VictoriaMetrics:
Why and how we built fast and scalable Open Source Time Series Database

Aliaksandr Valialkin
Roman Khavronenko
Monitoring applications
Old fashion

- Application prints logs
Old fashion

- Application prints logs
- Developer reads logs
Old fashion

- Application prints logs
- Developer reads logs
- Check logs when application breaks
New era

- Know the actual state of any application in any moment of time
New era

- Know the actual state of any application at any time
- Prove your theories with metrics, not just words
New era

- Know the actual state of any application at any time
- Prove your theories with metrics, not just words
- Visualize application state
New era

- Know the actual state of any application at any time
- Prove you theories with metrics not just words
- Visualize application state
- Setup alerting systems to control app state daily and nightly
Common beginner mistakes

- Wrong metric types
Common beginner mistakes

- Wrong metric types
- Metrics churn and high cardinality
Common beginner mistakes

- Wrong metric types
- Metrics churn and high cardinality
- Unused or forgotten metrics
Common beginner mistakes

- Wrong metric types
- Metrics churn and high cardinality
- Unused or forgotten metrics
- Complex, heavy, overloaded dashboards
Common beginner mistakes

- Wrong metric types
- Metrics churn and high cardinality
- Unused or forgotten metrics
- Complex, heavy, overloaded dashboards
- Meaningless false-positive alerts
With great efforts comes great success

- All mistakes learned
- All anomalies and memory leaks found and eliminated
- You show dashboard to you mom - let her to check out this masterpiece!
- Just-in-time alert saved your company from incident
Share knowledge!

- Monitoring is great
- Everybody should expose and collect metrics
- Everybody should write alerts and build dashboards
- ...
- ... and they do
Overload

- Everybody using metrics now
- There are billions of metrics ingested and fetched every day
- Common beginner mistakes again
- Heavy and slow queries
- Teams rely on business-critical metrics and alerts
With half the city feeding you sonar, you can image all of Gotham.
This is wrong.
<table>
<thead>
<tr>
<th>Prometheus</th>
<th>Alerts</th>
<th>Graph</th>
<th>Status</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage.tsdb.min-block-duration</td>
<td>2h</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage.tsdb.no-lockfile</td>
<td>false</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage.tsdb.path</td>
<td>/prometheus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage.tsdb.retention</td>
<td>0s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage.tsdb.retention.size</td>
<td>0B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>storage.tsdb.retention.time</td>
<td>15d</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prometheus Benchmark - 2.7.x

Metrics useful for benchmarking and load testing Prometheus itself. Designed primarily for Prometheus 2.7.x.
Last updated: 5 months ago
Becoming an expert
Problems to solve

- Long-term storage
- On-disk compression
- Query speed
- Scalability
- Global query view
Inspired by ClickHouse

- Awesome performance
- Simple and beautiful architecture
- Outstanding compression
- Scaling out of the box
Problems to solve

- Long-term storage
- On-disk compression
- Query speed
- Scalability
- Global query view
Long-term storage

- Arbitrary retention supported
- No performance degradation on big retentions
Improved on-disk compression

- Start with Gorilla encoding
Improved on-disk compression

- Start with Gorilla encoding
- Customize it for decimal floating-point numbers
Improved on-disk compression

- Start with Gorilla encoding
- Customize it for decimal floating-point numbers
- Use specialized encodings for counters and gauges
Improved on-disk compression

- Start with Gorilla encoding
- Customize it for decimal floating-point numbers
- Use specialized encodings for counters and gauges
- Compress the data with zstd
Improved on-disk compression

- Start with Gorilla encoding
- Customize it for decimal floating-point numbers
- Use specialized encodings for counters and gauges
- Compress the data with zstd
- The end result: up to 70x better compression than competitor
Disk usage, MB

- VictoriaMetrics v1.13.0: 377 MB
- InfluxDB v1.7.5: 566 MB
- TimescaleDB v1.2.2: 29000 MB
Disk write bandwidth usage, MB/s (lower is better)

- VictoriaMetrics v1.13.0
- InfluxDB v1.7.5
- TimescaleDB v1.2.2

- n1-standard-1: VictoriaMetrics 2, InfluxDB 13, TimescaleDB 70
- n1-standard-2: VictoriaMetrics 4, InfluxDB 19, TimescaleDB 95
- n1-standard-4: VictoriaMetrics 8, InfluxDB 32, TimescaleDB 152
- n1-standard-8: VictoriaMetrics 18, InfluxDB 62, TimescaleDB 260
- n1-standard-16: VictoriaMetrics 38, InfluxDB 100, TimescaleDB 260
- n1-standard-32: VictoriaMetrics 63, InfluxDB 164, TimescaleDB 260
- n1-standard-64: VictoriaMetrics 137, InfluxDB 190, TimescaleDB 260
Make queries faster

- Inverted index lookups
- Data scan
Make queries faster: inverted index

- MergeSet for fast lookups by label filters
Make queries faster: inverted index

- MergeSet for fast lookups by label filters
- Tune inverted index lookups for high number of time series
Make queries faster: inverted index

- MergeSet for fast lookups by label filters
- Tune inverted index lookups for high number of time series
- Index scan speed: up to 100 million items per second per CPU core
Make queries faster: data

- Minimize disk IO for data scans
Make queries faster: data

- Minimize disk IO for data scans
- Utilize the full power of CPU
Make queries faster: data

- Minimize disk IO for data scans
- Utilize the full power of CPU
- Parallelize data scans on all the CPU cores
Make queries faster: data

- Minimize disk IO for data scans
- Utilize the full power of CPU
- Parallelize data scans on all the CPU cores
- Smart caching
Make queries faster: data

- Minimize disk IO for data scans
- Utilize the full power of CPU
- Parallelize data scans on all the CPU cores
- Smart caching
- The end result: up to 50 million data points per CPU core
Scalability

- Vertical
- Horizontal
Vertical scalability

- Scale to available CPU cores
Vertical scalability

- Scale to available CPU cores
- Use available RAM for caches
Vertical scalability

- Scale to available CPU cores
- Use available RAM for caches
- End result: performance linearly scales for available CPU cores
Ingestion rate, thousands of data points / sec (higher is better)

- **VictoriaMetrics v1.13.0**
- **InfluxDB v1.7.5**
- **TimescaleDB v1.2.2**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>VictoriaMetrics</th>
<th>InfluxDB</th>
<th>TimescaleDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1-standard-1</td>
<td>510</td>
<td>200</td>
<td>388</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>510</td>
<td>320</td>
</tr>
<tr>
<td></td>
<td>3150</td>
<td>1130</td>
<td>1930</td>
</tr>
<tr>
<td></td>
<td>5920</td>
<td>2150</td>
<td>1920</td>
</tr>
<tr>
<td></td>
<td>11040</td>
<td>2300</td>
<td>2220</td>
</tr>
<tr>
<td></td>
<td>19100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
double-groupby-1, clients=vCPUs, rpm (higher is better)

- VictoriaMetrics v1.13.0
- InfluxDB v1.7.5
- TimescaleDB v1.2.2

- n1-standard-1: VictoriaMetrics = 71, InfluxDB = 7, TimescaleDB = 0
- n1-standard-2: VictoriaMetrics = 94, InfluxDB = 12, TimescaleDB = 0
- n1-standard-4: VictoriaMetrics = 185, InfluxDB = 23, TimescaleDB = 0
- n1-standard-8: VictoriaMetrics = 375, InfluxDB = 46, TimescaleDB = 0
- n1-standard-16: VictoriaMetrics = 722, InfluxDB = 69, TimescaleDB = 0
- n1-standard-32: VictoriaMetrics = 1303, InfluxDB = 162, TimescaleDB = 0
- n1-standard-64: VictoriaMetrics = 1350, InfluxDB = 159, TimescaleDB = 206
Horizontal scalability
**Clients**

*vmselect fully supports PromQL and can be used as Prometheus datasource in Grafana*

**Stateless**

*vmselect fetches and merges data from vmstorage during queries*

**Statefull**

*vmstorage stores time series data*

**Stateless**

*vminsert spreads time series across available vmstorage nodes*

**Writers**

*Multiple Prometheus instances may write data to VictoriaMetrics cluster. There is support for other ingestion protocols*
Horizontal scalability

- Cluster components scale independently
Horizontal scalability

- Cluster components scale independently
- Minimize network bandwidth usage between cluster components
Horizontal scalability

- Cluster components scale independently
- Minimize network bandwidth usage between cluster components
- The end result: cluster scales almost linearly up to 50 nodes.
Global view over infrastructure

- Collect data from many Prometheus instances
Global view over infrastructure

- Collect data from many Prometheus instances
- Collect data from other sources by InfluxDB, Graphite or OpenTSDB protocols
Global view over infrastructure

- Collect data from many Prometheus instances
- Collect data from other sources by InfluxDB, Graphite or OpenTSDB protocols
- Global query view
Global view over infrastructure

- Collect data from many Prometheus instances
- Collect data from other sources by InfluxDB, Graphite or OpenTSDB protocols
- Global query view
- Multi-tenancy
Get benefits from extended PromQL
Get benefits from extended PromQL

- Sometimes standard PromQL isn’t enough
Get benefits from extended PromQL

- Sometimes standard PromQL isn’t enough
- Additional functions: `label_set(q, label_name, label_value)`
Get benefits from extended PromQL

- Sometimes standard PromQL isn’t enough
- Additional functions: `label_set(q, label_name, label_value)`
- Additional operators: `q default 0`
Get benefits from extended PromQL

- Sometimes standard PromQL isn’t enough
- Additional functions: `label_set(q, label_name, label_value)`
- Additional operators: `q default 0`
- Optional square brackets for range functions: `rate(q)`
VictoriaMetrics: the final result

- Free open-source time series database
- Single-node and cluster versions
- Easy setup and operation
- Resource-efficient - uses low RAM, CPU, storage and network
- Good integration with Prometheus+Grafana stack
Questions?

Speakers:
Aliaksandr Valialkin, founder, CTO
Roman Khavronenko, co-founder, COO

victoriametrics.com  github.com/VictoriaMetrics
info@victoriametrics.com  slack.victoriametrics.com