efficiency of the concurrency control scheme of OctopusDB;
- evaluate the memory footprint of histograms and sketches;
- extend Blinktopus architecture to support transactional model;
- extend query classes by implementing sample-based data synopses.

## Demonstration

# Thank you!

# Questions? Recommendations? Remarks?