

Large-scale logging made easy

FrOSCon 2025

Aliaksandr Valialkin, CTO at VictoriaMetrics

Let's meet

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)
- I'm a software engineer

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)
- I'm a software engineer
- I write code in Go

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)
- I'm a software engineer
- I write code in Go
- I like writing useful open-source software, which works out of the box

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)
- I'm a software engineer
- I write code in Go
- I like writing useful open-source software, which works out of the box
- I like creating servers optimized for speed and low resource usage

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)
- I'm a software engineer
- I write code in Go
- I like writing useful open-source software, which works out of the box
- I like creating servers optimized for speed and low resource usage
- I created VictoriaMetrics - open-source database for metrics

Let's meet

- I'm valyala (aka Aliaksandr Valialkin)
- I'm a software engineer
- I write code in Go
- I like writing useful open-source software, which works out of the box
- I like creating servers optimized for speed and low resource usage
- I created VictoriaMetrics - open-source database for metrics
- I created VictoriaLogs - open-source database for logs

What is logging?



What is logging?

- Producing logs

What is logging?

- Producing logs
 - Server logs

What is logging?

- Producing logs
 - Server logs
 - Client application logs

What is logging?

- Producing logs
 - Server logs
 - Client application logs
 - Web event logs

What is logging?

- Producing logs
 - Server logs
 - Client application logs
 - Web event logs
 - Industrial logs

What is logging?

- Producing logs
 - Server logs
 - Client application logs
 - Web event logs
 - Industrial logs
- Collecting and saving logs into a centralized database

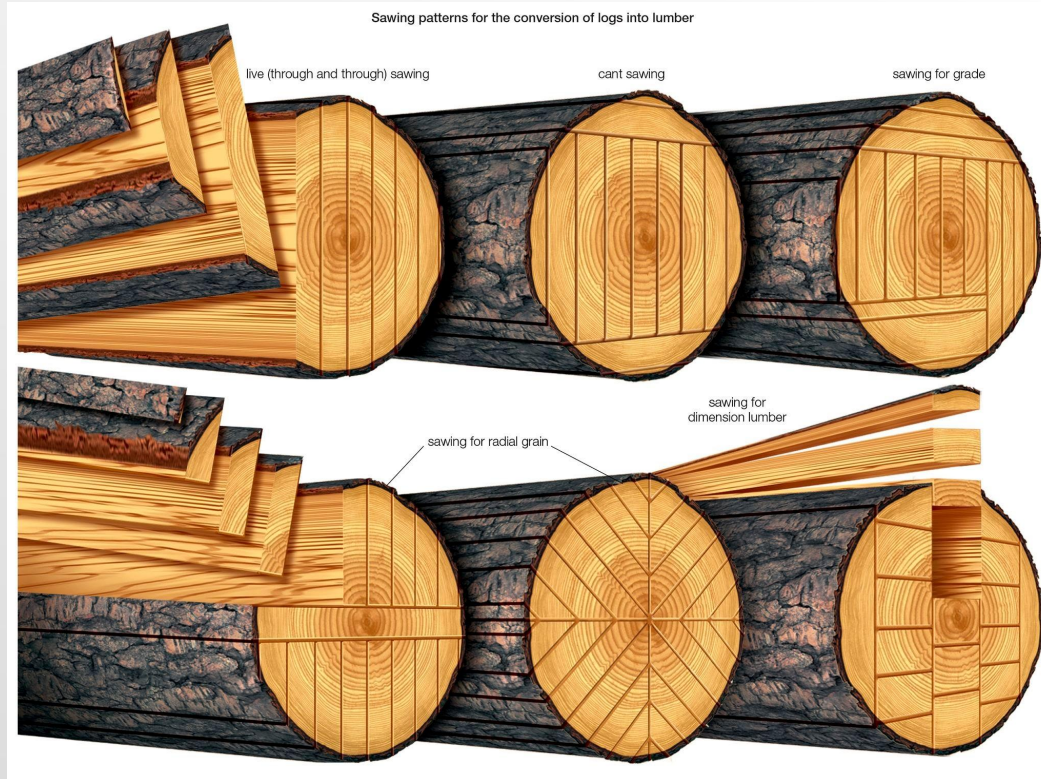
What is logging?

- Producing logs
 - Server logs
 - Client application logs
 - Web event logs
 - Industrial logs
- Collecting and saving logs into a centralized database
- Querying the collected logs

What is logging?

- Producing logs
 - Server logs
 - Client application logs
 - Web event logs
 - Industrial logs
- Collecting and saving logs into a centralized database
- Querying the collected logs
- Archiving old logs

Types of logs



Types of logs

- Plaintext logs

Types of logs

- Plaintext logs

```
2025-08-14T19:01:45.523Z      info    app/victoria-logs/main.go:43    starting VictoriaLogs at "[:9428]"...
2025-08-14T19:01:45.523Z      info    app/vlstorage/main.go:124      opening storage at -storageDataPath=victoria-logs-
data
2025-08-14T19:01:45.526Z      info    vendor/github.com/VictoriaMetrics/VictoriaMetrics/lib/memory/memory.go:42    li
miting caches to 19445563392 bytes, leaving 12963708928 bytes to the OS according to -memory.allowedPercent=60
2025-08-14T19:01:45.530Z      info    app/vlstorage/main.go:130      successfully opened storage in 0.007 seconds; smal
lParts: 8; bigParts: 10; smallPartBlocks: 7967; bigPartBlocks: 18526; smallPartRows: 2265816; bigPartRows: 27654184; small
PartSize: 302824584 bytes; bigPartSize: 3419219775 bytes
2025-08-14T19:01:45.530Z      info    app/victoria-logs/main.go:55    started VictoriaLogs in 0.007 seconds; see https:/
/docs.victoriametrics.com/victorialogs/
2025-08-14T19:01:45.530Z      info    vendor/github.com/VictoriaMetrics/VictoriaMetrics/lib/httpserver/httpserver.go:145
started server at http://0.0.0.0:9428/
2025-08-14T19:01:45.530Z      info    vendor/github.com/VictoriaMetrics/VictoriaMetrics/lib/httpserver/httpserver.go:147
pprof handlers are exposed at http://0.0.0.0:9428/debug/pprof/
```

Types of logs

- Plaintext logs
- Structured logs

Types of logs

- Plaintext logs
- Structured logs

```
{
  "_msg": "35.191.238.181:52096 - 0f400800-28bb-47c8-b5ba-277c8d8ba872 - - [2025/08/14 22:03:08] 10.71.11.7 GET - \"/ready\" HTTP/1.1 \"GoogleHC/1.0\" 2>
  "_stream": "{kubernetes.container_name=\"oauth2-proxy\",kubernetes.pod_name=\"vmlogs-single-victoria-logs-single-server-0\",kubernetes.pod_namespace=\\>
  "_stream_id": "0000000000000000000000c56bebb8b0c9bda967541bf348deb44c",
  "_time": "2025-08-14T22:03:08.673870959Z",
  "file": "/var/log/pods/vmlogs_vmlogs-single-victoria-logs-single-server-0_afddf5f1-3b80-4a60-b5e4-84acfc8f05da/oauth2-proxy/0.log",
  "kubernetes.container_id": "containerd://3d9e32d14efa63a101e5537cc3531043382bc2625033a0110f6968dd53f76d62",
  "kubernetes.container_image": "quay.io/oauth2-proxy/oauth2-proxy:v7.11.0",
  "kubernetes.container_image_id": "quay.io/oauth2-proxy/oauth2-proxy@sha256:be154ec9c7e710a99cfd497da3ae387a8a21657302faff4dc1ad5a1ce7ebc529",
  "kubernetes.container_name": "oauth2-proxy",
  "kubernetes.namespace_labels.kubernetes.io/metadata.name": "vmlogs",
  "kubernetes.pod_ip": "10.71.11.7",
  "kubernetes.pod_ips": "[\"10.71.11.7\"]",
  "kubernetes.pod_name": "vmlogs-single-victoria-logs-single-server-0",
  "kubernetes.pod_namespace": "vmlogs",
  "kubernetes.pod_node_name": "gke-sandbox-e2-standard-8-20250715071-5b0a2ce9-kgb8",
  "kubernetes.pod_owner": "StatefulSet/vmlogs-single-victoria-logs-single-server",
  "kubernetes.pod_uid": "afddf5f1-3b80-4a60-b5e4-84acfc8f05da",
  "source_type": "kubernetes_logs",
  "stream": "stdout"
}
```

Types of logs

- Plaintext logs
- Structured logs
- Wide events

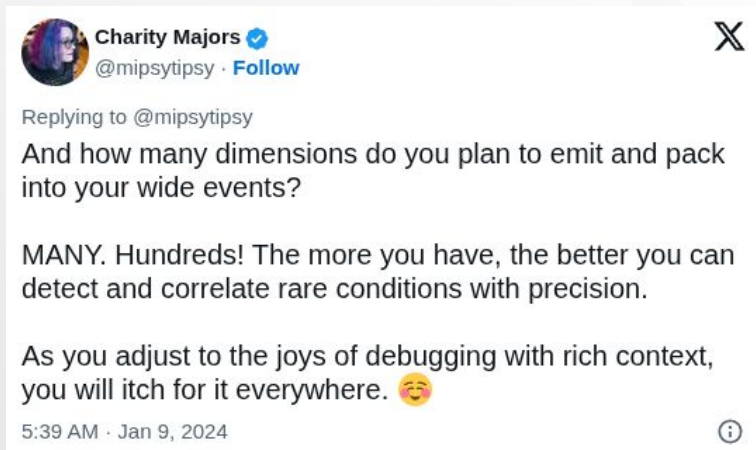
Types of logs

- Plaintext logs
- Structured logs
- Wide events
 - Logs with hundreds of fields

<https://jeremymorrell.dev/blog/a-practitioners-guide-to-wide-events/>

Types of logs

- Plaintext logs
- Structured logs
- Wide events
 - Logs with hundreds of fields



<https://jeremymorrell.dev/blog/a-practitioners-guide-to-wide-events/>

When logging becomes large-scale?



When logging becomes large-scale?

- A gigabyte of logs per day?

When logging becomes large-scale?

- A gigabyte of logs per day?
- A terabyte of logs per day?

When logging becomes large-scale?

- A gigabyte of logs per day?
- A terabyte of logs per day?
- A petabyte of logs per day?

When logging becomes large-scale?

- When logs cannot fit a single computer

When logging becomes large-scale?

- When logs cannot fit a single computer
- This depends on the size of the computer and the database used

When logging becomes large-scale?

- When logs cannot fit a single computer
- This depends on the size of the computer and the database used
 - This can be a gigabyte of logs stored in PostgreSQL on a low-end Raspberry PI

When logging becomes large-scale?

- When logs cannot fit a single computer
- This depends on the size of the computer and the database used
 - This can be a gigabyte of logs stored in PostgreSQL on a low-end Raspberry PI
 - This can be a petabyte of logs stored in a specialized database on a high-end computer

How to store a petabyte of logs on a single computer?

How to store a petabyte of logs on a single computer?



How to store a petabyte of logs on a single computer?

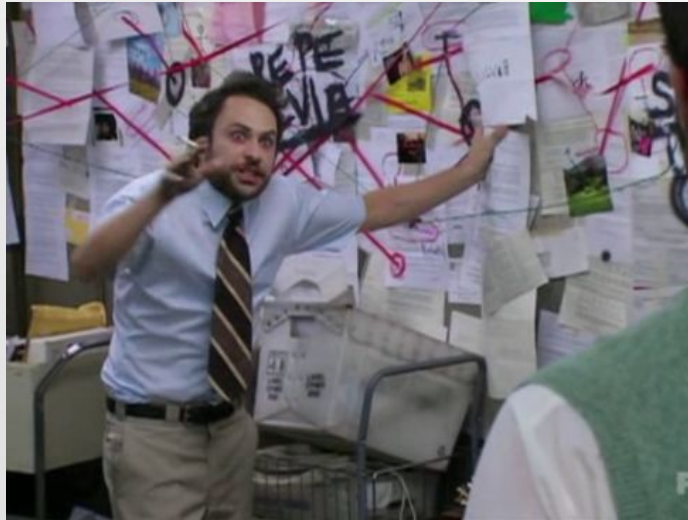
- The maximum available disk size for a single computer in Google Cloud is 64TB

How to store a petabyte of logs on a single computer?

- The maximum available disk size for a single computer in Google Cloud is 64TB
- 1PB of logs needs $1\text{PB} / 64\text{TB} =$ sixteen 64TB disks

How to store a petabyte of logs on a single computer?

- The maximum available disk size for a single computer in Google Cloud is 64TB
- 1PB of logs needs $1\text{PB} / 64\text{TB} = \text{sixteen } 64\text{TB}$ disks
- ???



How to store a petabyte of logs on a single computer?

- The maximum available disk size for a single computer in Google Cloud is 64TB
- 1PB of logs needs $1\text{PB} / 64\text{TB} =$ sixteen 64TB disks
- ???
- Compress logs by more than 16x times!

How to store a petabyte of logs on a single computer?

- The maximum available disk size for a single computer in Google Cloud is 64TB
- 1PB of logs needs $1\text{PB} / 64\text{TB} =$ sixteen 64TB disks
- ???
- Compress logs by more than 16x times!
- The typical compression ratio for logs stored in specialized databases for logs is in the range 8x-50x

How to store a petabyte of logs on a single computer?

- The maximum available disk size for a single computer in Google Cloud is 64TB
- 1PB of logs needs $1\text{PB} / 64\text{TB} =$ sixteen 64TB disks
- ???
- Compress logs by more than 16x times!
- The typical compression ratio for logs stored in specialized databases for logs is in the range 8x-50x
- It is possible to store a petabyte of logs (and more) on a single computer!

How to store a petabyte of logs on a single computer?

- The maximum available disk size for a single computer in Google Cloud is 64TB
- 1PB of logs needs $1\text{PB} / 64\text{TB} =$ sixteen 64TB disks
- ???
- Compress logs by more than 16x times!
- The typical compression ratio for logs stored in specialized databases for logs is in the range 8x-50x
- It is possible to store a petabyte of logs (and more) on a single computer!
- When using specialized databases for logs instead of PostgreSQL and MySQL

How do specialized databases for logs achieve big compression ratios?



How do specialized databases for logs achieve big compression ratios?

- Column-oriented storage - values for the same log field are stored and compressed together

How do specialized databases for logs achieve big compression ratios?

- Column-oriented storage - values for the same log field are stored and compressed together
- Similar values for the same log field compress very well

How do specialized databases for logs achieve big compression ratios?

- Column-oriented storage - values for the same log field are stored and compressed together
- Similar values for the same log field compress very well
- Use specialized codecs depending on the type of log field values (consts, enums, numbers, timestamps, ip addresses, etc.)

How to improve query performance over a petabyte of logs?

How to improve query performance over a petabyte of logs?

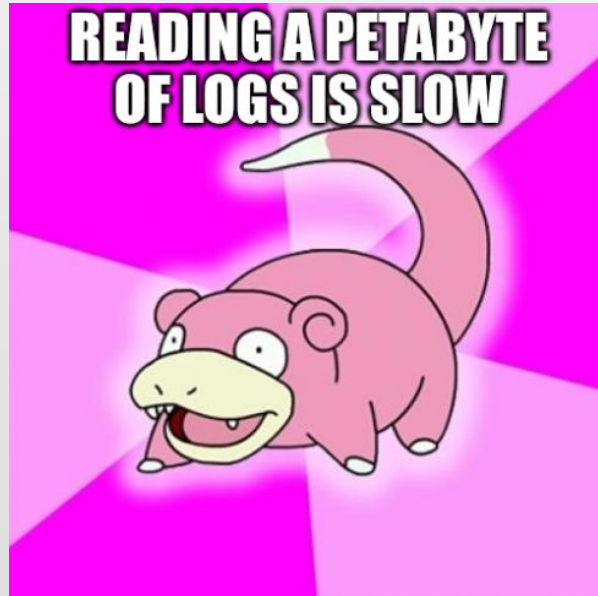
- The maximum read IO for modern disks is less than 4GB/s

How to improve query performance over a petabyte of logs?

- The maximum read IO for modern disks is less than 4GB/s
- Reading a petabyte of logs needs $1\text{PB} / 4\text{GB/s} = 250000\text{s} = \sim 70 \text{ hours}$

How to improve query performance over a petabyte of logs?

- The maximum read IO for modern disks is less than 4GB/s
- Reading a petabyte of logs needs $1\text{PB} / 4\text{GB/s} = 250000\text{s} = \sim 70 \text{ hours}$



How to improve query performance over a petabyte of logs?

- The maximum read IO for modern disks is less than 4GB/s
- Reading a petabyte of logs needs $1\text{PB} / 4\text{GB/s} = 250000\text{s} = \sim 70 \text{ hours}$
- OK, we compressed the logs to 64TB

How to improve query performance over a petabyte of logs?

- The maximum read IO for modern disks is less than 4GB/s
- Reading a petabyte of logs needs $1\text{PB} / 4\text{GB/s} = 250000\text{s} = \sim 70$ hours
- OK, we compressed the logs to 64TB
- Reading 64TB needs $64\text{TB} / 4\text{GB/s} = 16000\text{s} = 4$ hours 27 minutes

How to improve query performance over a petabyte of logs?

- The maximum read IO for modern disks is less than 4GB/s
- Reading a petabyte of logs needs $1\text{PB} / 4\text{GB/s} = 250000\text{s} = \sim 70$ hours
- OK, we compressed the logs to 64TB
- Reading 64TB needs $64\text{TB} / 4\text{GB/s} = 16000\text{s} = 4$ hours 27 minutes
- How to reduce the query time even more?

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time
 - Data size to read: $4\text{TB} / (30 / 3) = 400\text{GB}$

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time
 - Data size to read: $4\text{TB} / (30 / 3) = 400\text{GB}$
 - Query response time: $400\text{GB} / 4\text{GB/s} = 1$ minute 40 seconds

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time
 - Data size to read: $4\text{TB} / (30 / 3) = 400\text{GB}$
 - Query response time: $400\text{GB} / 4\text{GB/s} = 1$ minute 40 seconds
- Partition logs by log streams and read only the data for the needed streams

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time
 - Data size to read: $4\text{TB} / (30 / 3) = 400\text{GB}$
 - Query response time: $400\text{GB} / 4\text{GB/s} = 1$ minute 40 seconds
- Partition logs by log streams and read only the data for the needed streams
 - Read only the logs for 100 log streams out of 1000 log streams

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time
 - Data size to read: $4\text{TB} / (30 / 3) = 400\text{GB}$
 - Query response time: $400\text{GB} / 4\text{GB/s} = 1$ minute 40 seconds
- Partition logs by log streams and read only the data for the needed streams
 - Read only the logs for 100 log streams out of 1000 log streams
 - Data size to read: $400\text{GB} / (1000 / 100) = 40\text{GB}$

How to improve query performance over a petabyte of logs?

- Skip reading data for unneeded log fields
 - Read only 4TB of compressed log messages and skip reading the rest of log fields data
 - Query response time: $4\text{TB} / 4\text{GB/s} = \sim 17$ minutes
- Partition logs by time and read only the data for the needed time range
 - Read only the logs for the last 3 days out of 30 days logs' retention time
 - Data size to read: $4\text{TB} / (30 / 3) = 400\text{GB}$
 - Query response time: $400\text{GB} / 4\text{GB/s} = 1$ minute 40 seconds
- Partition logs by log streams and read only the data for the needed streams
 - Read only the logs for 100 log streams out of 1000 log streams
 - Data size to read: $400\text{GB} / (1000 / 100) = 40\text{GB}$
 - Query response time: $40\text{GB} / 4\text{GB/s} = 10\text{s}$

How to improve query performance over a petabyte of logs?

- Techniques shown on the previous slides allowed reducing query time over a petabyte of logs from 70 hours to 10 seconds!

How to improve query performance over a petabyte of logs?

- Techniques shown on the previous slides allowed reducing query time over a petabyte of logs from 70 hours to 10 seconds!
- But we can do even better for the “search for logs containing some unique or rarely seen phrase over a petabyte of logs” type of queries! For example, search for logs with the given `trace_id`, `user_id` or `ip`.

Bloom filters



Bloom filters

- Probabilistic data structure, which helps determining whether the given item presents in the given set of items

Bloom filters

- Probabilistic data structure, which helps determining whether the given item presents in the given set of items
- Needs less storage space than the summary length of items in the set

Bloom filters

- Probabilistic data structure, which helps determining whether the given item presents in the given set of items
- Needs less storage space than the summary length of items in the set



Bloom filters

- Probabilistic data structure, which helps determining whether the given item presents in the given set of items
- Needs less storage space than the summary length of items in the set
- Let's split logs into word tokens and store the words in bloom filters!

Bloom filters

- Probabilistic data structure, which helps determining whether the given item presents in the given set of items
- Needs less storage space than the summary length of items in the set
- Let's split logs into word tokens and store the words in bloom filters!
- Let's consult bloom filters before reading the actual logs

Bloom filters

- Probabilistic data structure, which helps determining whether the given item presents in the given set of items
- Needs less storage space than the summary length of items in the set
- Let's split logs into word tokens and store the words in bloom filters!
- Let's consult bloom filters before reading the actual logs
- If bloom filters say “there is no such word in this data block”, then just skip reading the data block and inspecting individual logs in it

Bloom filters

- This reduces the amounts of data, which needs to be read from disk, by 10-100 times in practical cases

Bloom filters

- This reduces the amounts of data, which needs to be read from disk, by 10-100 times in practical cases
- This improves “search for a needle in a haystack” query performance by 10-100 times

Bloom filters

- This reduces the amounts of data, which needs to be read from disk, by 10-100 times in practical cases
- This improves “search for a needle in a haystack” query performance by 10-100 times
- So we can scan logs for some unique `trace_id` at the raw speed of:

Bloom filters

- This reduces the amounts of data, which needs to be read from disk, by 10-100 times in practical cases
- This improves “search for a needle in a haystack” query performance by 10-100 times
- So we can scan logs for some unique trace_id at the raw speed of:
 - $\text{disk_read_speed} * \text{compression_rate} * \text{bloom_filters_multiplier}$

Bloom filters

- This reduces the amounts of data, which needs to be read from disk, by 10-100 times in practical cases
- This improves “search for a needle in a haystack” query performance by 10-100 times
- So we can scan logs for some unique trace_id at the raw speed of:
 - $\text{disk_read_speed} * \text{compression_rate} * \text{bloom_filters_multiplier}$
 - $4\text{GB/s} * 16 * 100 = 6.4\text{TB/s!}$

Bloom filters

- This reduces the amounts of data, which needs to be read from disk, by 10-100 times in practical cases
- This improves “search for a needle in a haystack” query performance by 10-100 times
- So we can scan logs for some unique trace_id at the raw speed of:
 - $\text{disk_read_speed} * \text{compression_rate} * \text{bloom_filters_multiplier}$
 - $4\text{GB/s} * 16 * 100 = 6.4\text{TB/s!}$
 - On a single computer!

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?



Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They store row fields close to each other (aka row-oriented storage)

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They store row fields close to each other (aka row-oriented storage)
 - This requires reading data from disk for all the row fields during queries, even if only a few of fields are really needed

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They store row fields close to each other (aka row-oriented storage)
 - This requires reading data from disk for all the row fields during queries, even if only a few of fields are really needed
- They do not store the related rows close to each other on disk

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They store row fields close to each other (aka row-oriented storage)
 - This requires reading data from disk for all the row fields during queries, even if only a few of fields are really needed
- They do not store the related rows close to each other on disk
 - This increases the number of disk seeks for reading individual rows

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They store row fields close to each other (aka row-oriented storage)
 - This requires reading data from disk for all the row fields during queries, even if only a few of fields are really needed
- They do not store the related rows close to each other on disk
 - This increases the number of disk seeks for reading individual rows
 - This increases the amounts of data read from disk, since disks read data in 4KB blocks

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They store row fields close to each other (aka row-oriented storage)
 - This requires reading data from disk for all the row fields during queries, even if only a few of fields are really needed
- They do not store the related rows close to each other on disk
 - This increases the number of disk seeks for reading individual rows
 - This increases the amounts of data read from disk, since disks read data in 4KB blocks
 - This becomes extremely slow and inefficient when millions of individual rows must be read from different locations on disk

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database
 - These indexes must fit the available RAM in order to be fast during data ingestion and querying

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database
 - These indexes must fit the available RAM in order to be fast during data ingestion and querying
 - Computers with terabytes of RAM are rare and very expensive

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database
 - These indexes must fit the available RAM in order to be fast during data ingestion and querying
 - Computers with terabytes of RAM are rare and very expensive
 - These indexes are slow when millions of individual rows must be located on disk in a single query

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database
 - These indexes must fit the available RAM in order to be fast during data ingestion and querying
 - Computers with terabytes of RAM are rare and very expensive
 - These indexes are slow when millions of individual rows must be located on disk in a single query
- They overwrite the existing data on disk in small chunks

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database
 - These indexes must fit the available RAM in order to be fast during data ingestion and querying
 - Computers with terabytes of RAM are rare and very expensive
 - These indexes are slow when millions of individual rows must be located on disk in a single query
- They overwrite the existing data on disk in small chunks
 - B-trees need frequent updates for their nodes during data ingestion

Why PostgreSQL and MySQL aren't good for storing a petabyte of logs per a single computer?

- They use b-tree indexes for fast access to the given row on disk
 - The size of these indexes is usually up to 30% comparing to the size of the data stored in the database
 - These indexes must fit the available RAM in order to be fast during data ingestion and querying
 - Computers with terabytes of RAM are rare and very expensive
 - These indexes are slow when millions of individual rows must be located on disk in a single query
- They overwrite the existing data on disk in small chunks
 - B-trees need frequent updates for their nodes during data ingestion
 - Small overwrites significantly increase both disk read and write IO, since SSD and NVMe disks update data in e.g. erase blocks of 512KB size. E.g. a single-byte update on disk leads to a 512KB read and a 512KB write

What to do when a single computer cannot handle more logs?

What to do when a single computer cannot handle more logs?

- Migrate to a specialized database for storing more logs per computer

What to do when a single computer cannot handle more logs?

- Migrate to a specialized database for storing more logs per computer
- Scale horizontally to multiple computers

What to do when a single computer cannot handle more logs?

- Migrate to a specialized database for storing more logs per computer
- Scale horizontally to multiple computers



Which open-source specialized databases for logs to use?

Which open-source specialized databases for logs to use?

- Elasticsearch

Which open-source specialized databases for logs to use?

- Elasticsearch
- Loki

Which open-source specialized databases for logs to use?

- Elasticsearch
- Loki
- ClickHouse

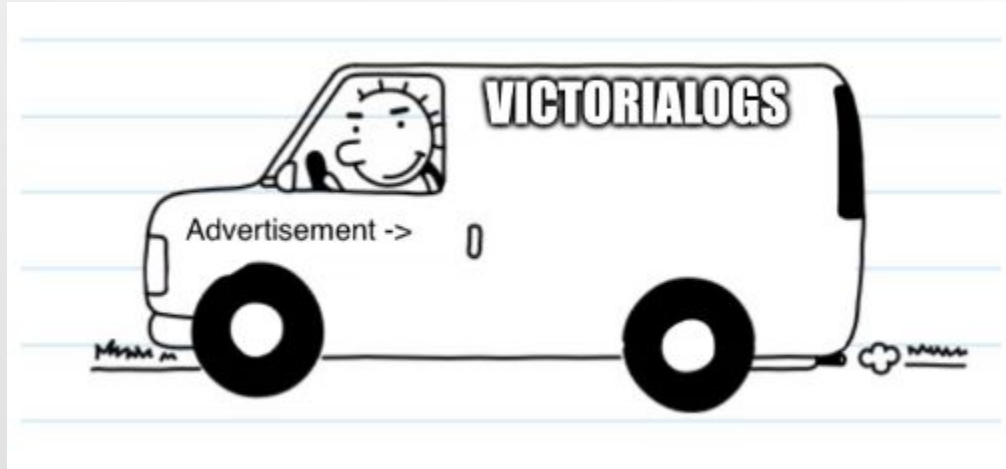
Which open-source specialized databases for logs to use?

- Elasticsearch
- Loki
- ClickHouse
- VictoriaLogs

Which open-source specialized databases for logs to use?

- Elasticsearch - needs a ton of RAM
- Loki - hard to configure and operate
- ClickHouse - needs careful tuning of table schema for high performance
- VictoriaLogs!

Why VictoriaLogs?



Why VictoriaLogs?

- Open-source database optimized for logs

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides
- Can handle hundreds of terabytes of logs per computer

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides
- Can handle hundreds of terabytes of logs per computer
- Can scale horizontally to hundreds of computers for handling tens of petabytes of logs

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides
- Can handle hundreds of terabytes of logs per computer
- Can scale horizontally to hundreds of computers for handling tens of petabytes of logs
- Supports popular data ingestion protocols for logs

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides
- Can handle hundreds of terabytes of logs per computer
- Can scale horizontally to hundreds of computers for handling tens of petabytes of logs
- Supports popular data ingestion protocols for logs
- Very easy to configure and operate - zero-config and schemaless

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides
- Can handle hundreds of terabytes of logs per computer
- Can scale horizontally to hundreds of computers for handling tens of petabytes of logs
- Supports popular data ingestion protocols for logs
- Very easy to configure and operate - zero-config and schemaless
- Runs efficiently on both Raspberry PI and computers with hundreds of CPU cores and terabytes of RAM

Why VictoriaLogs?

- Open-source database optimized for logs
- Includes all the optimizations mentioned on the previous slides
- Can handle hundreds of terabytes of logs per computer
- Can scale horizontally to hundreds of computers for handling tens of petabytes of logs
- Supports popular data ingestion protocols for logs
- Very easy to configure and operate - zero-config and schemaless
- Runs efficiently on both Raspberry PI and computers with hundreds of CPU cores and terabytes of RAM
- Try it in your homelab and in your large-scale production environment!

VictoriaLogs reviews from real users

VictoriaLogs reviews from real users



← Back



fukawi2

@phs@aus.social

Seriously impressed with Victoria Logs.

We're replacing our ElasticSearch cluster (27 nodes, ~588 CPU Cores, with 4656gb RAM) with a single VL Node (8 CPU Cores, 64gb RAM).

We shipped 100m logs to VL in the last hour - box is basically idling with only a couple of GB of RAM used. Any random search query returns in a fraction of a second.

Seriously seriously impressive.

May 28, 2025, 07:59 AM · 🌐 · Web

Last edited May 28, 08:02 AM

74 boosts · 156 favorites

<https://aus.social/@phs/114583927679254536>

On the Mini, I'm running VictoriaLogs, a high performance log database. It's not your average generic indexer like Elasticsearch and its ilk, nor is it anything like a traditional relational database. It shows some resemblance to document stores, but only on the surface. It is specifically built for logs. Structured data is indexed and tokenized. Due to tokenization, the on-disk (and thus, in-memory) data is a lot smaller, and many queries are considerably cheaper, even if performing a full scan: rather than comparing arbitrary long strings, it will compare much smaller, bounded integer tokens in most cases. Feel free to read about how VictoriaLogs works, it's a very educational read!

<https://chronicles.mad-scientist.club/tales/grepping-logs-remains-terrible/>

👉 **Key takeaway:** VictoriaLogs delivered **3×** higher ingestion speed while consuming **72% less CPU** and **87% less memory** compared to Loki.

VictoriaLogs stays comfortably below its **4 vCPU / 8 GiB** limits even during ingestion bursts

TL;DR of the Numbers

- **70–94 % faster** across common search patterns.
- **≈40 % smaller** on disk with the same retention window.
- **Half the compute**, freeing a full vCPU and ~1–2GiB RAM on our smallest nodes.

Bottom line: For a log-heavy, search-centric use case, VictoriaLogs lets us answer questions in seconds instead of minutes while cutting infra costs.

Join VictoriaLogs (and VictoriaMetrics) community!

Join VictoriaLogs (and VictoriaMetrics) community!

- Slack: inviter (<https://slack.victoriametrics.com/>) and channel (<https://victoriametrics.slack.com/>)

Join VictoriaLogs (and VictoriaMetrics) community!

- Slack: inviter (<https://slack.victoriametrics.com/>) and channel (<https://victoriametrics.slack.com/>)
- GitHub: <https://github.com/VictoriaMetrics/VictoriaLogs/>

Join VictoriaLogs (and VictoriaMetrics) community!

- Slack: inviter (<https://slack.victoriametrics.com/>) and channel (<https://victoriametrics.slack.com/>)
- GitHub: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- Mastodon: <https://mastodon.social/@victoriametrics/>

Join VictoriaLogs (and VictoriaMetrics) community!

- Slack: inviter (<https://slack.victoriametrics.com/>) and channel (<https://victoriametrics.slack.com/>)
- GitHub: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- Mastodon: <https://mastodon.social/@victoriametrics/>
- Reddit: <https://www.reddit.com/r/VictoriaMetrics/>

Join VictoriaLogs (and VictoriaMetrics) community!

- Slack: inviter (<https://slack.victoriametrics.com/>) and channel (<https://victoriametrics.slack.com/>)
- GitHub: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- Mastodon: <https://mastodon.social/@victoriametrics/>
- Reddit: <https://www.reddit.com/r/VictoriaMetrics/>
- Telegram: https://t.me/VictoriaMetrics_en

Join VictoriaLogs (and VictoriaMetrics) community!

- Slack: inviter (<https://slack.victoriametrics.com/>) and channel (<https://victoriametrics.slack.com/>)
- GitHub: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- Mastodon: <https://mastodon.social/@victoriametrics/>
- Reddit: <https://www.reddit.com/r/VictoriaMetrics/>
- Telegram: https://t.me/VictoriaMetrics_en
- Twitter: <https://x.com/VictoriaMetrics/>

Useful links

Useful links

- VictoriaLogs sources: <https://github.com/VictoriaMetrics/VictoriaLogs/>

Useful links

- VictoriaLogs sources: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- VictoriaLogs docs: <https://docs.victoriametrics.com/victorialogs/>

Useful links

- VictoriaLogs sources: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- VictoriaLogs docs: <https://docs.victoriametrics.com/victorialogs/>
- Follow me: <http://github.com/valyala/>, <https://x.com/valyala> and <https://www.linkedin.com/in/valyala/>

Questions?

- VictoriaLogs sources: <https://github.com/VictoriaMetrics/VictoriaLogs/>
- VictoriaLogs docs: <https://docs.victoriametrics.com/victorialogs/>
- Follow me: <http://github.com/valyala/>, <https://x.com/valyala> and <https://www.linkedin.com/in/valyala/>