Korean Ver. (slide.1 - 170)

VictoriaMetrics: 시계열 데이터 대혼돈의 멀티버스

손주식, 이선규
NAVER Search

NAVER DEVIEW 2023

CONTENTS

1. Background

1.1 Monitoring NAVER Search System

NAVER DEVIEW 2023

NAVER Search: A Large-Scale Distributed System

- 수십억 개의 검색 요청
- 수만 대의 검색 서버
- 수백 개의 검색 서비스

1.1 Monitoring NAVER Search System

NAVER DEVIEW 2023

NAVER Search: A Large-Scale Distributed System

- 수십억 개의 검색 요청
- 수만 대의 검색 서버
- 수백 개의 검색 서비스

Search SRE: Keeping NAVER Search reliable!!

- NAVER Search의 수 많은 서비스, 시스템을 모니터링
- 장애 예방과 대응

1.1 Monitoring NAVER Search System

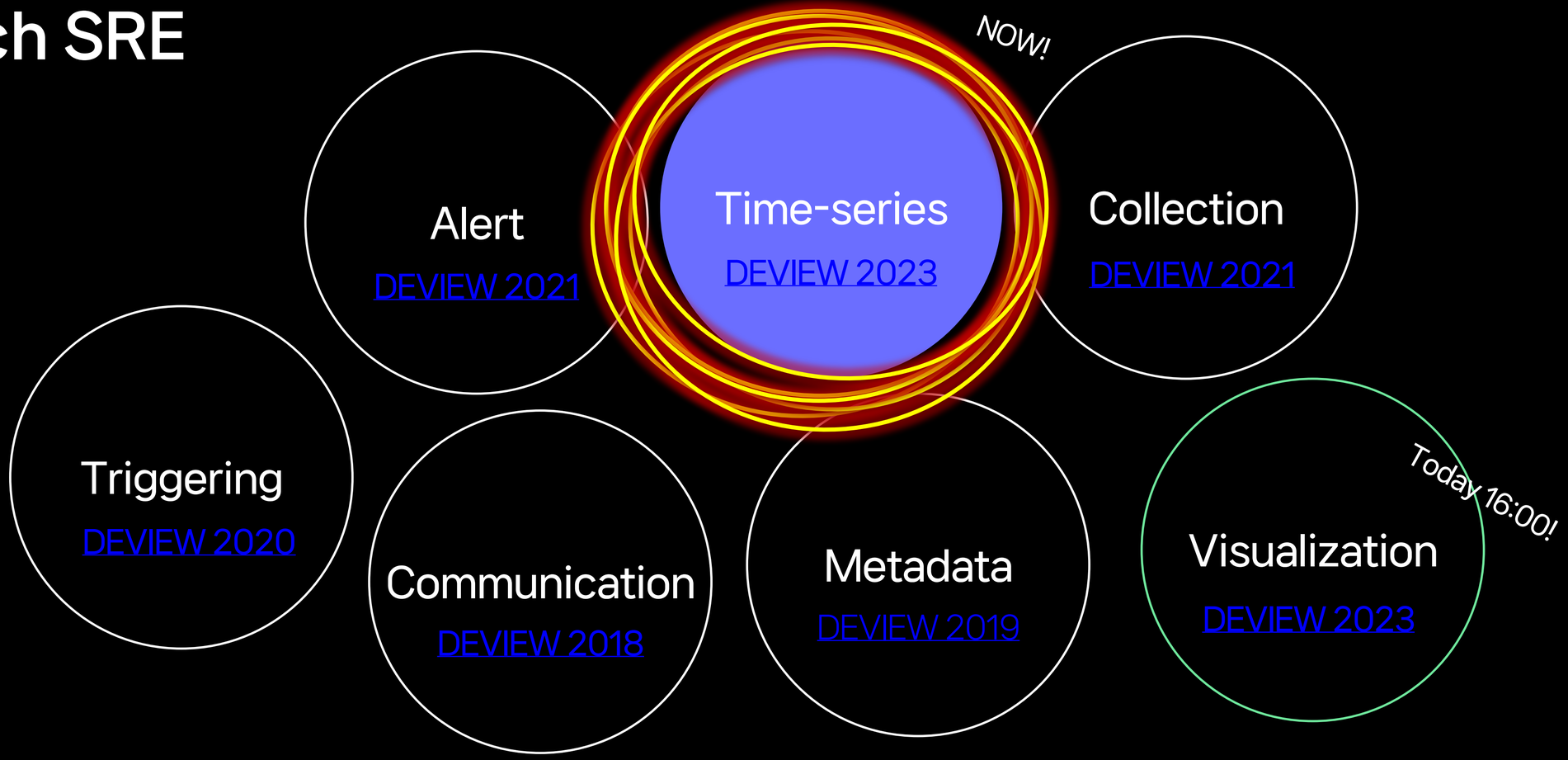
NAVER DEVIEW 2023

Search SRE

1.1 Monitoring NAVER Search System

NAVER DEVIEW 2023

Search SRE



1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is time series data?

- 시간 순서대로 나열된 숫자 데이터

1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is time series data?

- 시간 순서대로 나열된 숫자 데이터

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	…
A	190	170	250	160	220	200	240	200	…
B	250	195	185	260	175	180	200	240	…
C	201	203	205	208	210	222	224	226	…
…	…	…	…	…	…	…	…	…	…

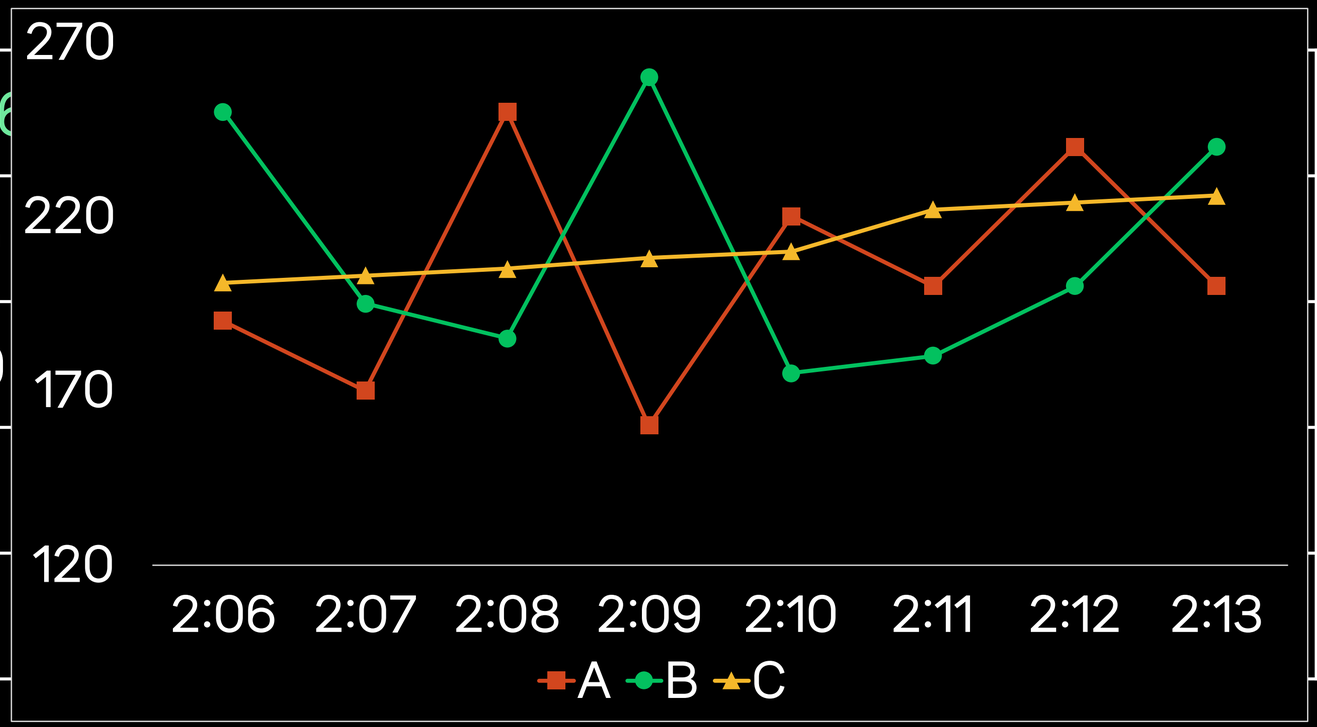
1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is time series data?

- 시간 순서대로 나열된 숫자 데이터

	02:06		02:13	...
A	190		200	...
B	250		240	...
C	201		226	...
...	...		...	...



1.2 Large-scale time series data

NAVER
DEVIEW
2023

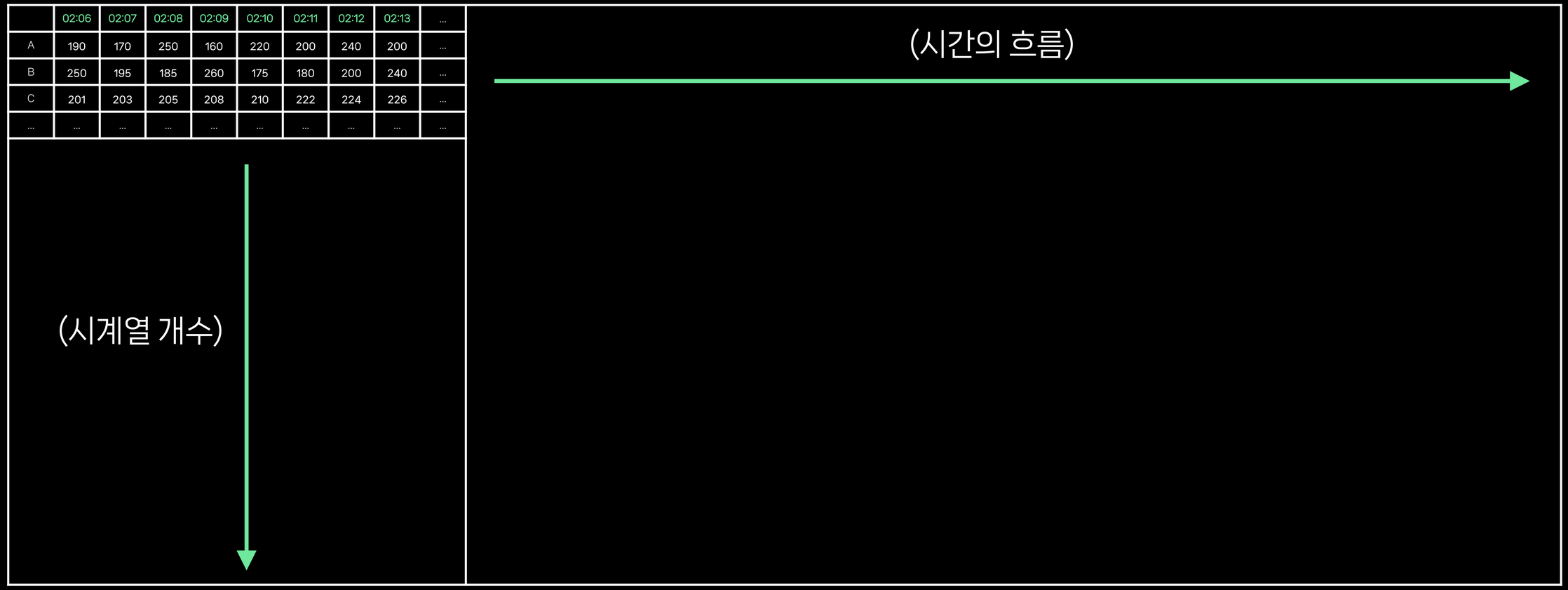
What is large-scale time series data?

1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is large-scale time series data?

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	...
A	190	170	250	160	220	200	240	200	...
B	250	195	185	260	175	180	200	240	...
C	201	203	205	208	210	222	224	226	...
...	...	...	...	...	...	...	...	...	...



1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is large-scale time series data?

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	...
A	190	170	250	160	220	200	240	200	...
B	250	195	185	260	175	180	200	240	...
C	201	203	205	208	210	222	224	226	...
...	...	...	...	...	...	...	...	...	...

(시계열 개수)

(시간의 흐름)

전체 데이터 규모 = (데이터 보관 기간) X (시계열 개수)

데이터를 오래 보관할수록,

시계열의 개수가 많을수록,

1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is large-scale time series data?

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	...
A	190	170	250	160	220	200	240	200	...
B	250	195	185	260	175	180	200	240	...
C	201	203	205	208	210	222	224	226	...
...	...	...	...	...	...	...	...	...	...

(시계열 개수)

(시간의 흐름)

전체 데이터 규모 = (데이터 보관 기간) X (시계열 개수)

데이터를 오래 보관할수록,

시계열의 개수가 많을수록,

대용량 시계열 데이터

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems

참고 : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems
- Stage 1 : 백만 개 → Distributed systems

참고 : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems
- Stage 1 : 백만 개 → Distributed systems
- Stage 2 : 2 백만 개 → Poplular platforms

참고 : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems
- Stage 1 : 백만 개 → Distributed systems
- Stage 2 : 2 백만 개 → Poplular platforms
- Stage 3 : 5 백만 개 → Custom applications

참고 : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems
- Stage 1 : 백만 개 → Distributed systems
- Stage 2 : 2 백만 개 → Poplular platforms
- Stage 3 : 5 백만 개 → Custom applications
- Stage 4 : 수 천만 개 → Microservices

참고 : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems
- Stage 1 : 백만 개 → Distributed systems
- Stage 2 : 2 백만 개 → Poplular platforms
- Stage 3 : 5 백만 개 → Custom applications
- Stage 4 : 수 천만 개 → Microservices
- Stage 5 : 수 십억 개 → Kubernetes

참고 : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

서버 모니터링 환경에서는 얼마나 많은 시계열이 만들어지는가?

- Stage 0 : 백만 개 이하 → Legacy systems
- Stage 1 : 백만 개 → Distributed systems
- Stage 2 : 2 백만 개 → Poplular platforms
- Stage 3 : 5 백만 개 → Custom applications
- Stage 4 : 수 천만 개 → Microservices
- Stage 5 : 수 십억 개 → Kubernetes

} 대용량 시계열 → 별도의 솔루션 필요

참고 : The cost of scale in Prometheus ecosystem

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus와 Gorilla 압축 알고리즘

- 2012년 Prometheus 등장 → De-facto monitoring system

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus와 Gorilla 압축 알고리즘

- 2012년 Prometheus 등장
- 2015년 Gorilla 압축 알고리즘 등장 → Facebook의 Time series 특화 압축 기술

(잠시 후 소개 예정)

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus와 Gorilla 압축 알고리즘

- 2012년 Prometheus 등장
- 2015년 Gorilla 압축 알고리즘 등장
- 수백만 개(Stage 3) 규모의 시계열 데이터는 이제 Prometheus 하나로 해결 가능

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus와 Gorilla 압축 알고리즘

- 2012년 Prometheus 등장
- 2015년 Gorilla 압축 알고리즘 등장
- **수백만 개**(Stage 3) 규모의 시계열 데이터는 이제 Prometheus 하나로 해결 가능
- 그러나, 여전히 수천만 개(Stage 4, 5)를 넘어서는 규모를 다루기는 어려움

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus와 Gorilla 압축 알고리즘

- 2012년 Prometheus 등장
- 2015년 Gorilla 압축 알고리즘 등장
- **수백만 개**(Stage 3) 규모의 시계열 데이터는 이제 Prometheus 하나로 해결 가능
- 그러나, 여전히 수천만 개(Stage 4, 5)를 넘어서는 규모를 다루기는 어려움

대용량 시계열

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus 의 한계를 극복하기 위한 Scalable Solutions

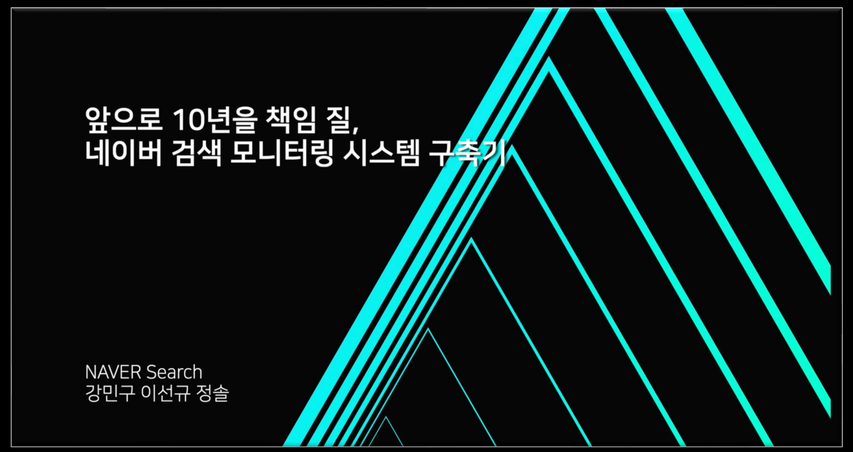
- Thanos
- Cortex
- Grafana Mimir
- M3DB
- Promscale

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus 의 한계를 극복하기 위한 Scalable Solutions

- Thanos
- Cortex
- Grafana Mimir
- M3DB
- Promscale
- VictoriaMetrics ← 표준 호환, 고성능, 쉬운 운영



자세한 이야기는 지난 DEVIEW 2021 에서 소개

참고 : The cost of scale in Prometheus ecosystem

2. A Deep-dive into Time series DB

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

(Request 형태)

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

Time series Name

UNIX Timestamp Value

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

(Request 형태)

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

(저장소에 저장된 형태)

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06					
	190					

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
```

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07				
	190	170				

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
```

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07	02:08			
	190	170	250			

2.1 Data Model

NAVER
DEVIEW
2023

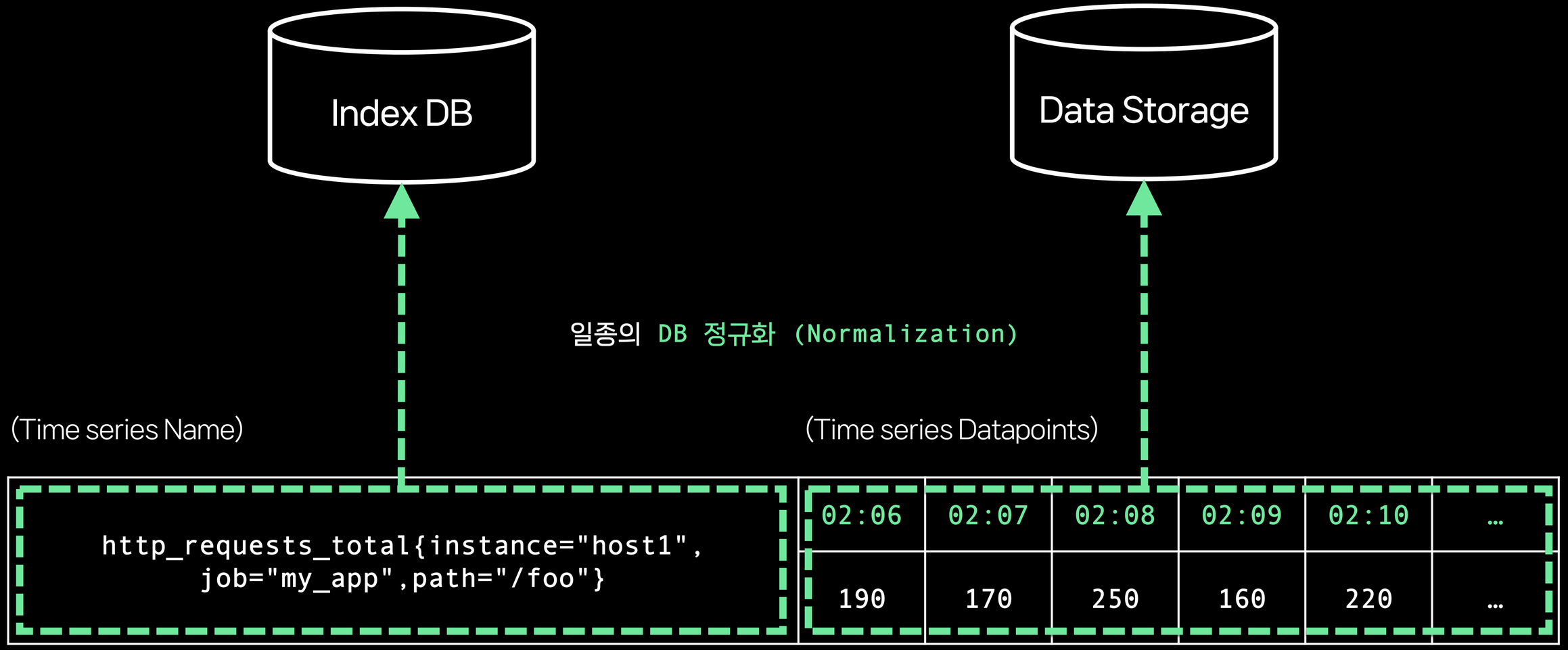
Write Requests Example

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271340  160
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271400  220
…
```

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…

2.1 Data Model

NAVER
DEVIEW
2023



2.1 Data Model

NAVER
DEVIEW
2023

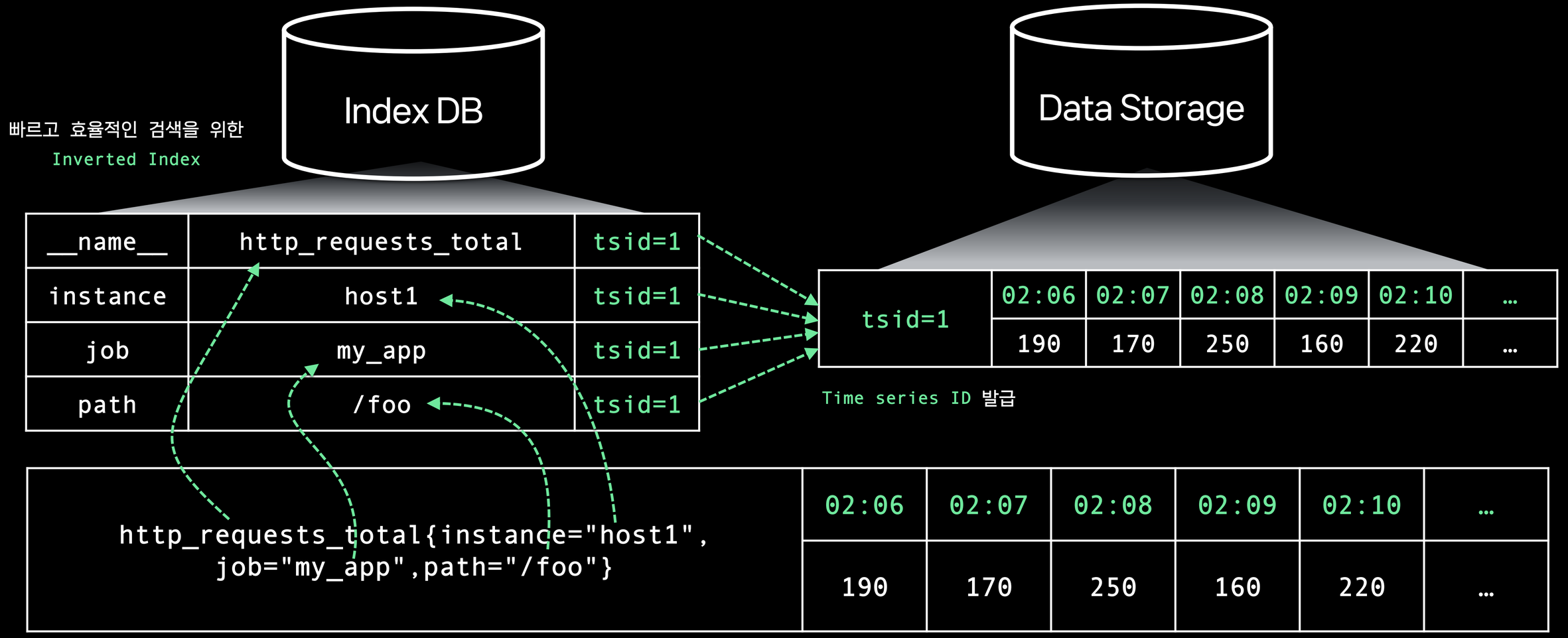
tsid=1	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…

Time series ID 발급

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…

2.1 Data Model

NAVER
DEVIEW
2023



2.1 Data Model

NAVER
DEVIEW
2023

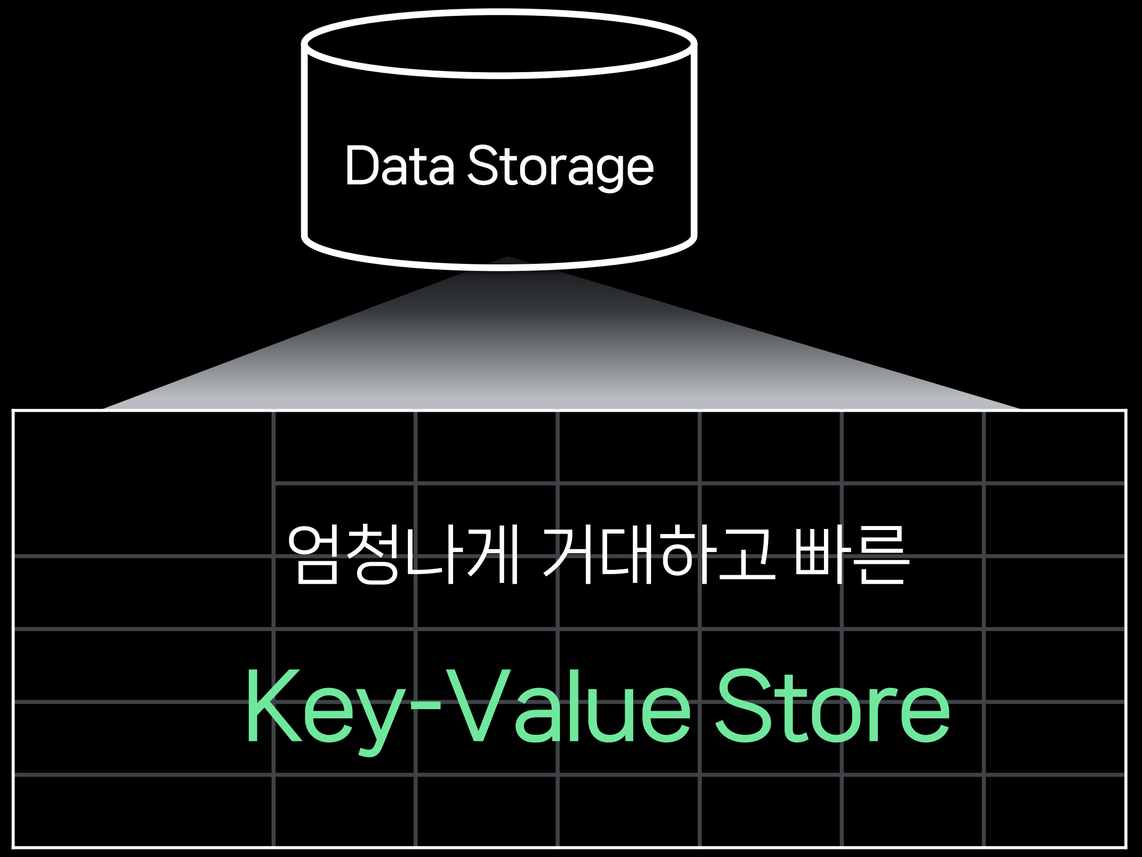
__name__	http_requests_total	tsid=1
instance	host1	tsid=1
job	my_app	tsid=1
path	/foo	tsid=1
		tsid=2
		tsid=2
		tsid=2
…	…	…

tsid=1	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…
tsid=2	...					
tsid=3						
tsid=4						
...						

시계열이 추가될 때마다 빠르게 증가

2.1 Data Model

NAVER
DEVIEW
2023

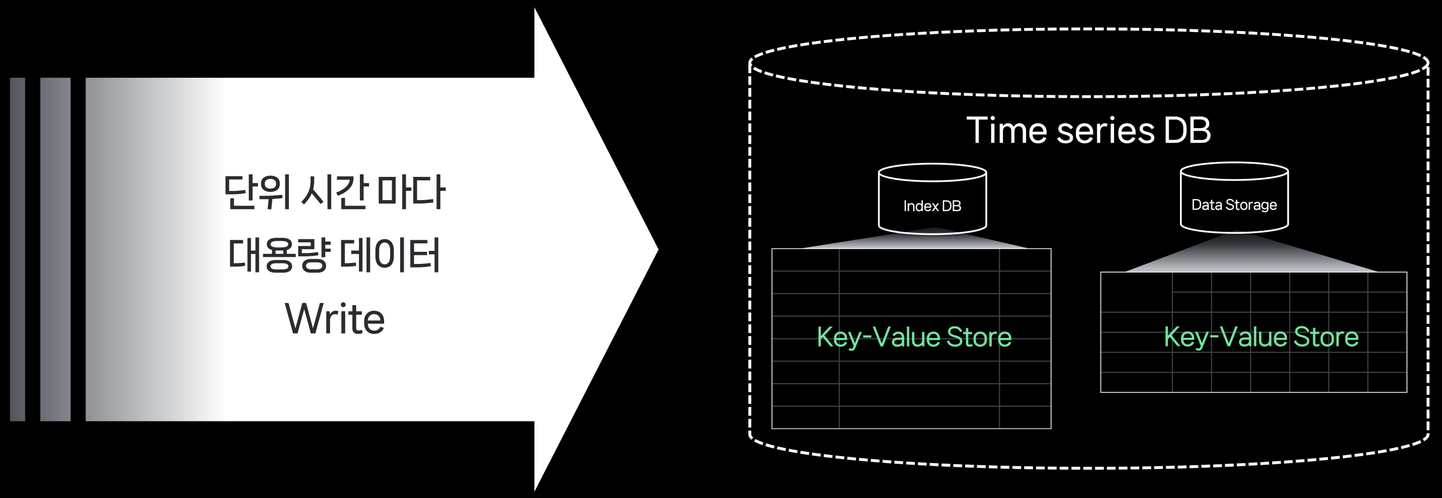


2.2 Key Requirements

NAVER
DEVIEW
2023

Large-scale Write

- 단위 시간 마다 수천만 개 이상의 데이터 유입 → 빠른 대용량 Write 성능 필요

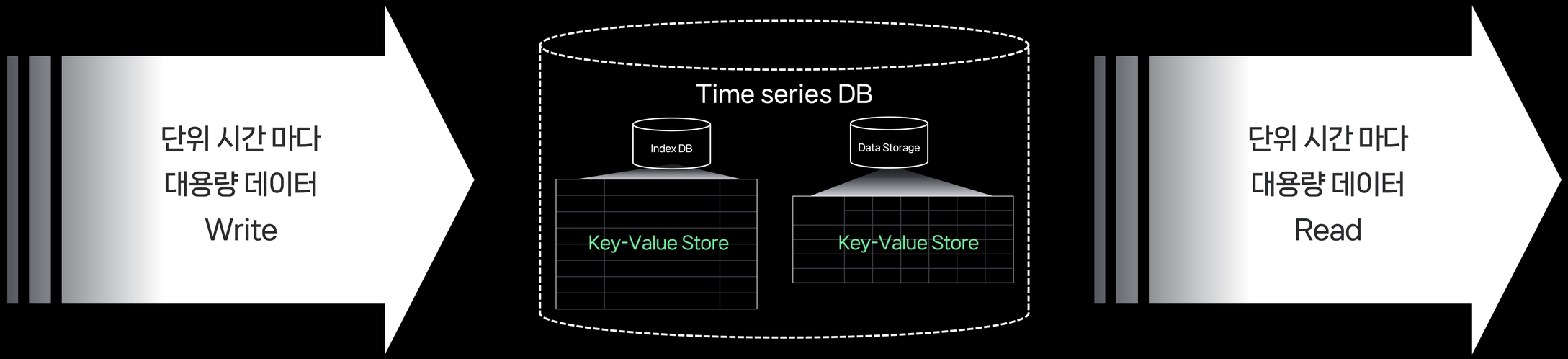


2.2 Key Requirements

NAVER
DEVIEW
2023

Large-scale Write / Read

- 단위 시간 마다 수천만 개 이상의 데이터 유입 → 빠른 대용량 Write 성능 필요
- 단위 시간마다 광범위한 데이터를 조회하여 이상 탐지 → 빠른 대용량 Read 성능 필요



2.2 Key Requirements

NAVER
DEVIEW
2023

Write, Read 둘 다 빠르게?

- Edit가 아닌 Append 위주의 데이터 추가 → Write : O(1)

2.2 Key Requirements

NAVER
DEVIEW
2023

Write, Read 둘 다 빠르게?

- Edit가 아닌 Append 위주의 데이터 추가 → Write : O(1)
- 데이터가 항상 정렬되어 있도록 유지 → Read : O(log n) - O(n)

2.2 Key Requirements

NAVER
DEVIEW
2023

Write, Read 둘 다 빠르게?

- Edit가 아닌 Append 위주의 데이터 추가 → Write : O(1)
- 데이터가 항상 정렬되어 있도록 유지 → Read : O(log n) ~ O(n)

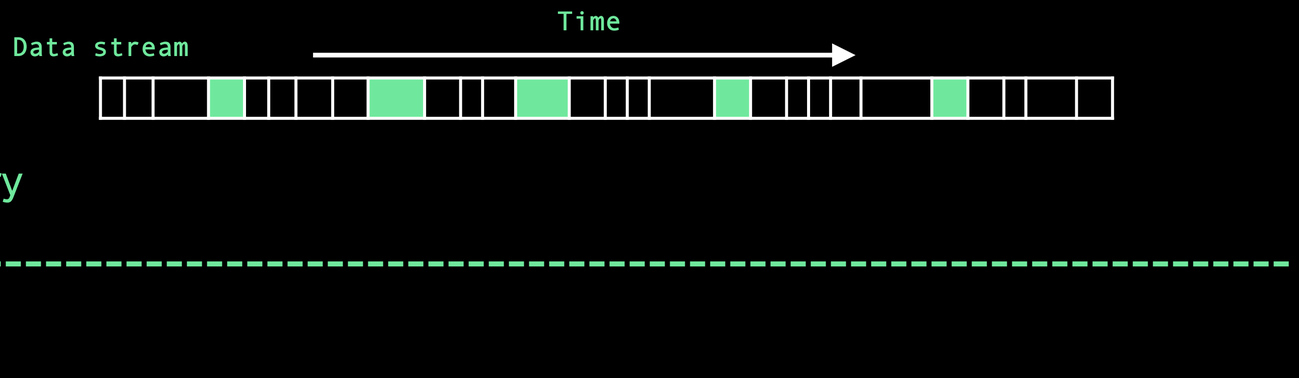
(HBase, Cassandra 등 NoSQL에서 사용하는...)

LSM(Log Structured Merge) Tree!

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Write 과정



[참고] Log Structured Merge Trees

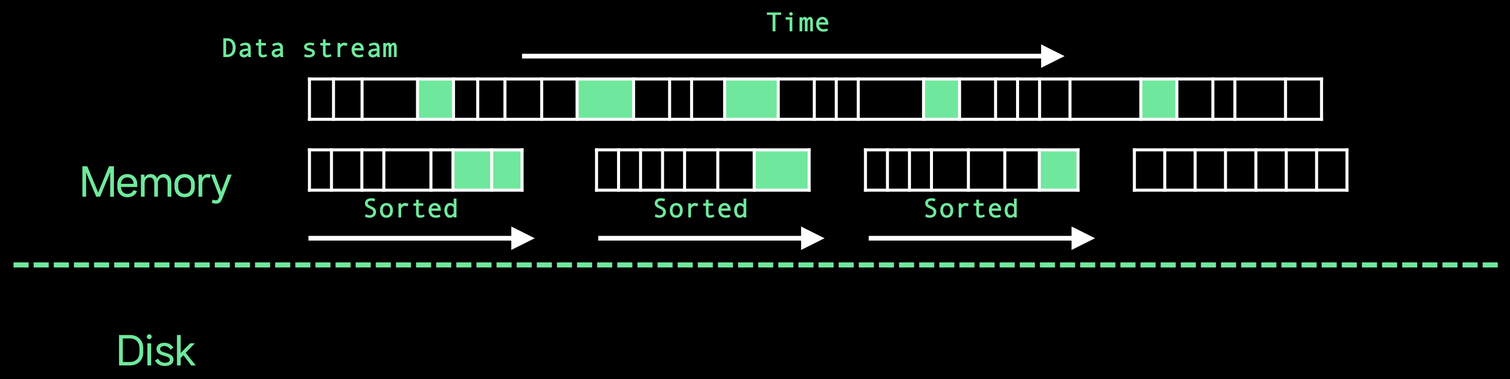
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Write 과정

- 데이터 스트림을 받아서 **작은 조각 단위**로 메모리 상에서 **정렬**



[참고] Log Structured Merge Trees

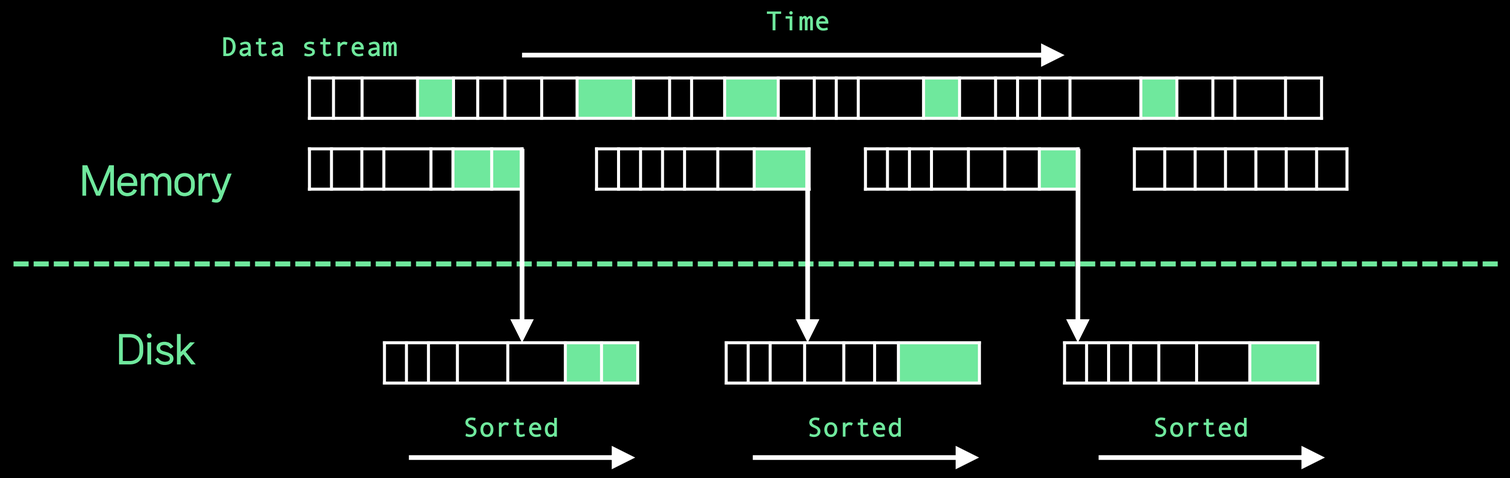
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Write 과정

- 데이터 스트림을 받아서 **작은 조각 단위**로 메모리 상에서 **정렬**
- 정렬된 데이터는 주기적으로 **파일 저장** (Flush)



[참고] Log Structured Merge Trees

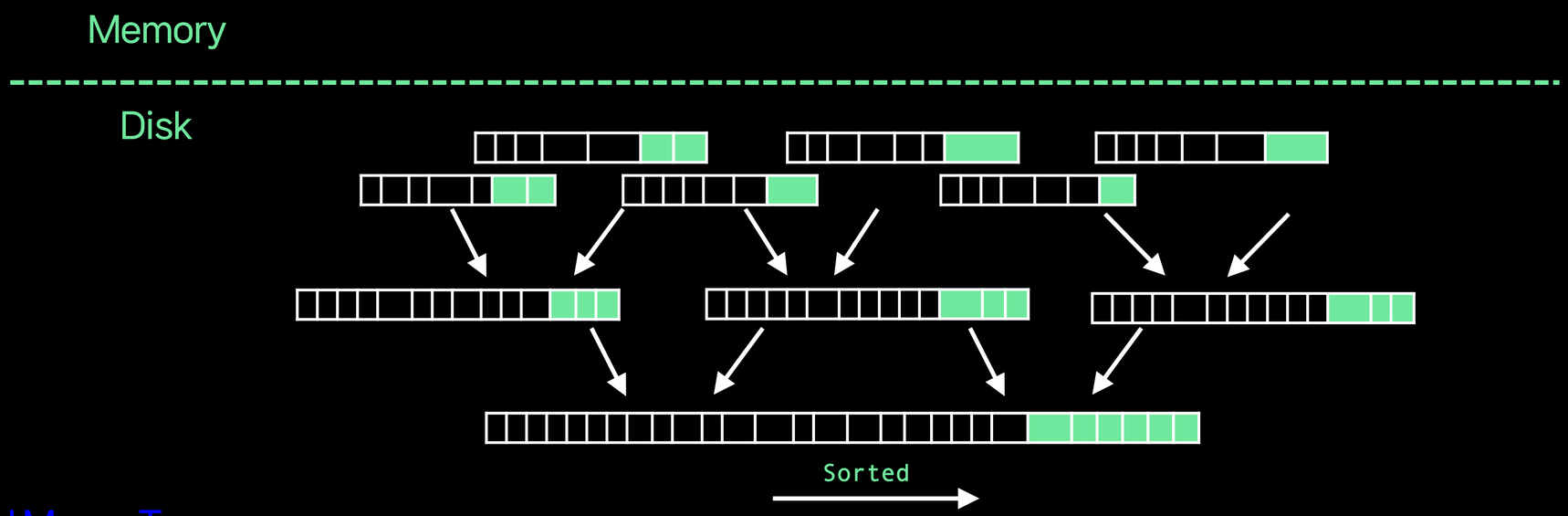
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Merge 과정

- 작은 조각 파일들은 주기적으로 Merge



[참고] Log Structured Merge Trees

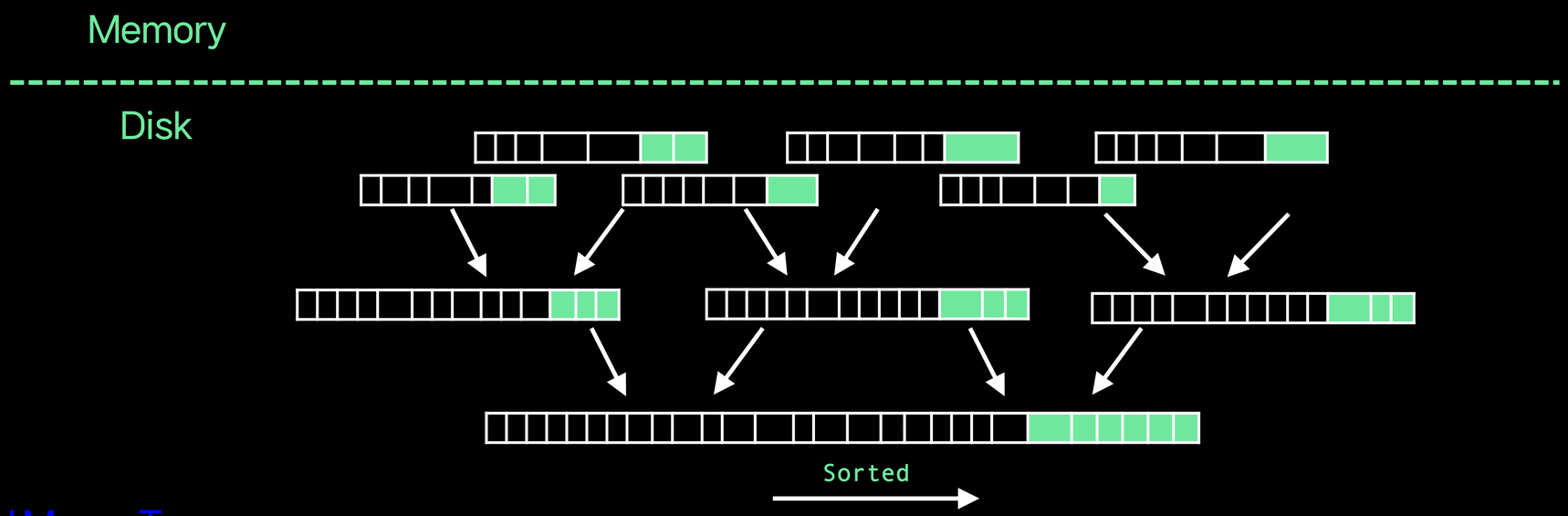
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Merge 과정

- 작은 조각 파일들은 주기적으로 Merge
- 이미 정렬된 데이터이므로 불변 상태 유지 하면서 빠르게 처리



[참고] Log Structured Merge Trees

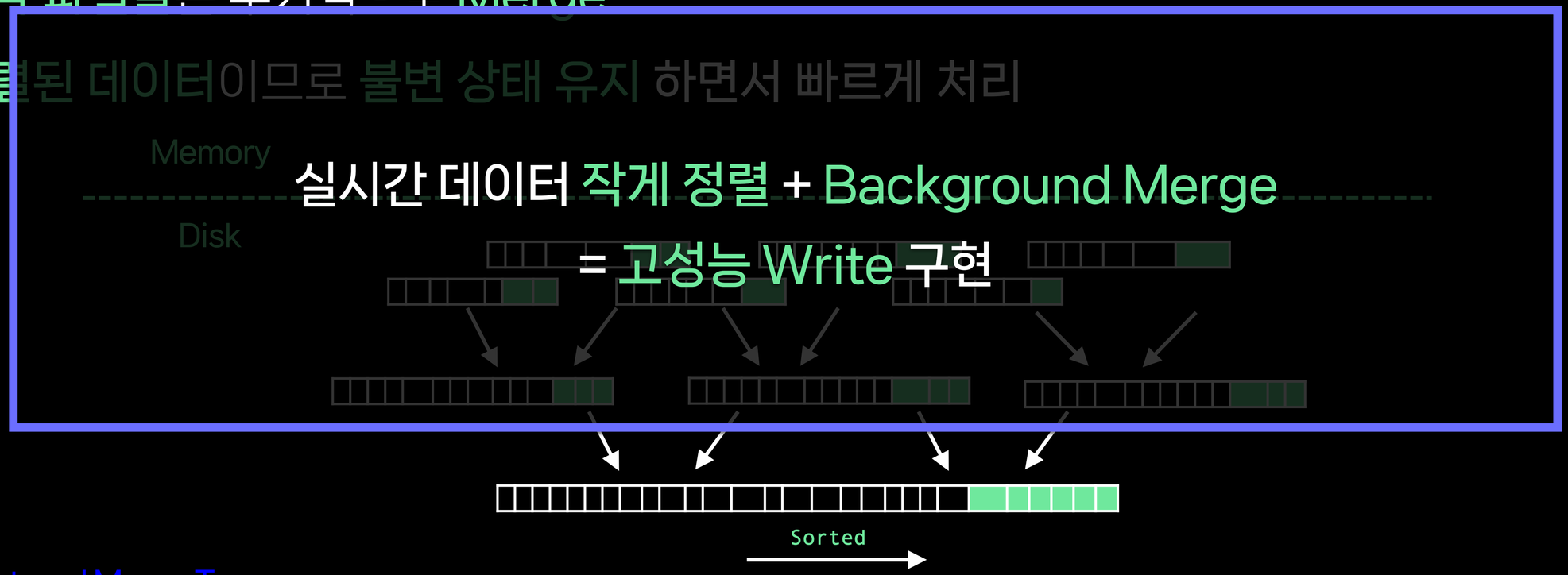
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Merge 과정

- 작은 조각 파일들은 주기적으로 Merge
- 이미 정렬된 데이터이므로 불변 상태 유지 하면서 빠르게 처리



[참고] Log Structured Merge Trees

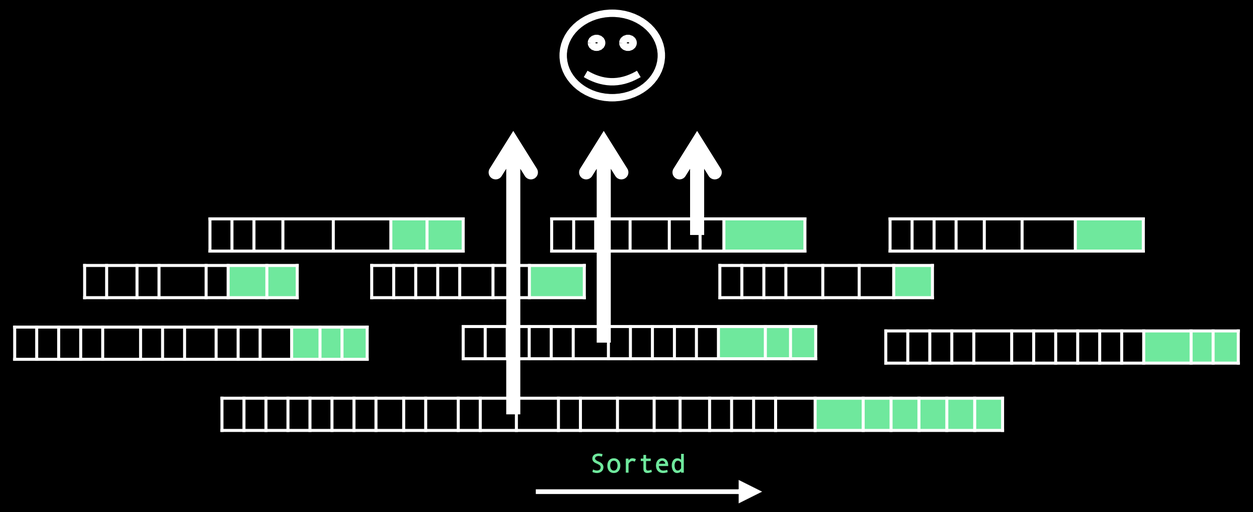
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Read 과정

- 이진 탐색으로 빠르게 데이터 조회



[참고] Log Structured Merge Trees

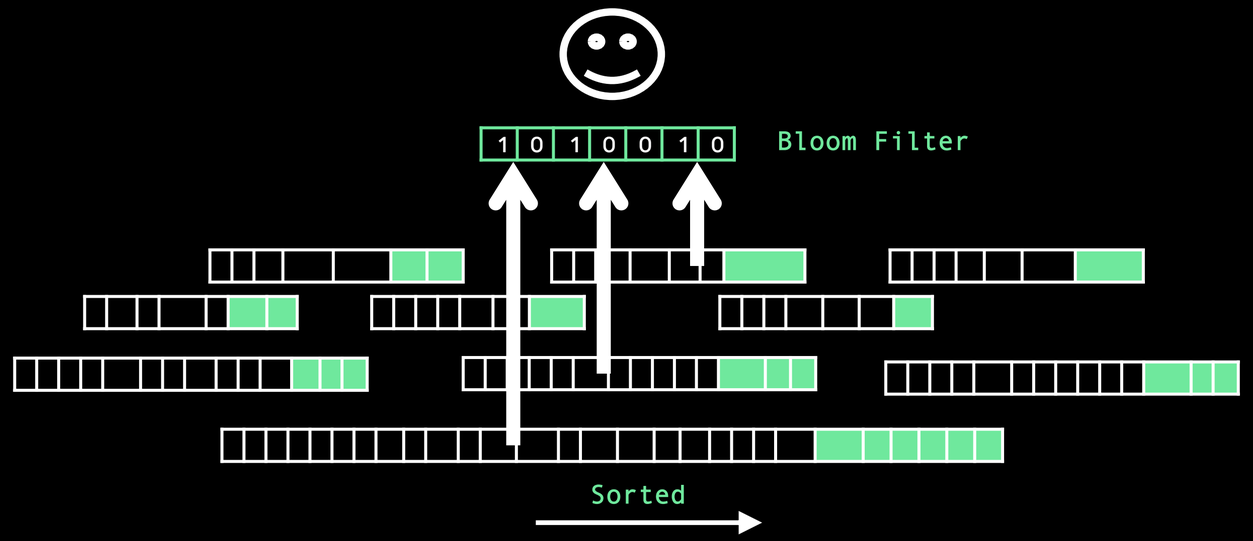
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Read 과정

- 이진 탐색으로 빠르게 데이터 조회
- 여러 파일에서 조회를 한다는 단점은 Bloom Filter를 통해 완화



[참고] Log Structured Merge Trees

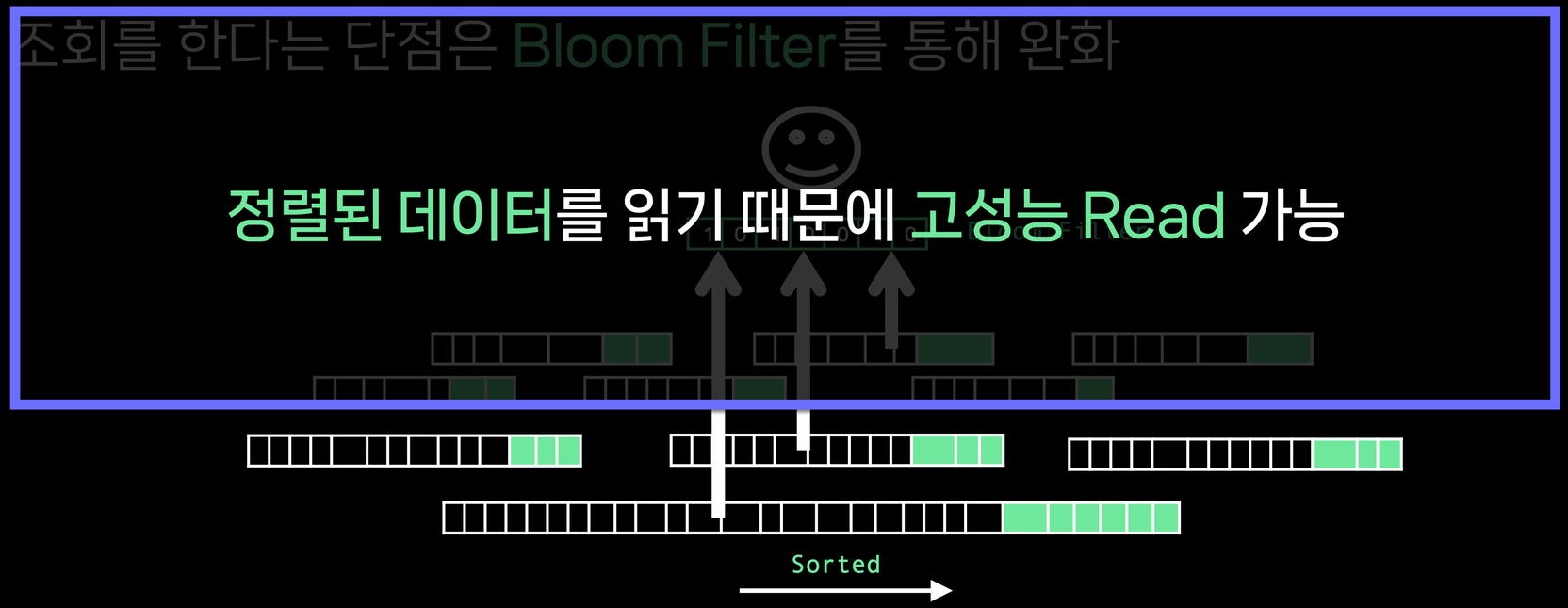
[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Read 과정

- 이진 탐색으로 빠르게 데이터 조회
- 여러 파일에서 조회를 한다는 단점은 Bloom Filter를 통해 완화

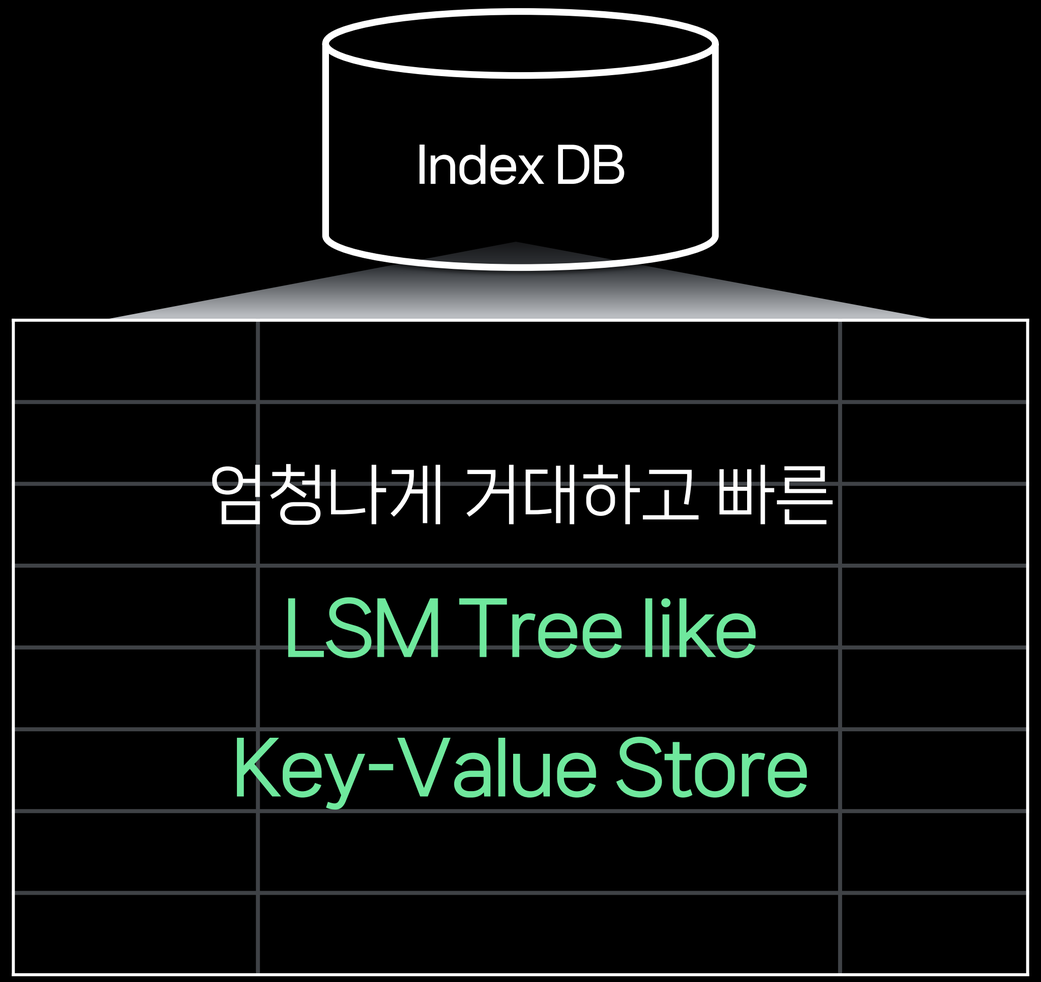


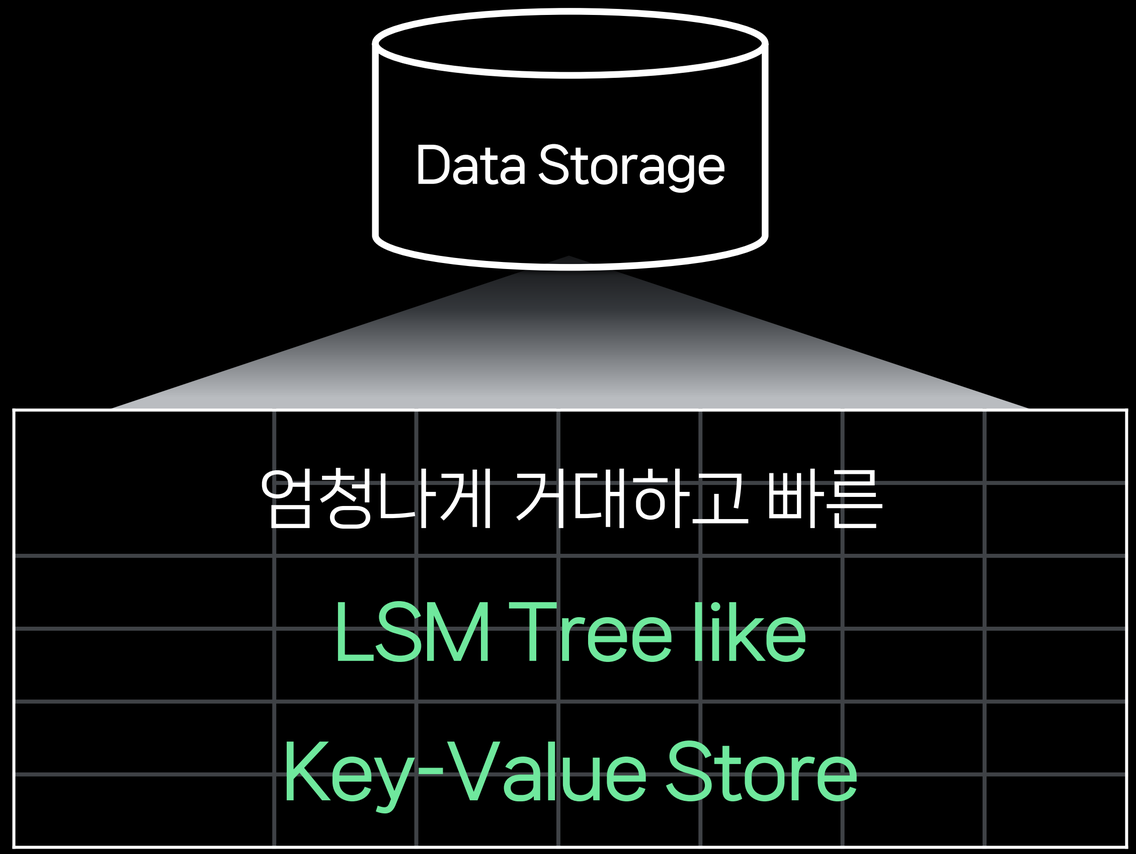
[참고] Log Structured Merge Trees

[참고] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023





2.4 Index DB

NAVER
DEVIEW
2023

Index DB 동작 과정 소개

2.4 Index DB

NAVER
DEVIEW
2023

Write Request

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

2.4 Index DB

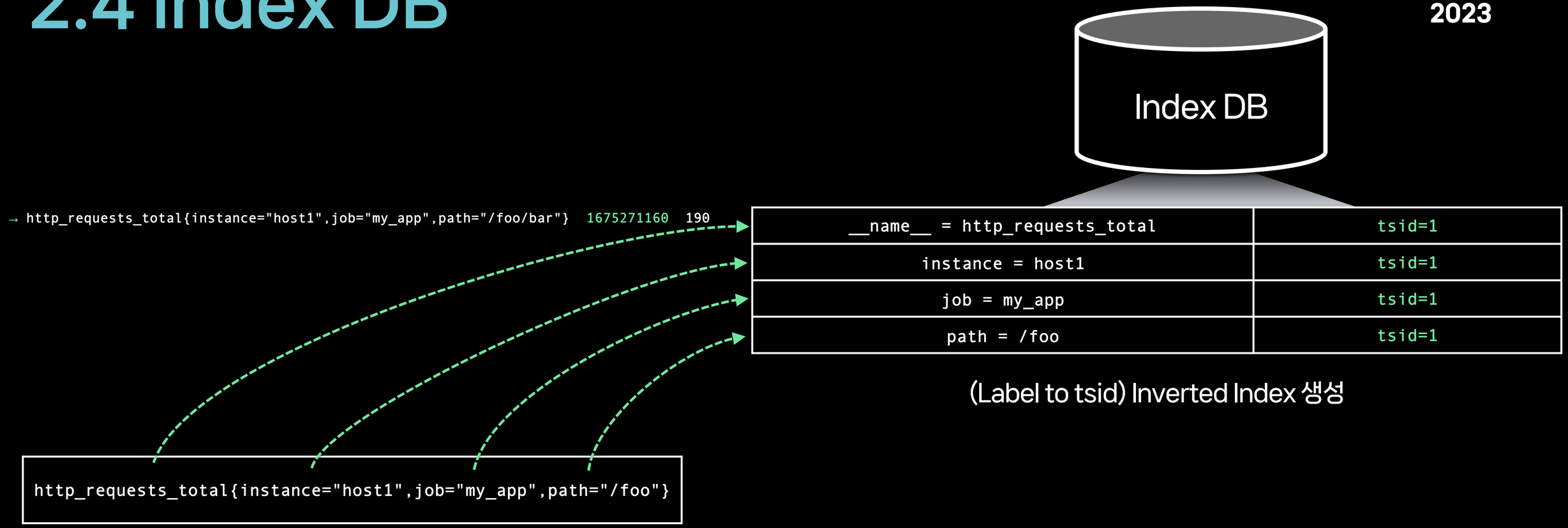
NAVER
DEVIEW
2023


```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

Time series name with labels

2.4 Index DB

NAVER
DEVIEW
2023



(Label to tsid) Inverted Index 생성

2.4 Index DB

NAVER
DEVIEW
2023


```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

```
http_requests_total{instance="host1",job="my_app",path="/foo"}
```

Per-day Index 생성

2.4 Index DB

NAVER
DEVIEW
2023


```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

```
http_requests_total{instance="host1",job="my_app",path="/foo"}
```

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

빠른 조회를 위한 Metric Name Index 생성

2.4 Index DB

NAVER
DEVIEW
2023


```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

`http_requests_total{instance="host1",job="my_app",path="/foo"}`

역방향 Index 생성

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

tsid=1	http_requests_total{instance="host1",job="my_app",path="/foo/bar"}

2.4 Index DB

NAVER
DEVIEW
2023

→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"} 1675271160 190

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1
2023-02-28	tsid=1
2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

http_requests_total{instance="host1",job="my_app",path="/foo"}

꽤 절차가 많은 편

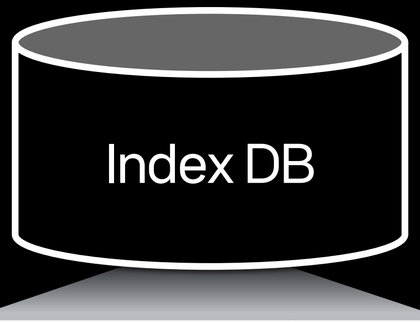
시계열이 처음 Write 될 때만 수행되는 Slow Insert

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

tsid=1	http_requests_total{instance="host1",job="my_app",path="/foo/bar"}

2.4 Index DB

NAVER
DEVIEW
2023




```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271340  160
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271400  220
```

같은 시계열에 추가 데이터 유입

http_requests_total{instance="host1",job="my_app",path="/foo"}

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

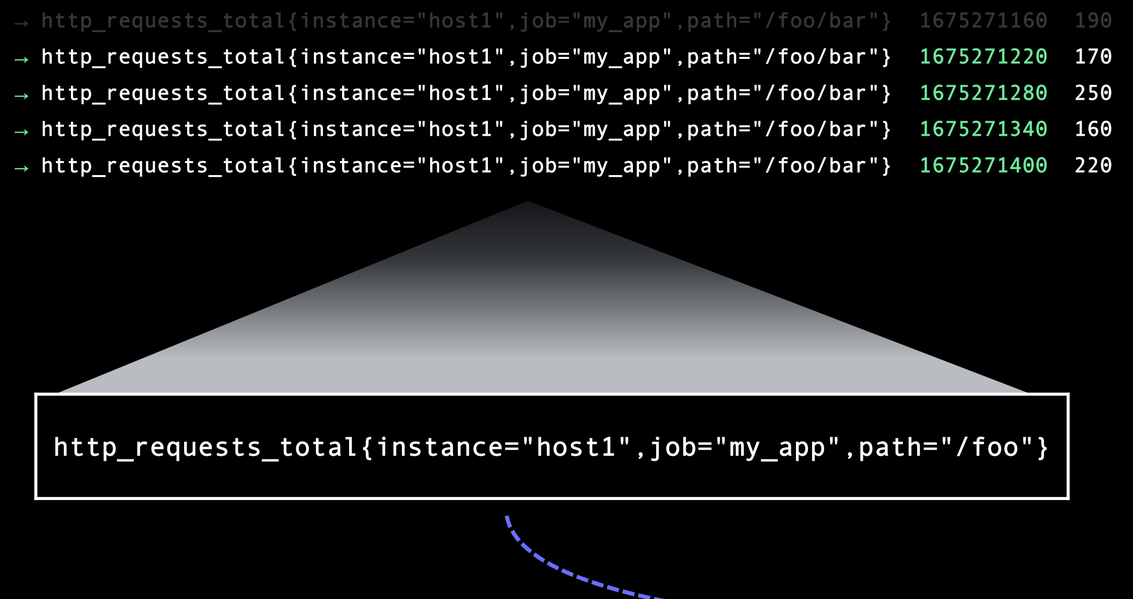
2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

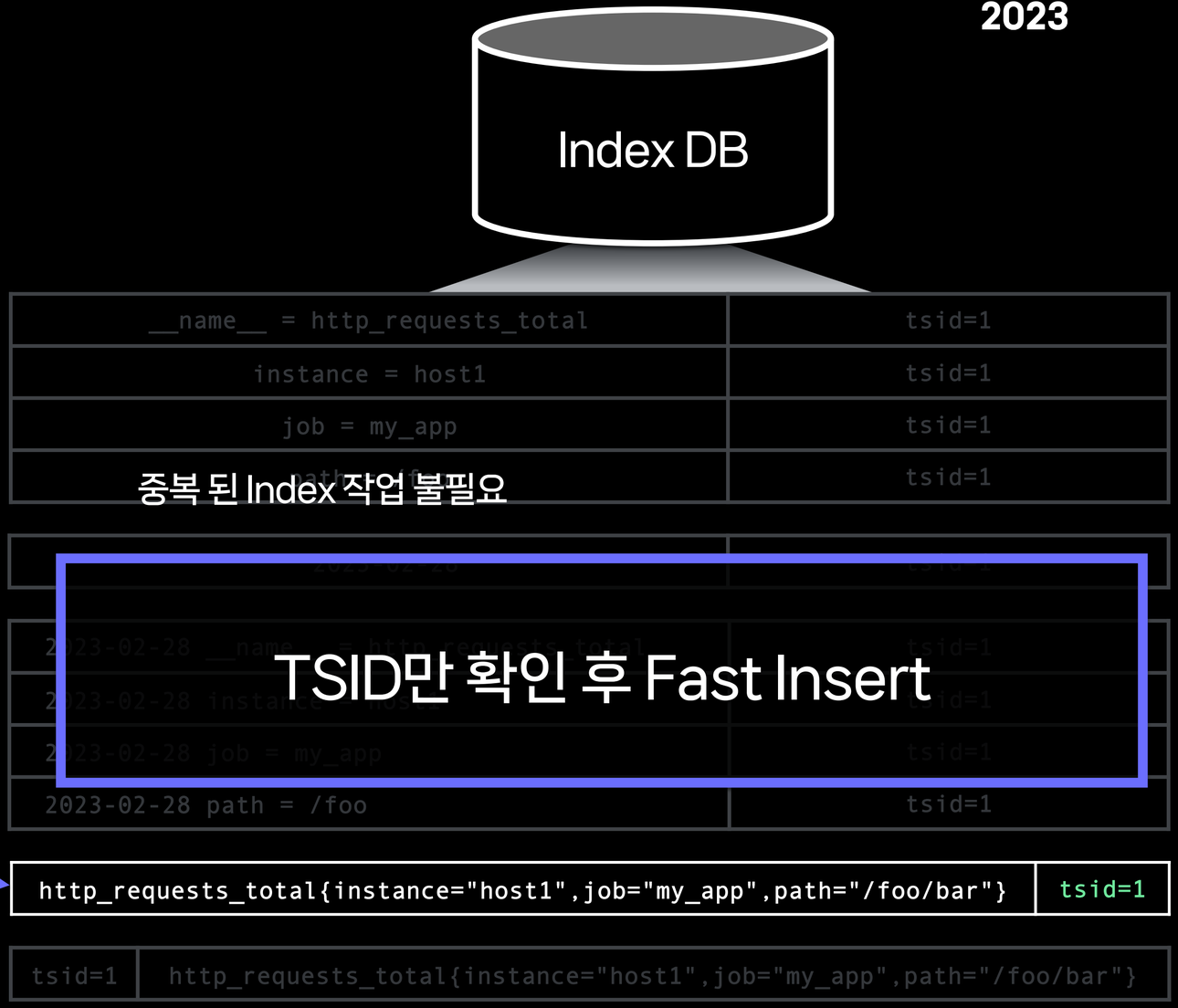
http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

tsid=1	http_requests_total{instance="host1",job="my_app",path="/foo/bar"}

2.4 Index DB

NAVER
DEVIEW
2023





2.5 Data Storage

NAVER
DEVIEW
2023

Data Storage 동작 과정 소개

2.5 Data Storage

NAVER
DEVIEW
2023

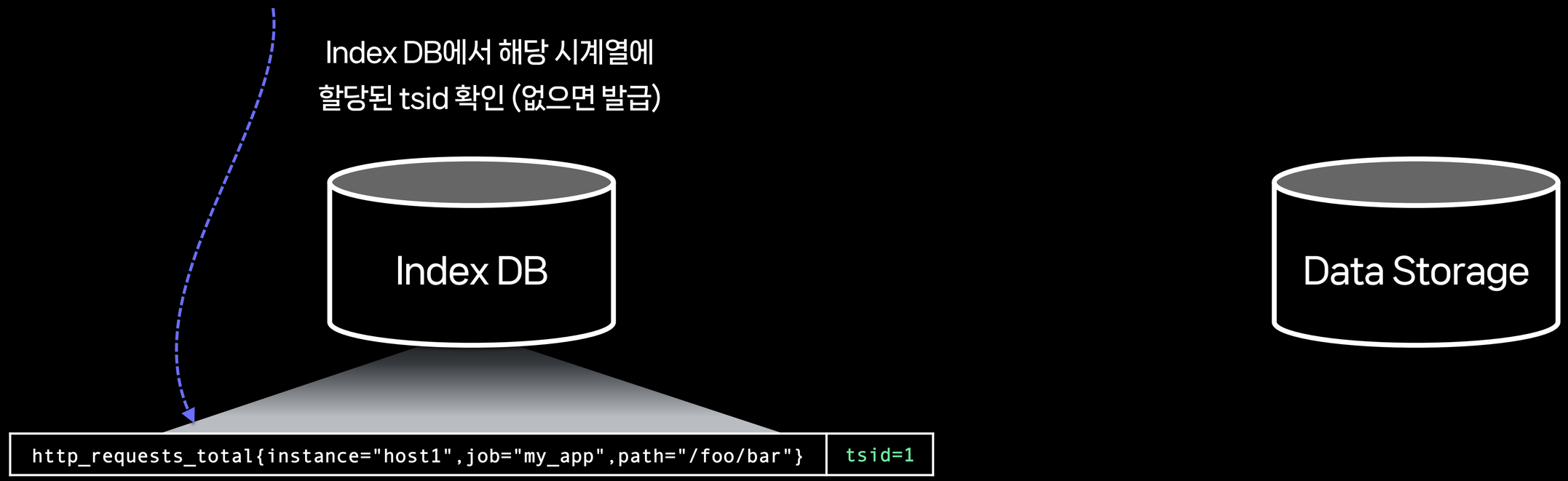
(Write Request)

Metric Name	UNIX Timestamp	Value
→ `http_requests_total{instance="host1",job="my_app",path="/foo/bar"}`	`1675271160`	`190`

2.5 Data Storage

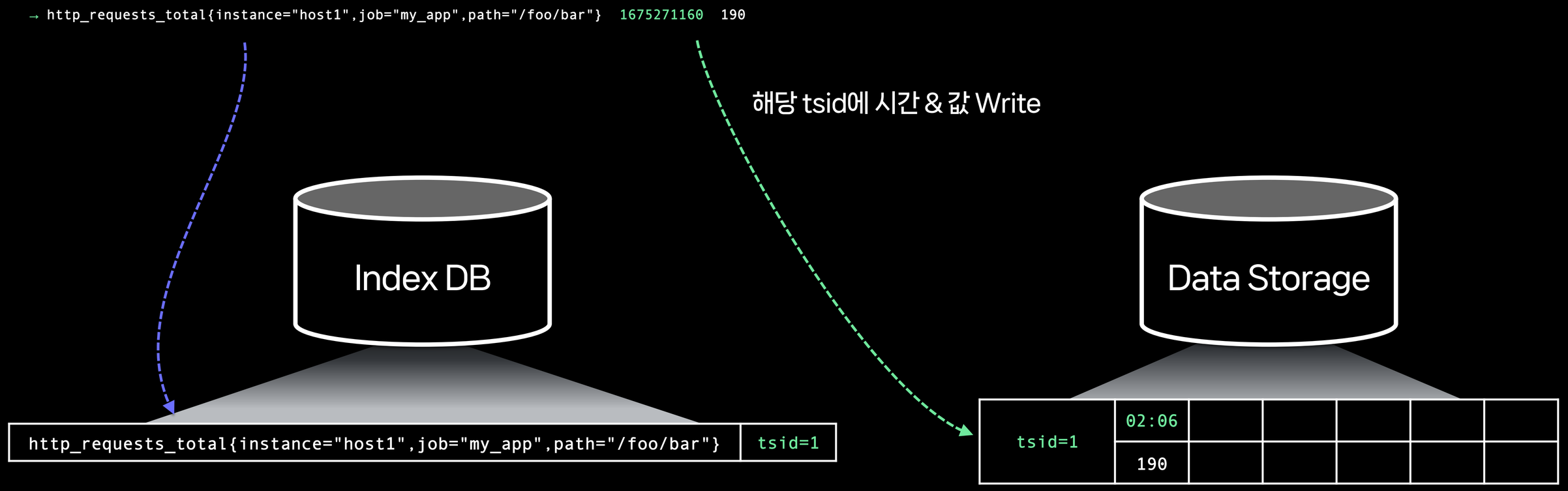
NAVER
DEVIEW
2023

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```



2.5 Data Storage

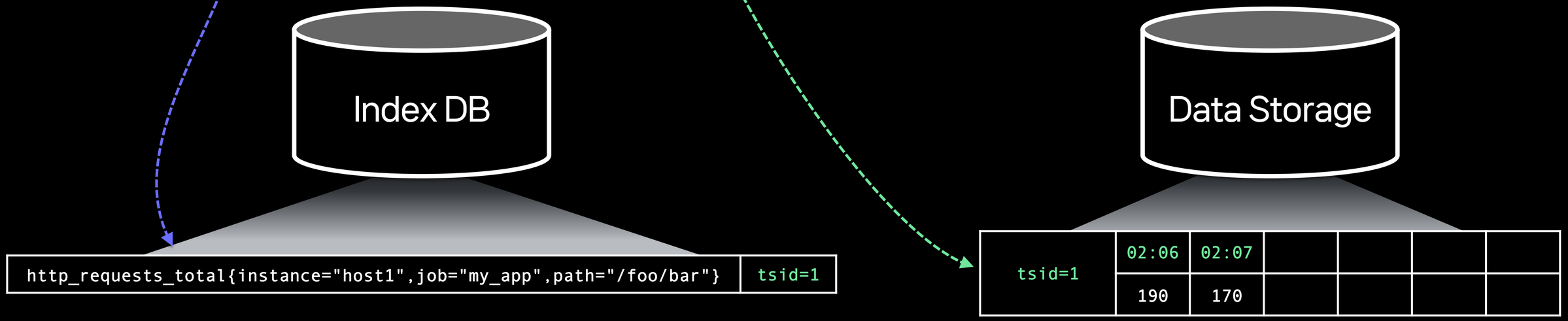
NAVER
DEVIEW
2023



2.5 Data Storage

NAVER
DEVIEW
2023

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
```

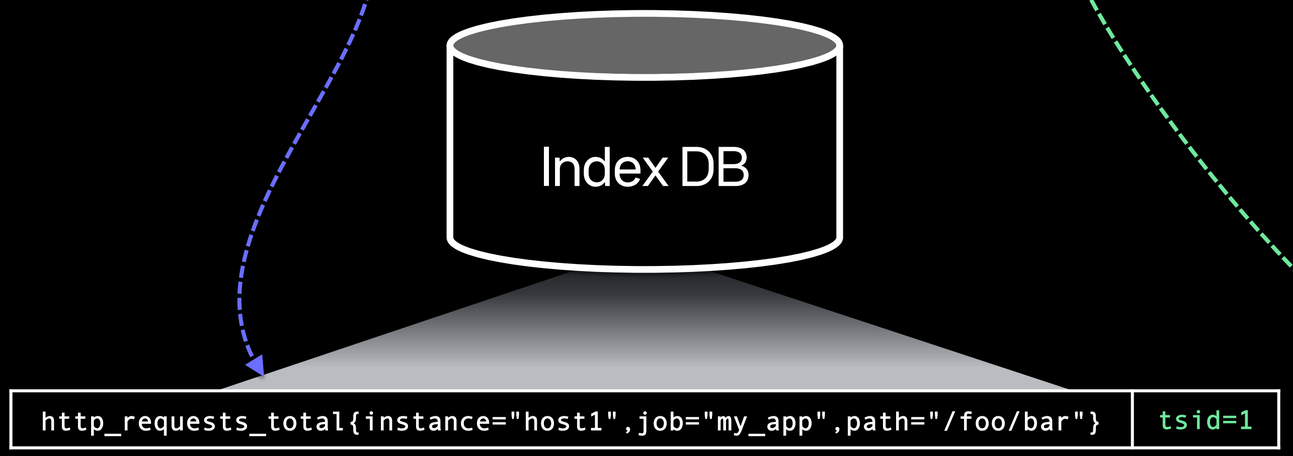


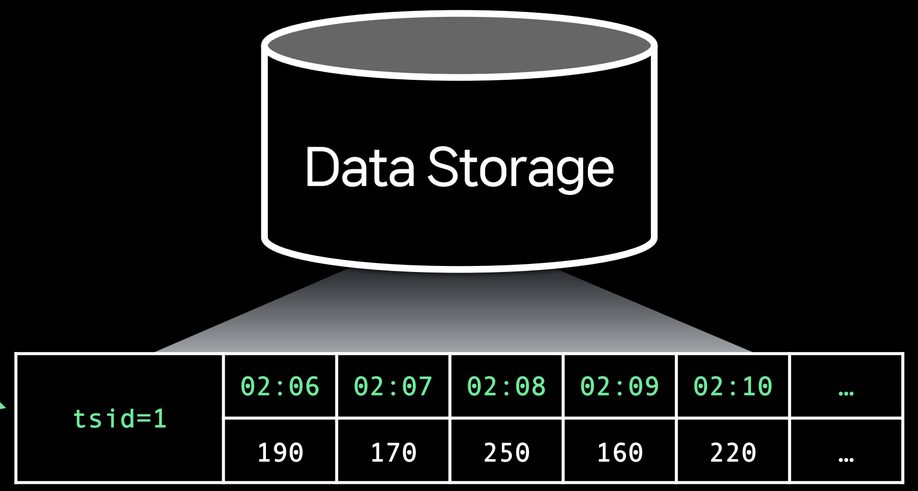
Data Storage에 차곡차곡 저장되는 시계열 데이터

2.5 Data Storage

NAVER
DEVIEW
2023

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271340  160
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271400  220
```

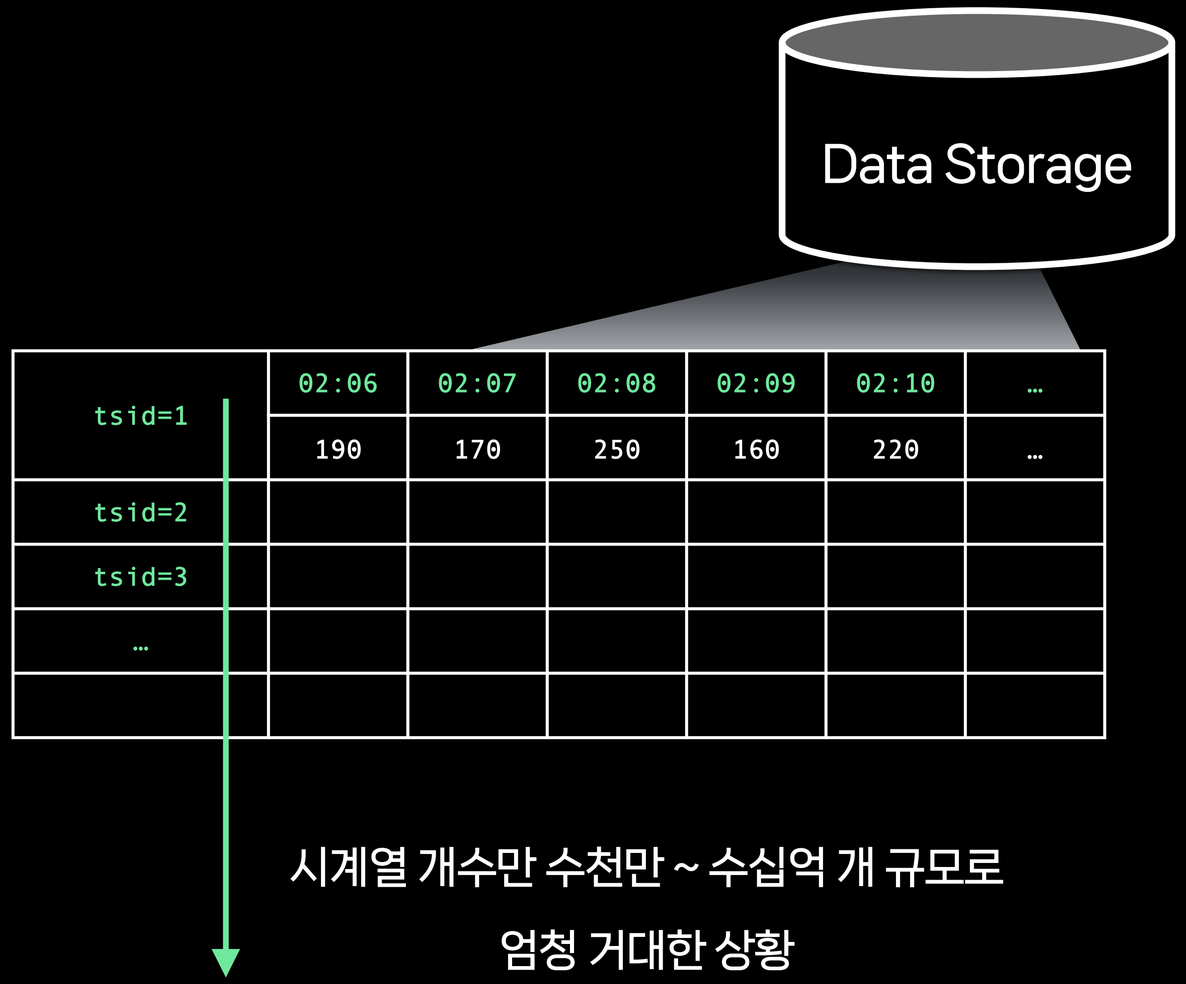




Data Storage에 차곡차곡 저장되는 시계열 데이터

2.5 Data Storage

NAVER
DEVIEW
2023

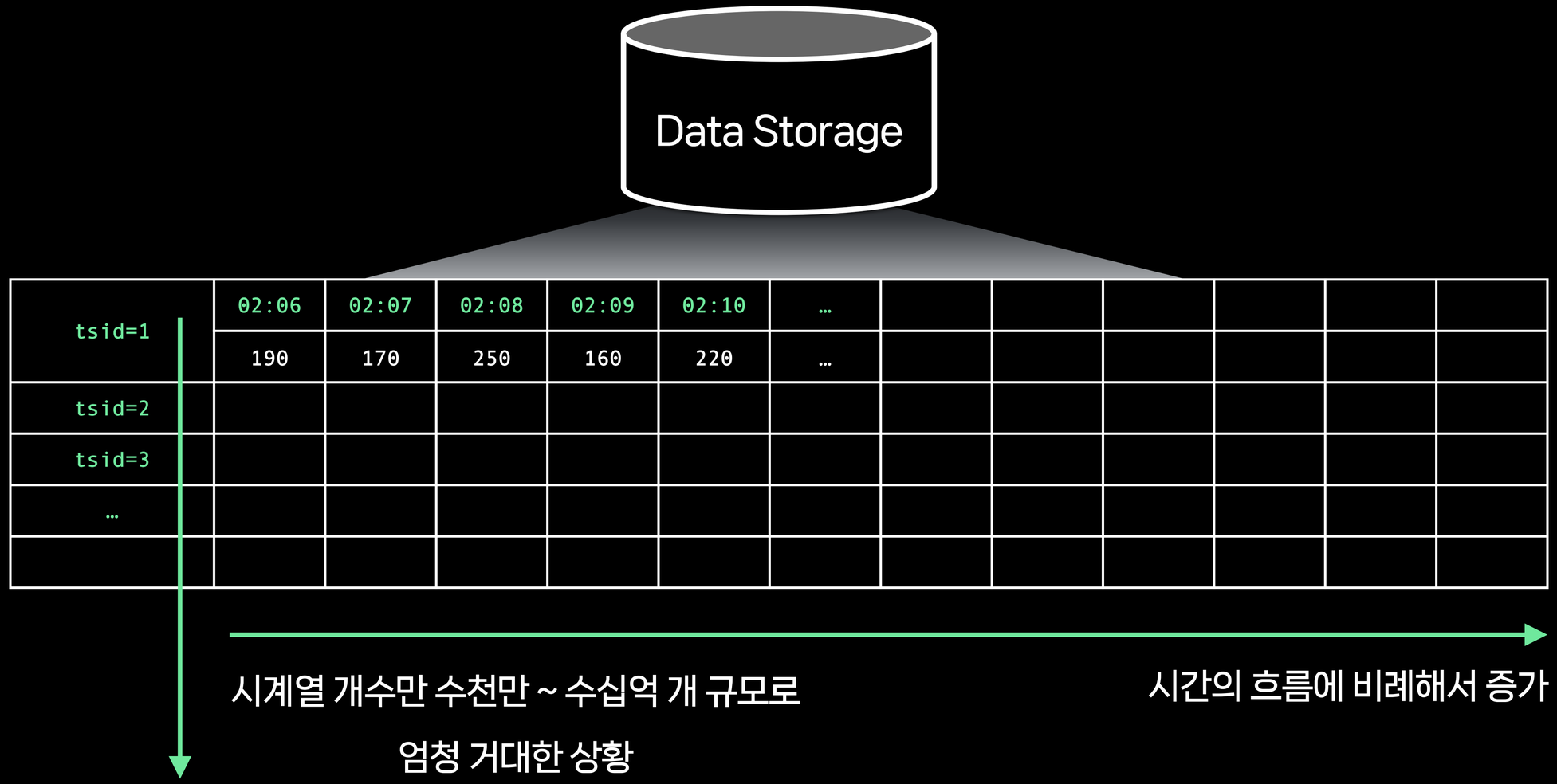


시계열 개수만 수천만 ~ 수십억 개 규모로

엄청 거대한 상황

2.5 Data Storage

NAVER
DEVIEW
2023



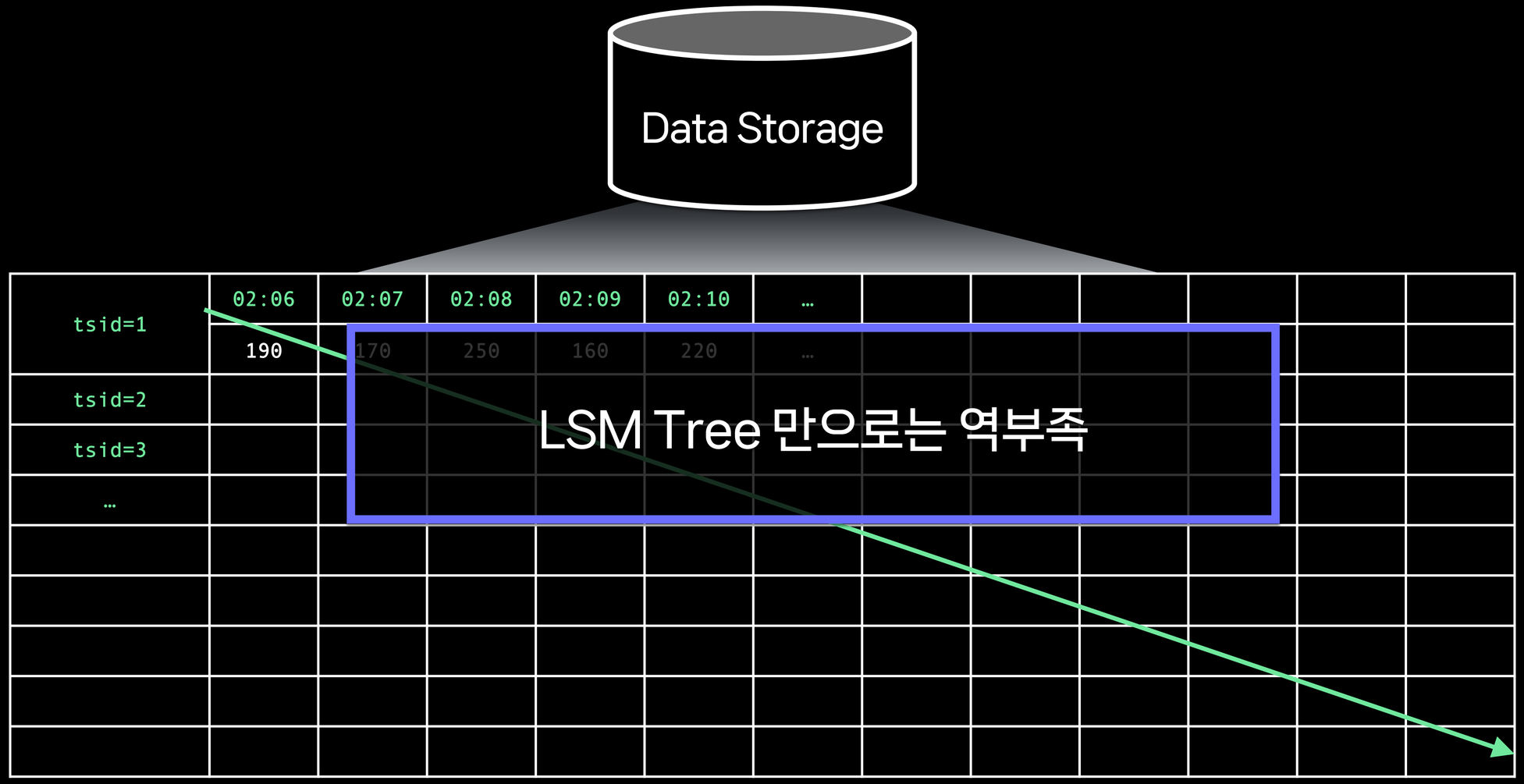
2.5 Data Storage

NAVER
DEVIEW
2023

전체로 보면 엄청나게 빠른 속도로 증가하는 데이터 규모

2.5 Data Storage

NAVER
DEVIEW
2023



전체로 보면 엄청나게 빠른 속도로 증가하는 데이터 규모

2.5 Data Storage

NAVER
DEVIEW
2023

전체로 보면 엄청나게 빠른 속도로 증가하는 데이터 규모

2.6 Gorilla Compression

NAVER
DEVIEW
2023

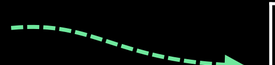
(Write Requests)

UNIX Timestamp	Value
1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220



하나의 Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

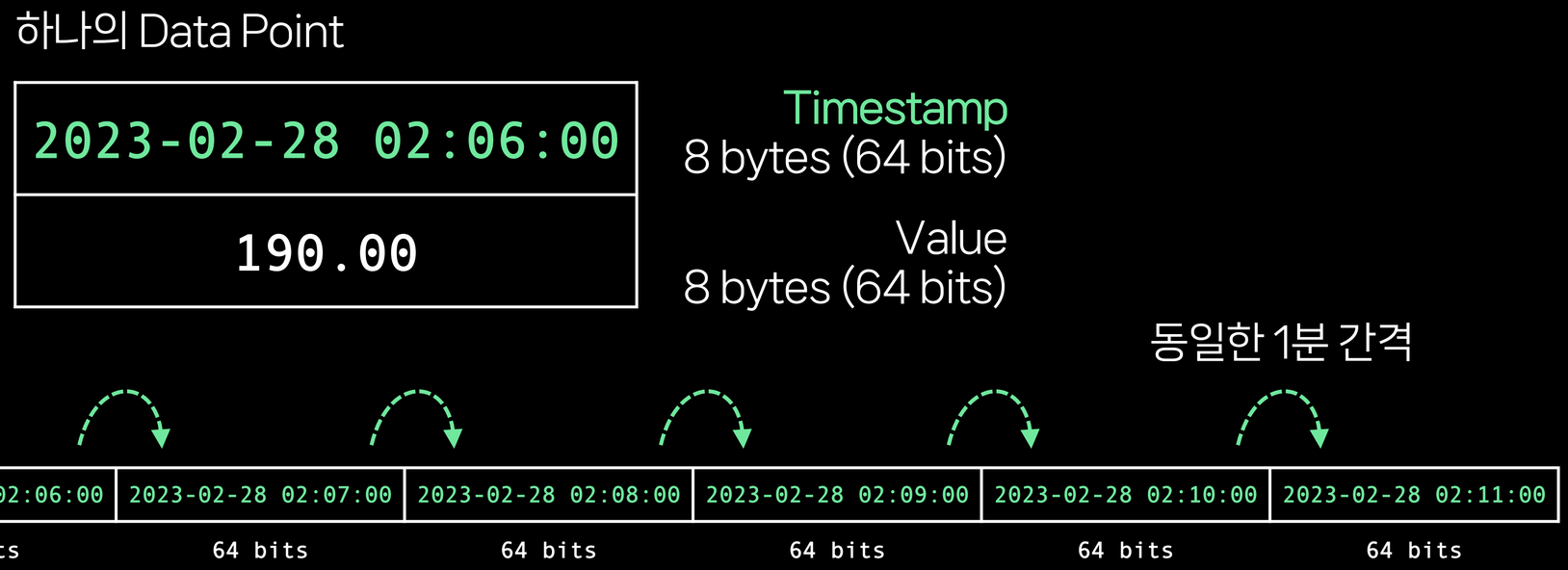
Value
8 bytes (64 bits)

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220



https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

사실은 거의 중복된 값

일일이 다 저장할 필요가 없다!

하나의 Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

동일한 1분 간격

2023-02-28 02:06:00	2023-02-28 02:07:00	2023-02-28 02:08:00	2023-02-28 02:09:00	2023-02-28 02:10:00	2023-02-28 02:11:00
64 bits	64 bits	64 bits	64 bits	64 bits	64 bits

2023-02-28 02:06:00	+ 1분	동일	동일	동일	동일
64 bits	14 bits	1 bit	1 bit	1 bit	1 bit

Delta of Delta Encoding으로

Timestamp 대폭 압축 가능

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

하나의 Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Previous Value	190.00	0x4067c00000000000
Current Value	170.00	0x4065400000000000

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

하나의 Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Previous Value	190.00	0x4067c00000000000
Current Value	170.00	0x4065400000000000

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

하나의 Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Previous Value	190.00	0x4067c00000000000
Current Value	170.00	0x4065400000000000
XOR	-	0x0002800000000000

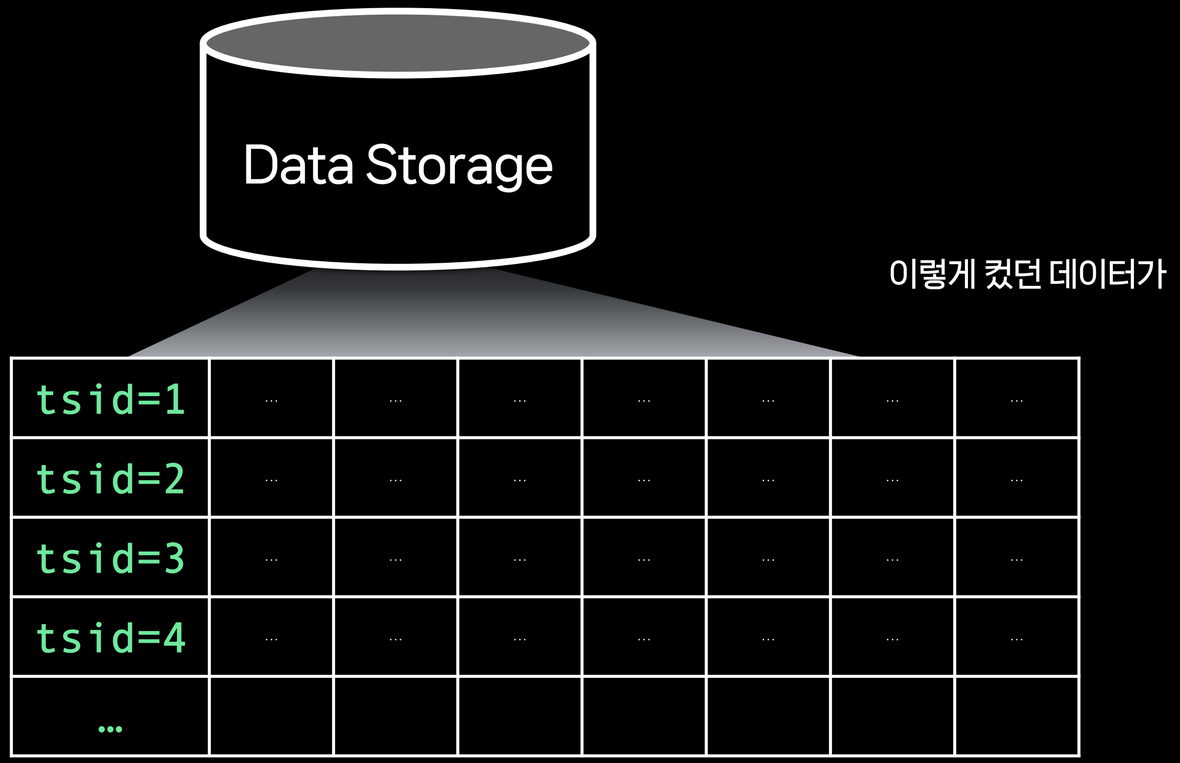
Value 또한 이진수 관점으로 보면 대부분이 중복

XOR 연산 시 `0` → Delta of Delta Encoding으로 압축 가능

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

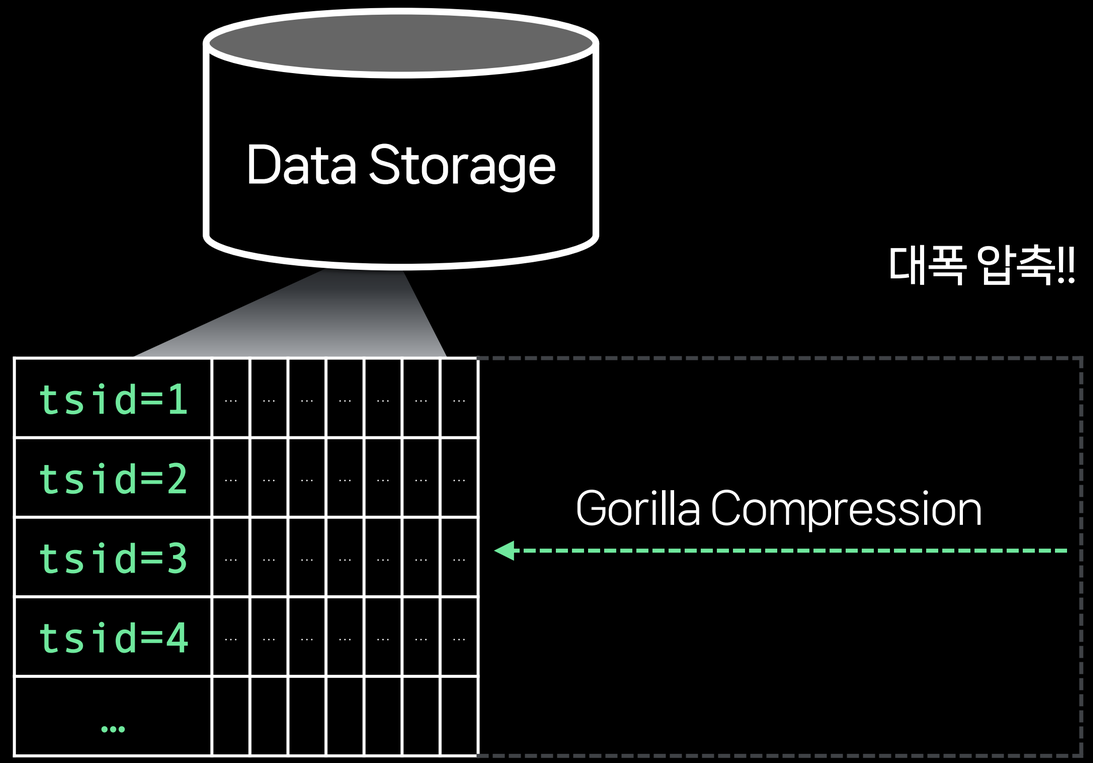
NAVER
DEVIEW
2023



https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

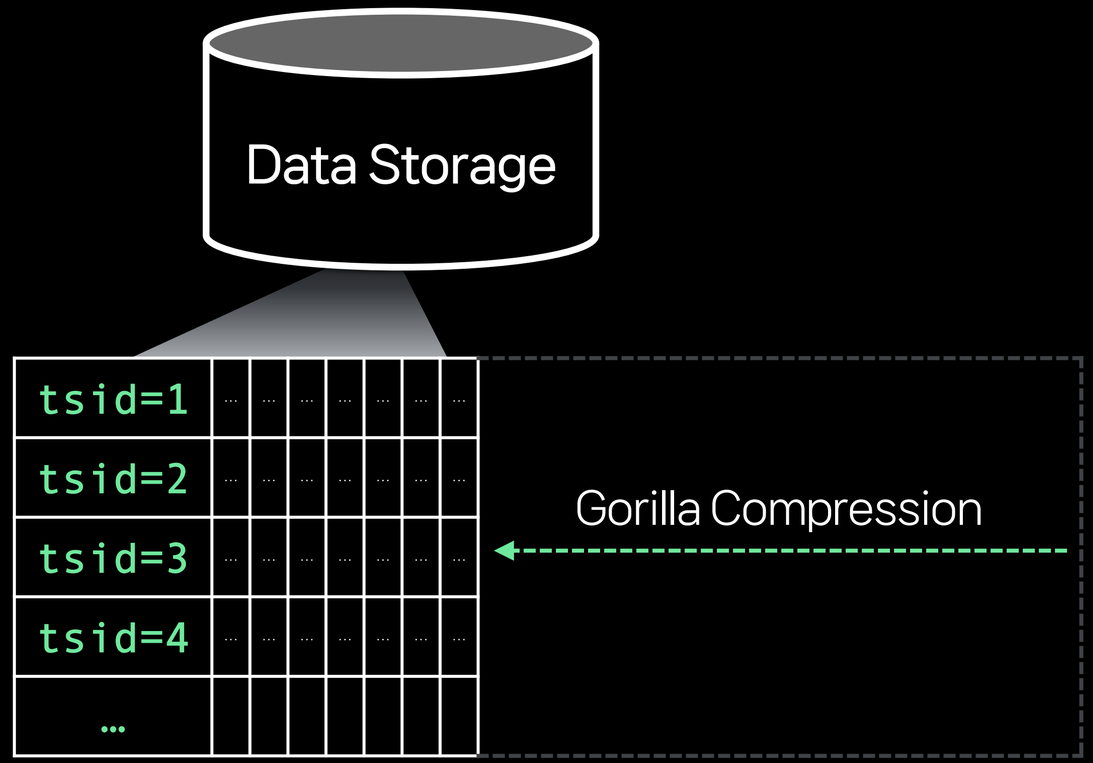
NAVER
DEVIEW
2023



https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023



(실제 대시보드 스크린샷)

16 Bytes 분량의 정보를 0.47 Byte 수준으로 압축!

메모리에 훨씬 많은 데이터가 올라가기 때문에

빠른 대용량 처리 가능

https://dl.acm.org/doi/10.14778/2824032.2824078

2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

High Churn Rate

Churn Rate: 고객 이탈률

- Time series가 계속 새로 만들어지기만 하고 오래 쌓이지 않는 상황

2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

High Churn Rate

Churn Rate: 고객 이탈률

- Time series가 계속 새로 만들어지기만 하고 오래 쌓이지 않는 상황
- 불필요하게 값이 자주 변하는 Label 존재

```
http_requests_total{instance="host1",job="my_app",path="/foo”,tag="i73w3e2h4asdf8q23jha"}
```

2.7 Importance of Churn Rate

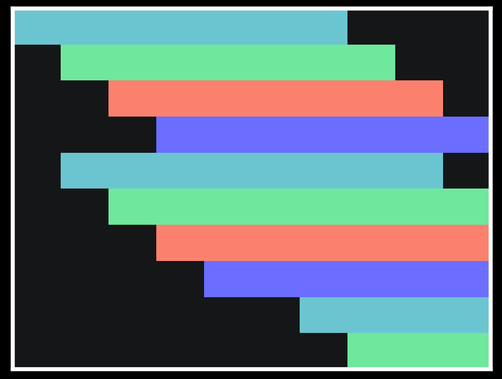
NAVER
DEVIEW
2023

High Churn Rate

Churn Rate: 고객 이탈률

- Time series가 계속 새로 만들어지기만 하고 오래 쌓이지 않는 상황
- 불필요하게 값이 자주 변하는 Label 존재

```
http_requests_total{instance="host1",job="my_app",path="/foo”,tag="i73w3e2h4asdf8q23jha"}
```



일반적인 시계열 데이터 분포

2.7 Importance of Churn Rate

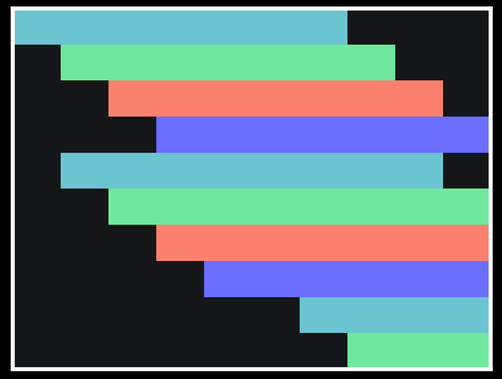
NAVER
DEVIEW
2023

High Churn Rate

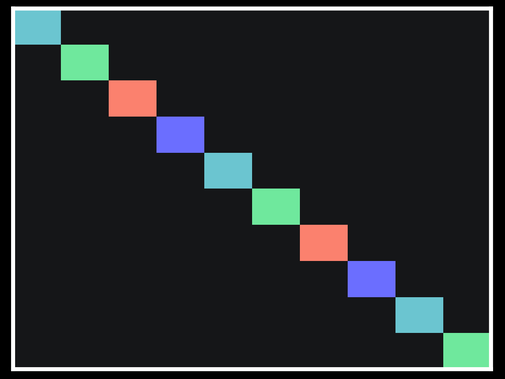
Churn Rate: 고객 이탈률

- Time series가 계속 새로 만들어지기만 하고 오래 쌓이지 않는 상황
- 불필요하게 값이 자주 변하는 Label 존재

```
http_requests_total{instance="host1",job="my_app",path="/foo”,tag="i73w3e2h4asdf8q23jha"}
```



일반적인 시계열 데이터 분포



Churn Rate가 높은 시계열 데이터 분포

2.7 Importance of Churn Rate

NAVER DEVIEW 2023

High Churn Rate

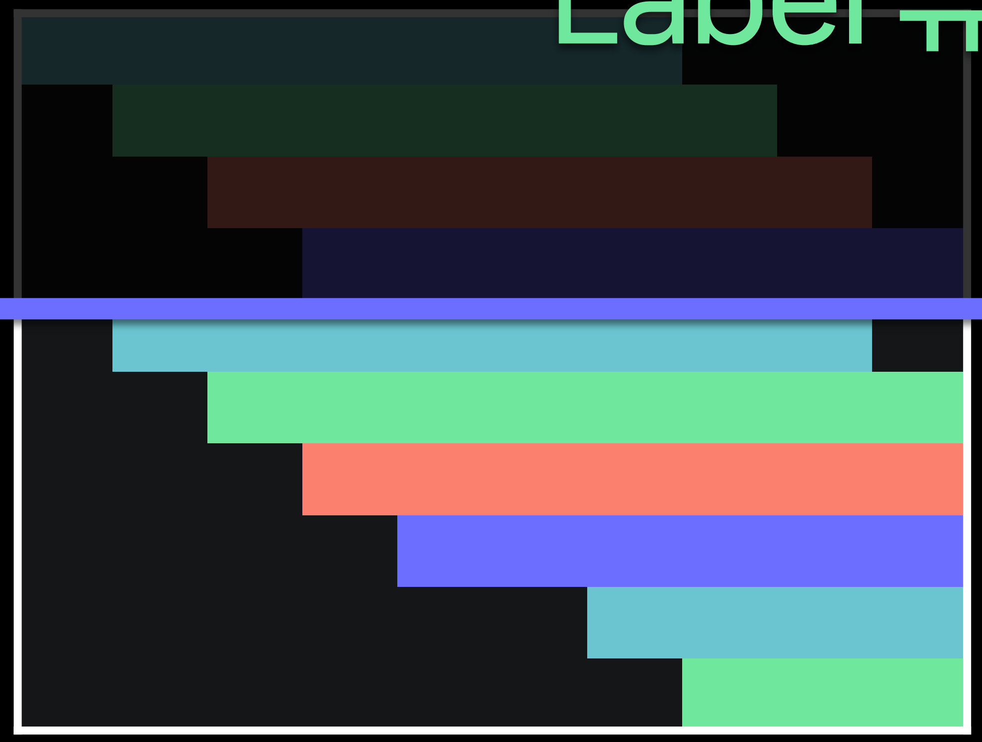
Churn Rate: 고객 이탈률

- Time series가 계속 새로 만들어지기만 하고 오래 쌓이지 않는 상황
- 불필요하게 값이 자주 변하는 Label 존재

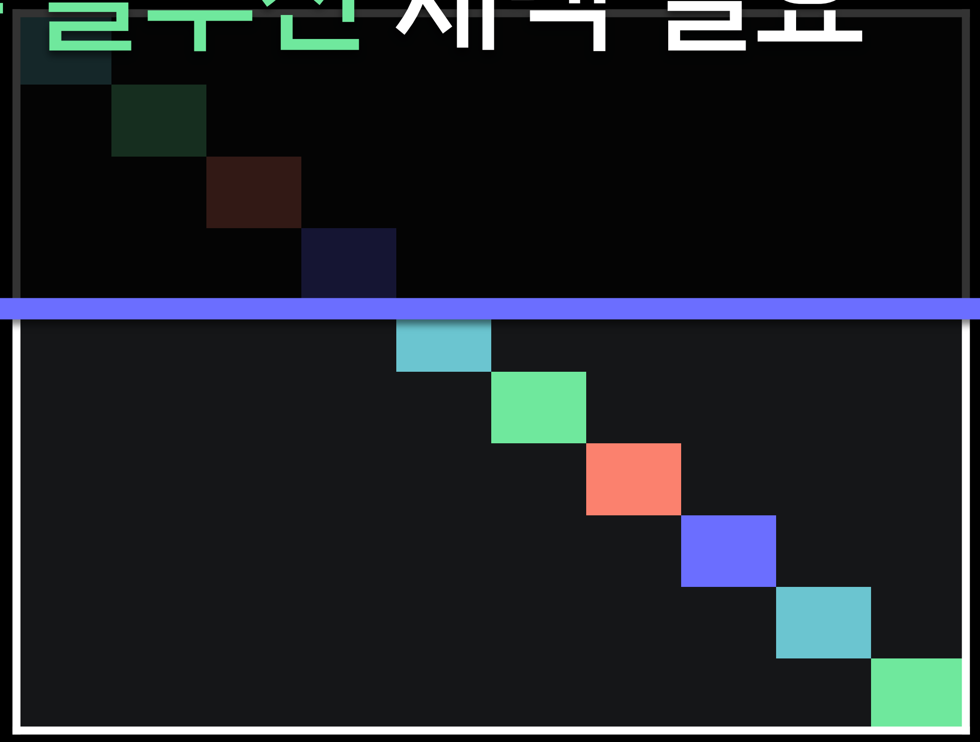
```
http_requests_total{ ... q23jha"}
```

시계열 DB로 처리하기에 적합하지 않은 데이터 구조

Label 튜닝 or 다른 솔루션 채택 필요



일반적인 시계열 데이터 분포



Churn Rate가 높은 시계열 데이터 분포

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage
- LSM Tree 구조를 통해 Read / Write 모두 빠르게 처리

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage
- LSM Tree 구조를 통해 Read / Write 모두 빠르게 처리
- Gorilla Compression 을 활용하여 압축률 향상 & 메모리 효율 증가

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage
- LSM Tree 구조를 통해 Read / Write 모두 빠르게 처리
- Gorilla Compression 을 활용하여 압축률 향상 & 메모리 효율 증가
- High Churn Rate 상황은 피하는 것이 중요

3. Time series in the Multiverse of Madness

3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 하나로 모든 기능 제공

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*

~/VictoriaMetrics/single ··· ✔ 〈 at 10:22:04
```

3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 하나로 모든 기능 제공
- 추가 최적화로 Prometheus보다 2~10배 빠름

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*

~/VictoriaMetrics/single ✔ at 10:22:04
```

3.1 Happy one universe

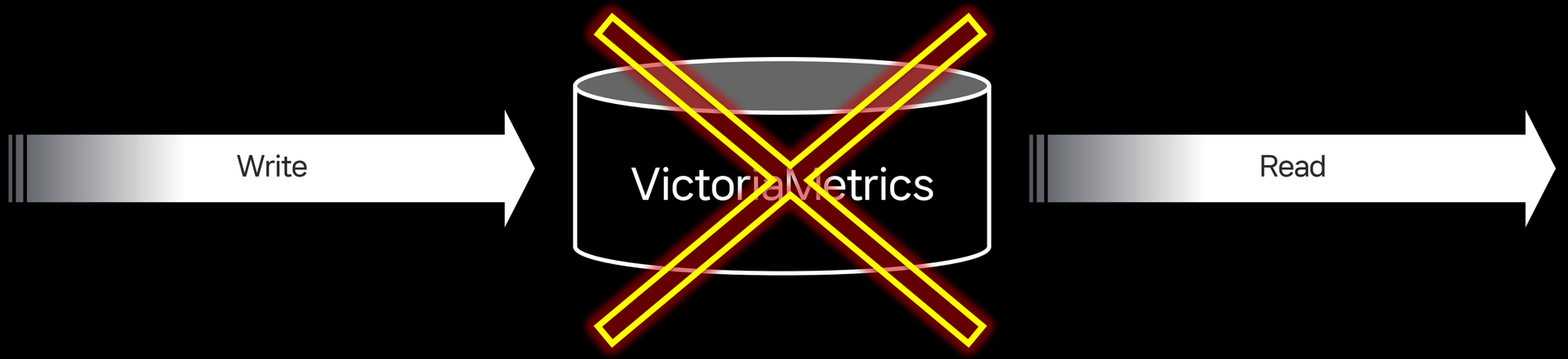
NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 하나로 모든 기능 제공
- 추가 최적화로 Prometheus보다 2~10배 빠름
- 단점 : Single Point of Failure

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*

~/VictoriaMetrics/single ····· ✔ 〈 at 10:22:04
```



3.1 Happy one universe

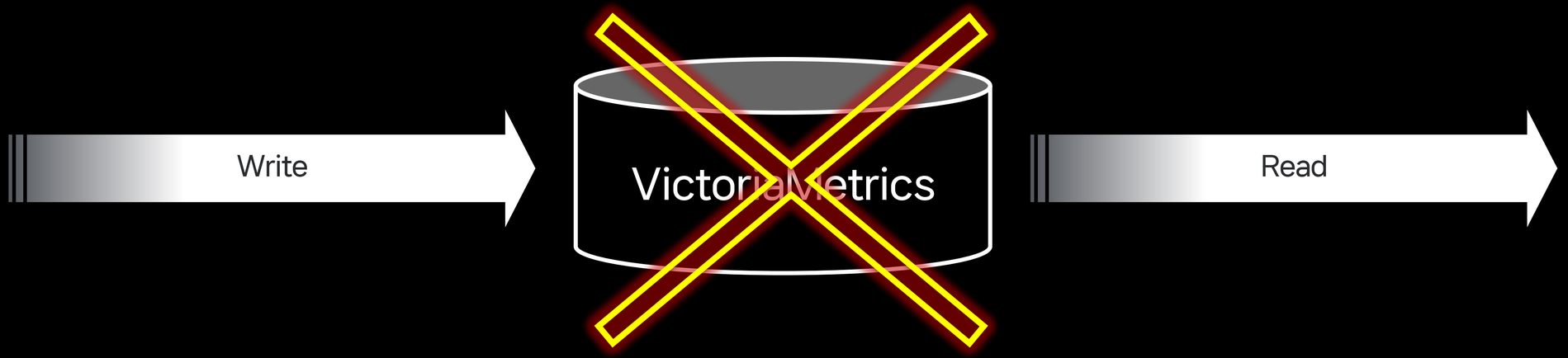
NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 하나로 모든 기능 제공
- 추가 최적화로 Prometheus보다 2~10배 빠름
- 단점 : Single Point of Failure

수 백명의 개발자가 담당하는
수 많은 서비스 모니터링 중단

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*
VictoriaMetrics/single ····· ✔ 〈 at 10:22:04
```



3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

3.1 Happy one universe

NAVER
DEVIEW
2023

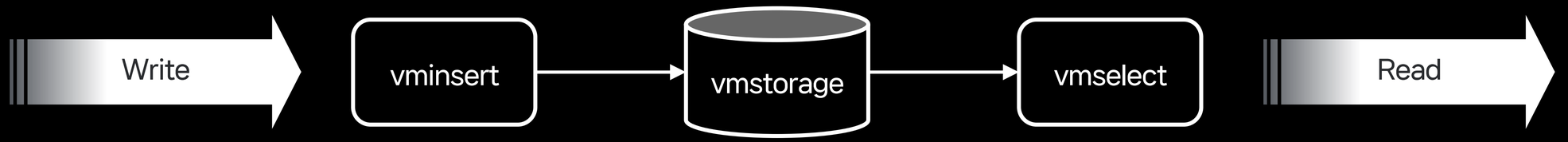
Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 3개

  (Thanos, Cortex, Mimir 등 다른 제품과 차이나는 부분)

```
❯ ls -alFh
total 70976
drwxr-xr-x  5 user  staff   160B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:45 vminsert-prod*
-rwxr-xr-x@ 1 user  staff    13M Feb  2 07:46 vmselect-prod*
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:46 vmstorage-prod*

~/VictoriaMetrics/cluster ✔ at 10:23:09
```



3.1 Happy one universe

NAVER
DEVIEW
2023

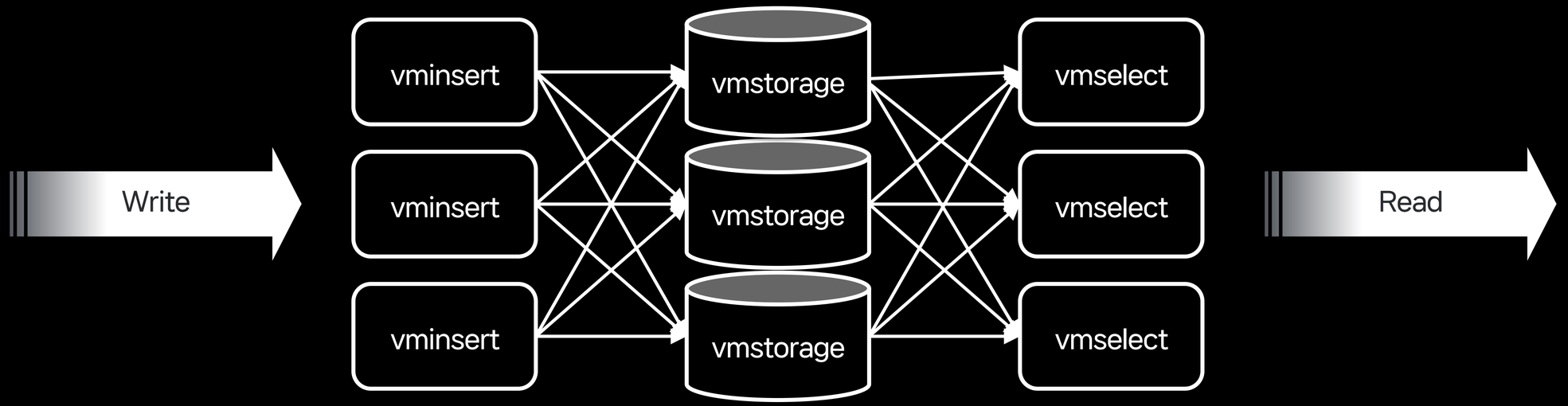
Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 3개
- 각 컴포넌트는 손쉽게 수평 확장 가능

(Prometheus의 가장 큰 한계 극복)

```
❯ ls -alFh
total 70976
drwxr-xr-x  5 user  staff   160B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:45 vminsert-prod*
-rwxr-xr-x@ 1 user  staff    13M Feb  2 07:46 vmselect-prod*
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:46 vmstorage-prod*

~/VictoriaMetrics/cluster ··· ✔ at 10:23:09
```



3.1 Happy one universe

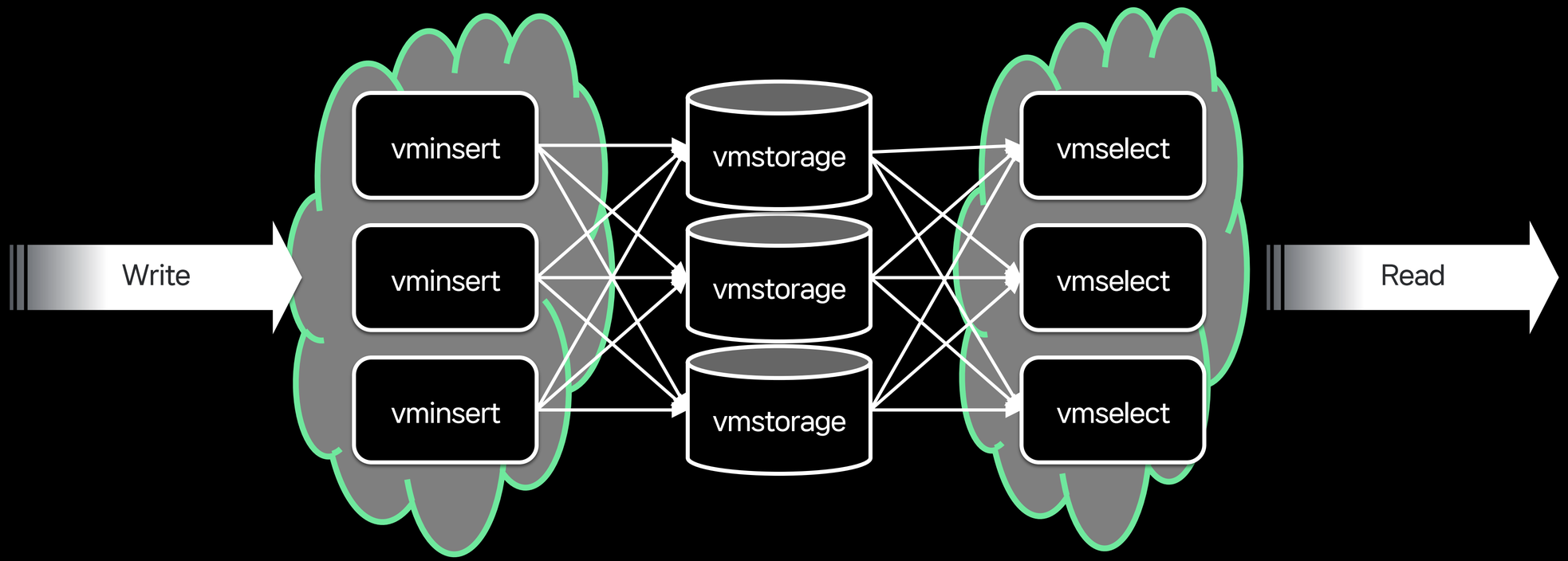
NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- 외부 의존성 없이 바이너리 파일 3개
- 각 컴포넌트는 손쉽게 수평 확장 가능

- Stateful / Stateless 구분

(Physical Machine & Kubernetes 혼용)

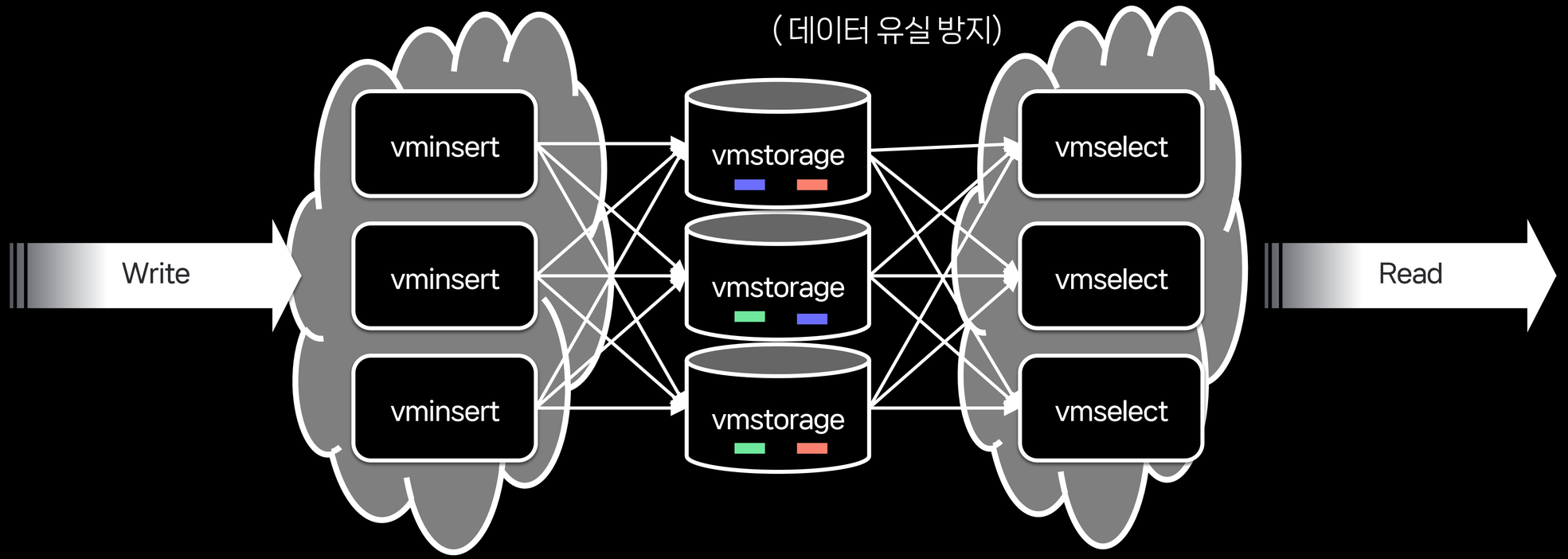


3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

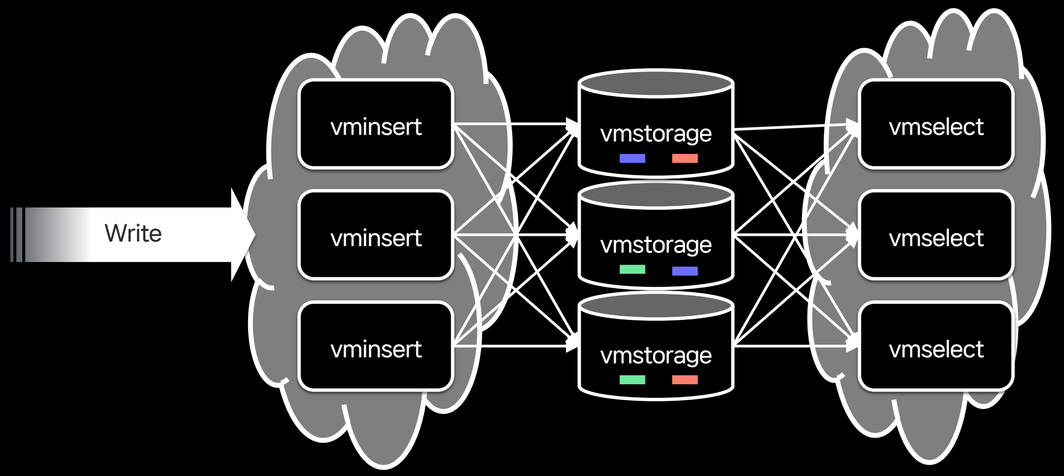
- 외부 의존성 없이 바이너리 파일 3개
- 각 컴포넌트는 손쉽게 수평 확장 가능
- Stateful / Stateless 구분
- ReplicationFactor 활용 가능



3.1 Happy one universe

NAVER
DEVIEW
2023

Happy Moments...



Read

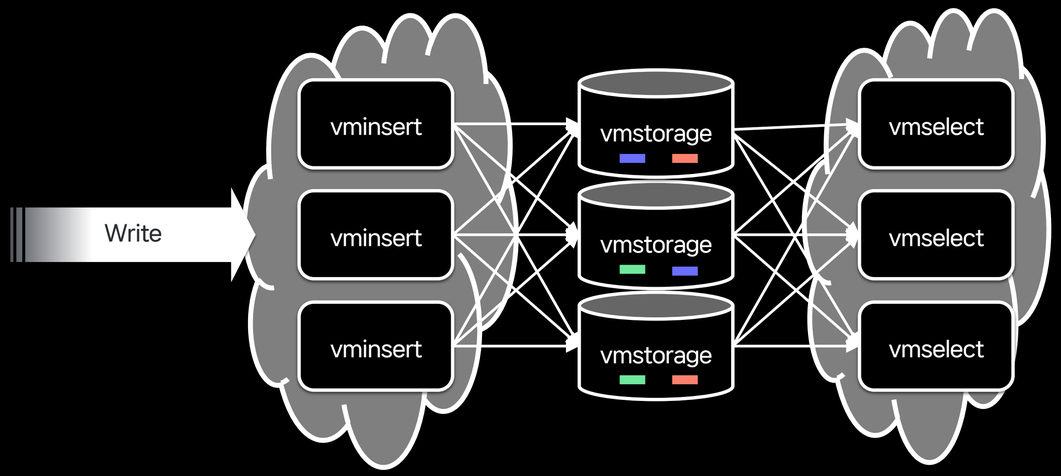
Service-level Monitoring Dashboard

대시보드에서 유용하게 사용

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Happy Moments...



Read

Service-level Monitoring Dashboard

Read

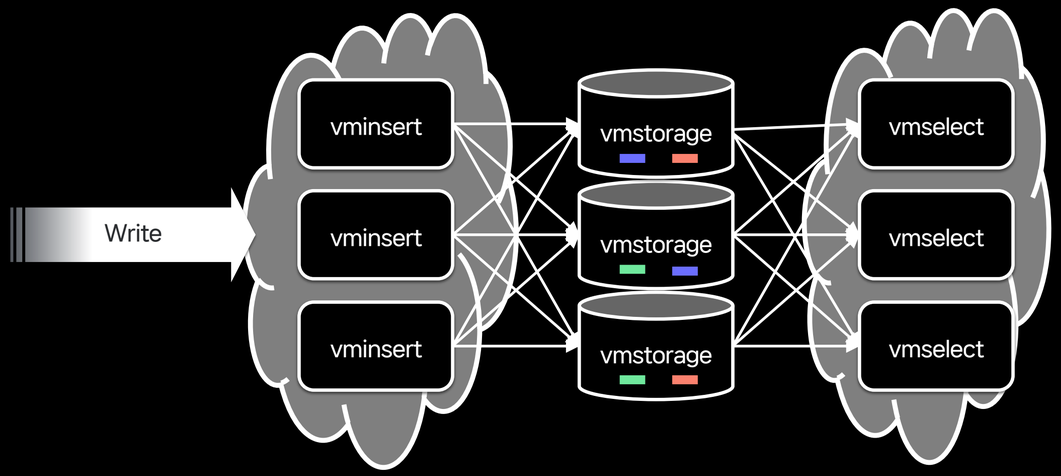
Service-level Alert System

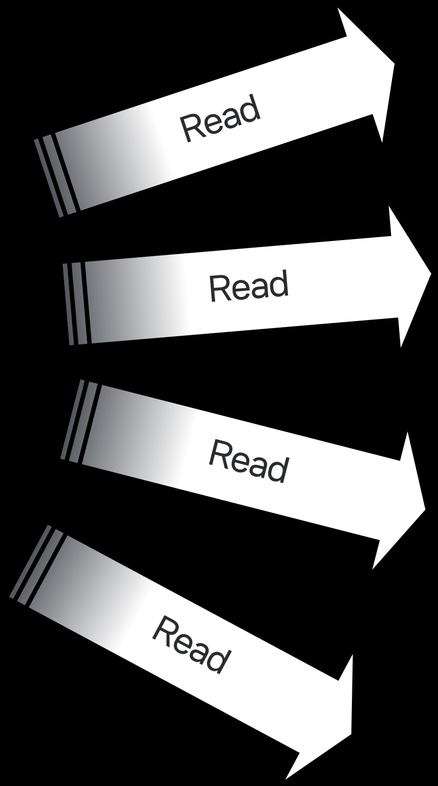
경보 시스템에도 적용

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Happy Moments...





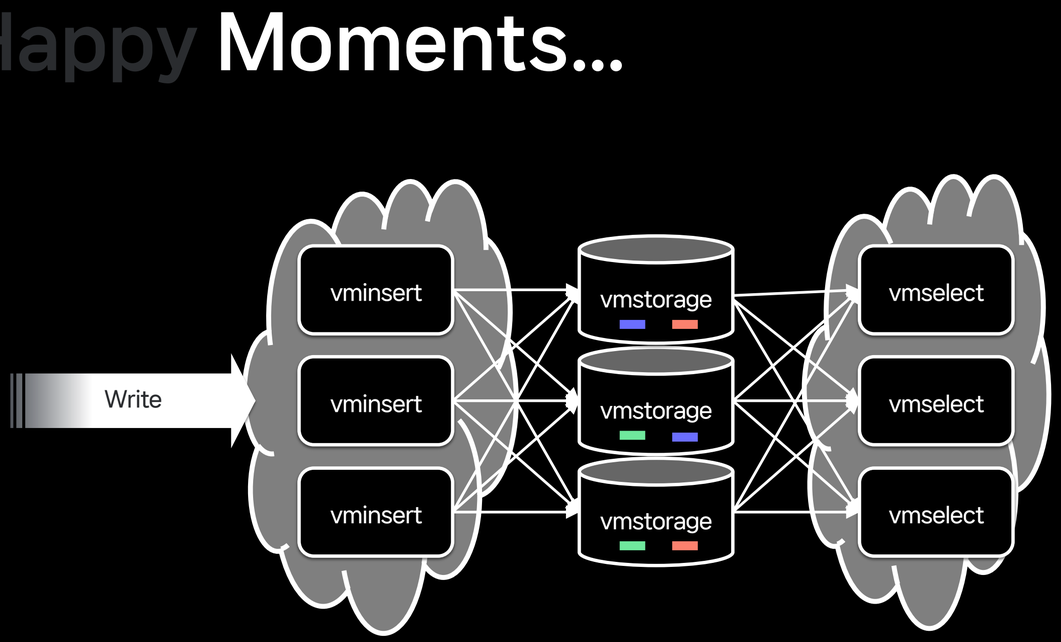
- Service-level Monitoring Dashboard
- Host-level Monitoring Dashboard
- Service-level Alert System
- Host-level Alert System

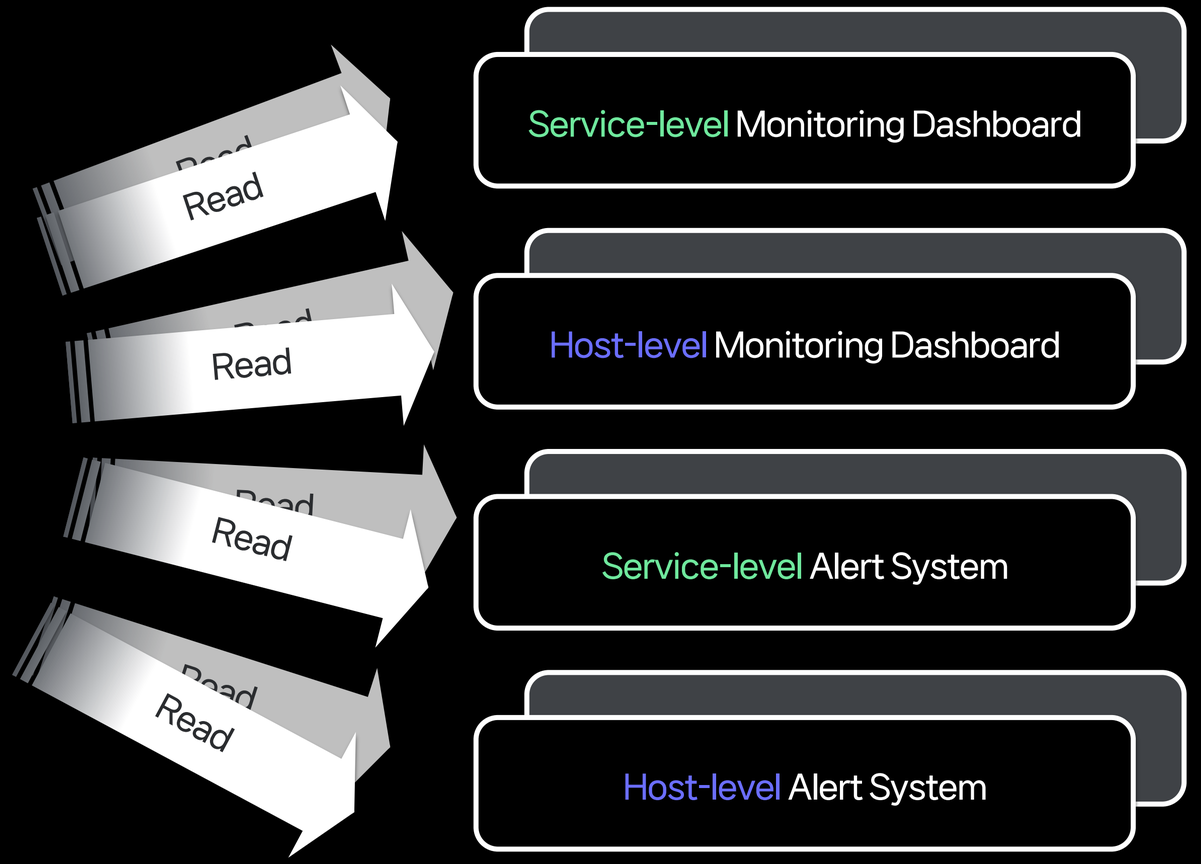
다른 종류의 대시보드와 경보 시스템에도 적용

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Happy Moments...

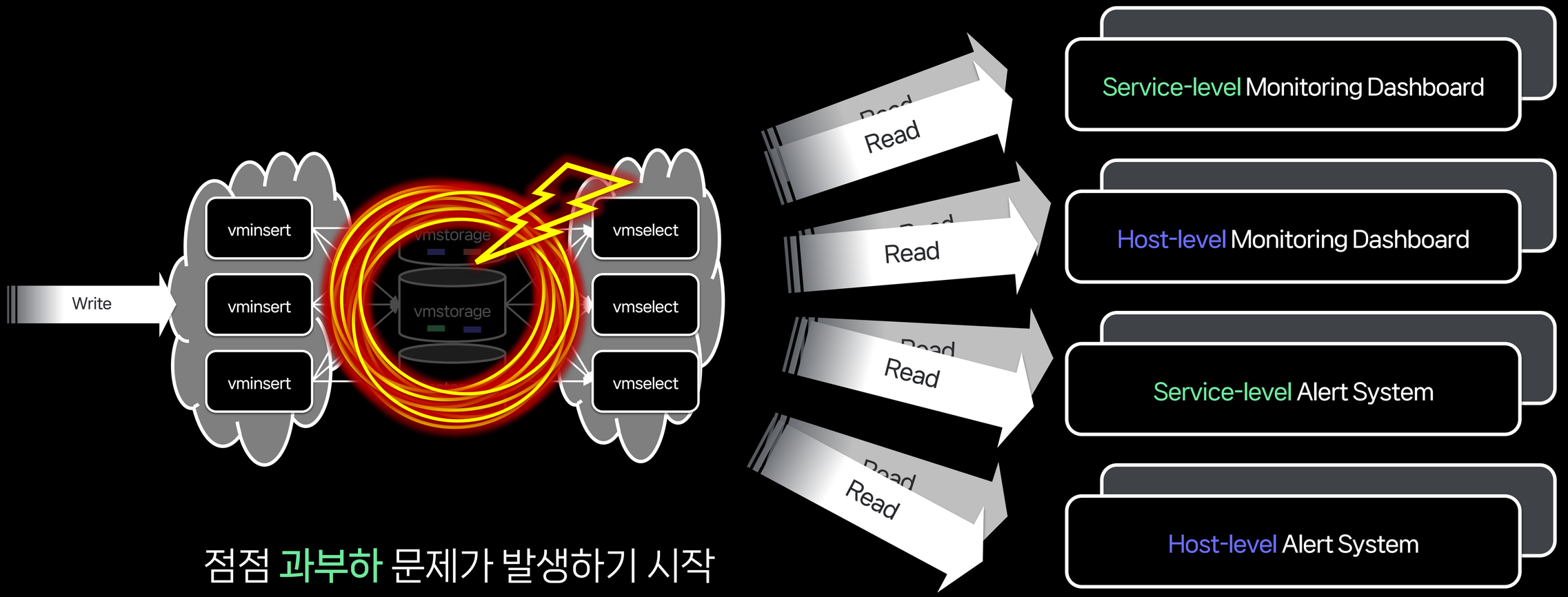




개발 & 테스트용 시스템에서도 사용

3.2 The Rise of The Multiverse

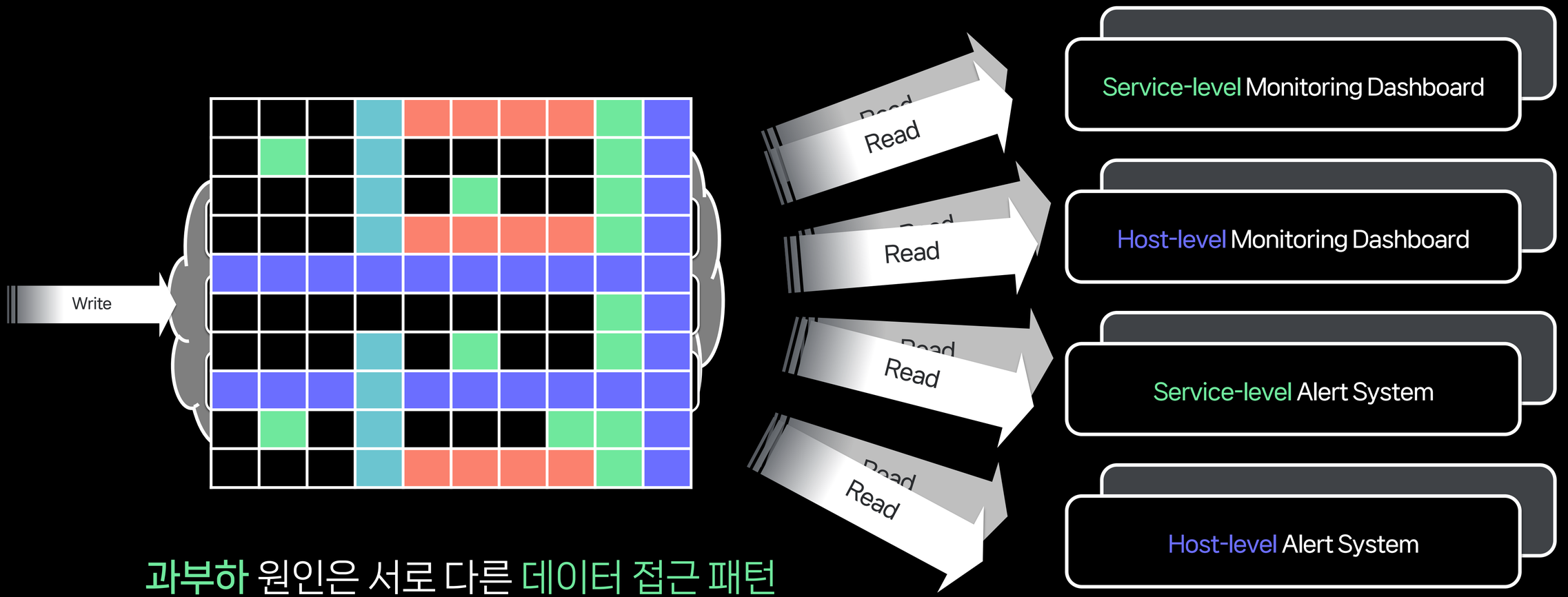
NAVER
DEVIEW
2023



점점 **과부하** 문제가 발생하기 시작

3.2 The Rise of The Multiverse

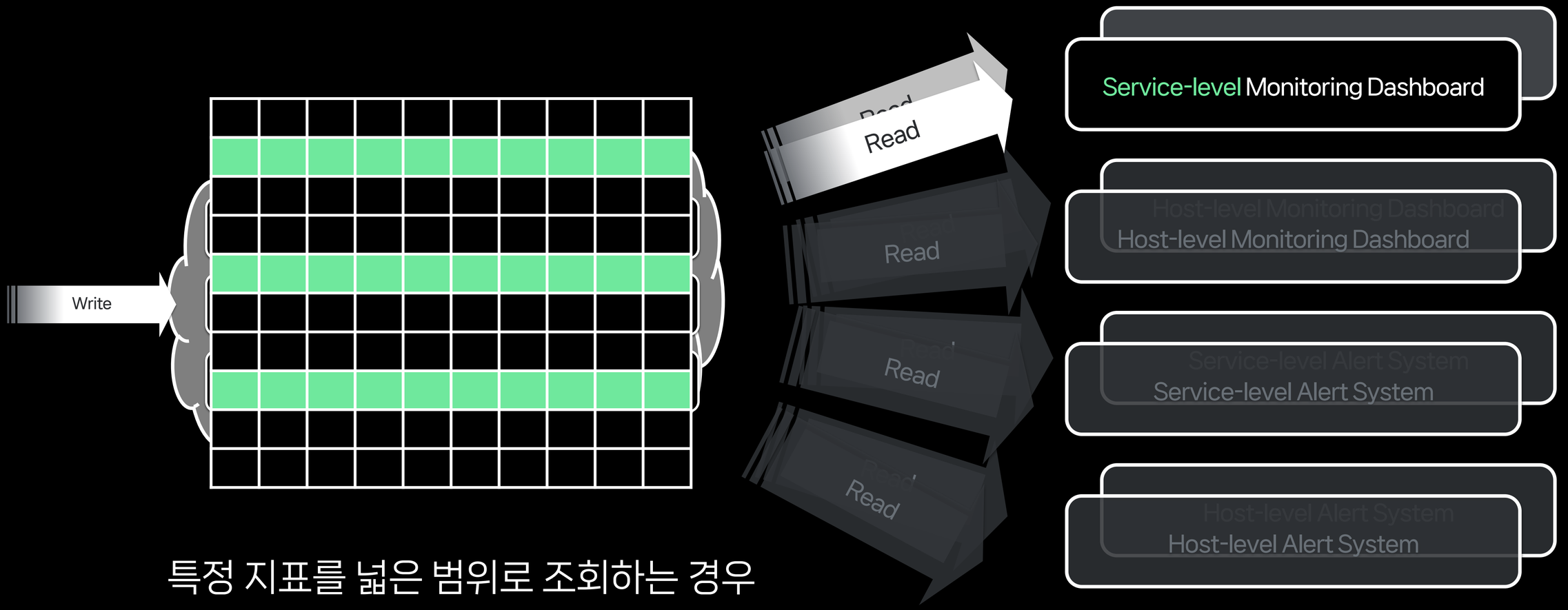
NAVER
DEVIEW
2023



과부하 원인은 서로 다른 데이터 접근 패턴

3.2 The Rise of The Multiverse

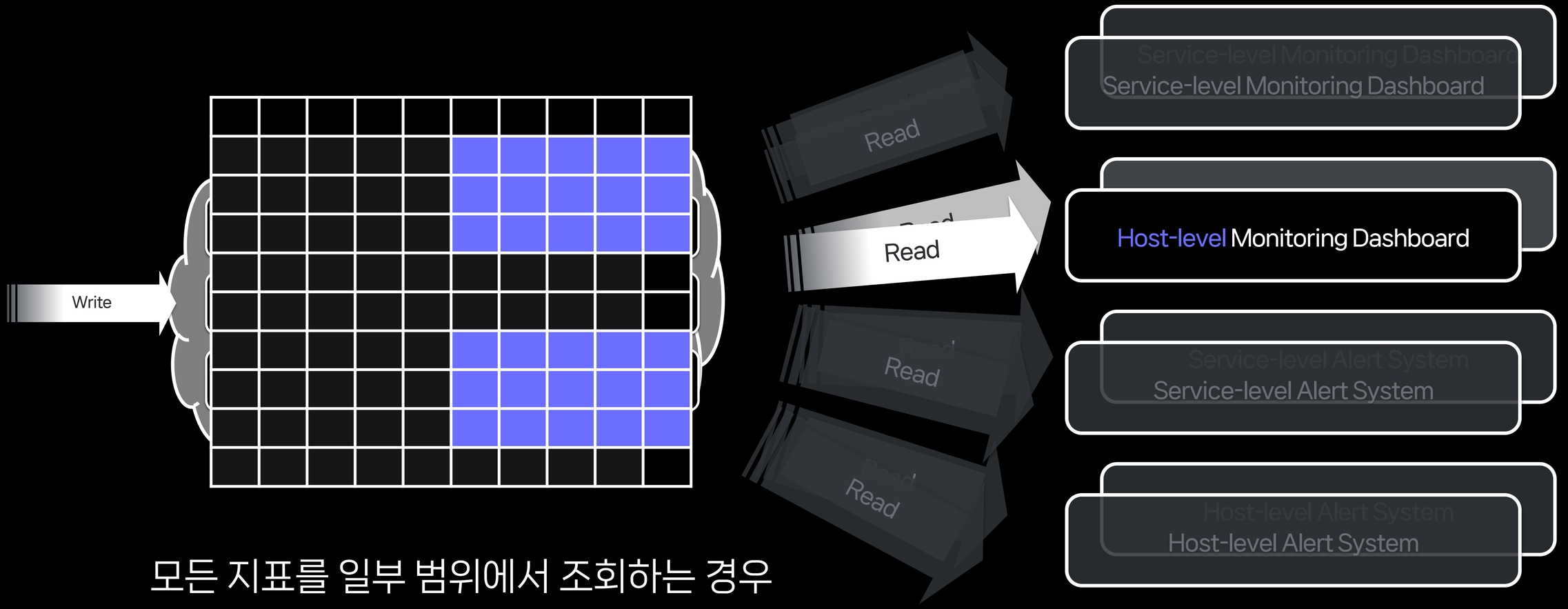
NAVER
DEVIEW
2023



특정 지표를 넓은 범위로 조회하는 경우

3.2 The Rise of The Multiverse

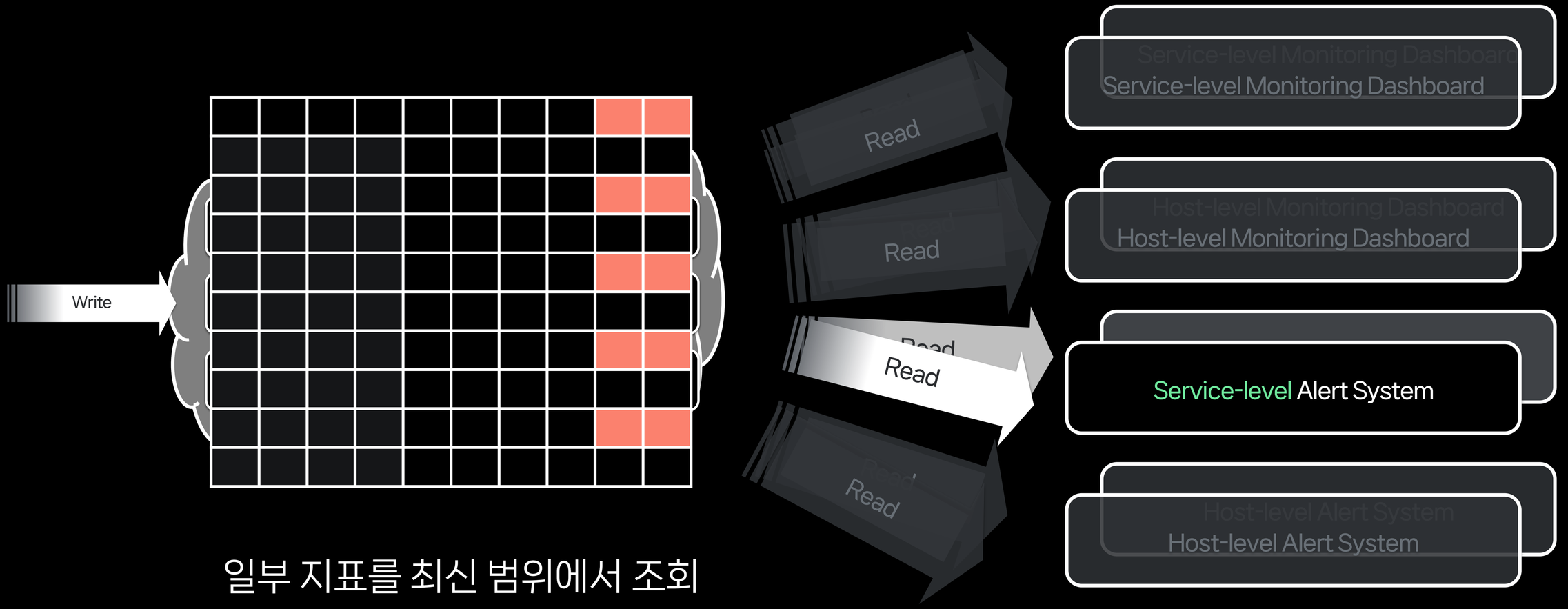
NAVER
DEVIEW
2023



모든 지표를 일부 범위에서 조회하는 경우

3.2 The Rise of The Multiverse

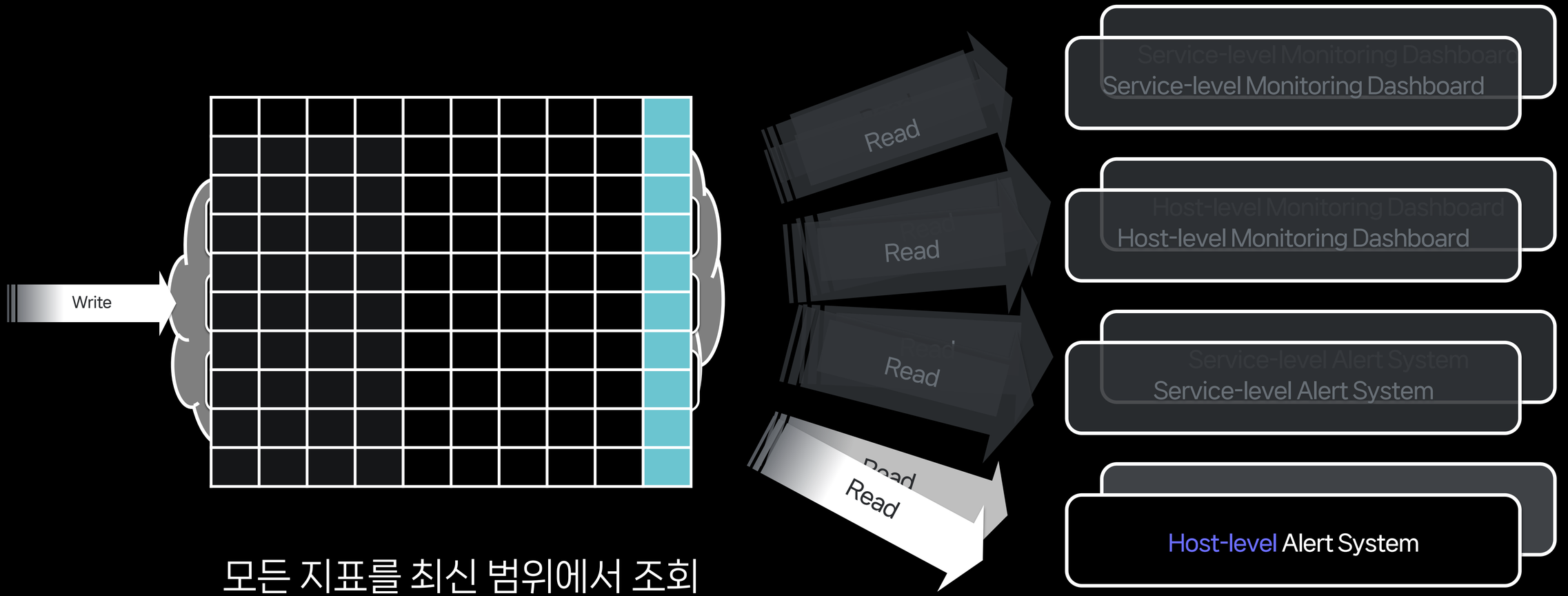
NAVER
DEVIEW
2023



일부 지표를 최신 범위에서 조회

3.2 The Rise of The Multiverse

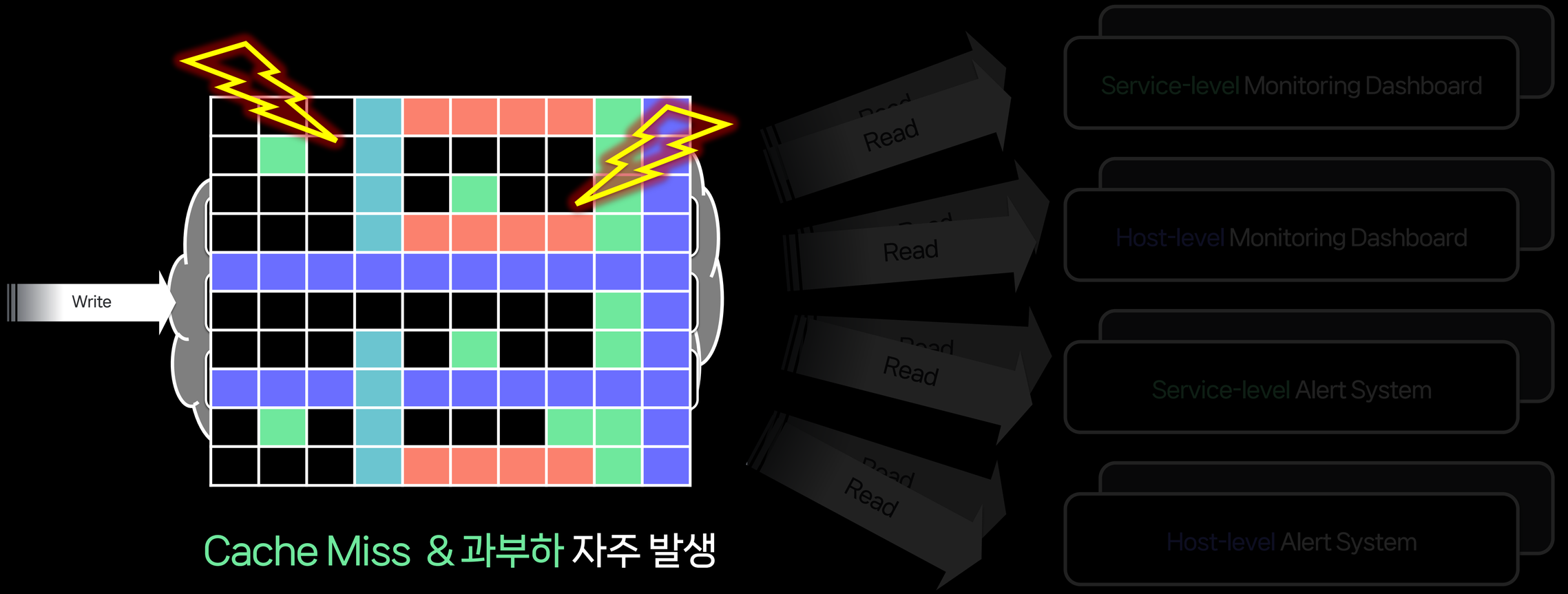
NAVER
DEVIEW
2023



모든 지표를 최신 범위에서 조회

3.2 The Rise of The Multiverse

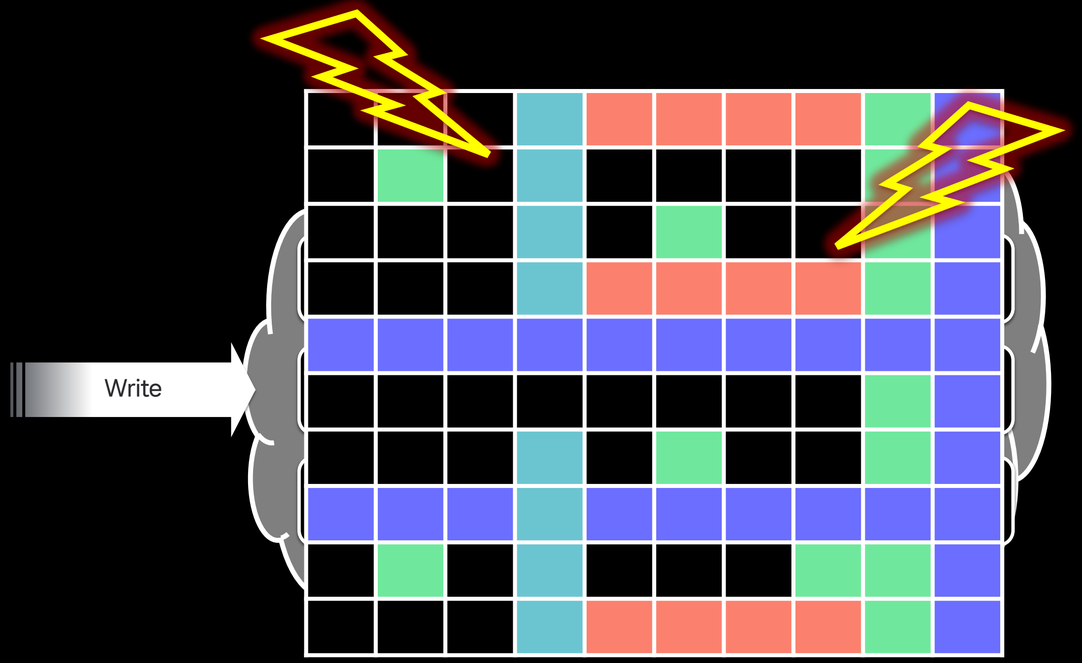
NAVER
DEVIEW
2023



Cache Miss & 과부하 자주 발생

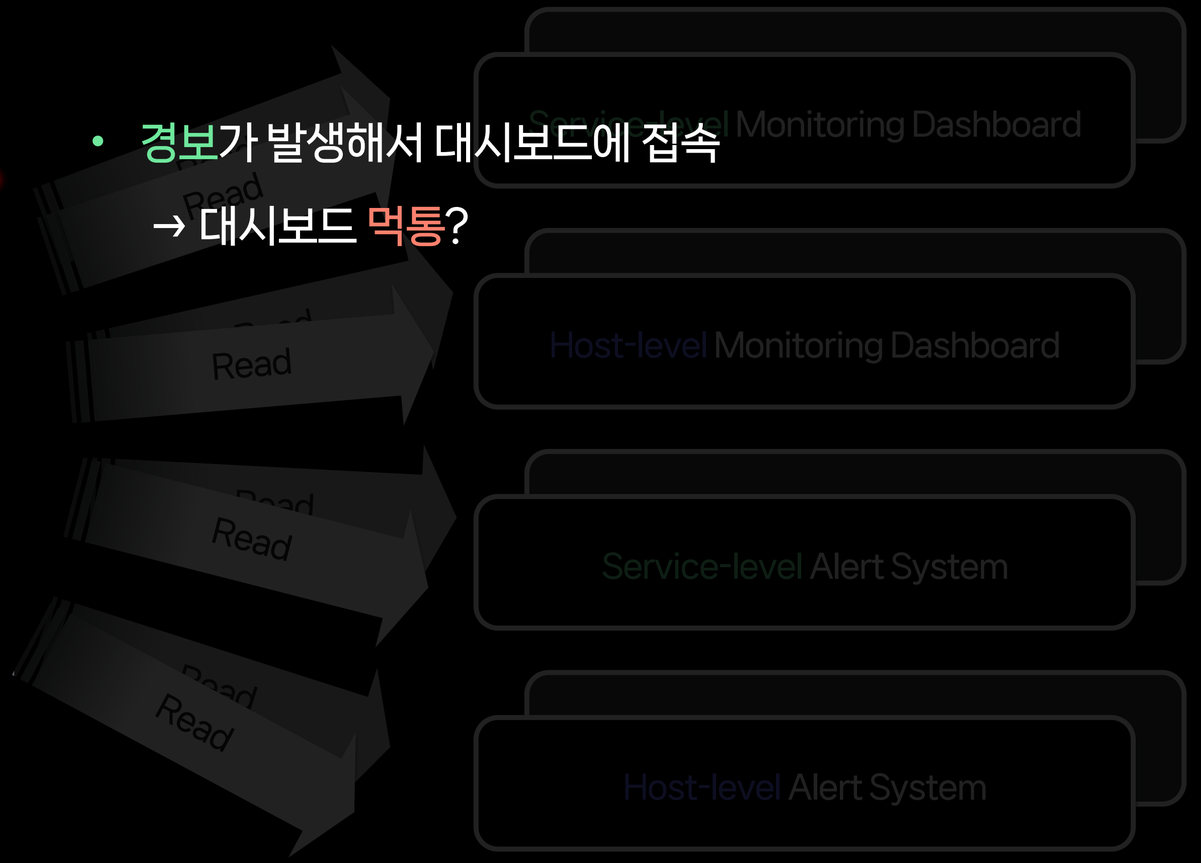
3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023



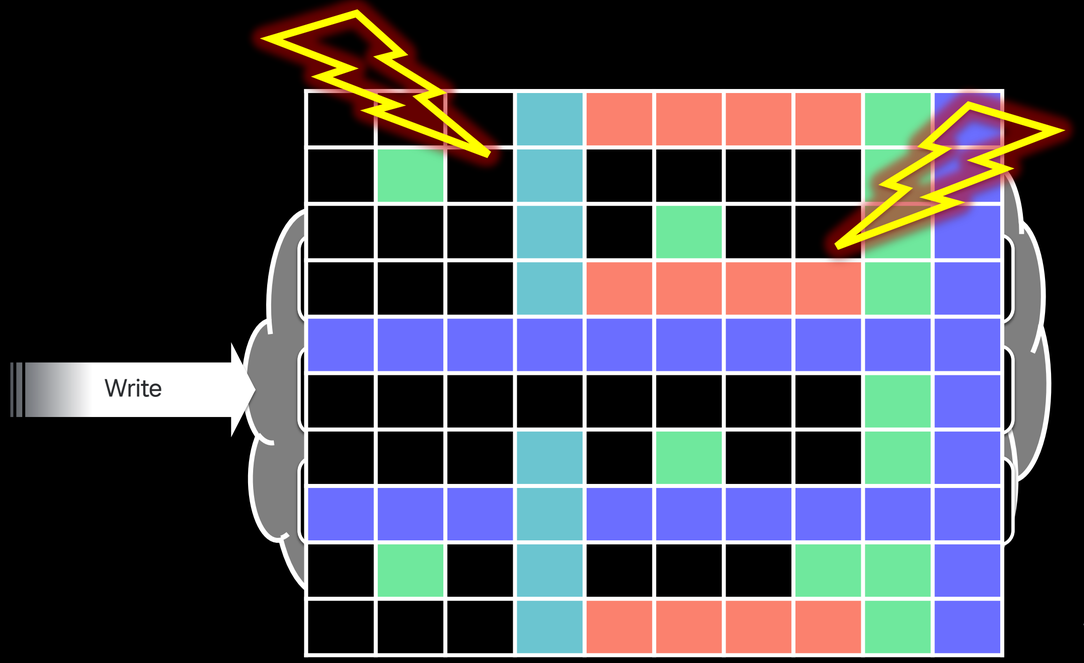
- 경보가 발생해서 대시보드에 접속

  → 대시보드 먹통?

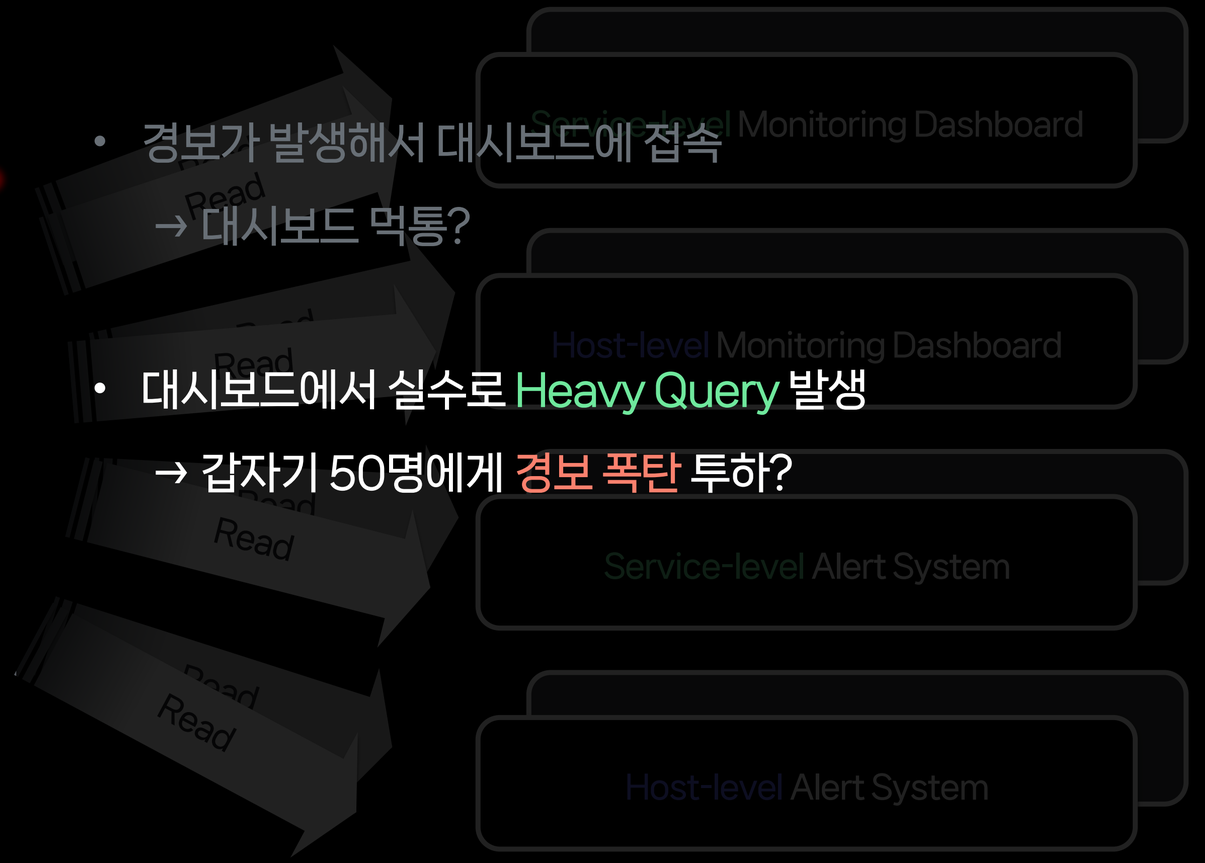


3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

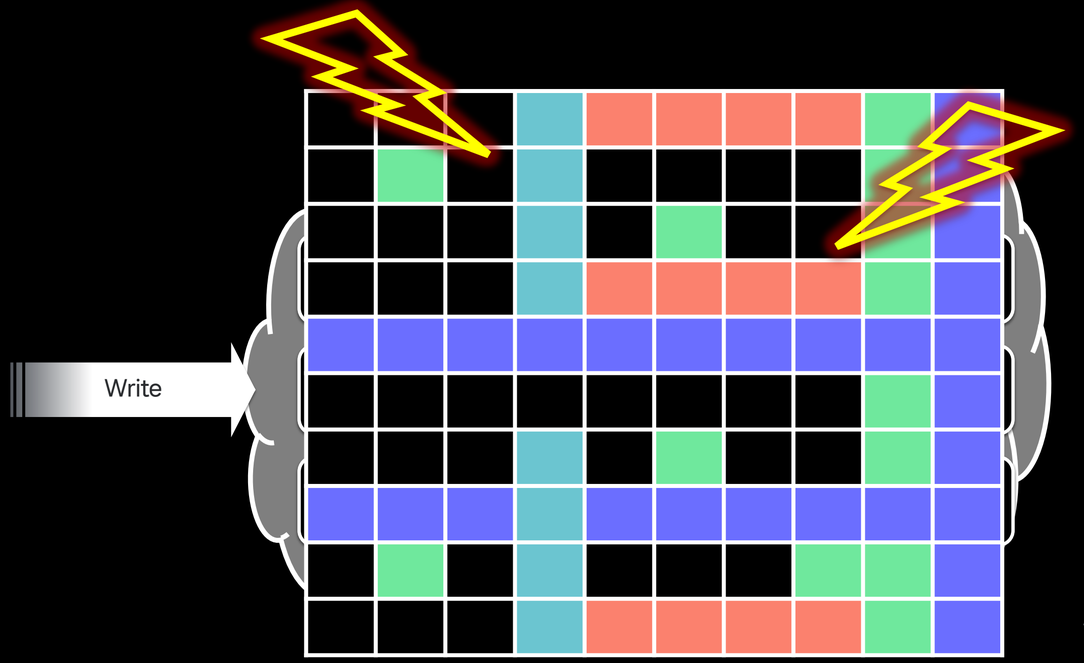


- 경보가 발생해서 대시보드에 접속
  → 대시보드 먹통?
- 대시보드에서 실수로 Heavy Query 발생
  → 갑자기 50명에게 경보 폭탄 투하?

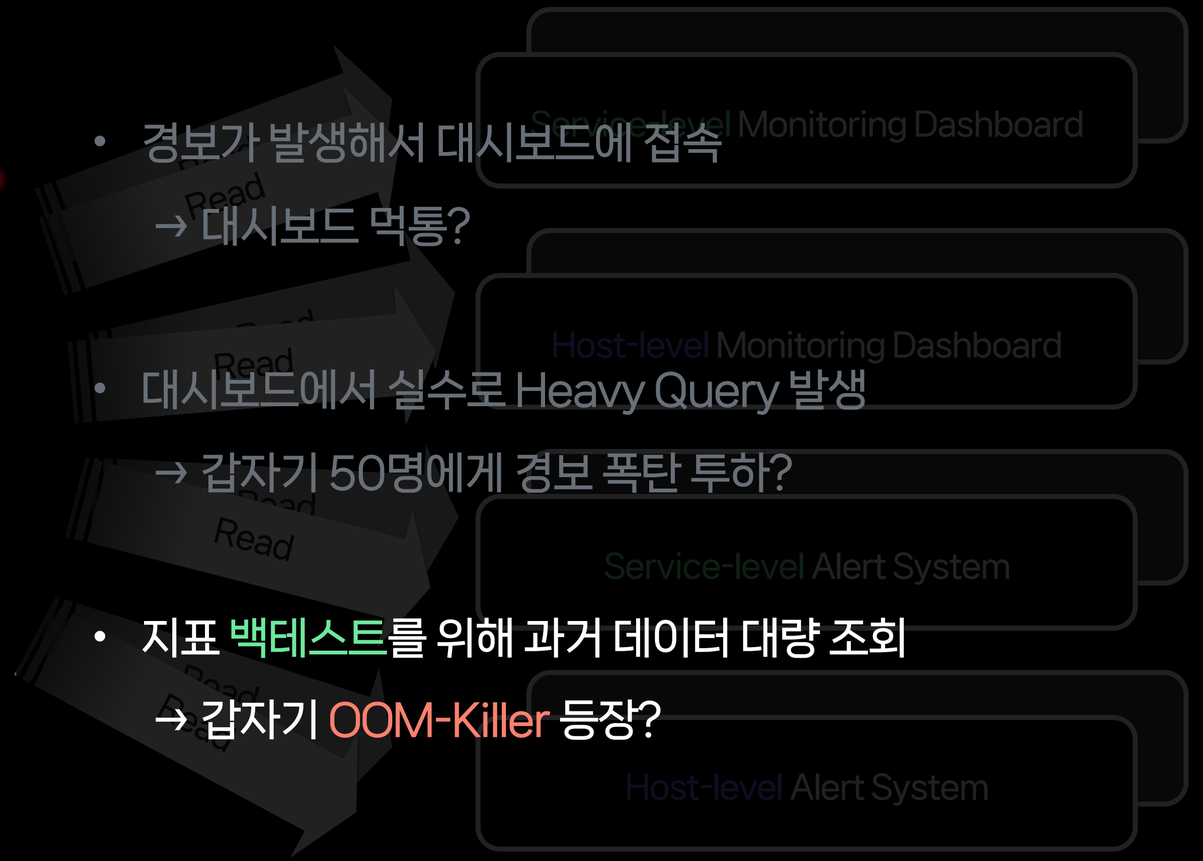


3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

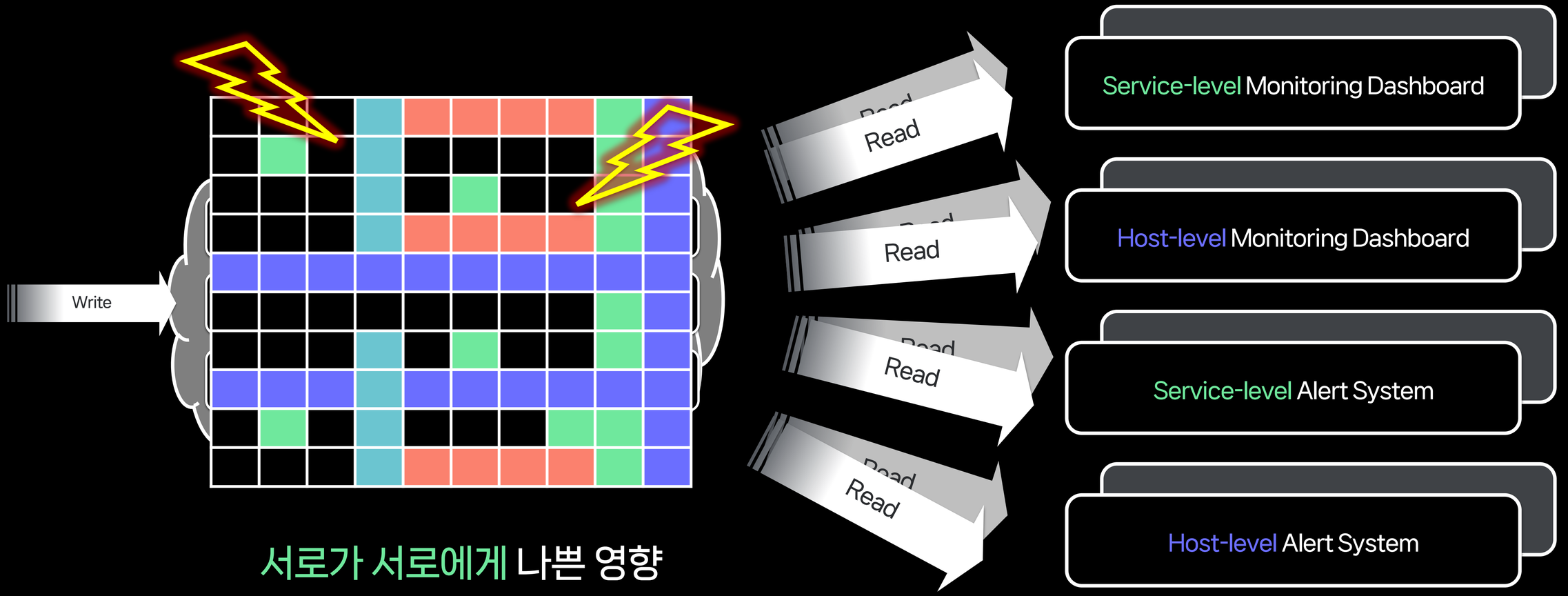


- 경보가 발생해서 대시보드에 접속
  → 대시보드 먹통?
- 대시보드에서 실수로 Heavy Query 발생
  → 갑자기 50명에게 경보 폭탄 투하?
- 지표 백테스트를 위해 과거 데이터 대량 조회
  → 갑자기 OOM-Killer 등장?



3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023



서로가 서로에게 나쁜 영향

3.2 The Rise of The Multiverse

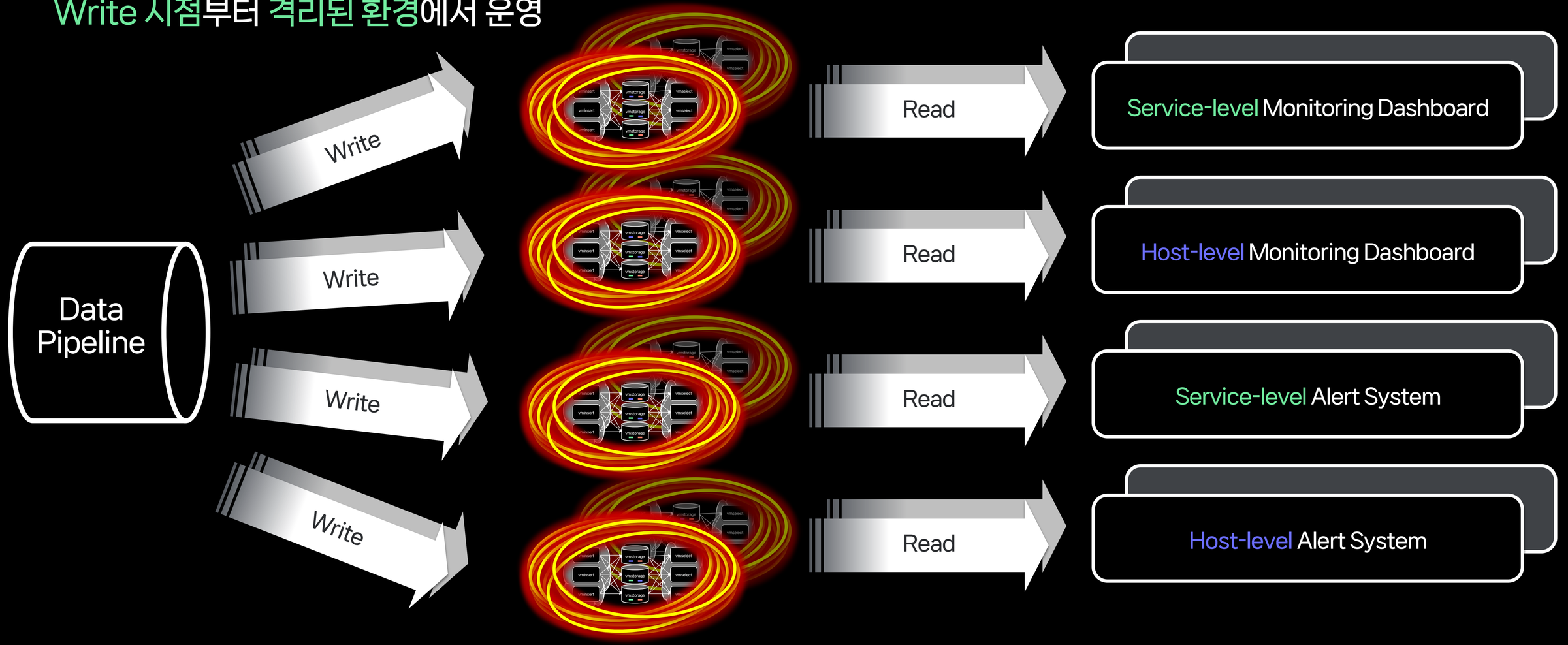
NAVER
DEVIEW
2023

Multiverse로 분리!

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Write 시점부터 격리된 환경에서 운영



3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Write 시점부터 격리된 환경에서 운영

대규모 클러스터 N개 운영 중

Stage 3 (시계열 수 백만 개) X 4 + α

Stage 4 (시계열 수 천만 개) X 4 + α

3. Time series in the Multiverse of Madness

NAVER
DEVIEW
2023

Summary

- Single Point of Failure 문제를 해결하기 위해 Cluster Mode 사용

3. Time series in the Multiverse of Madness

NAVER
DEVIEW
2023

Summary

- Single Point of Failure 문제를 해결하기 위해 Cluster Mode 사용
- 다양한 Application의 서로 다른 접근 패턴으로 과부하 문제 발생

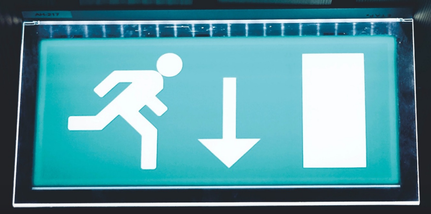
3. Time series in the Multiverse of Madness

NAVER
DEVIEW
2023

Summary

- Single Point of Failure 문제를 해결하기 위해 Cluster Mode 사용
- 다양한 Application의 서로 다른 접근 패턴으로 과부하 문제 발생
- Application들이 서로 영향을 주지 않도록 Multiverse 형태로 분리

4. Lessons Learned



- 모니터링 및 경보 시스템은 서비스의 비상등과 같은 역할

- 모니터링 및 경보 시스템은 서비스의 비상등과 같은 역할
- 서비스에 장애가 발생하더라도 모니터링 시스템은 정상 동작 해야 함

- 모니터링 및 경보 시스템은 서비스의 **비상등**과 같은 역할
- 서비스에 **장애가 발생**하더라도 **모니터링 시스템은 정상 동작** 해야 함
- 서비스에 장애가 발생하지 **않으면** 경보를 울리지 **않아야** 함

NAVER
DEVIEW
2023

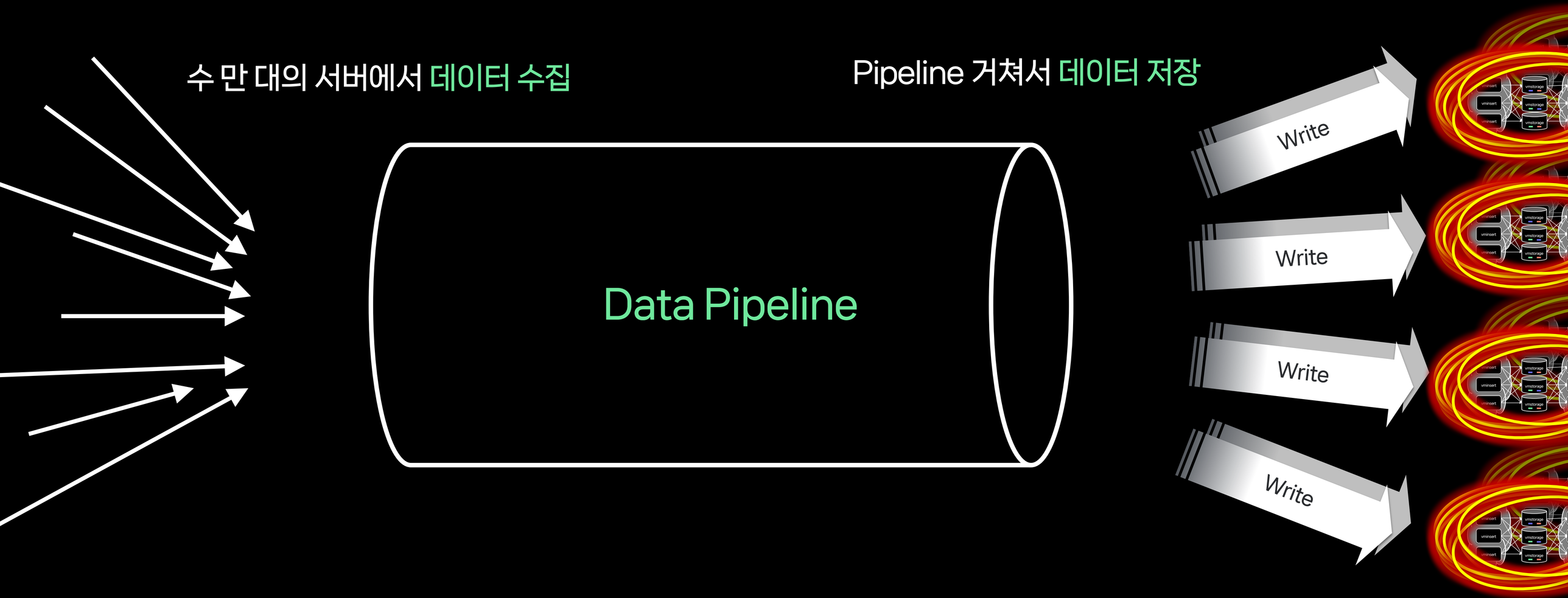
시계열 DB에서 데이터 유실(data loss)과 접속 중단(downtime)이

절대 발생하면 안되는 이유!

이미지 출처 : https://www.pexels.com/photo/lighted-running-signage-845265/

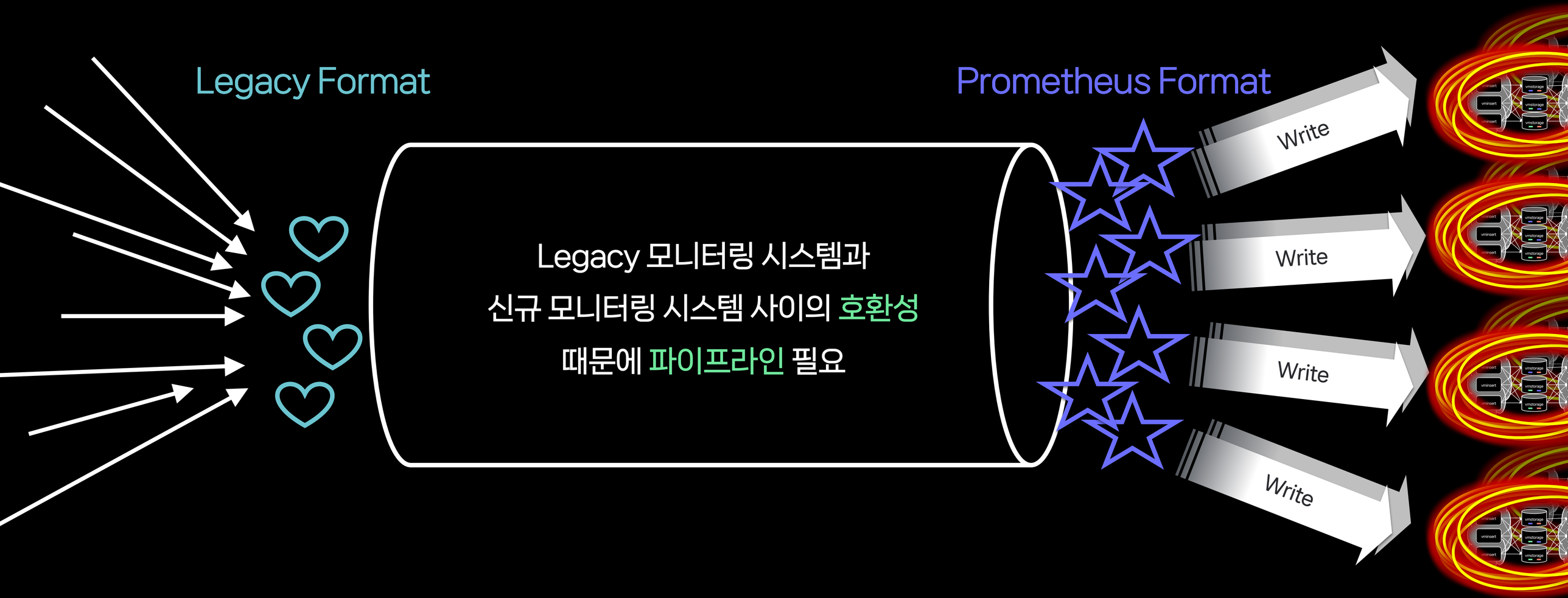
4.1 Write path for no data loss

NAVER
DEVIEW
2023



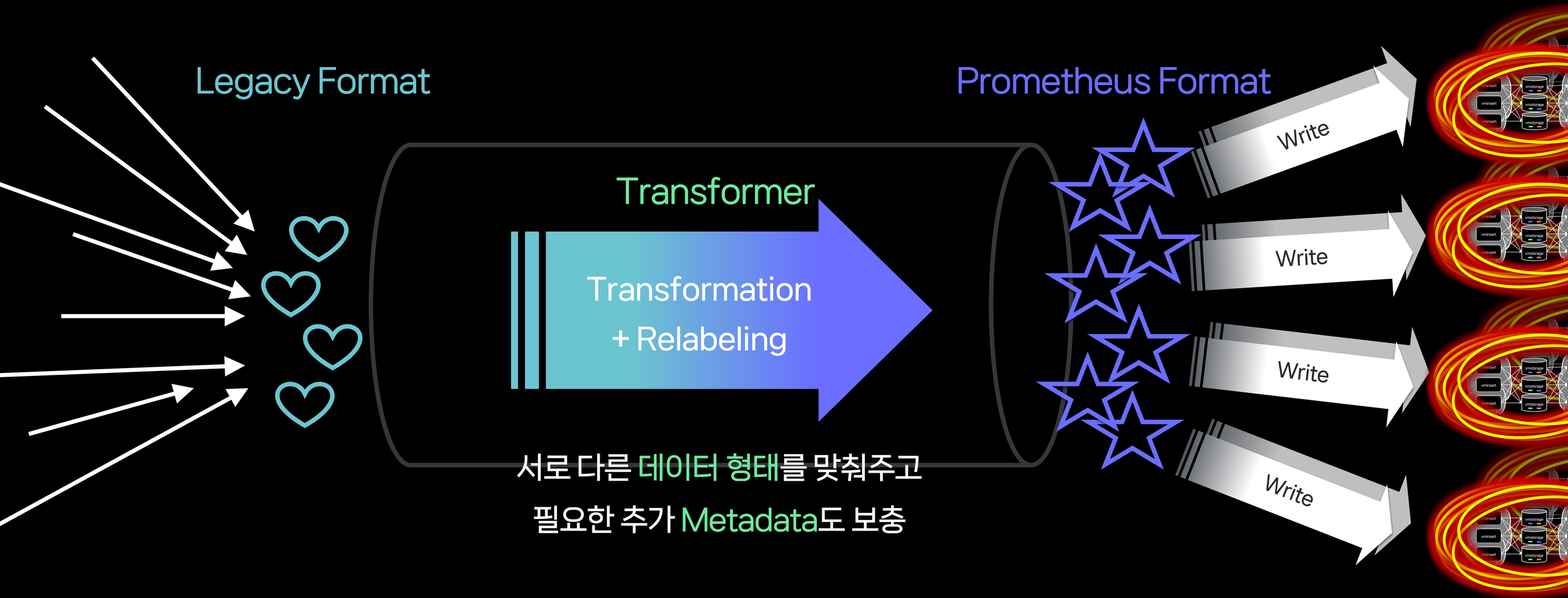
4.1 Write path for no data loss

NAVER
DEVIEW
2023



4.1 Write path for no data loss

NAVER
DEVIEW
2023

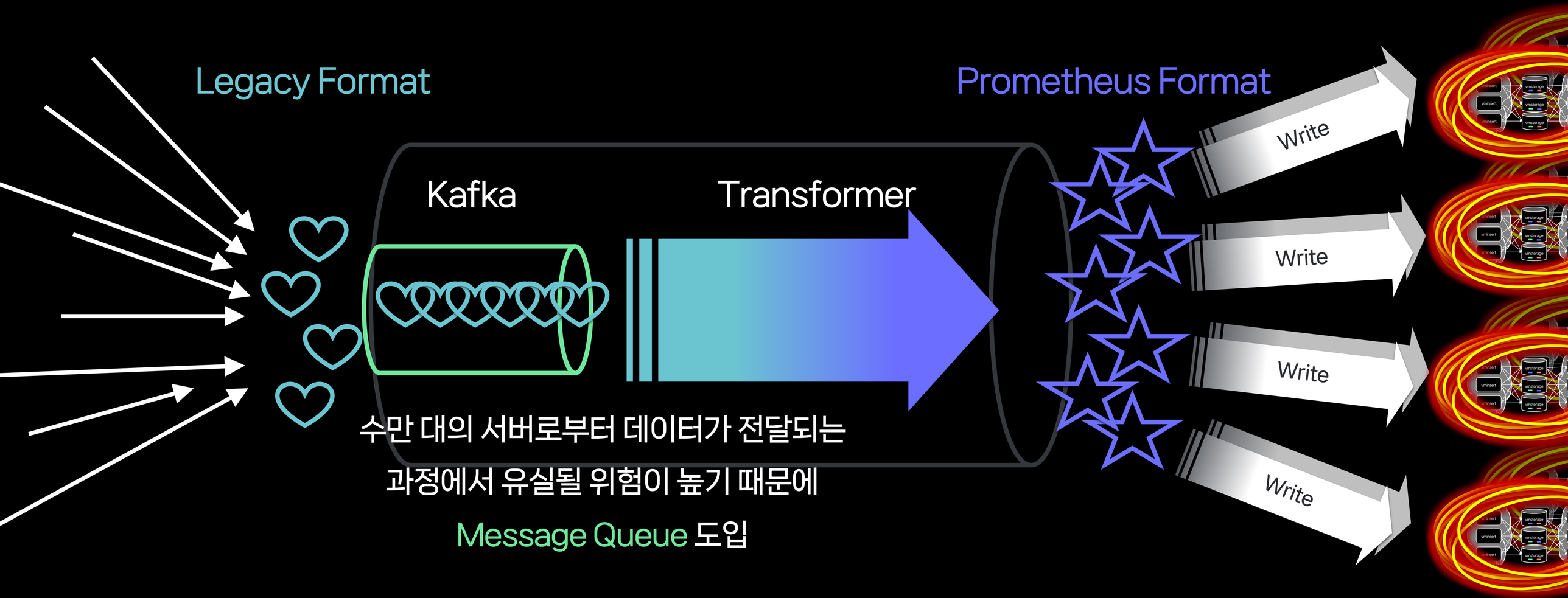


서로 다른 데이터 형태를 맞춰주고

필요한 추가 Metadata도 보충

4.1 Write path for no data loss

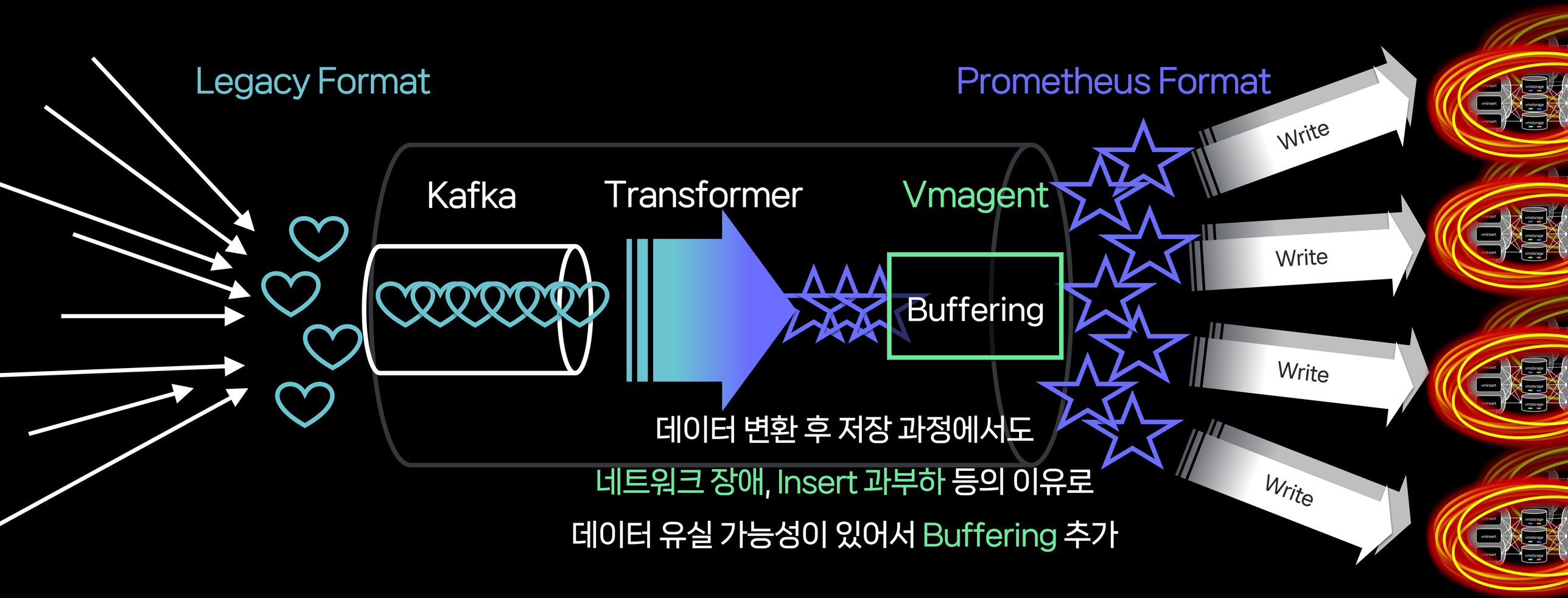
NAVER
DEVIEW
2023



수만 대의 서버로부터 데이터가 전달되는
과정에서 유실될 위험이 높기 때문에
Message Queue 도입

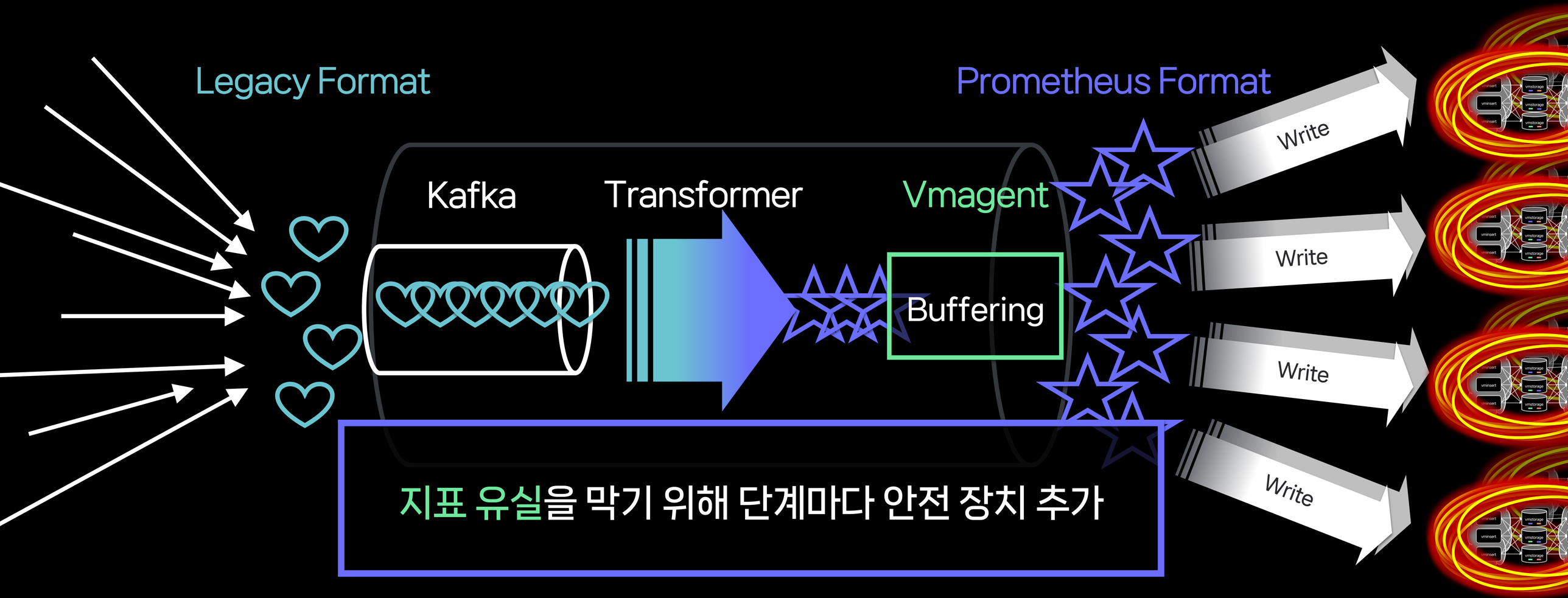
4.1 Write path for no data loss

NAVER
DEVIEW
2023



4.1 Write path for no data loss

NAVER
DEVIEW
2023

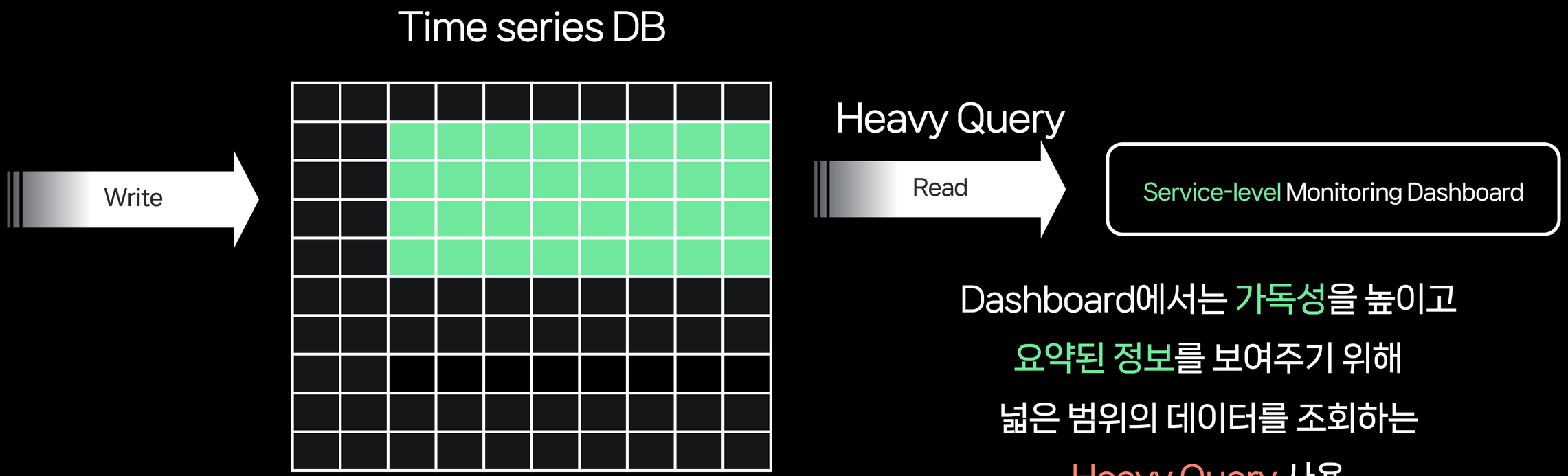


4.2 Read path for no downtime

NAVER
DEVIEW
2023

4.2 Read path for no downtime

NAVER
DEVIEW
2023



Dashboard에서는 가독성을 높이고

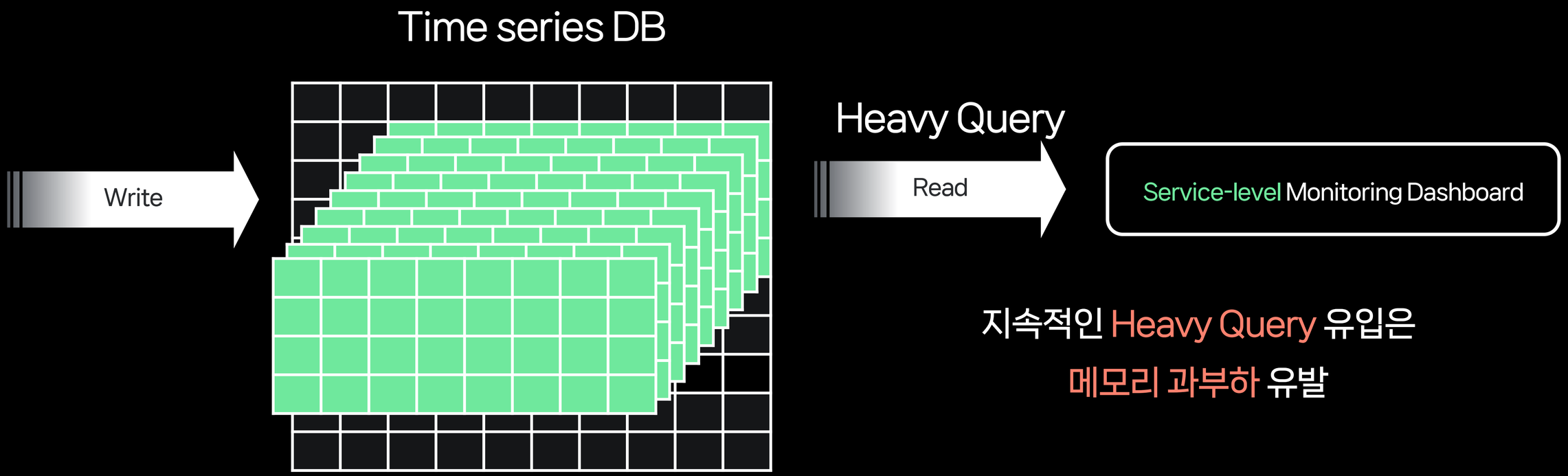
요약된 정보를 보여주기 위해

넓은 범위의 데이터를 조회하는

Heavy Query 사용

4.2 Read path for no downtime

NAVER
DEVIEW
2023

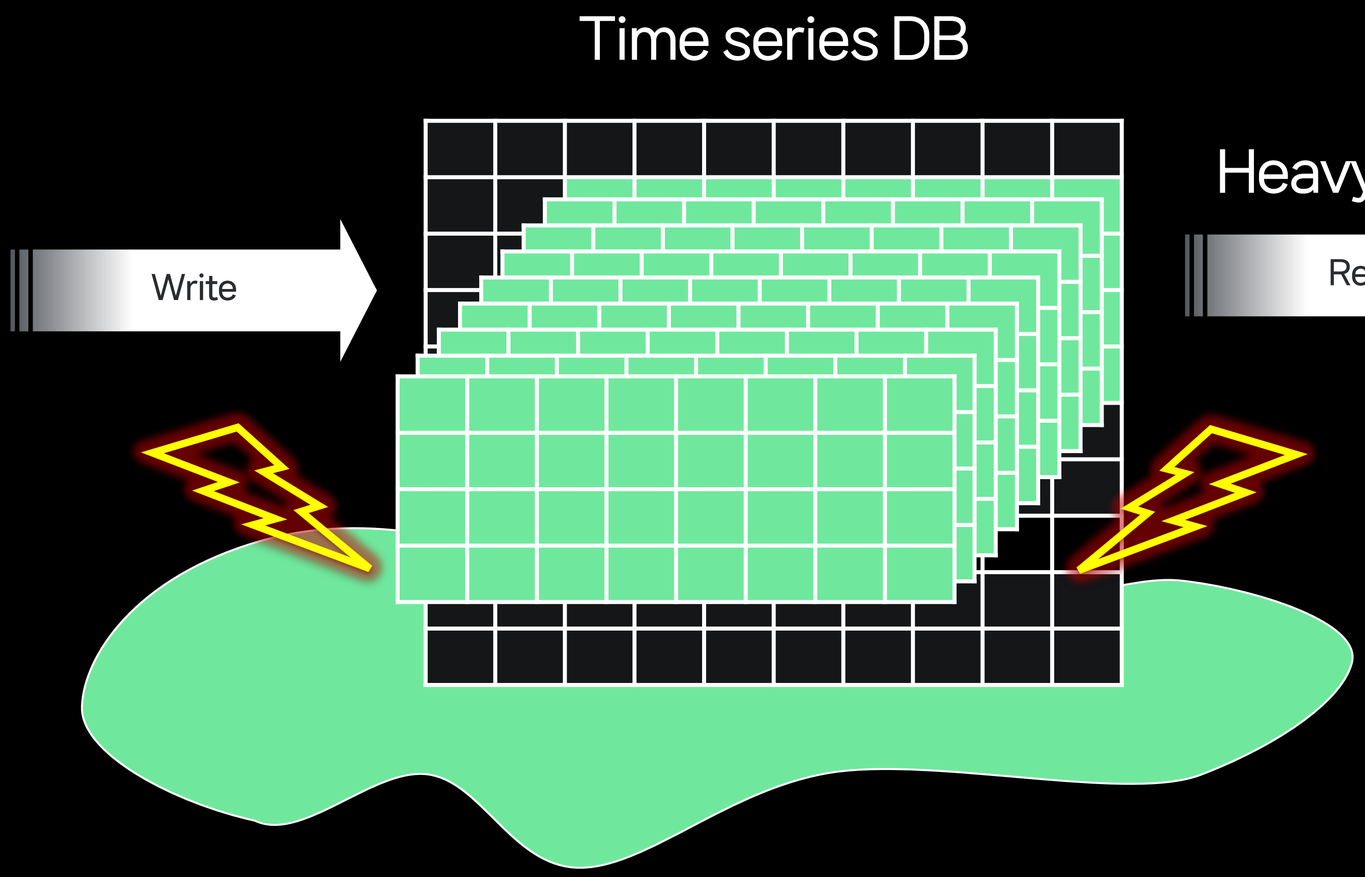


지속적인 Heavy Query 유입은

메모리 과부하 유발

4.2 Read path for no downtime

NAVER
DEVIEW
2023



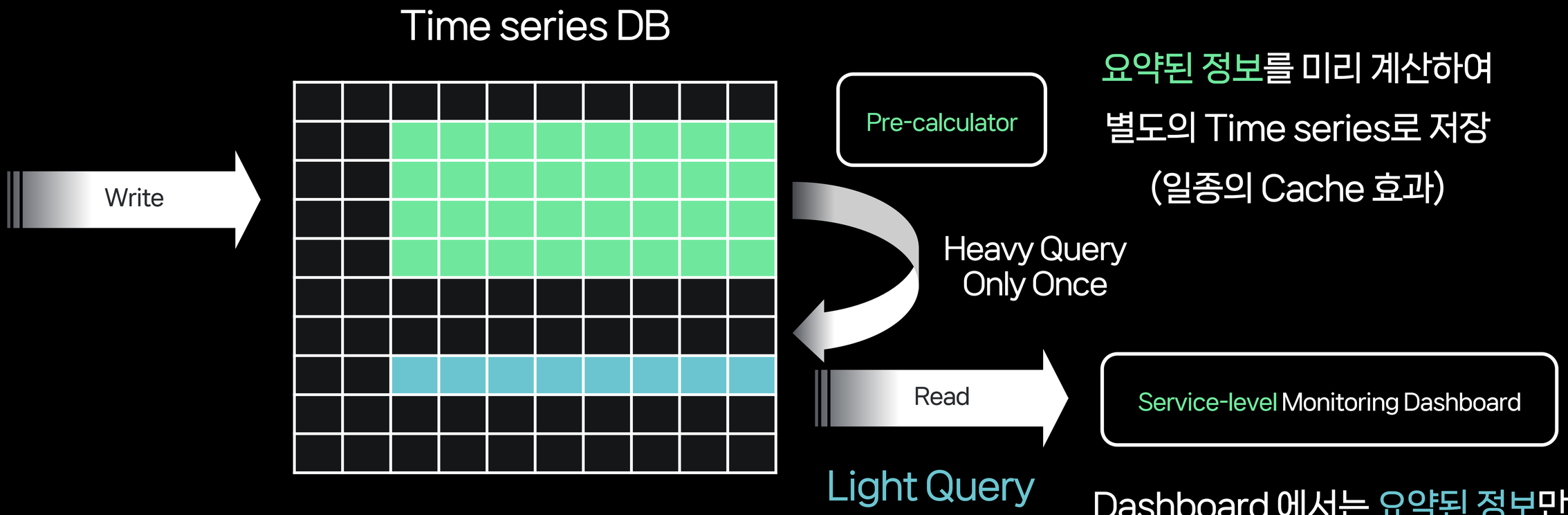
지속적인 Heavy Query 유입은

메모리 과부하 유발

→ OOM-Kill의 원인이 될 수 있음!

4.2 Read path for no downtime

NAVER
DEVIEW
2023

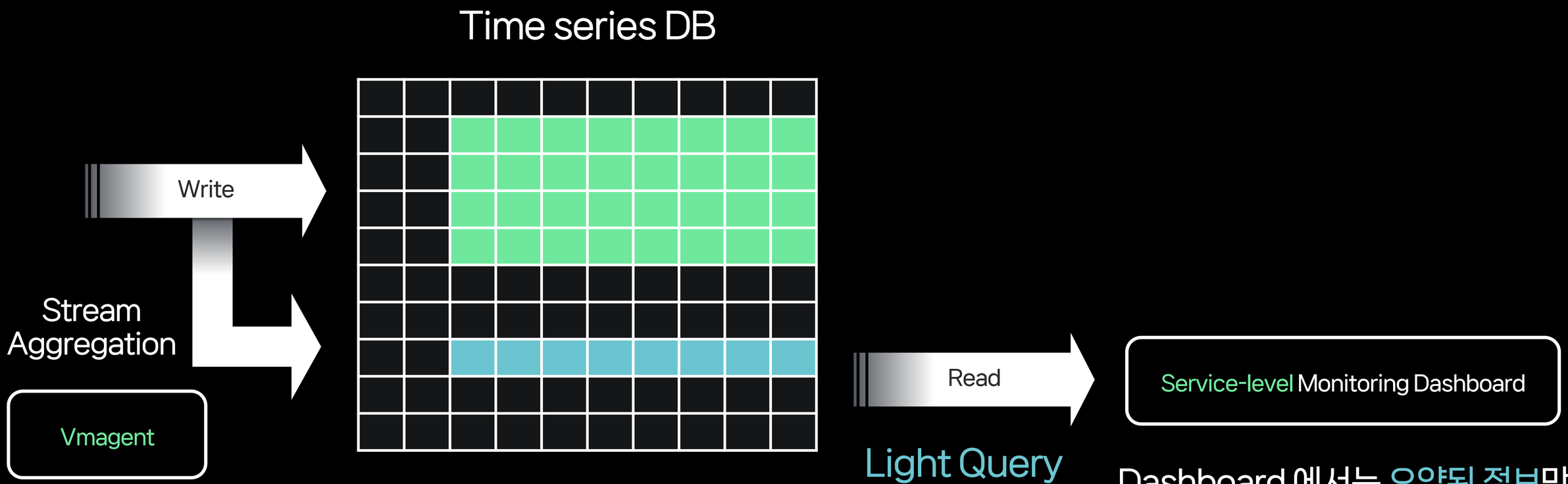


요약된 정보를 미리 계산하여
별도의 Time series로 저장
(일종의 Cache 효과)

Dashboard 에서는 요약된 정보만
요청하므로 훨씬 가벼운 Query 사용

4.2 Read path for no downtime

NAVER
DEVIEW
2023



처음부터 요약된 정보로 저장하는 방식 (적용 예정)

Dashboard 에서는 요약된 정보만
요청하므로 훨씬 가벼운 Query 사용

4.2 Read path for no downtime

NAVER
DEVIEW
2023

처음부터 요약된 정보로 저장하는 방식 (적용 예정)

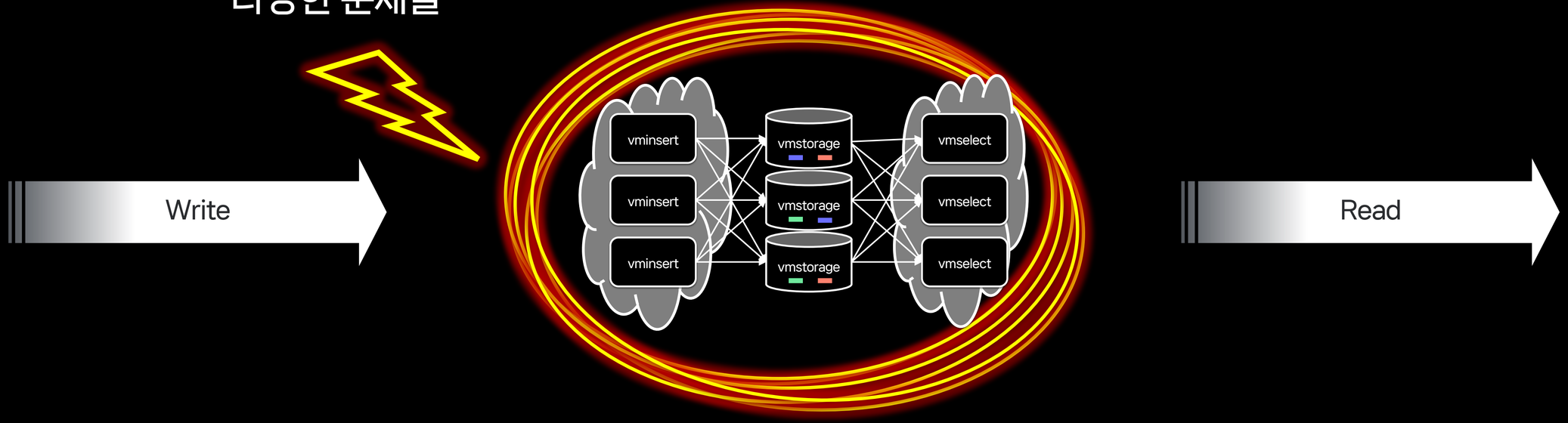
Dashboard 에서는 요약된 정보만
요청하므로 훨씬 가벼운 Query 사용

4.3 Multiverse Management

NAVER
DEVIEW
2023

그래도 끝나지 않는 고통

인프라 수준에서 발생하는
다양한 문제들



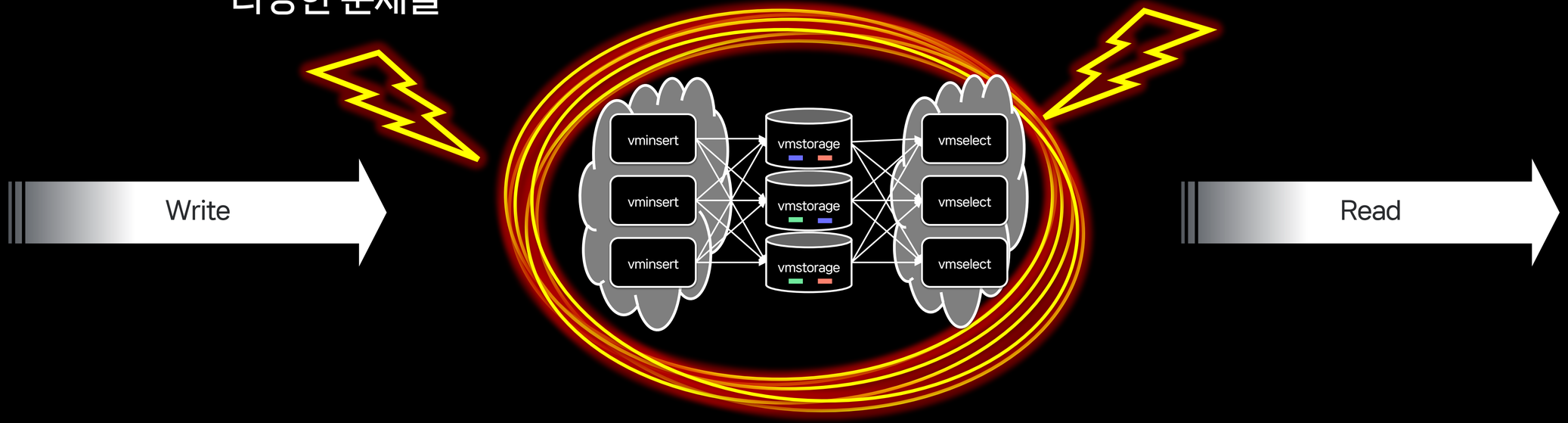
4.3 Multiverse Management

NAVER
DEVIEW
2023

그래도 끝나지 않는 고통

인프라 수준에서 발생하는
다양한 문제들

순간적인 과부하로
Write / Read 지연



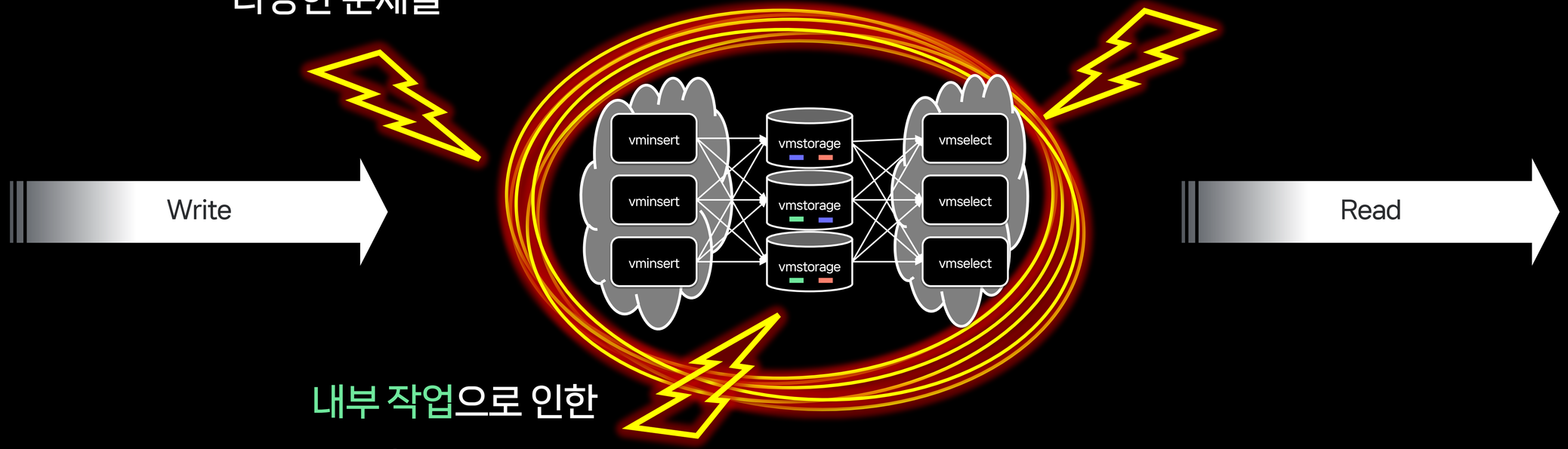
4.3 Multiverse Management

NAVER
DEVIEW
2023

그래도 끝나지 않는 고통

인프라 수준에서 발생하는
다양한 문제들

순간적인 과부하로
Write / Read 지연

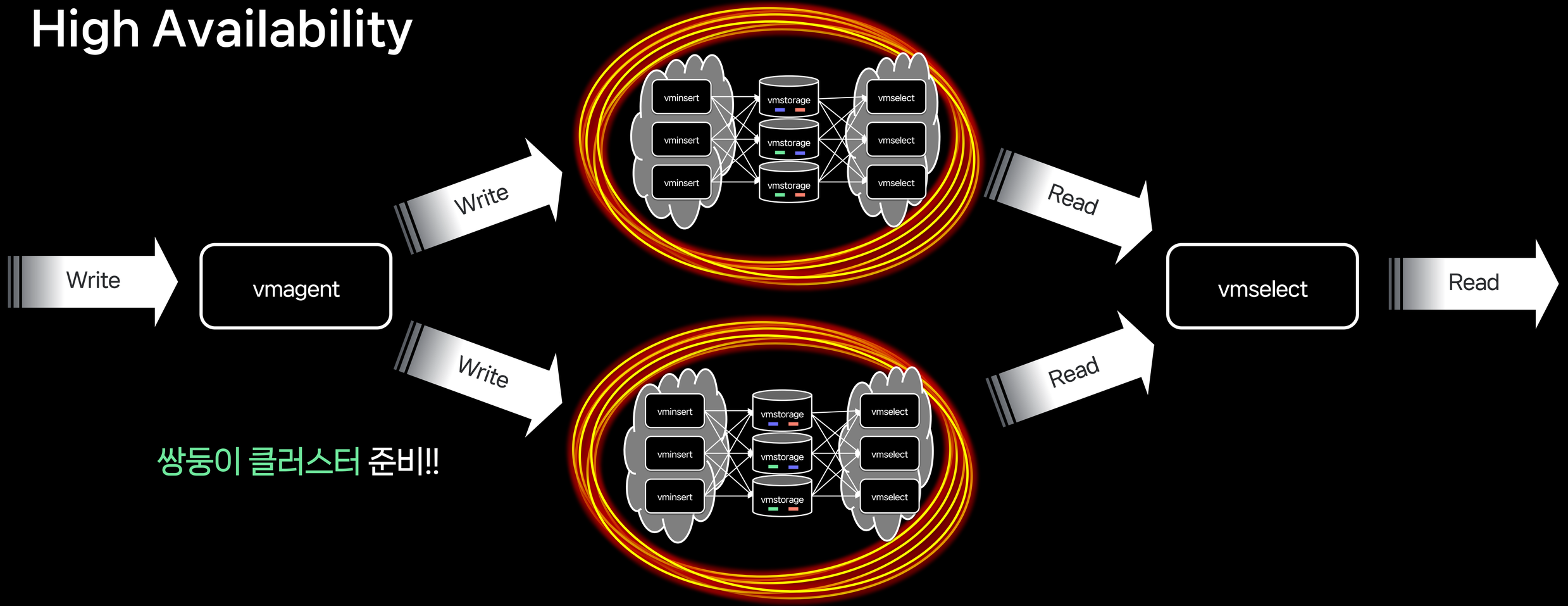


내부 작업으로 인한
리소스 부족

4.3 Multiverse Management

NAVER
DEVIEW
2023

High Availability

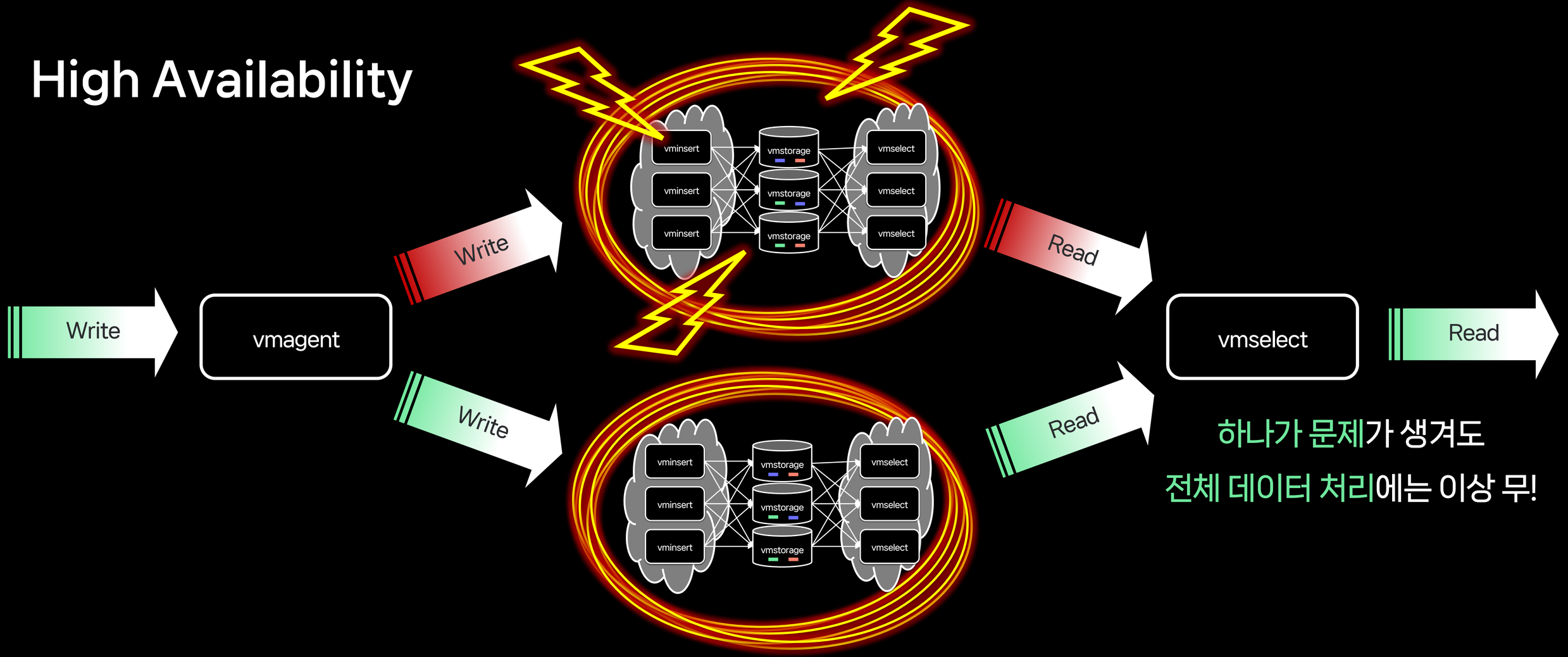


쌍둥이 클러스터 준비!!

4.3 Multiverse Management

NAVER
DEVIEW
2023

High Availability



하나가 문제가 생겨도

전체 데이터 처리에는 이상 무!

4.3 Multiverse Management

NAVER
DEVIEW
2023

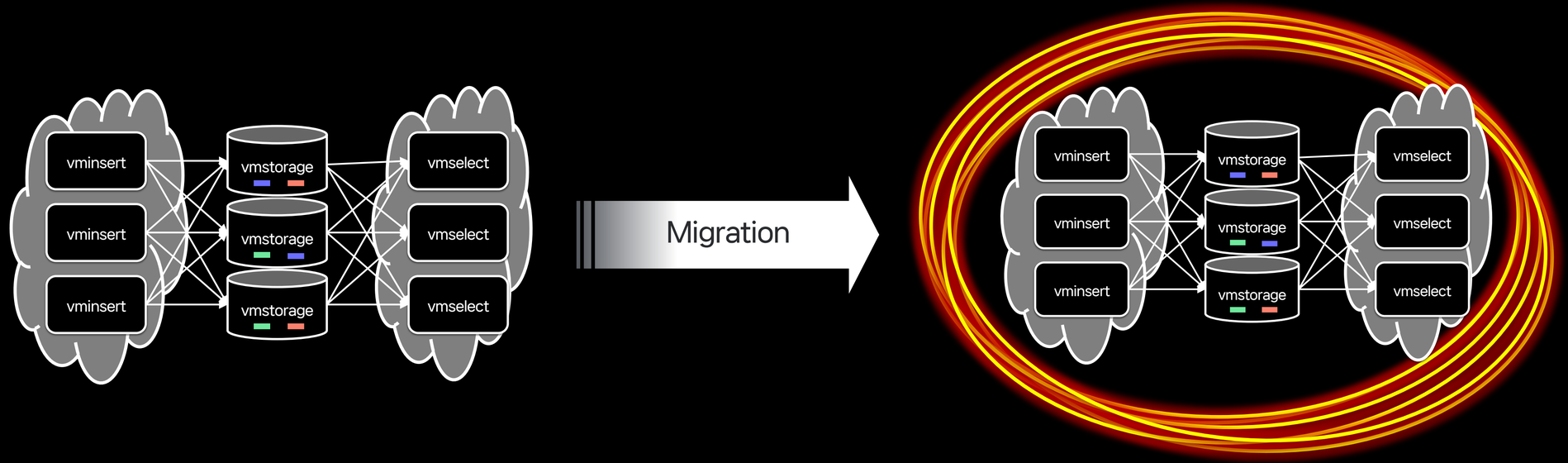
High Availability

4.3 Multiverse Management

NAVER
DEVIEW
2023

Data Migration

- 한 Universe에 저장된 데이터를 다른 Universe로 옮겨야 하는 경우

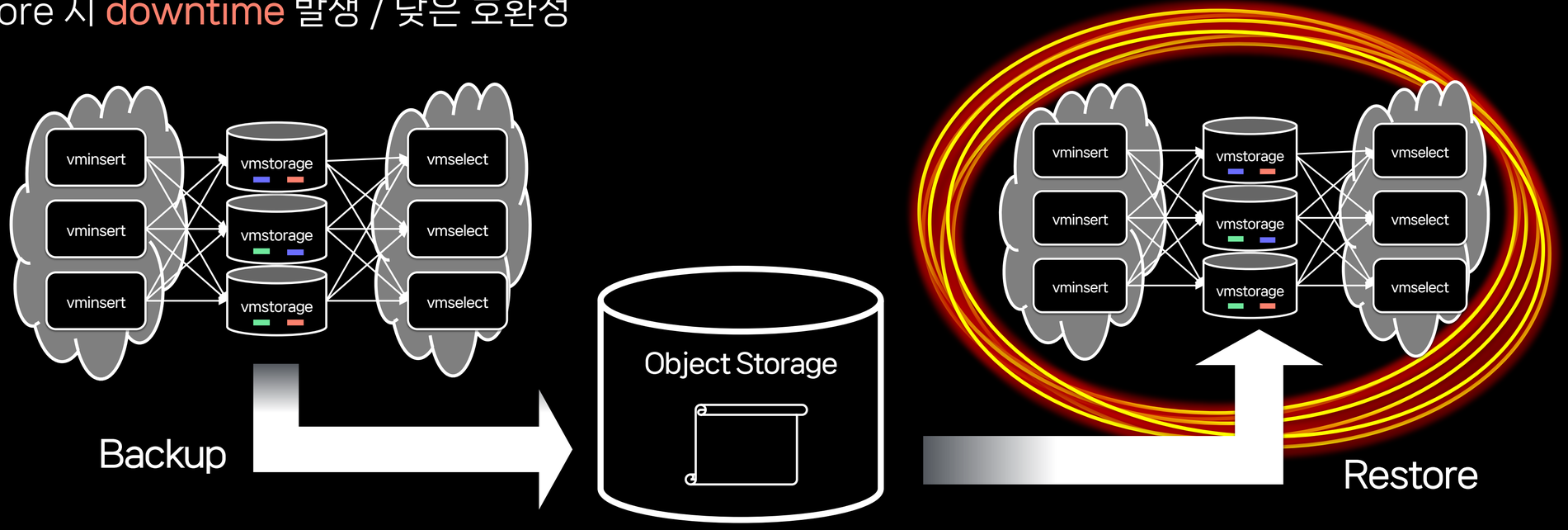


4.3 Multiverse Management

NAVER
DEVIEW
2023

Backup & Restore 방식

- 압축된 데이터 그대로 Dump 후 붙여넣는 방식 (storage to storage)
- 장점 : 빠른 속도 (Network Bandwidth가 허용되는 만큼 전송 가능)
- 단점 : Restore 시 downtime 발생 / 낮은 호환성

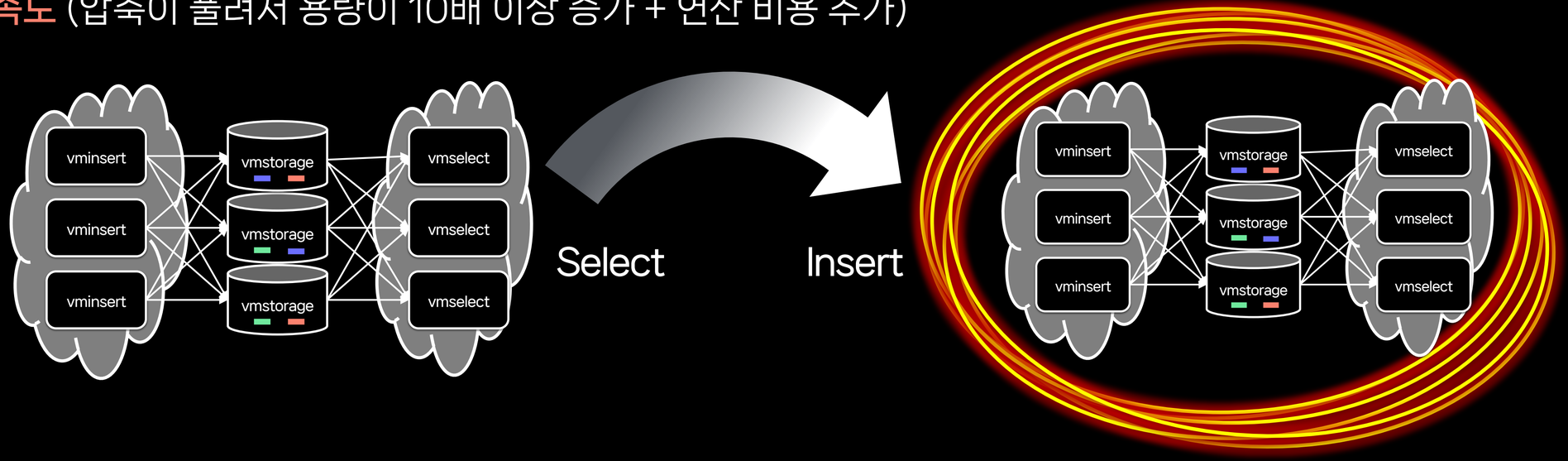


4.3 Multiverse Management

NAVER
DEVIEW
2023

Select & Insert 방식

- 데이터를 일단 Source 클러스터에서 Select 한 다음에 다시 Destination 클러스터에 Insert를 하는 방식
- 장점 : 뛰어난 호환성, 무중단 작업 가능
- 단점 : 느린 속도 (압축이 풀려서 용량이 10배 이상 증가 + 연산 비용 추가)



4.3 Multiverse Management

NAVER
DEVIEW
2023

Select & Insert 방식

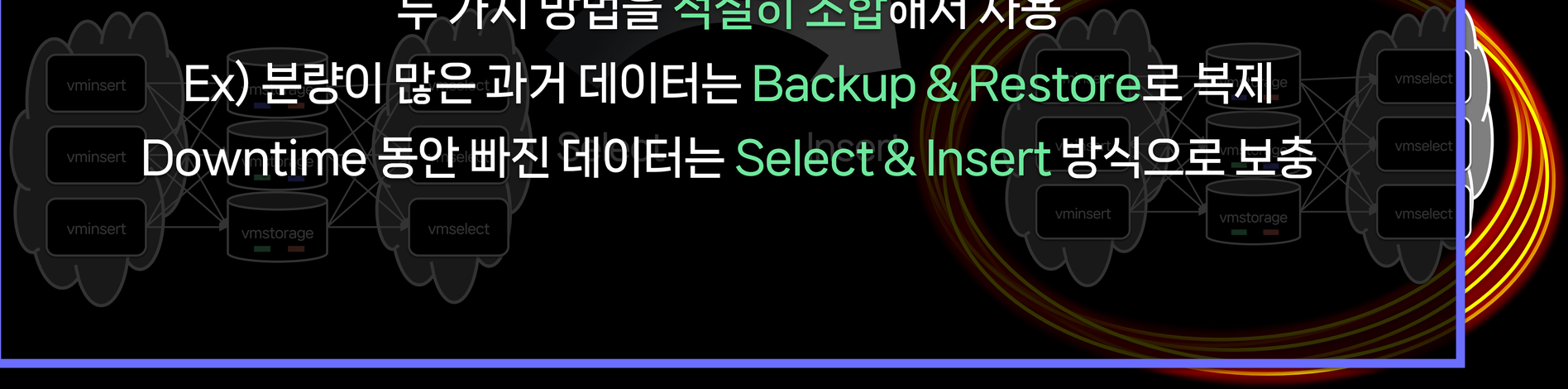
- 데이터를 일단 Source 클러스터에서 Select 한 다음에 다시 Destination 클러스터에 Insert를 하는 방식
- 장점 : 뛰어난 호환성, 무중단 작업 가능
- 단점 : 느린 속도

장단점이 분명하므로 실제 작업 시에는

두 가지 방법을 적절히 조합해서 사용

Ex) 분량이 많은 과거 데이터는 Backup & Restore로 복제

Downtime 동안 빠진 데이터는 Select & Insert 방식으로 보충



4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- 비상등 역할을 잘 하기 위해서 **지표 유실**과 **중단 시간 최소화** 필요

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- 비상등 역할을 잘 하기 위해서 지표 유실과 중단 시간 최소화 필요
- 지표 유실을 방지하기 위해 Message Queue와 Data Buffering 도입

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- 비상등 역할을 잘 하기 위해서 지표 유실과 중단 시간 최소화 필요
- 지표 유실을 방지하기 위해 Message Queue와 Data Buffering 도입
- 중단 시간 최소화를 위해 Pre-calculator를 통한 Heavy Query 완화

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- 비상등 역할을 잘 하기 위해서 지표 유실과 중단 시간 최소화 필요
- 지표 유실을 방지하기 위해 Message Queue와 Data Buffering 도입
- 중단 시간 최소화를 위해 Pre-calculator를 통한 Heavy Query 완화
- High Availability 구성을 위해 Multi-level Cluster 기능 도입

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- 비상등 역할을 잘 하기 위해서 지표 유실과 중단 시간 최소화 필요
- 지표 유실을 방지하기 위해 Message Queue와 Data Buffering 도입
- 중단 시간 최소화를 위해 Pre-calculator를 통한 Heavy Query 완화
- High Availability 구성을 위해 Multi-level Cluster 기능 도입
- 효과적인 Multiverse 운영을 위해 Data Migration 기능 활용

5. Takeaways

5. Takeaways

NAVER
DEVIEW
2023

Recall

참고 : The cost of scale in Prometheus ecosystem

Stage	# of Time series	Environment
0	백만 개 이하	Legacy systems
1	백만 개	Distributed systems
2	2 백만 개	Poplular platforms
3	5 백만 개	Custom applications
4	수 천만 개	Microservices
5	수 십억 개	Kubernetes

5. Takeaways

NAVER
DEVIEW
2023

Recall

참고 : The cost of scale in Prometheus ecosystem

Stage	# of Time series	Environment	Time series DB
0	백만 개 이하	Legacy systems	Prometheus
1	백만 개	Distributed systems	
2	2 백만 개	Poplular platforms	
3	5 백만 개	Custom applications	
4	수 천만 개	Microservices	**VictoriaMetrics Single or Cluster**
5	수 십억 개	Kubernetes	**VictoriaMetrics Cluster**

5. Takeaways

NAVER
DEVIEW
2023

No silver bullet

- VictoriaMetrics가 꽤 많은 문제를 해결해주는 것은 사실
- 그러나 무중단, 그리고 지표 유실 없이 오랜 기간 운영하고
  다양한 시나리오에 대응하려면 여전히 고민할 부분도 많고 운영 노하우가 필요

함께 고민해봅시다

- 기술 직무 안내 사이트
- 채용 문의

Q & A

Thank You

Appendix

NAVER
DEVIEW
2023

LSM Tree 관련

- The Secret Sauce Behind NoSQL: LSM Tree
- Log Structured Merge Trees
- Scaling Write-Intensive Key-Value Stores

VictoriaMetrics 관련

- VictoriaMetrics: scaling to 100 million metrics per second
- Specifics of Data Analysis in Time Series Databases
- The cost of scale in Prometheus ecosystem

Appendix

NAVER
DEVIEW
2023

다른 제품과 비교

- VictoriaMetrics VS ScyllaDB
- VictoriaMetrics VS Prometheus
- VictoriaMetrics VS OpenTSDB
- VictoriaMetrics VS Mimir

기타 참고 사례

- VictoriaMetrics + Monarch
- VictoriaMetrics + Thanos
- VictoriaMetrics 관련 문서 모음

English Ver. (slide.171 - 340)

VictoriaMetrics: Time series in the Multiverse of Madness

SON JOOSIK, LEE SEONKYU
NAVER Search

NAVER DEVIEW 2023

CONTENTS

1. Background

1.1 Monitoring NAVER Search System

NAVER DEVIEW 2023

NAVER Search: A Large-Scale Distributed System

- Billions of search requests
- Tens of thousands of search servers
- Hundreds of search services

1.1 Monitoring NAVER Search System

NAVER DEVIEW 2023

NAVER Search: A Large-Scale Distributed System

- Billions of search requests
- Tens of thousands of search servers
- Hundreds of search services

Search SRE: Keeping NAVER Search reliable!!

- Monitoring NAVER Search's many services, systems
- Outage prevention and response

1.1 Monitoring NAVER Search System

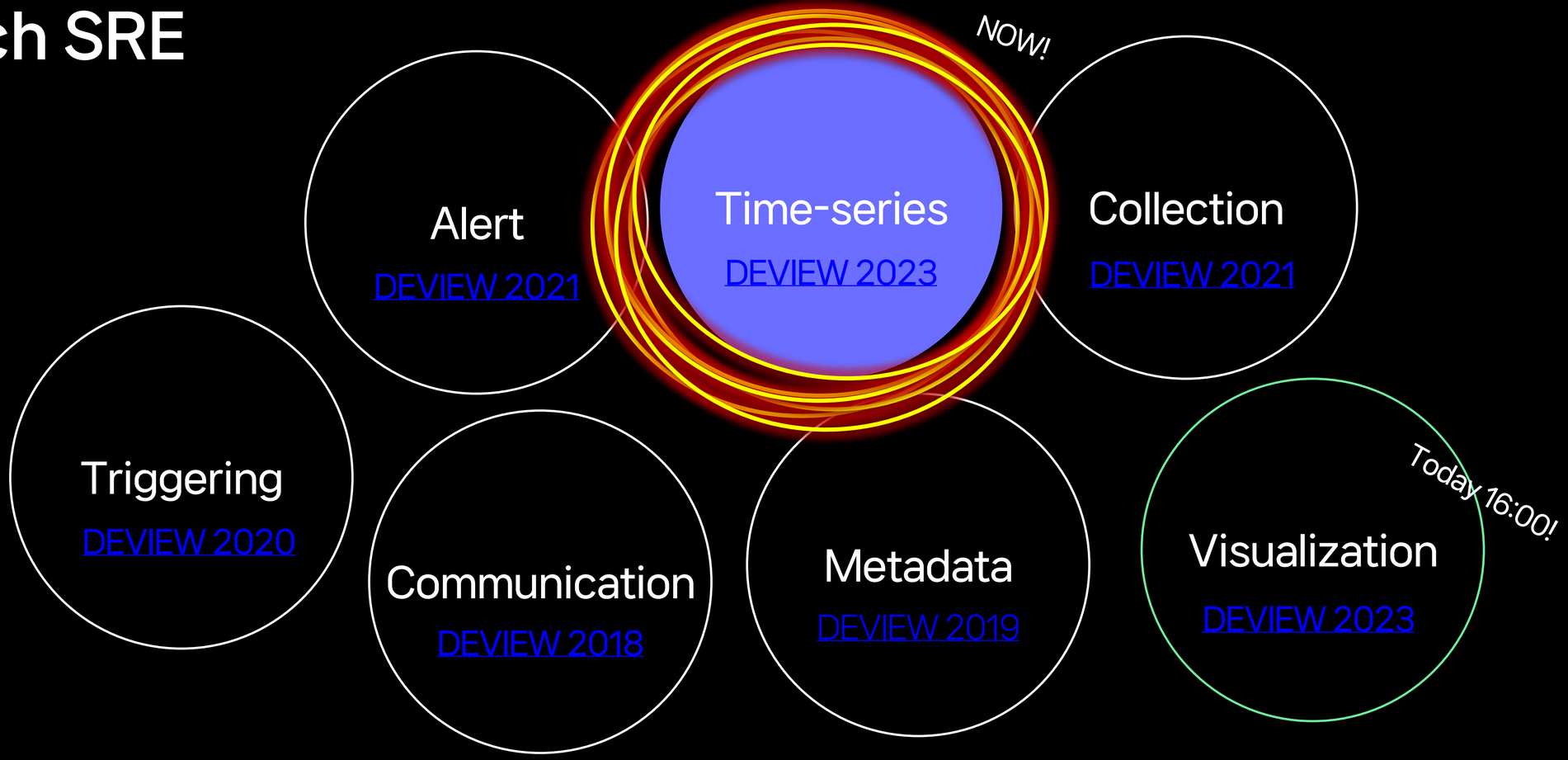
NAVER DEVIEW 2023

Search SRE

1.1 Monitoring NAVER Search System

NAVER DEVIEW 2023

Search SRE



1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is time series data?

- Numeric data in chronological order

1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is time series data?

- Numeric data in chronological order

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	…
A	190	170	250	160	220	200	240	200	…
B	250	195	185	260	175	180	200	240	…
C	201	203	205	208	210	222	224	226	…
…	…	…	…	…	…	…	…	…	…

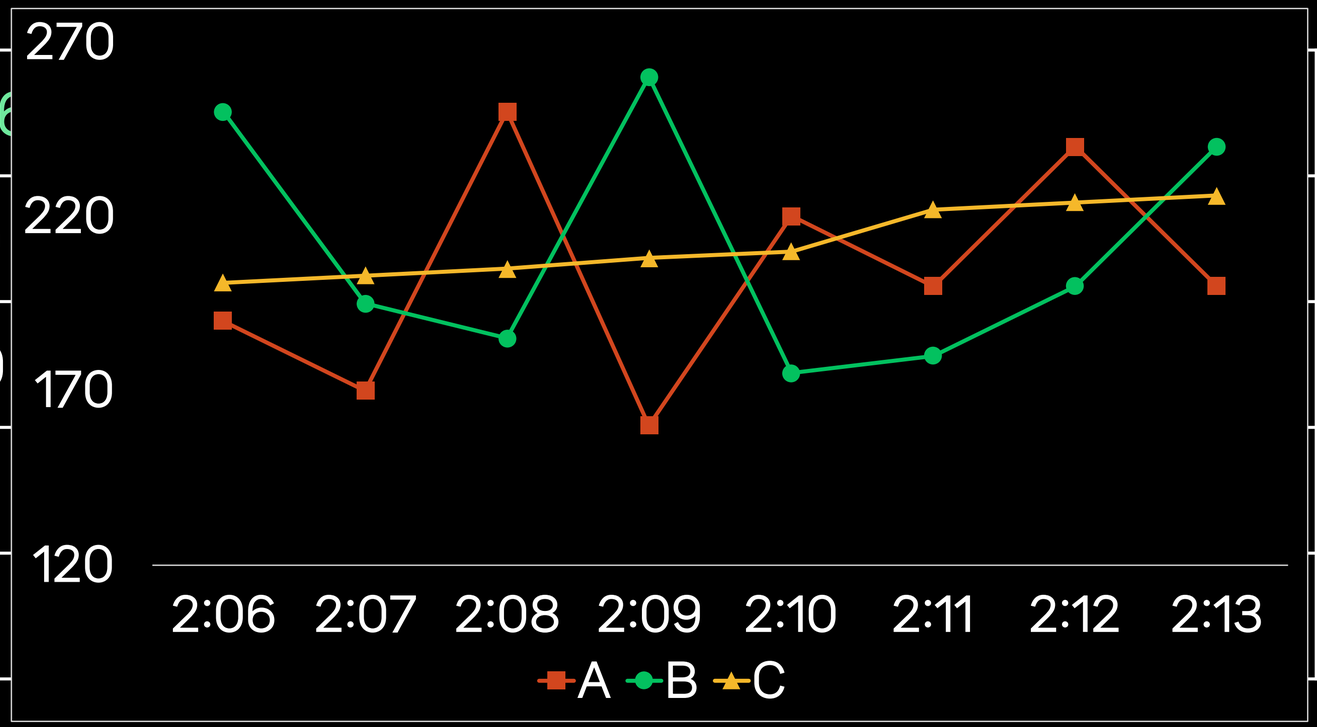
1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is time series data?

- Numeric data in chronological order

	02:06		02:13	...
A	190		200	...
B	250		240	...
C	201		226	...
...	...		...	...



1.2 Large-scale time series data

NAVER
DEVIEW
2023

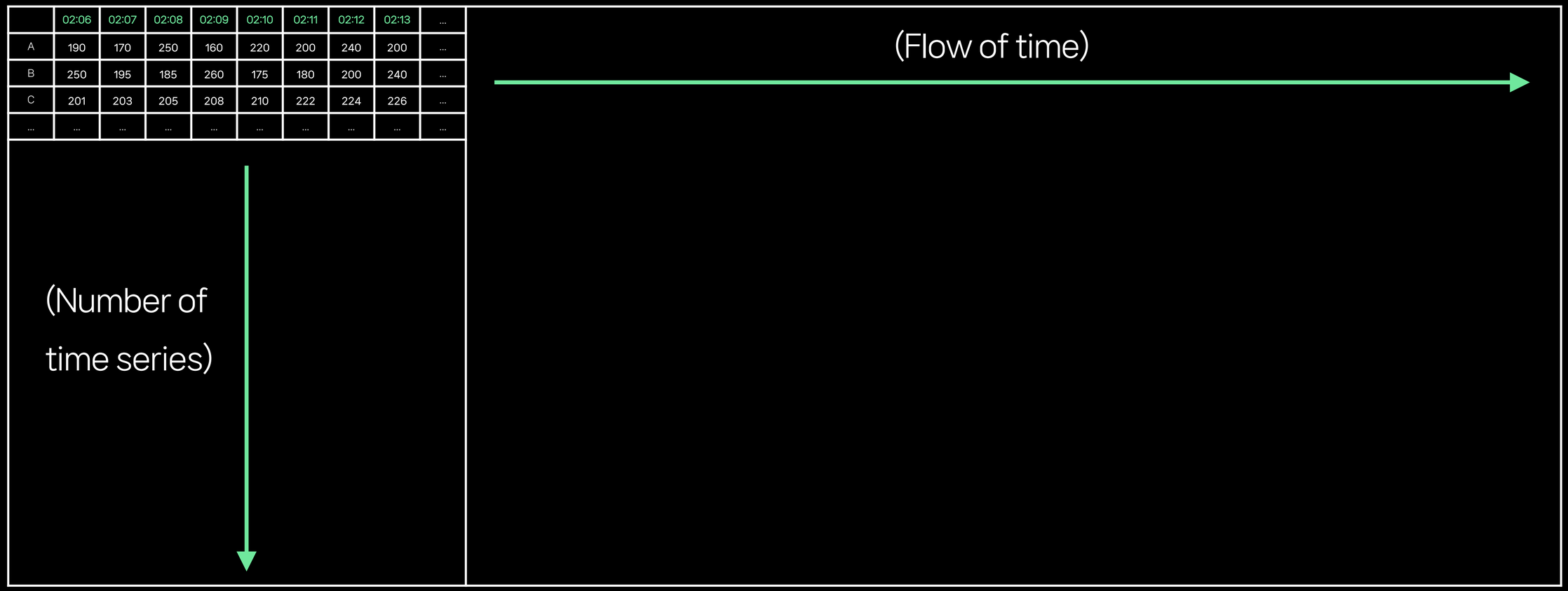
What is large-scale time series data?

1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is large-scale time series data?

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	...
A	190	170	250	160	220	200	240	200	...
B	250	195	185	260	175	180	200	240	...
C	201	203	205	208	210	222	224	226	...
...	...	...	...	...	...	...	...	...	...



1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is large-scale time series data?

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	...
A	190	170	250	160	220	200	240	200	...
B	250	195	185	260	175	180	200	240	...
C	201	203	205	208	210	222	224	226	...
...	...	...	...	...	...	...	...	...	...

(Number of time series)

(Flow of time)

Total data size = (data retention) X (number of time series)

The longer you keep your data,

the higher the number of time series,

1.2 Large-scale time series data

NAVER
DEVIEW
2023

What is large-scale time series data?

	02:06	02:07	02:08	02:09	02:10	02:11	02:12	02:13	...
A	190	170	250	160	220	200	240	200	...
B	250	195	185	260	175	180	200	240	...
C	201	203	205	208	210	222	224	226	...
...	...	...	...	...	...	...	...	...	...

(Number of time series)

(Flow of time)

Total data size = (data retention) X (number of time series)

The longer you keep your data,

the higher the number of time series,

Large Scale time series

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems

Ref. : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems
- Stage 1 : 1 million → Distributed systems

Ref. : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems
- Stage 1 : 1 million → Distributed systems
- Stage 2 : 2 million → Poplular platforms

Ref. : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems
- Stage 1 : 1 million → Distributed systems
- Stage 2 : 2 million → Poplular platforms
- Stage 3 : 5 million → Custom applications

Ref. : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems
- Stage 1 : 1 million → Distributed systems
- Stage 2 : 2 million → Poplular platforms
- Stage 3 : 5 million → Custom applications
- Stage 4 : Tens of millions → Microservices

Ref. : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems
- Stage 1 : 1 million → Distributed systems
- Stage 2 : 2 million → Poplular platforms
- Stage 3 : 5 million → Custom applications
- Stage 4 : Tens of millions → Microservices
- Stage 5 : Billions → Kubernetes

Ref. : The cost of scale in Prometheus ecosystem

1.2 Large-scale time series data

NAVER
DEVIEW
2023

How many time series are created in a server monitoring environment?

- Stage 0 : Less than a million → Legacy systems
- Stage 1 : 1 million → Distributed systems
- Stage 2 : 2 million → Poplular platforms
- Stage 3 : 5 million → Custom applications
- Stage 4 : Tens of millions → Microservices
- Stage 5 : Billions → Kubernetes

} Large scale time series:
Specialized solution required

Ref. : The cost of scale in Prometheus ecosystem

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus and Gorilla Compression Algorithms

- Prometheus appears in 2012 → De-facto monitoring system

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus and Gorilla Compression Algorithms

- Prometheus appears in 2012
- Gorilla compression algorithm in 2015

→ Facebook's time series-specific compression technology

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus and Gorilla Compression Algorithms

- Prometheus appears in 2012
- Gorilla compression algorithm in 2015
- Time series data in the millions (Stage 3) can now be solved with Prometheus

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus and Gorilla Compression Algorithms

- Prometheus appears in 2012
- Gorilla compression algorithm in 2015
- Time series data in the millions (Stage 3) can now be solved with Prometheus
- However, still difficult to handle scales beyond tens of millions (Stage 4, 5)

1.3 Time series DB History

NAVER
DEVIEW
2023

Prometheus and Gorilla Compression Algorithms

- Prometheus appears in 2012
- Gorilla compression algorithm in 2015
- Time series data in the millions (Stage 3) can now be solved with Prometheus
- However, still difficult to handle scales beyond tens of millions (Stage 4, 5)

Large scale Time Series

1.3 Time series DB History

NAVER
DEVIEW
2023

Scalable Solutions to Overcome the Limitations of Prometheus

- Thanos
- Cortex
- Grafana Mimir
- M3DB
- Promscale

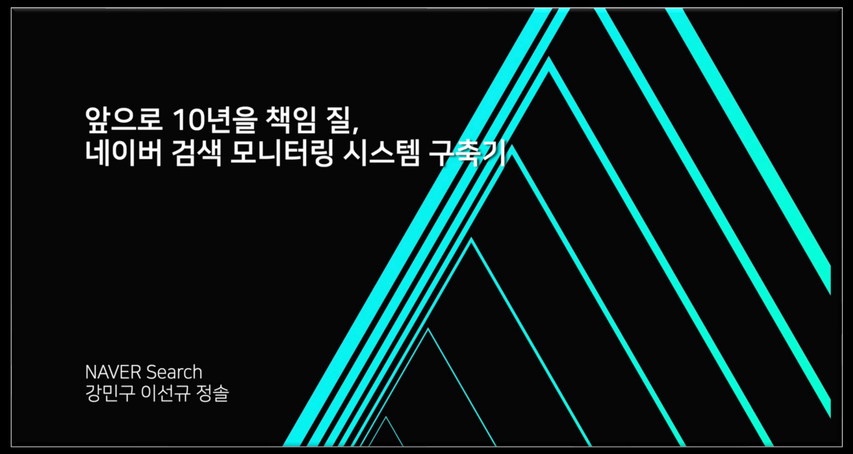
1.3 Time series DB History

NAVER
DEVIEW
2023

Scalable Solutions to Overcome the Limitations of Prometheus

- Thanos
- Cortex
- Grafana Mimir
- M3DB
- Promscale
- VictoriaMetrics ← Prometheus compatible,
  high performance, easy to operate

Ref. : The cost of scale in Prometheus ecosystem



You can read more about the story
in our previous DEVIEW 2021

2. A Deep-dive into Time series DB

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

(Request Format)

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

Time series Name　　UNIX Timestamp　Value

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

(Request Format)

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

(Format in storage)

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06					
	190					

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
```

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07				
	190	170				

2.1 Data Model

NAVER
DEVIEW
2023

Write Requests Example

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
```

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07	02:08			
	190	170	250			

2.1 Data Model

NAVER
DEVIEW
2023

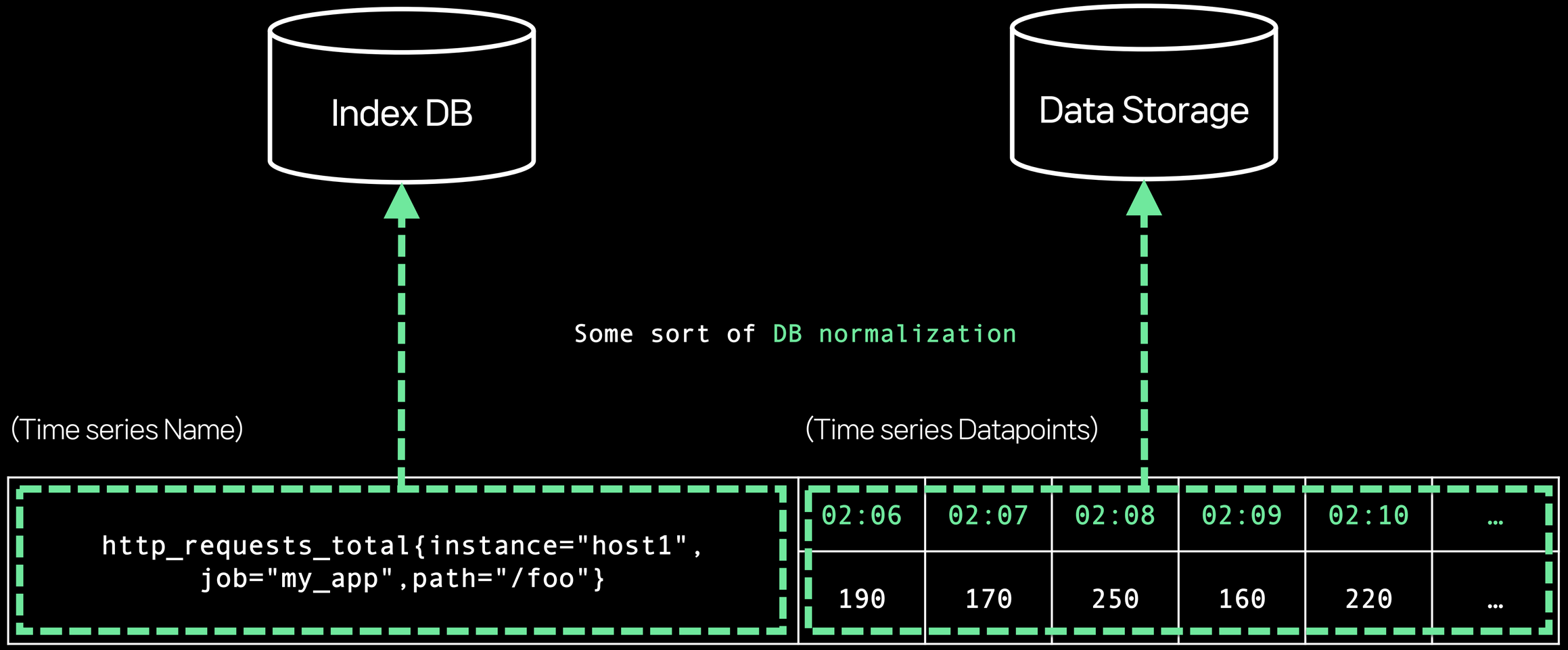
Write Requests Example

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271340  160
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271400  220
…
```

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…

2.1 Data Model

NAVER
DEVIEW
2023



2.1 Data Model

NAVER
DEVIEW
2023

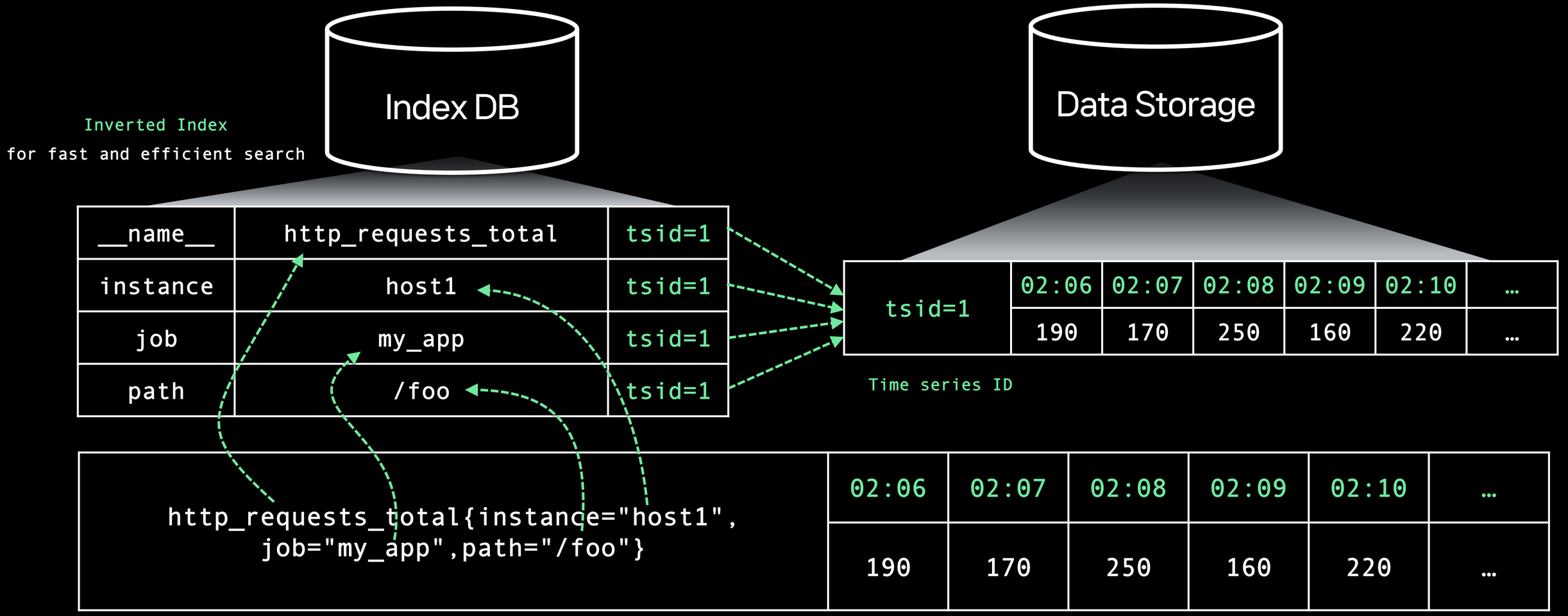
tsid=1	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…

Time series ID

http_requests_total{instance="host1", job="my_app",path="/foo"}	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…

2.1 Data Model

NAVER
DEVIEW
2023



2.1 Data Model

NAVER
DEVIEW
2023

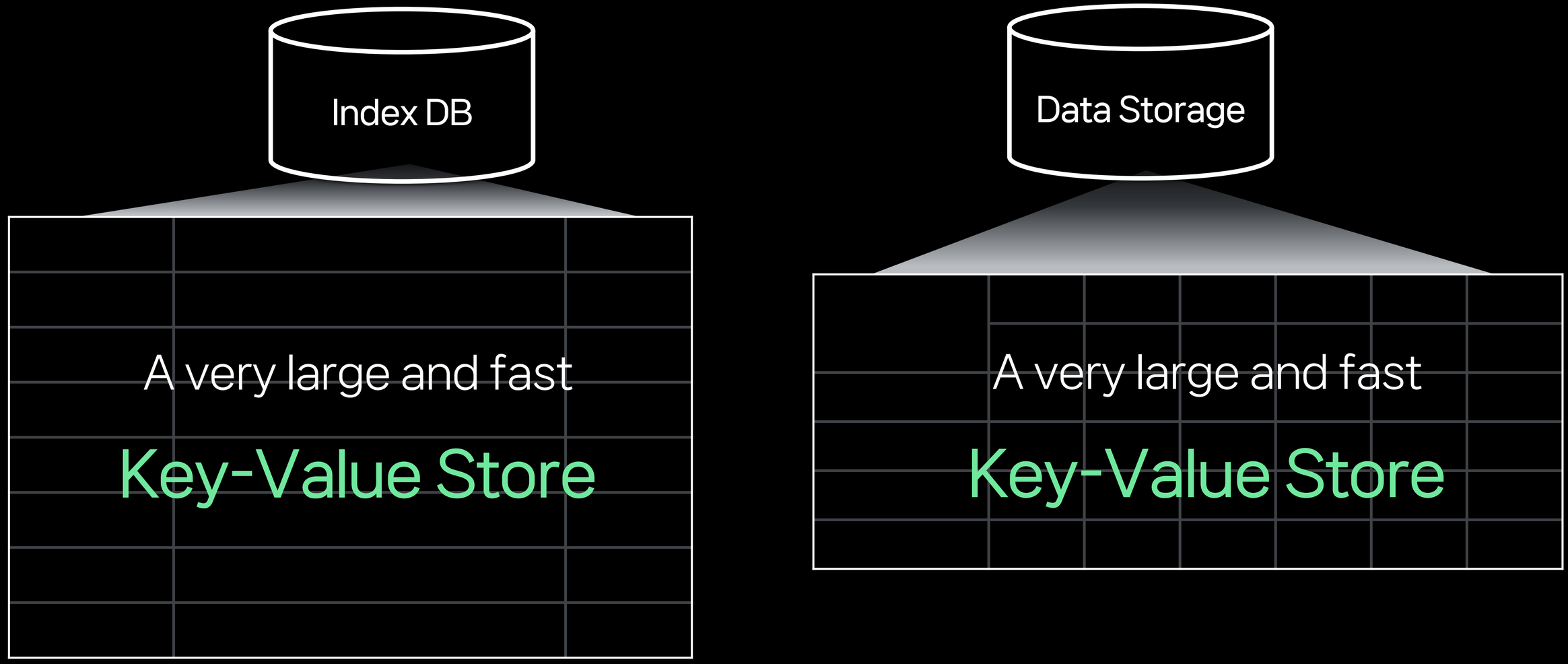
__name__	http_requests_total	tsid=1
instance	host1	tsid=1
job	my_app	tsid=1
path	/foo	tsid=1
		tsid=2
		tsid=2
		tsid=2
…	…	…

tsid=1	02:06	02:07	02:08	02:09	02:10	…
	190	170	250	160	220	…
tsid=2	...					
tsid=3						
tsid=4						
...						

Increases rapidly with each additional time series

2.1 Data Model

NAVER
DEVIEW
2023

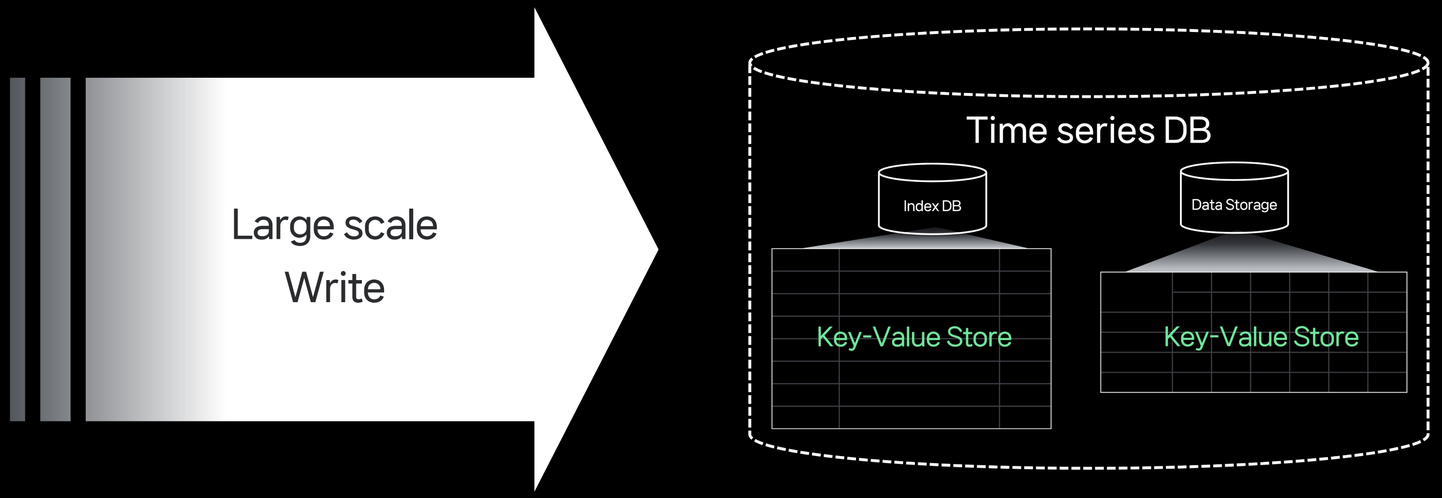


2.2 Key Requirements

NAVER
DEVIEW
2023

Large-scale Write

- Tens of millions of data flows per unit of time → Fast write performance required

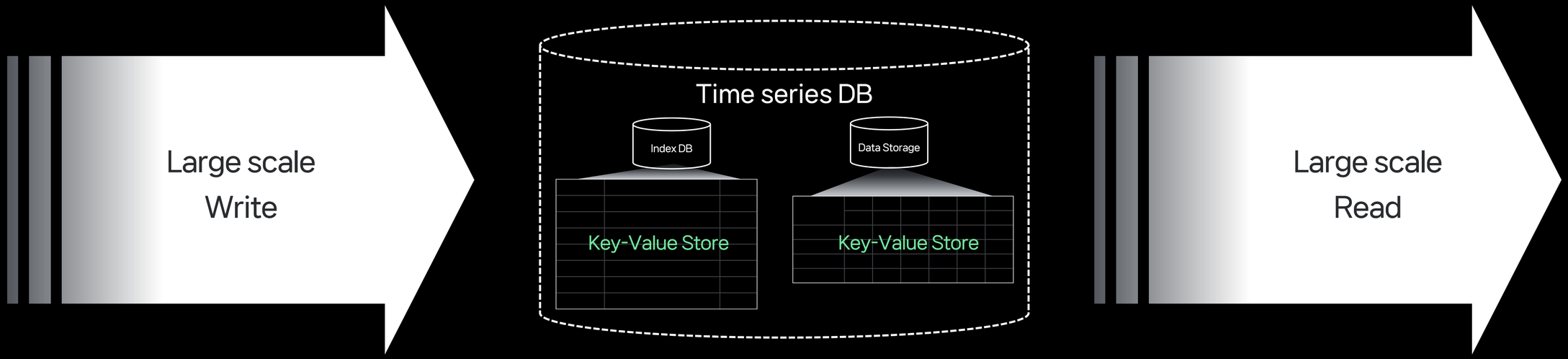


2.2 Key Requirements

NAVER
DEVIEW
2023

Large-scale Write / Read

- Tens of millions of data flows per unit of time → Fast write performance required
- Query a wide range of data to detect anomalies → Requires fast read performance



2.2 Key Requirements

NAVER
DEVIEW
2023

Write, read, both fast?

- Adding data with Append, not Edit → Write : O(1)

2.2 Key Requirements

NAVER
DEVIEW
2023

Write, read, both fast?

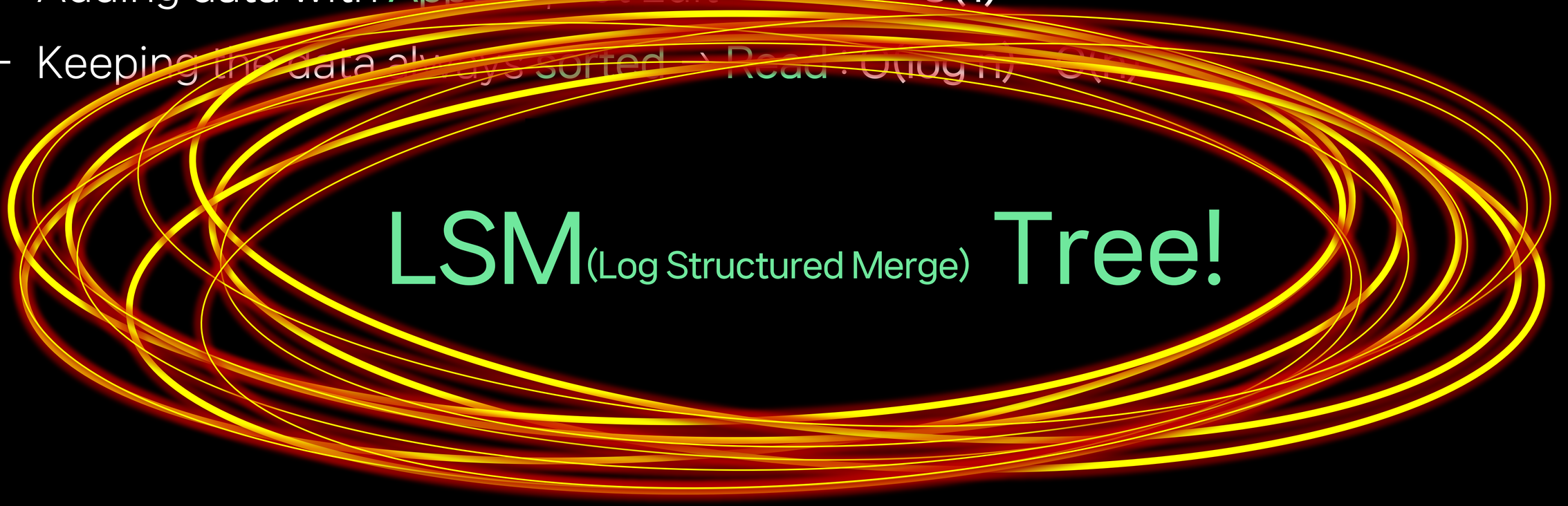
- Adding data with Append, not Edit → Write : O(1)
- Keeping the data always sorted → Read : O(log n) - O(n)

2.2 Key Requirements

NAVER
DEVIEW
2023

Write, read, both fast?

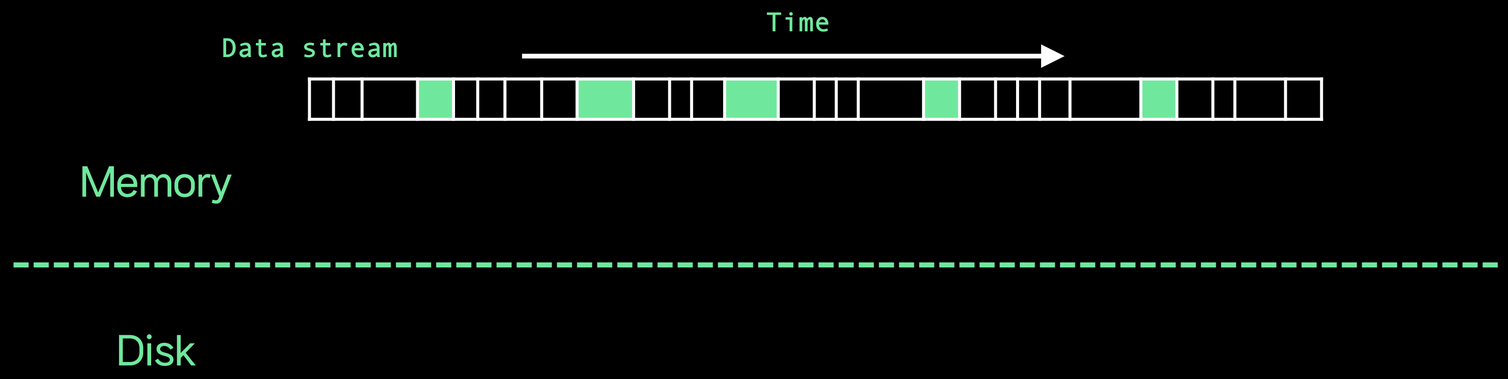
- Adding data with Append, not Edit → Write : O(1)
- Keeping the data always sorted → Read : O(log n) ~ O(1)



2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Write process



[Ref.] Log Structured Merge Trees

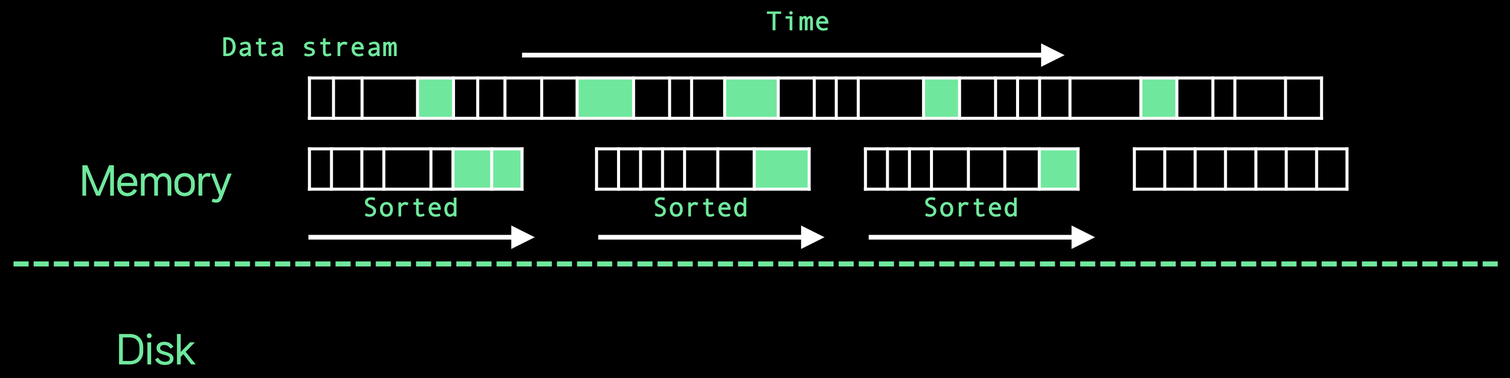
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Write process

- Data sorted in memory into small pieces



[Ref.] Log Structured Merge Trees

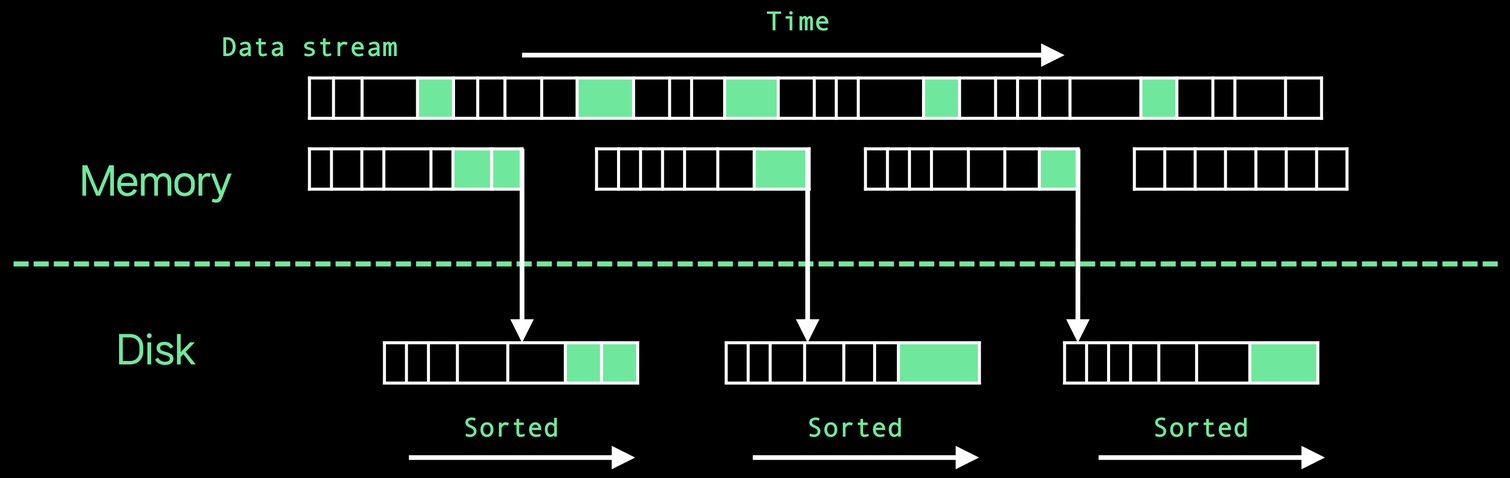
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Write process

- Data sorted in memory into small pieces
- Sorted data is periodically saved to a file (Flush)



[Ref.] Log Structured Merge Trees

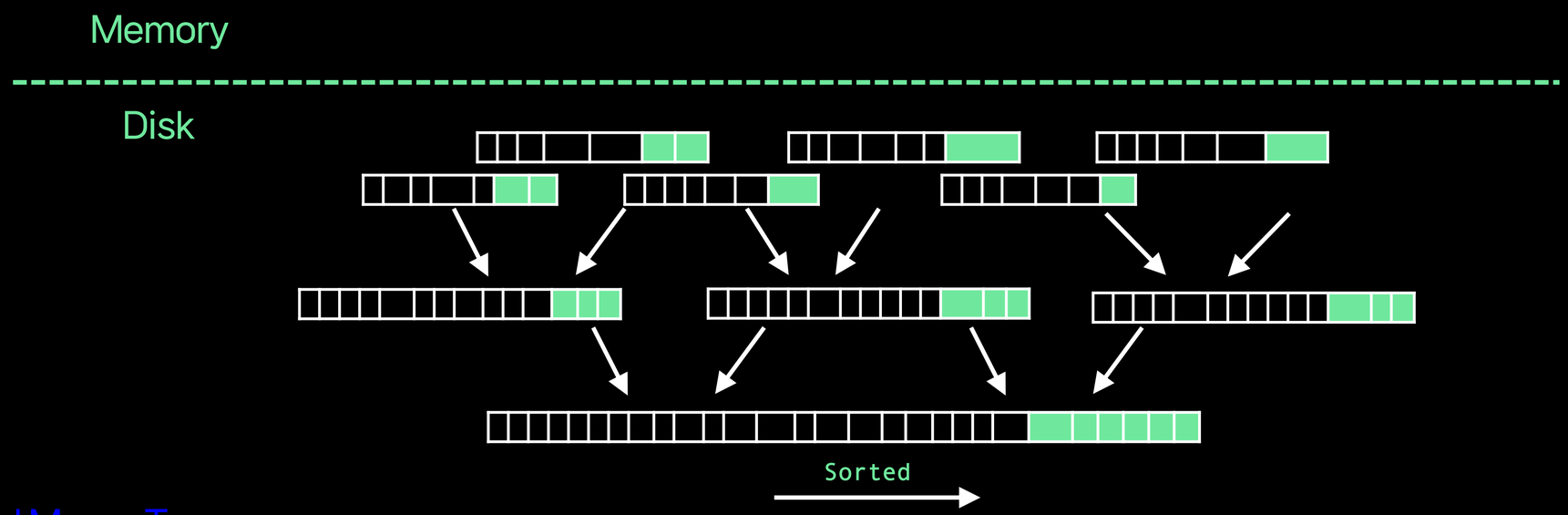
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Merge process

- Smaller fragmented files are periodically merged



[Ref.] Log Structured Merge Trees

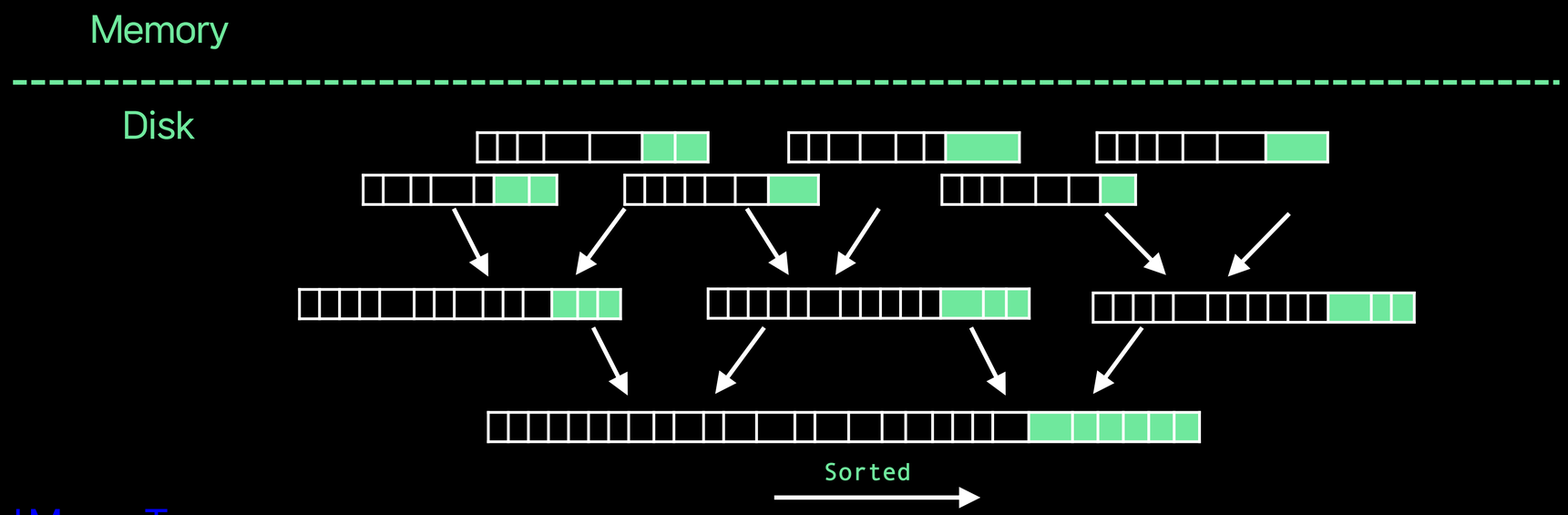
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Merge process

- Smaller fragmented files are periodically merged
- Already sorted data, so processed fast while keeping it immutable



[Ref.] Log Structured Merge Trees

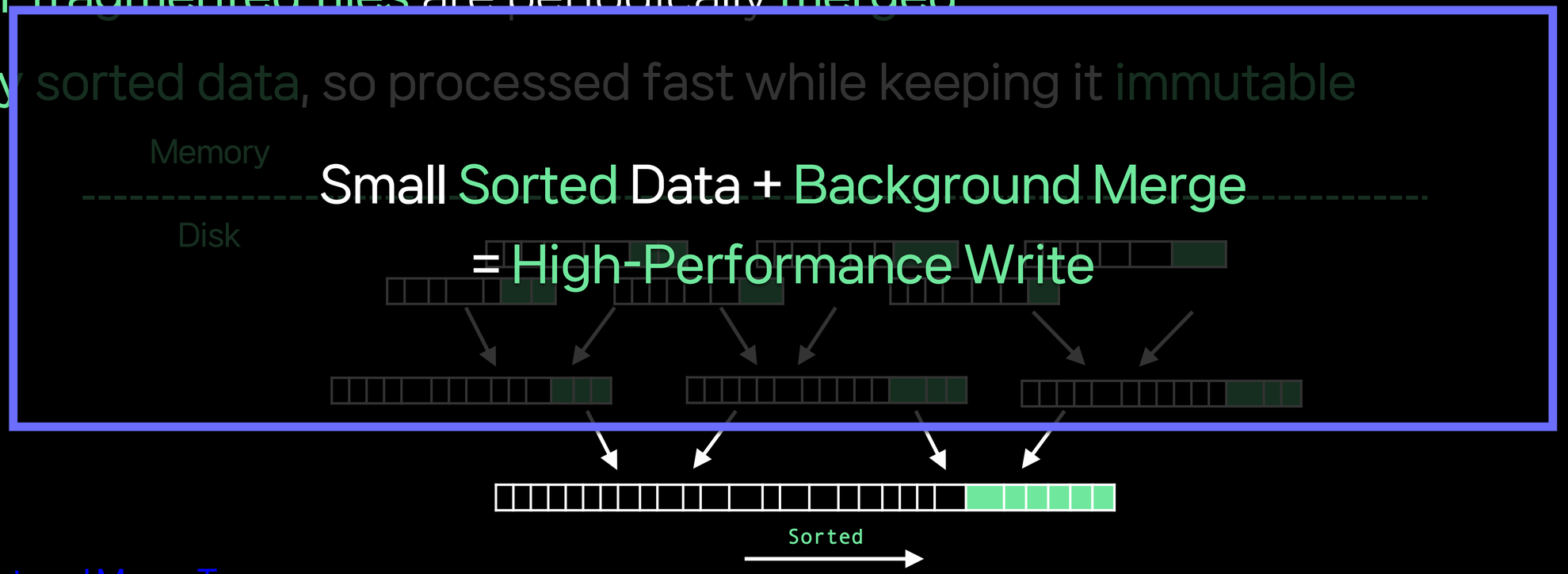
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Merge process

- Smaller fragmented files are periodically merged
- Already sorted data, so processed fast while keeping it immutable



[Ref.] Log Structured Merge Trees

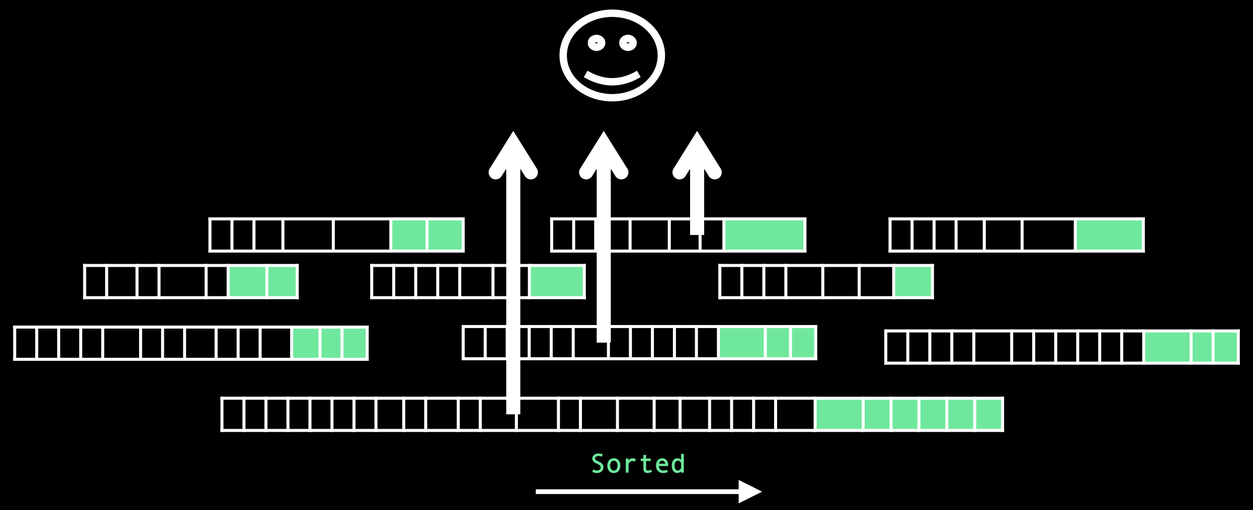
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Read process

- Quickly look up data with binary search



[Ref.] Log Structured Merge Trees

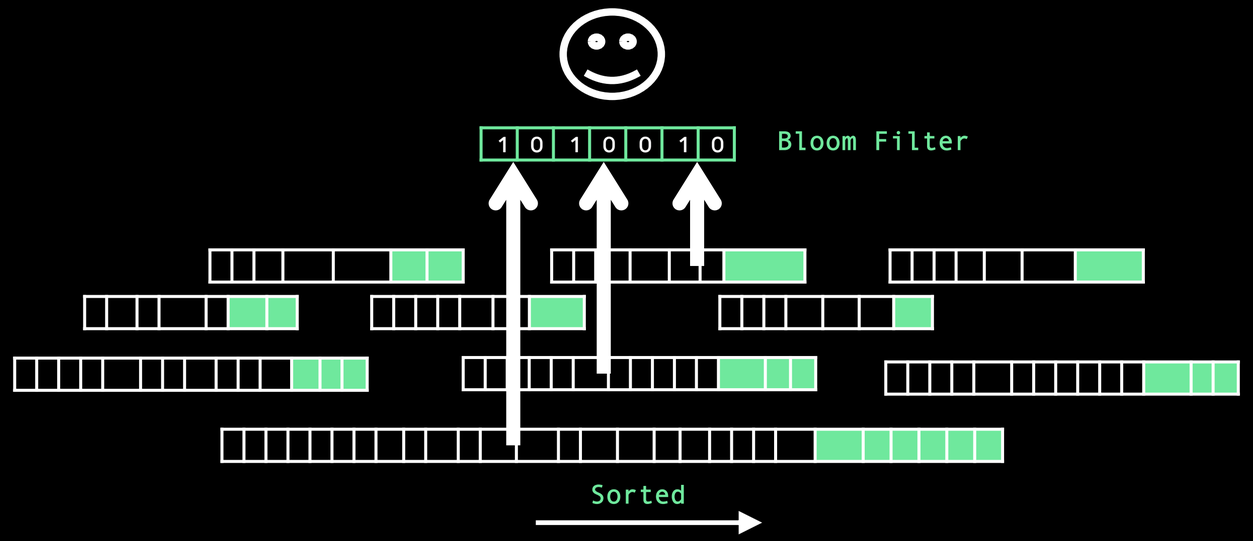
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Read process

- Quickly look up data with binary search
- The disadvantage of looking up in multiple files is mitigated by the Bloom Filter



[Ref.] Log Structured Merge Trees

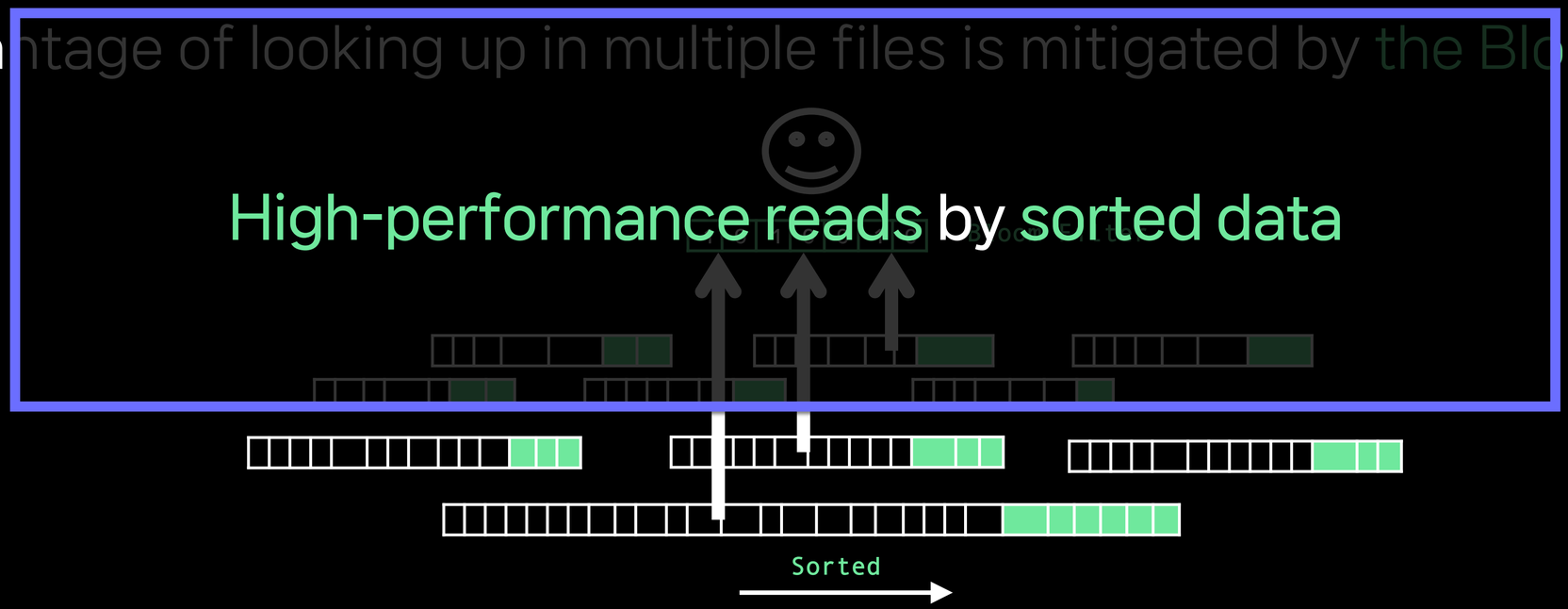
[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023

Read process

- Quickly look up data with binary search
- The disadvantage of looking up in multiple files is mitigated by the Bloom Filter

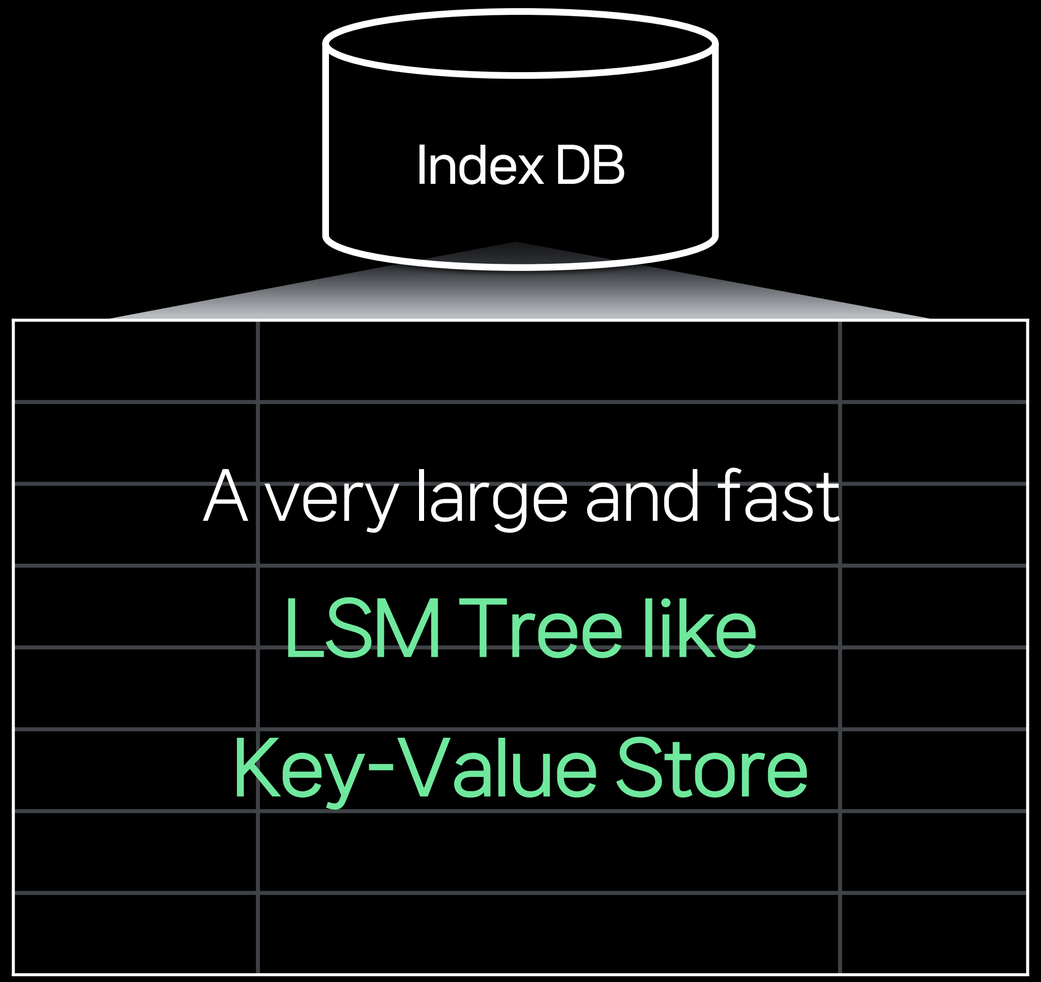


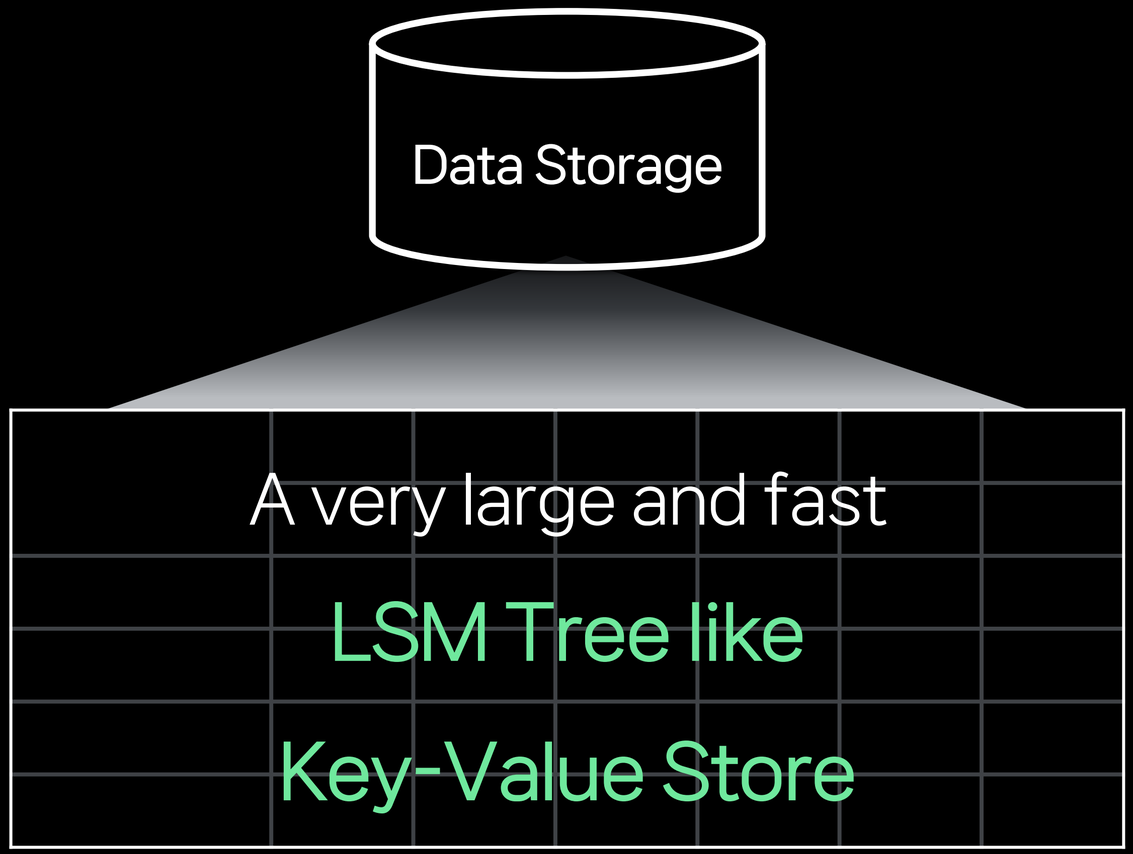
[Ref.] Log Structured Merge Trees

[Ref.] Scaling Write-Intensive Key-Value Stores

2.3 LSM(Log Structured Merge) Tree

NAVER
DEVIEW
2023





2.4 Index DB

NAVER
DEVIEW
2023

Index DB Operation Process

2.4 Index DB

NAVER
DEVIEW
2023

Write Request

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

2.4 Index DB

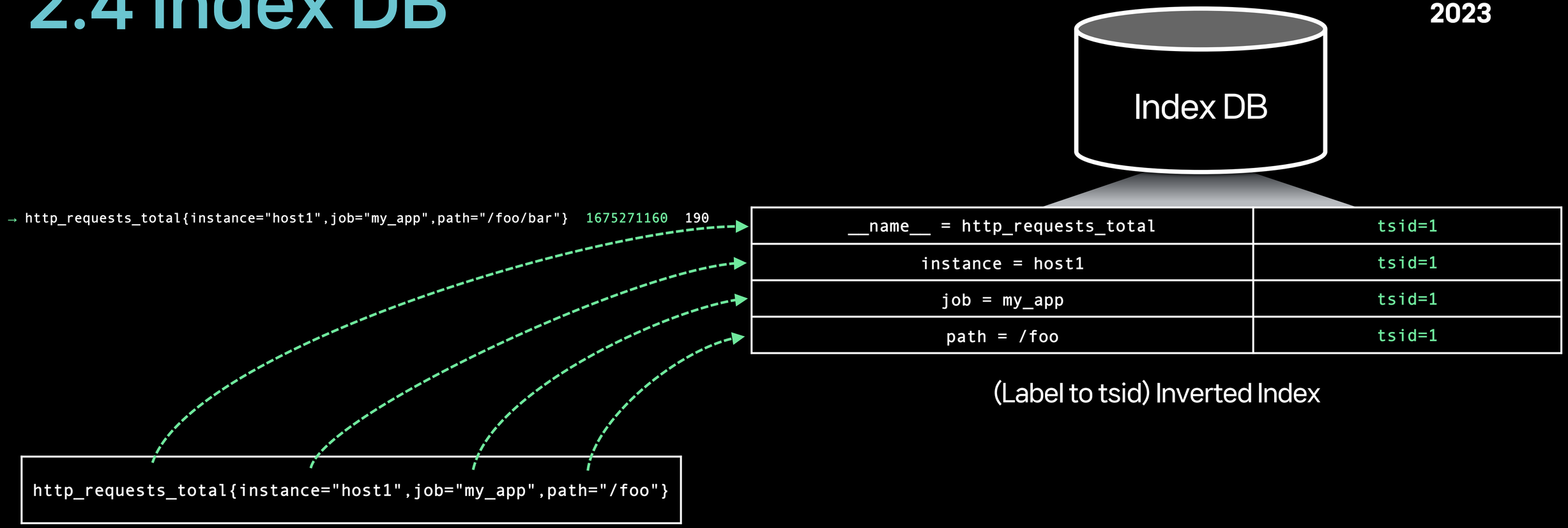
NAVER
DEVIEW
2023


```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

Time series name with labels

2.4 Index DB

NAVER
DEVIEW
2023



2.4 Index DB

NAVER DEVIEW 2023

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

http_requests_total{instance="host1",job="my_app",path="/foo"}

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

Per-day Index

2.4 Index DB

NAVER
DEVIEW
2023


```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

`http_requests_total{instance="host1",job="my_app",path="/foo"}`

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

Metric Name Index for fast look up

2.4 Index DB

NAVER DEVIEW 2023

→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"} 1675271160 190

http_requests_total{instance="host1",job="my_app",path="/foo"}

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

Inverted Index

tsid=1	http_requests_total{instance="host1",job="my_app",path="/foo/bar"}

2.4 Index DB

NAVER
DEVIEW
2023

→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"} 1675271160 190

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1
2023-02-28	tsid=1
2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

Pretty procedural

Slow Insert, only performed on first write

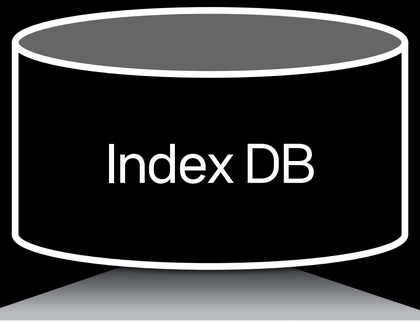
http_requests_total{instance="host1",job="my_app",path="/foo"}

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

tsid=1	http_requests_total{instance="host1",job="my_app",path="/foo/bar"}

2.4 Index DB

NAVER
DEVIEW
2023




```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271340  160
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271400  220
```

More data into the same time series

http_requests_total{instance="host1",job="my_app",path="/foo"}

__name__ = http_requests_total	tsid=1
instance = host1	tsid=1
job = my_app	tsid=1
path = /foo	tsid=1

2023-02-28	tsid=1

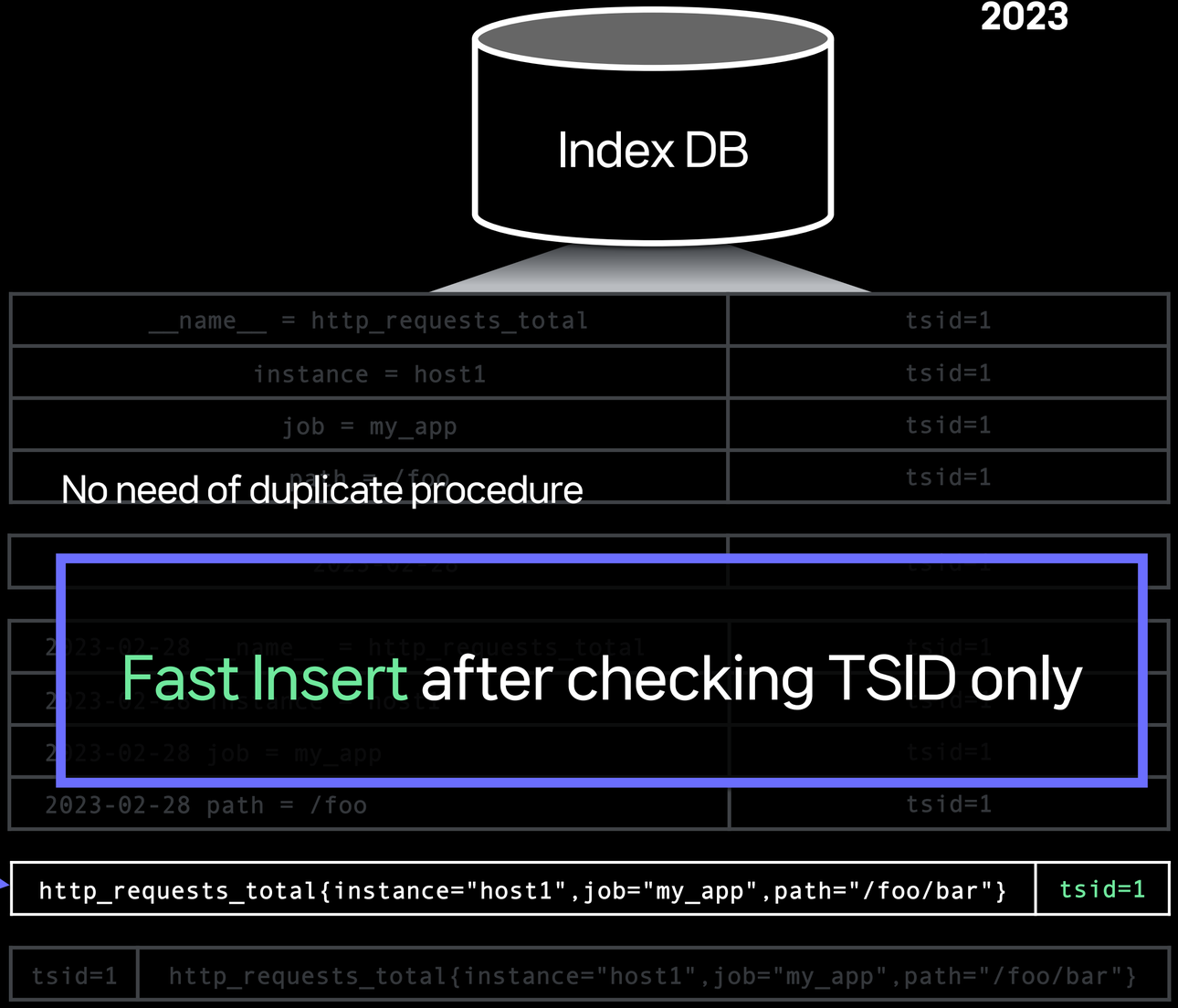
2023-02-28 __name__ = http_requests_total	tsid=1
2023-02-28 instance = host1	tsid=1
2023-02-28 job = my_app	tsid=1
2023-02-28 path = /foo	tsid=1

http_requests_total{instance="host1",job="my_app",path="/foo/bar"}	tsid=1

tsid=1	http_requests_total{instance="host1",job="my_app",path="/foo/bar"}

2.4 Index DB

NAVER
DEVIEW
2023



2.5 Data Storage

NAVER
DEVIEW
2023

Data Storage Operation Process

2.5 Data Storage

NAVER
DEVIEW
2023

(Write Request)

Metric Name	UNIX Timestamp	Value
→ `http_requests_total{instance="host1",job="my_app",path="/foo/bar"}`	`1675271160`	`190`

2.5 Data Storage

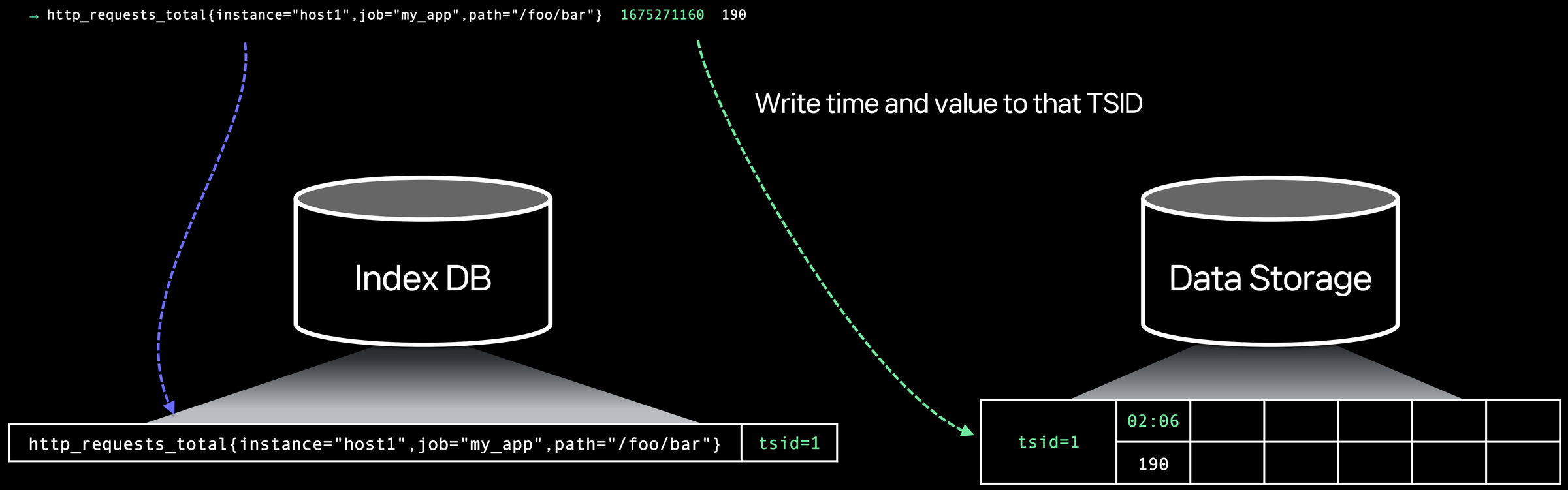
NAVER
DEVIEW
2023

```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
```

Check the tsid assigned to that time series in the Index DB
(issue if not present)

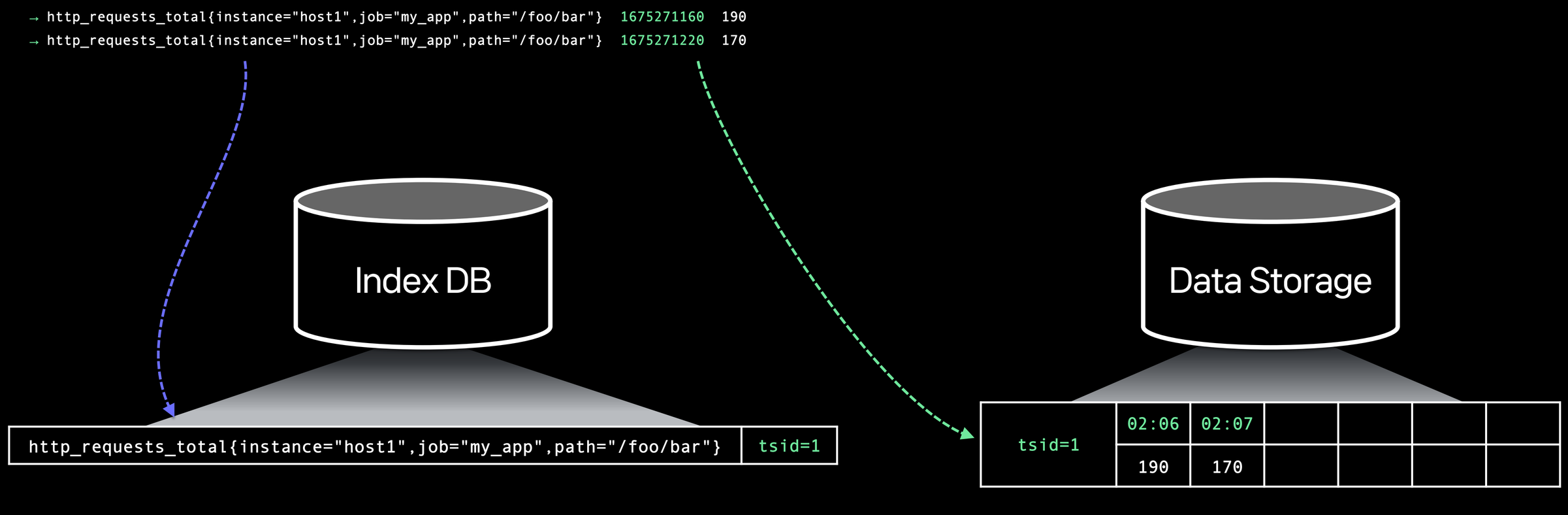
2.5 Data Storage

NAVER
DEVIEW
2023



2.5 Data Storage

NAVER
DEVIEW
2023

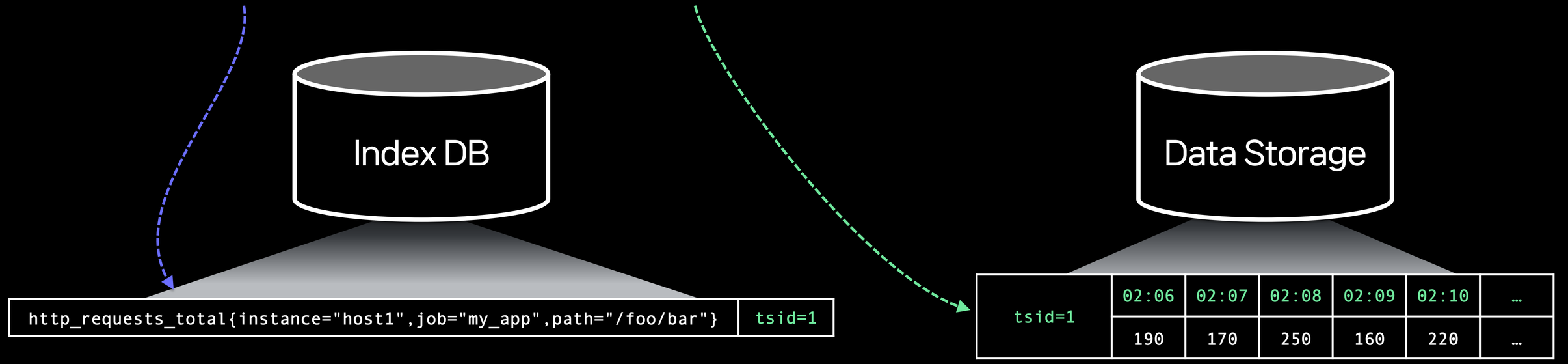


Time series data is stored in Data Storage

2.5 Data Storage

NAVER
DEVIEW
2023

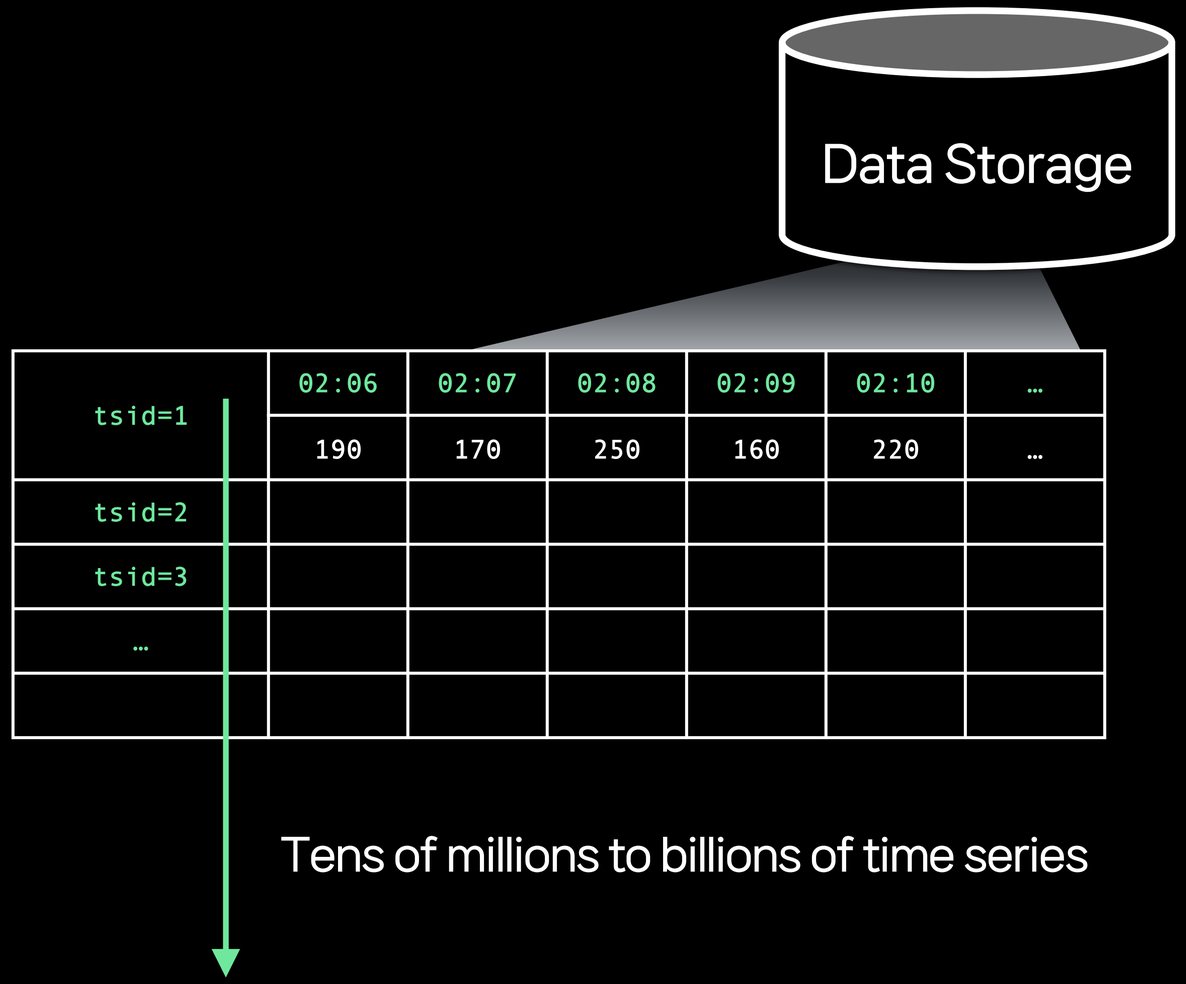
```
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271160  190
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271220  170
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271280  250
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271340  160
→ http_requests_total{instance="host1",job="my_app",path="/foo/bar"}  1675271400  220
```



Time series data is stored in Data Storage

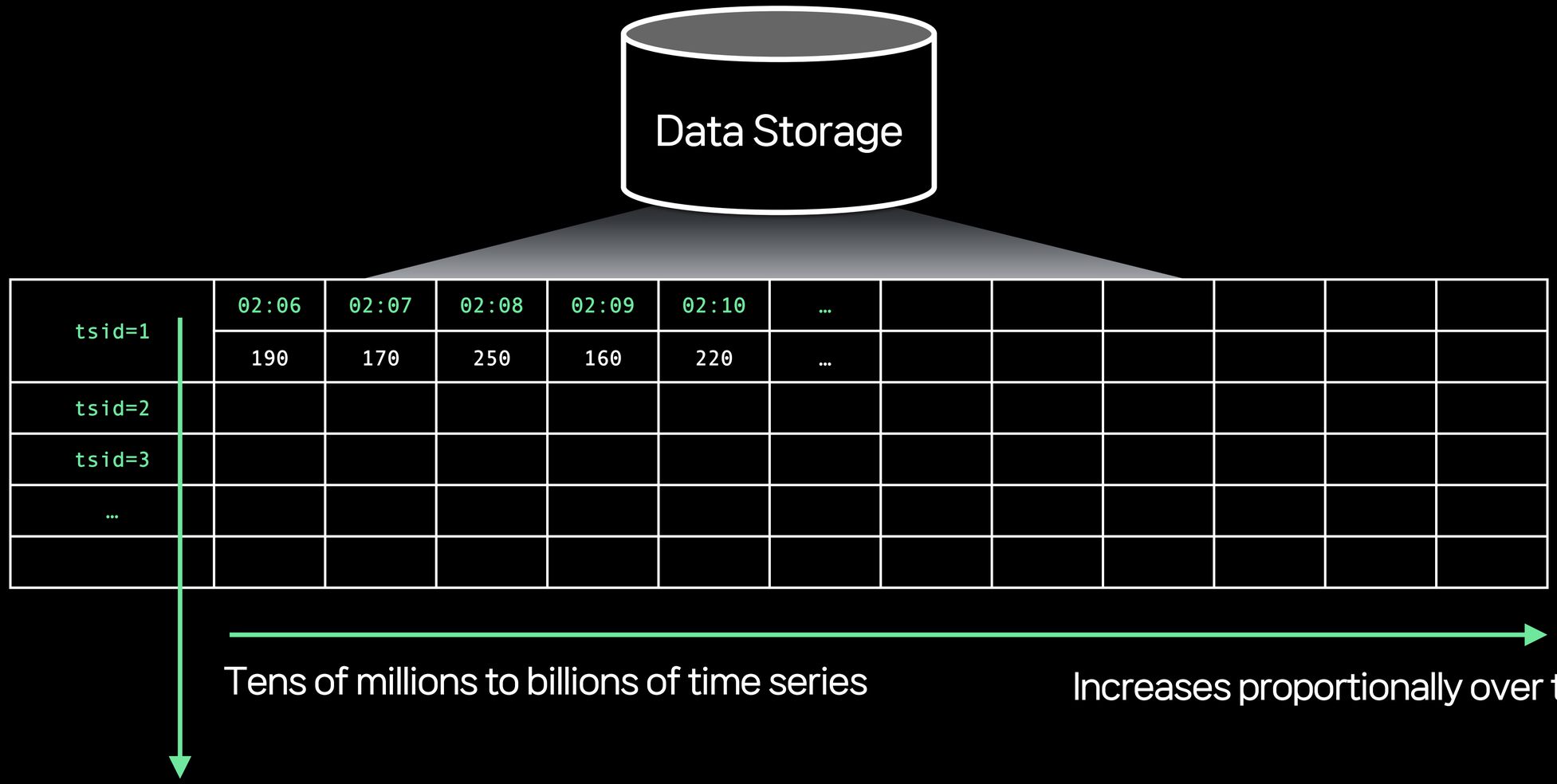
2.5 Data Storage

NAVER
DEVIEW
2023



2.5 Data Storage

NAVER
DEVIEW
2023



2.5 Data Storage

NAVER
DEVIEW
2023

Data volumes are growing at an incredibly rapid rate

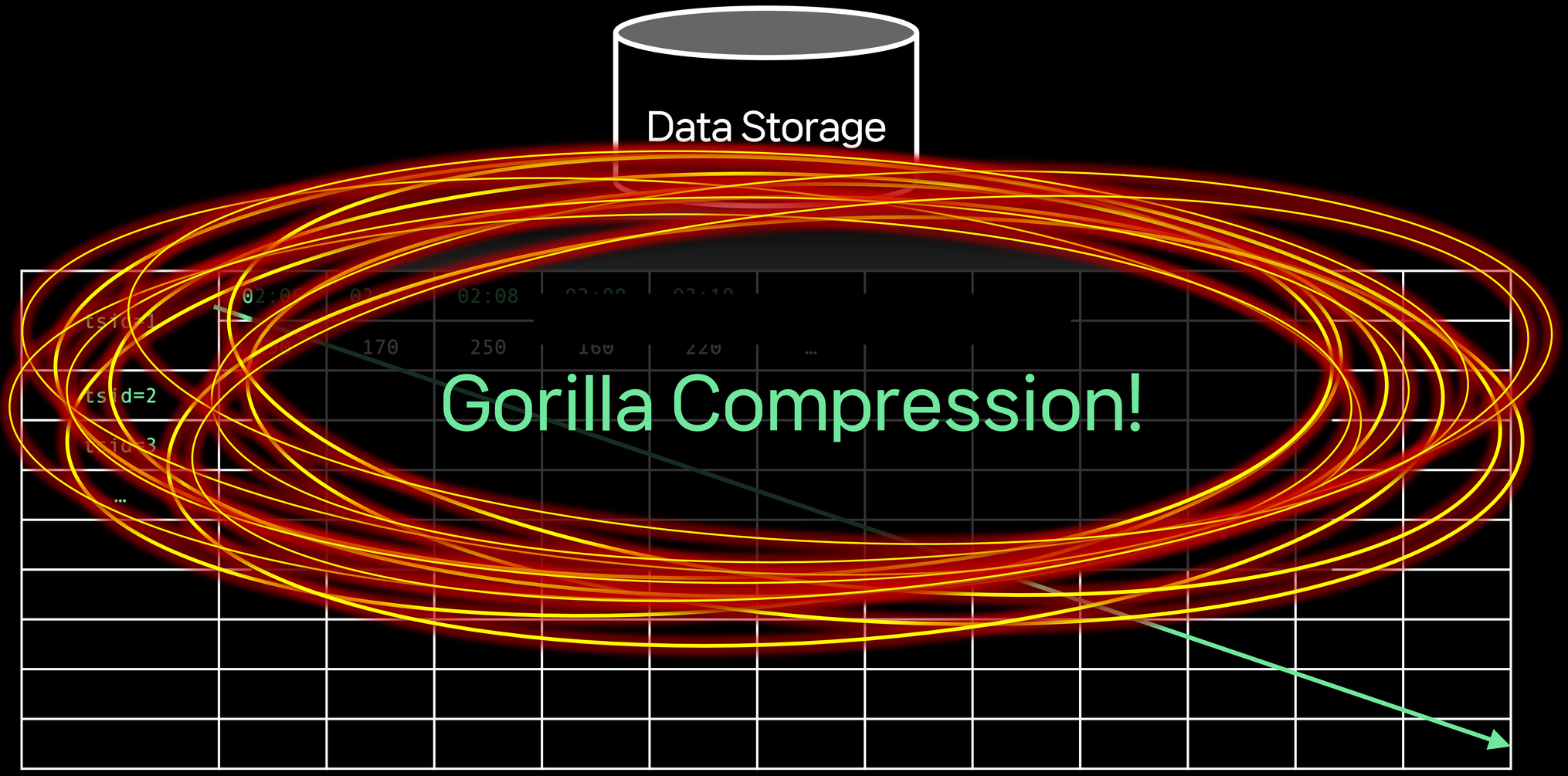
2.5 Data Storage

NAVER
DEVIEW
2023

Data volumes are growing at an incredibly rapid rate

2.5 Data Storage

NAVER
DEVIEW
2023



Data volumes are growing at an incredibly rapid rate

2.6 Gorilla Compression

NAVER
DEVIEW
2023

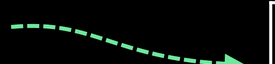
(Write Requests)

UNIX Timestamp	Value
1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220



Single Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

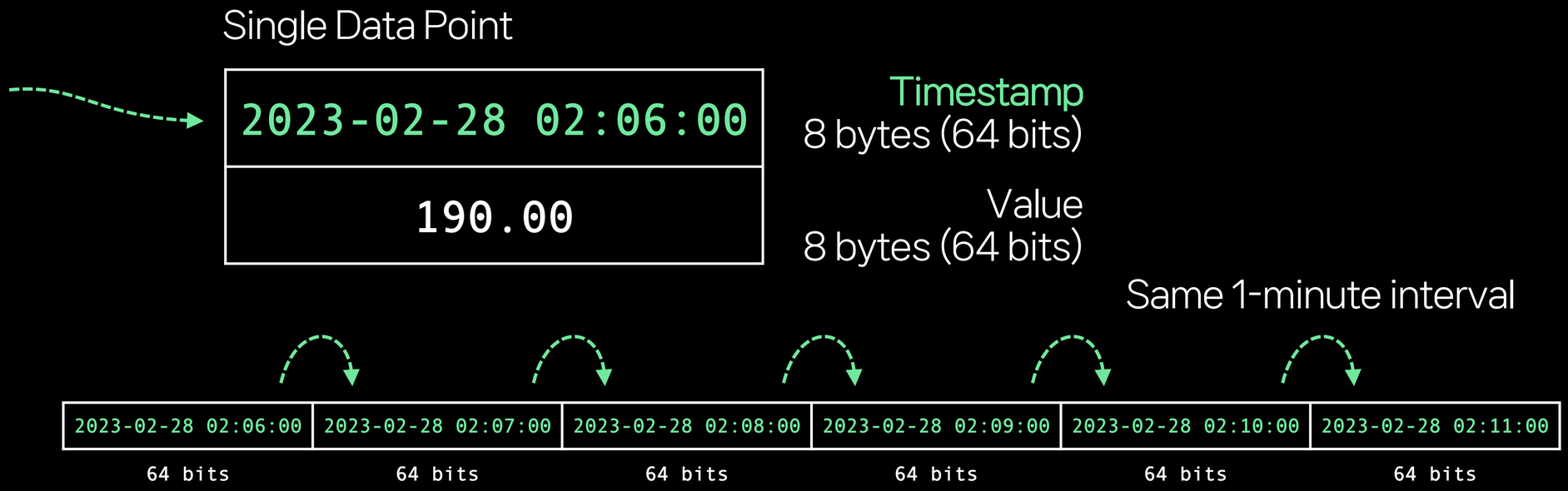
Value
8 bytes (64 bits)

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220



https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER DEVIEW 2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

Almost duplicate values

No need to store all of them

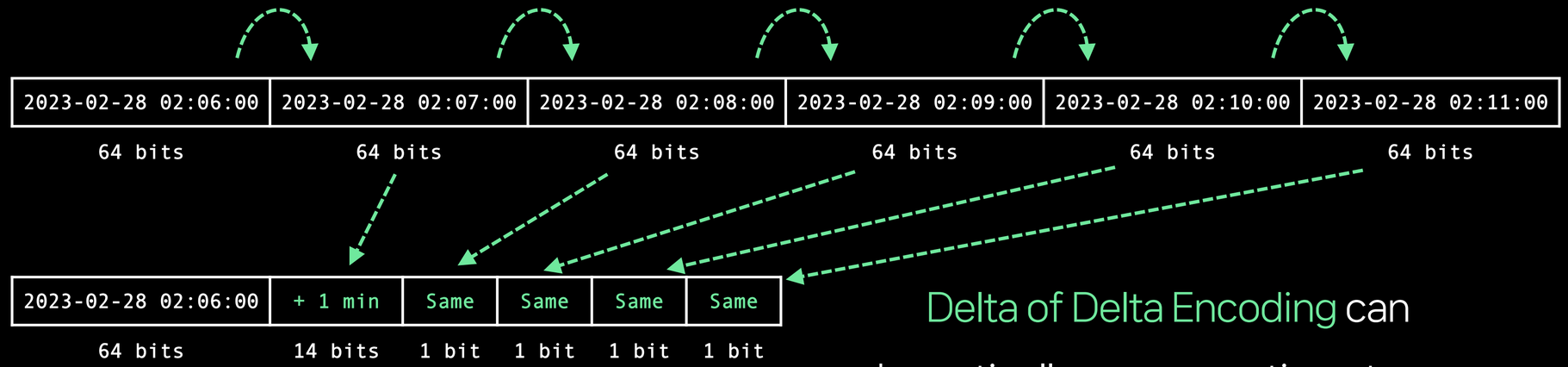
Single Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Same 1-minute interval



Delta of Delta Encoding can dramatically compress timestamps

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

Single Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Previous Value	190.00	0x4067c00000000000
Current Value	170.00	0x4065400000000000

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER DEVIEW 2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

Single Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Previous Value	190.00	0x4067c00000000000
Current Value	170.00	0x4065400000000000

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER DEVIEW 2023

1675271160	190
1675271220	170
1675271280	250
1675271340	160
1675271400	220

Single Data Point

2023-02-28 02:06:00
190.00

Timestamp
8 bytes (64 bits)

Value
8 bytes (64 bits)

Previous Value	190.00	0x4067c00000000000
Current Value	170.00	0x4065400000000000
XOR	-	0x0002800000000000

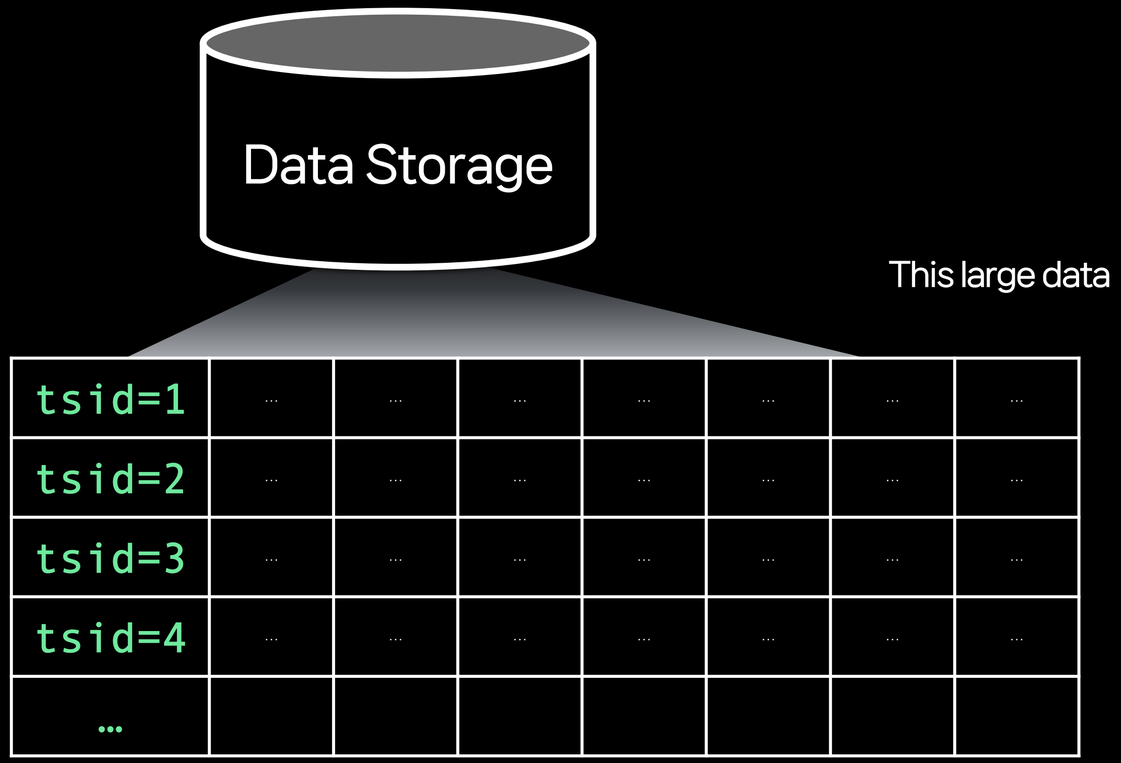
Value is also mostly redundant

XOR results `0` → Delta of Delta Encoding

https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

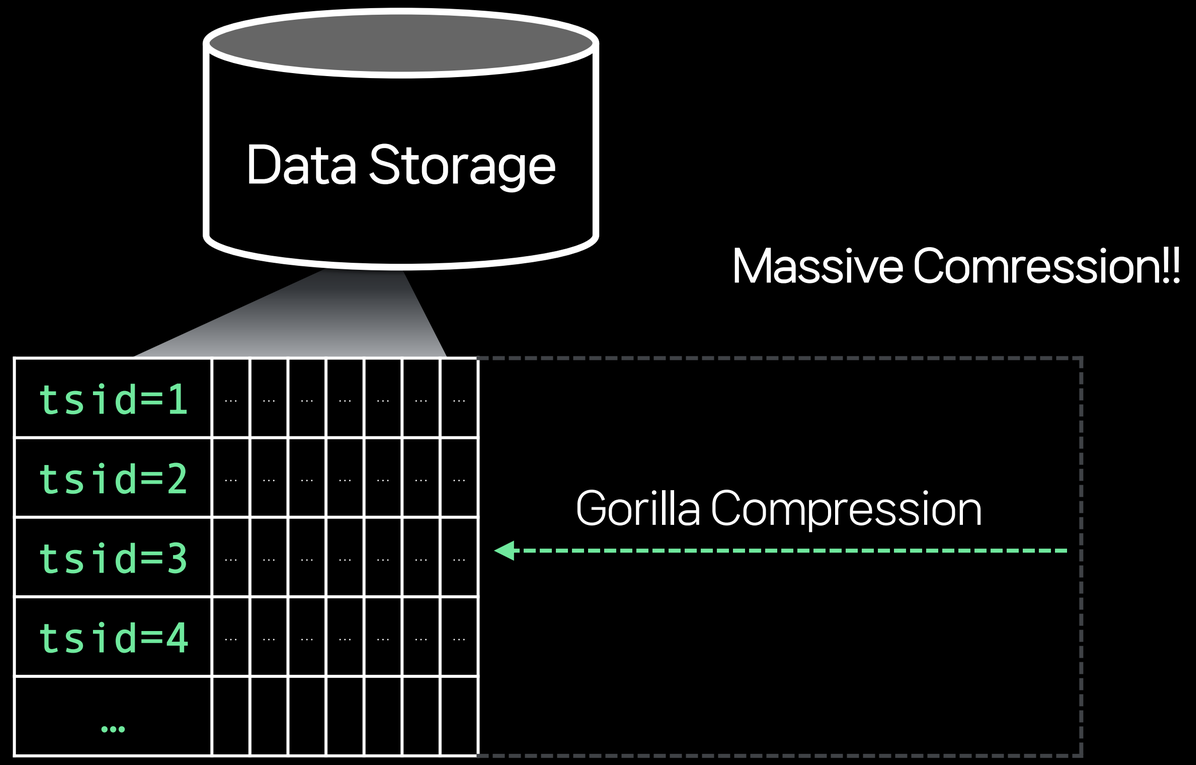
NAVER
DEVIEW
2023



https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

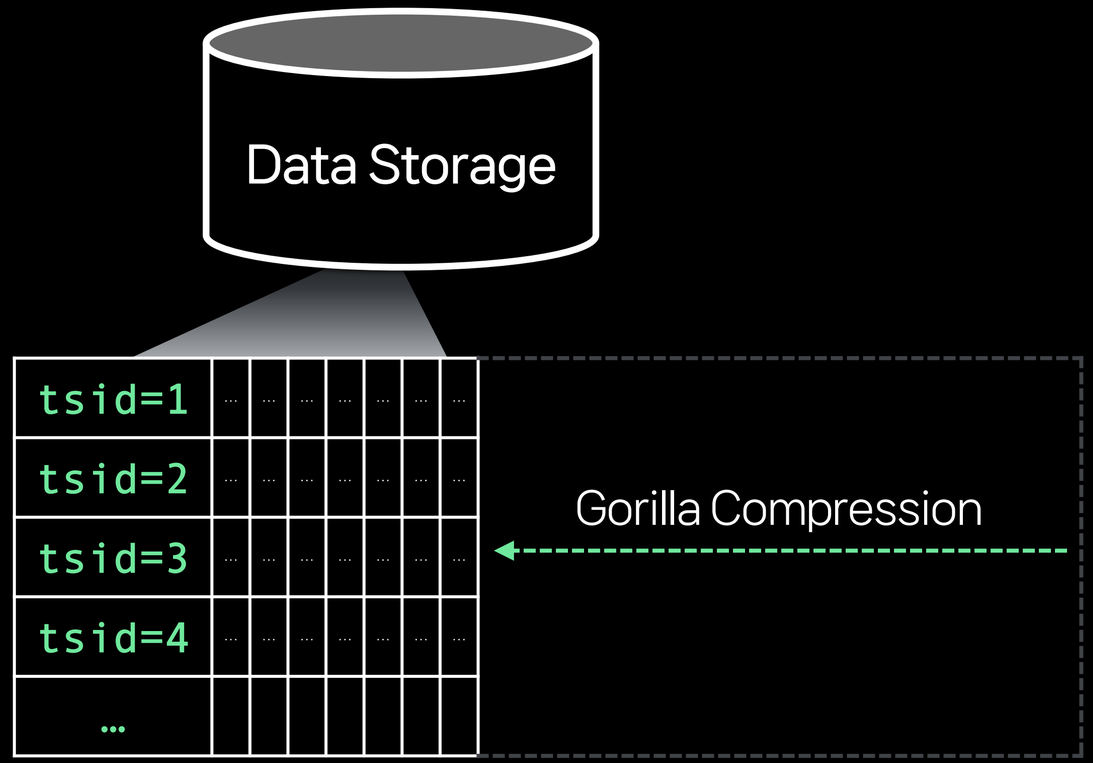
NAVER
DEVIEW
2023



https://dl.acm.org/doi/10.14778/2824032.2824078

2.6 Gorilla Compression

NAVER
DEVIEW
2023



(Actual dashboard screenshot)

Compressing 16 Bytes into 0.47 Bytes!

Much more data in memory

Faster processing of large volumes

https://dl.acm.org/doi/10.14778/2824032.2824078

2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

High Churn Rate

- Time series is constantly being created and doesn't accumulate over time.

2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

High Churn Rate

- Time series is constantly being created and doesn't accumulate over time.
- Existence of a Label whose value changes unnecessarily frequently

```
http_requests_total{instance="host1",job="my_app",path="/foo”,tag="i73w3e2h4asdf8q23jha"}
```

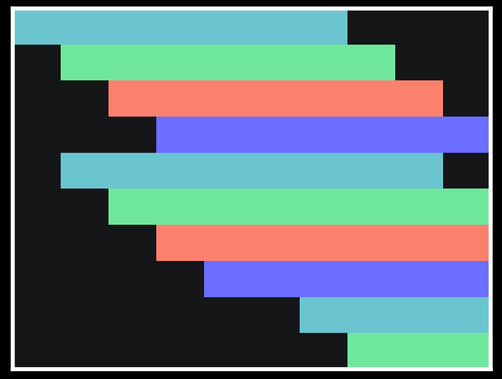
2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

High Churn Rate

- Time series is constantly being created and doesn't accumulate over time.
- Existence of a Label whose value changes unnecessarily frequently

```
http_requests_total{instance="host1",job="my_app",path="/foo”,tag="i73w3e2h4asdf8q23jha"}
```



Common time series data distributions

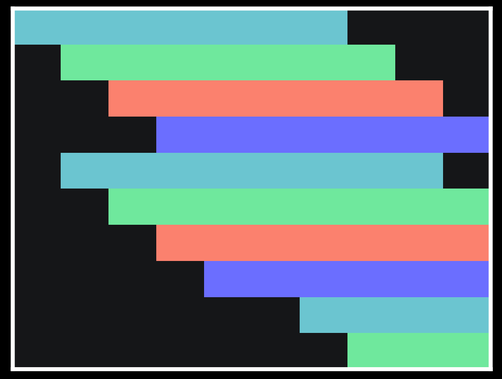
2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

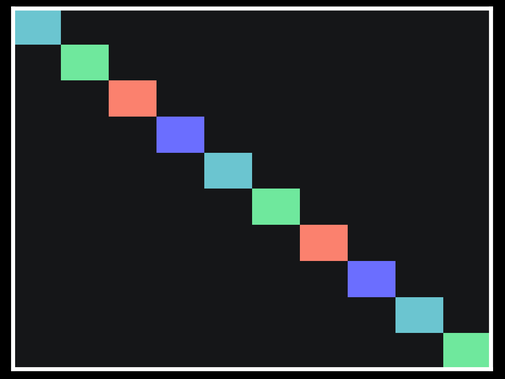
High Churn Rate

- Time series is constantly being created and doesn't accumulate over time.
- Existence of a Label whose value changes unnecessarily frequently

```
http_requests_total{instance="host1",job="my_app",path="/foo”,tag="i73w3e2h4asdf8q23jha"}
```



Common time series data distributions



Time series with high churn rate

2.7 Importance of Churn Rate

NAVER
DEVIEW
2023

High Churn Rate

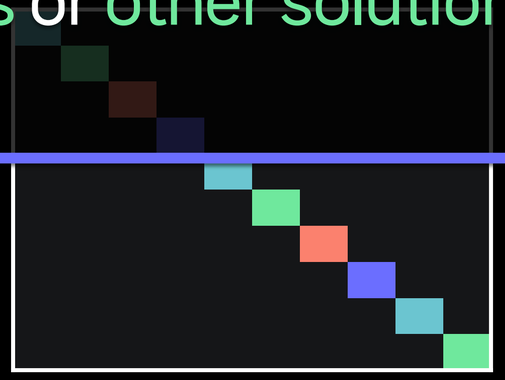
- Time series is constantly being created and doesn't accumulate over time.
- Existence of a Label whose value changes unnecessarily frequently

```
http_requests_total{instance="host:port", path="/foo", tag="i73w3e2h4asdfq23jha"}
```

Not suitable for Time Series DB

Need to tune labels or other solution

Common time series data distributions



Time series with high churn rate

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage
- LSM Tree speeds up both reads and writes

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage
- LSM Tree speeds up both reads and writes
- Gorilla Compression for better compression & increased memory efficiency

2. A Deep-dive into Time series DB

NAVER
DEVIEW
2023

Summary

- Time series DB = Index DB + Data Storage
- LSM Tree speeds up both reads and writes
- Gorilla Compression for better compression & increased memory efficiency
- It's important to avoid high churn rate situations

3. Time series in the Multiverse of Madness

3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- One binary file with no external dependencies

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*

~/VictoriaMetrics/single ✔ at 10:22:04
```

3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- One binary file with no external dependencies
- 2-10x faster than Prometheus

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*

 ~/VictoriaMetrics/single ···· ✔ 〈 at 10:22:04
```

3.1 Happy one universe

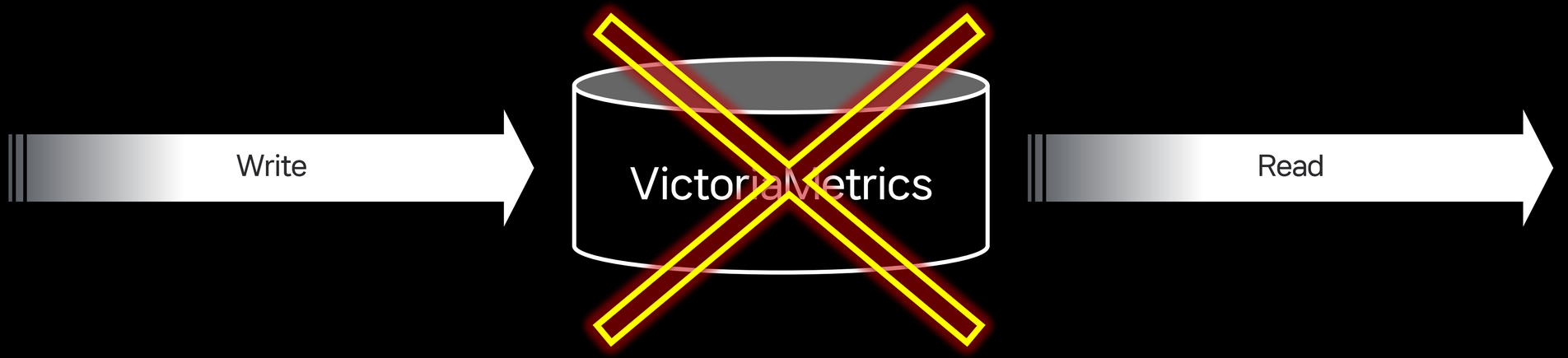
NAVER
DEVIEW
2023

Single Mode / Cluster Mode

- One binary file with no external dependencies
- 2-10x faster than Prometheus
- Disadvantages: Single Point of Failure

```
❯ ls -alFh
total 34912
drwxr-xr-x  3 user  staff    96B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    17M Feb  2 07:03 victoria-metrics-prod*

~/VictoriaMetrics/single ✔ at 10:22:04
```

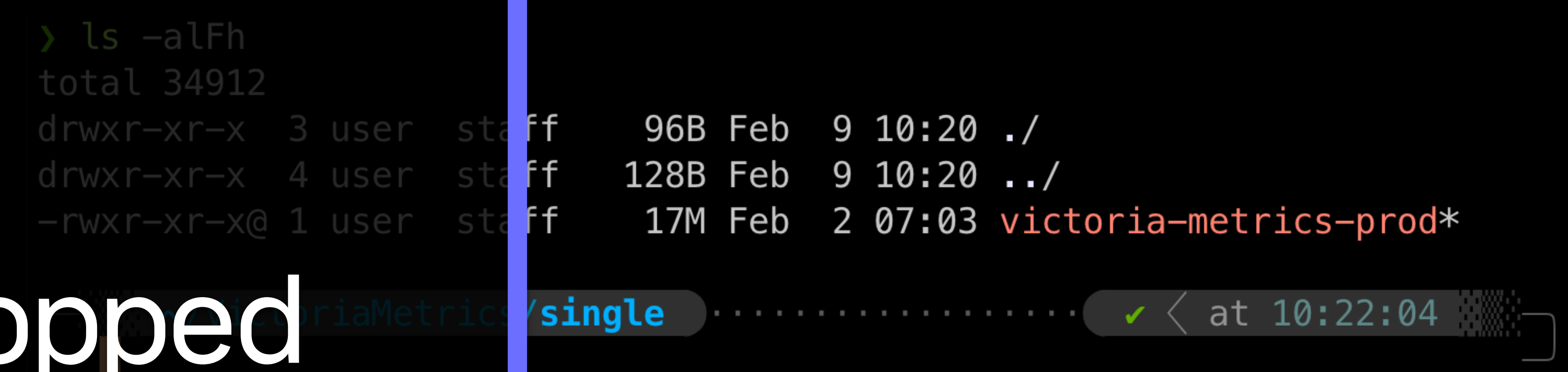


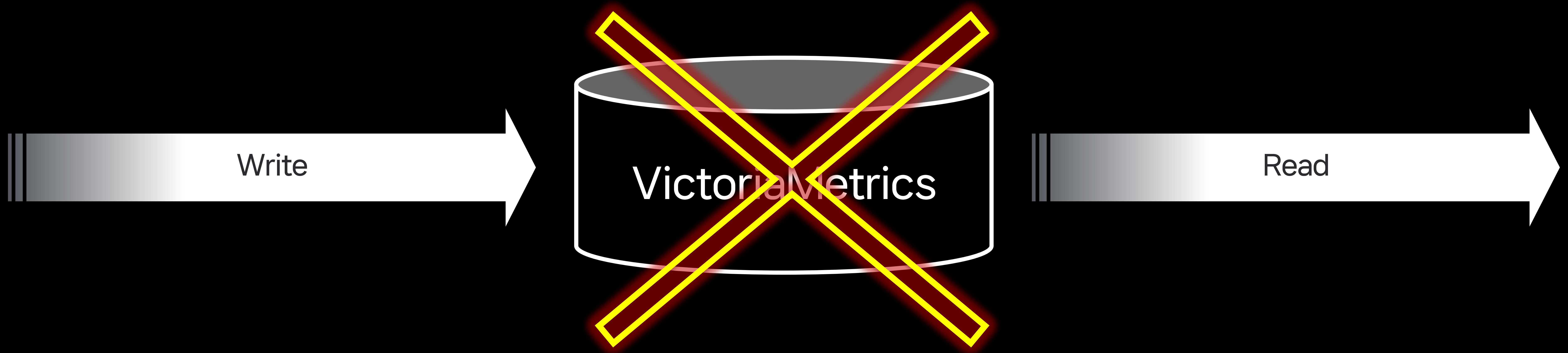
3.1 Happy one universe

Single Mode / Cluster Mode

- One binary file with no external dependencies
- 2-10x faster than Prometheus
- Disadvantages: Single Point of Failure

Monitoring for hundreds services stopped





3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / Cluster Mode

3.1 Happy one universe

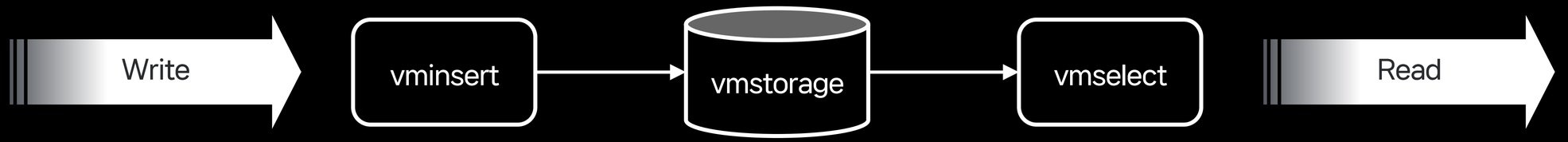
NAVER
DEVIEW
2023

Single Mode / **Cluster Mode**

- 3 binary files with no external dependencies
  (Different from Thanos, Cortex, Mimir, etc.)

```
❯ ls -alFh
total 70976
drwxr-xr-x  5 user  staff   160B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:45 vminsert-prod*
-rwxr-xr-x@ 1 user  staff    13M Feb  2 07:46 vmselect-prod*
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:46 vmstorage-prod*

~/VictoriaMetrics/cluster ✔ at 10:23:09
```



3.1 Happy one universe

NAVER
DEVIEW
2023

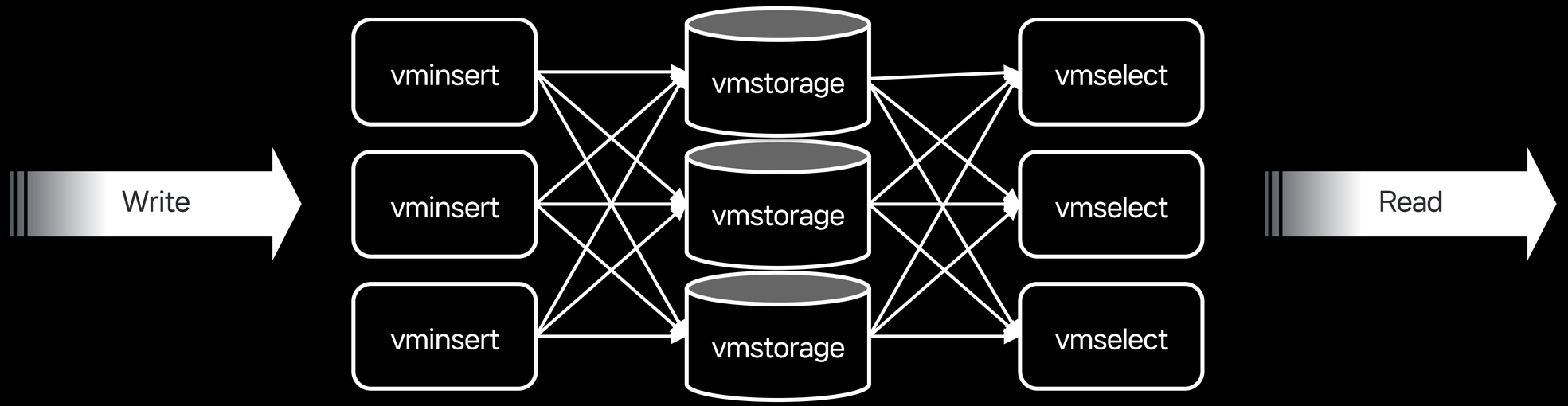
Single Mode / **Cluster Mode**

- 3 binary files with no external dependencies
- Each component is easily horizontally scalable

  (Overcoming Prometheus' biggest limitations)

```
❯ ls -alFh
total 70976
drwxr-xr-x  5 user  staff   160B Feb  9 10:20 ./
drwxr-xr-x  4 user  staff   128B Feb  9 10:20 ../
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:45 vminsert-prod*
-rwxr-xr-x@ 1 user  staff    13M Feb  2 07:46 vmselect-prod*
-rwxr-xr-x@ 1 user  staff    11M Feb  2 07:46 vmstorage-prod*

~/VictoriaMetrics/cluster ···· ✔ 〈 at 10:23:09
```



3.1 Happy one universe

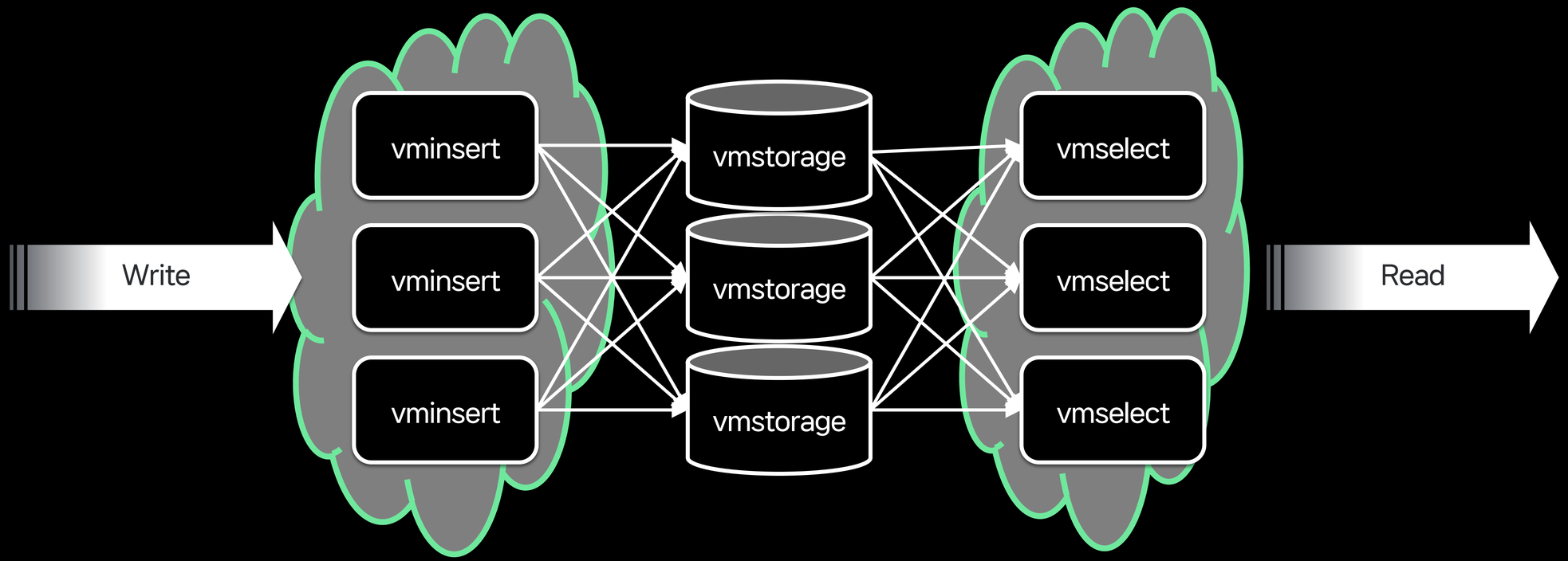
NAVER
DEVIEW
2023

Single Mode / **Cluster Mode**

- 3 binary files with no external dependencies
- Each component is easily horizontally scalable

- Stateful / Stateless Servers

(Physical Machine & Kubernetes)

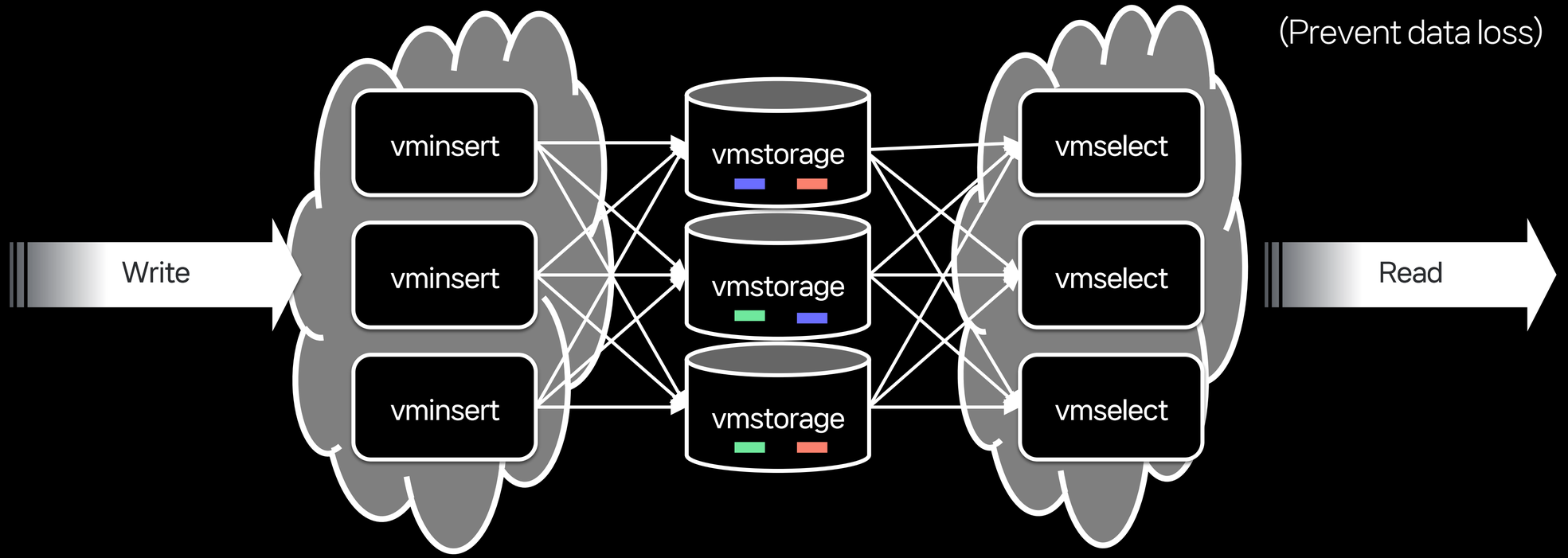


3.1 Happy one universe

NAVER
DEVIEW
2023

Single Mode / **Cluster Mode**

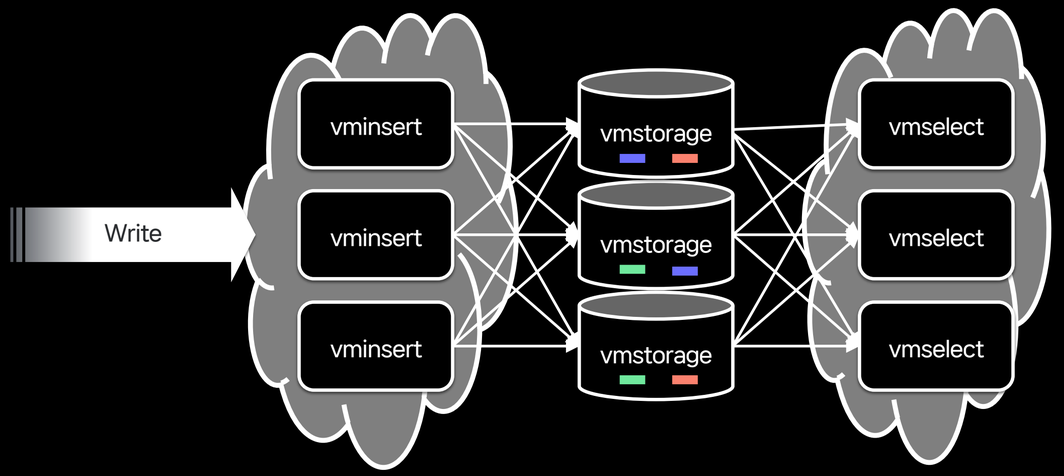
- 3 binary files with no external dependencies
- Each component is easily horizontally scalable
- Stateful / Stateless Servers
- ReplicationFactor
  (Prevent data loss)



3.1 Happy one universe

NAVER
DEVIEW
2023

Happy Moments...



Read

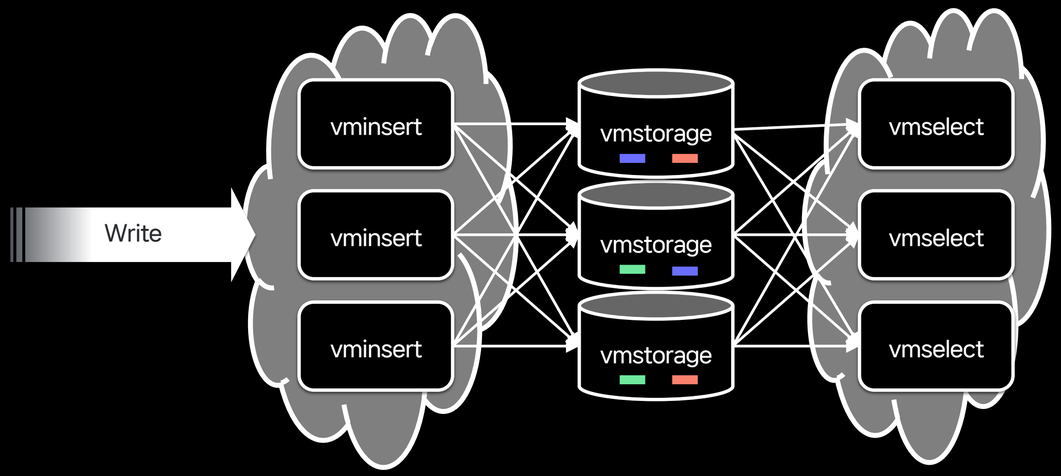
Service-level Monitoring Dashboard

Useful in dashboards

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Happy **Moments...**



Read

Service-level Monitoring Dashboard

Read

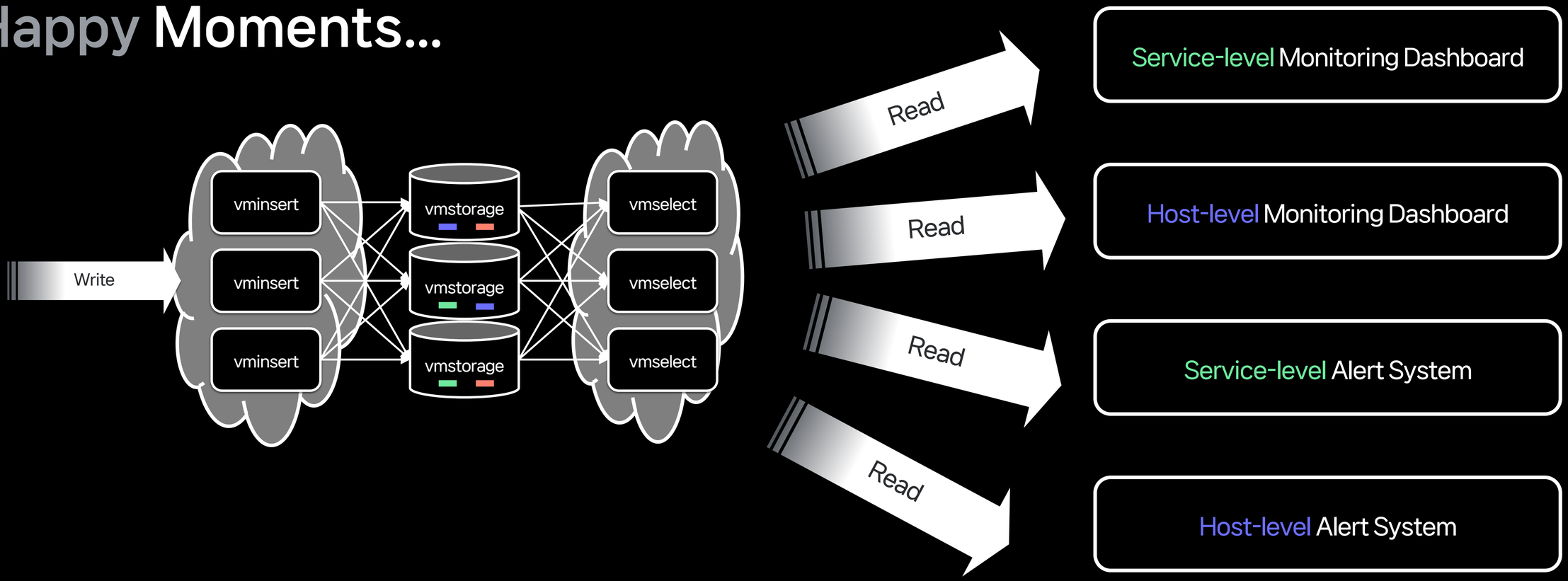
Service-level Alert System

Useful for alert systems

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Happy **Moments...**

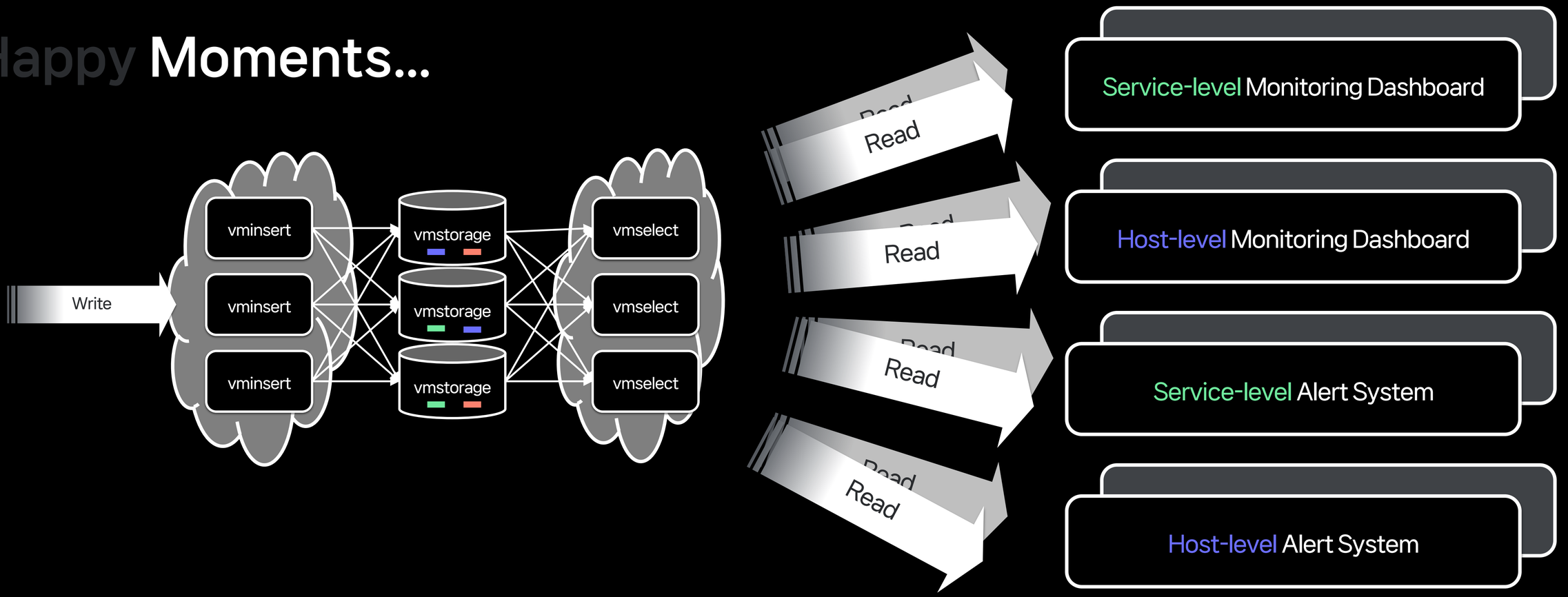


Other types of dashboards and alert systems

3.2 The Rise of The Multiverse

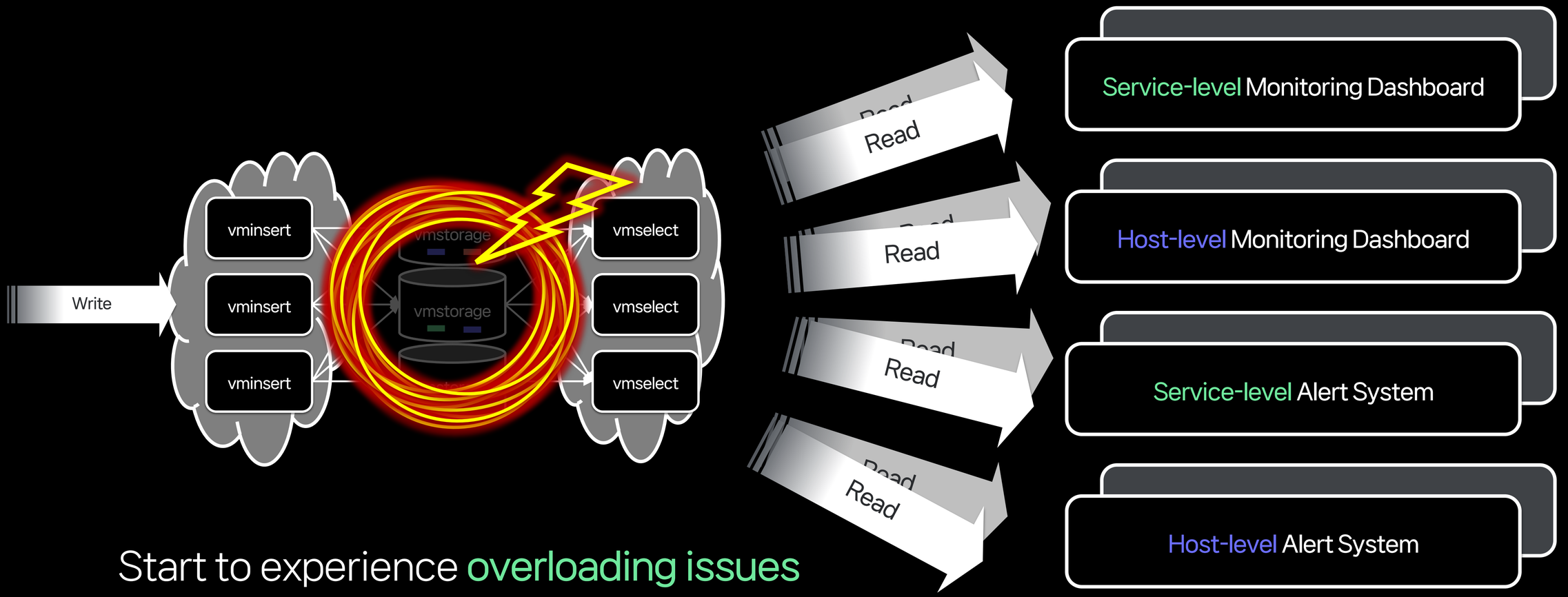
NAVER
DEVIEW
2023

Happy Moments...



3.2 The Rise of The Multiverse

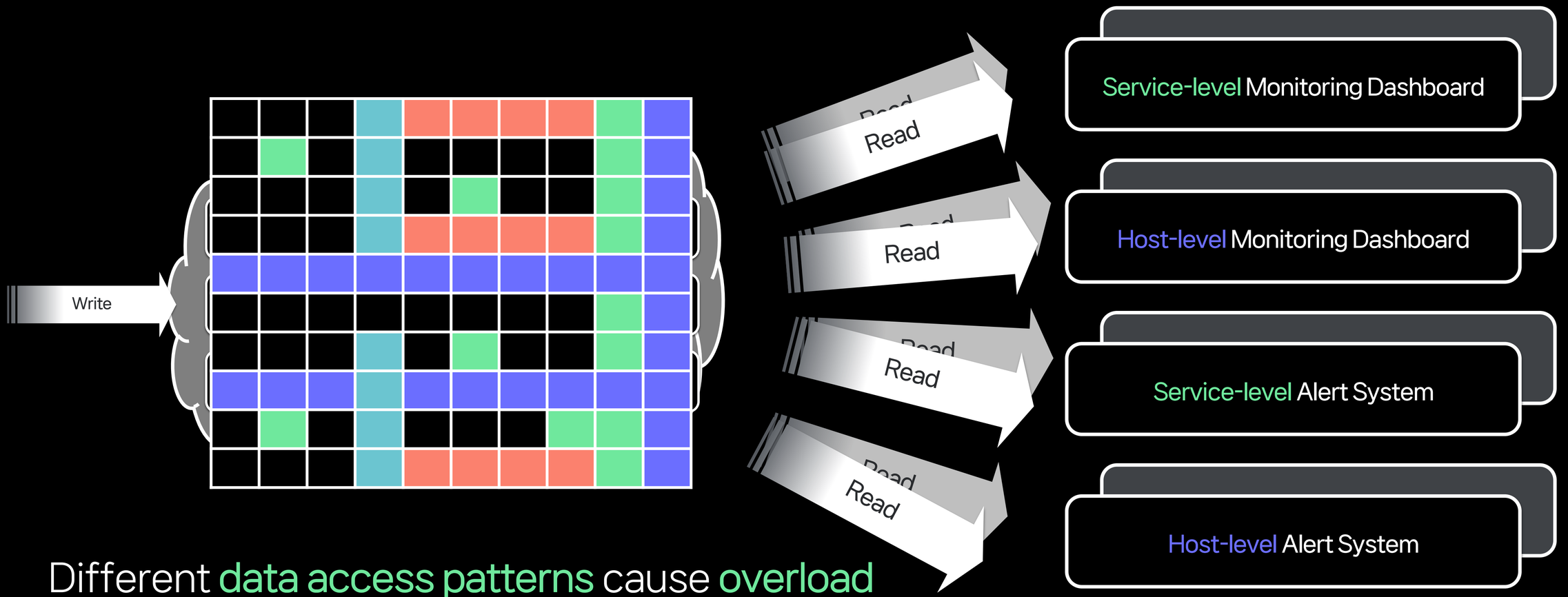
NAVER
DEVIEW
2023



Start to experience overloading issues

3.2 The Rise of The Multiverse

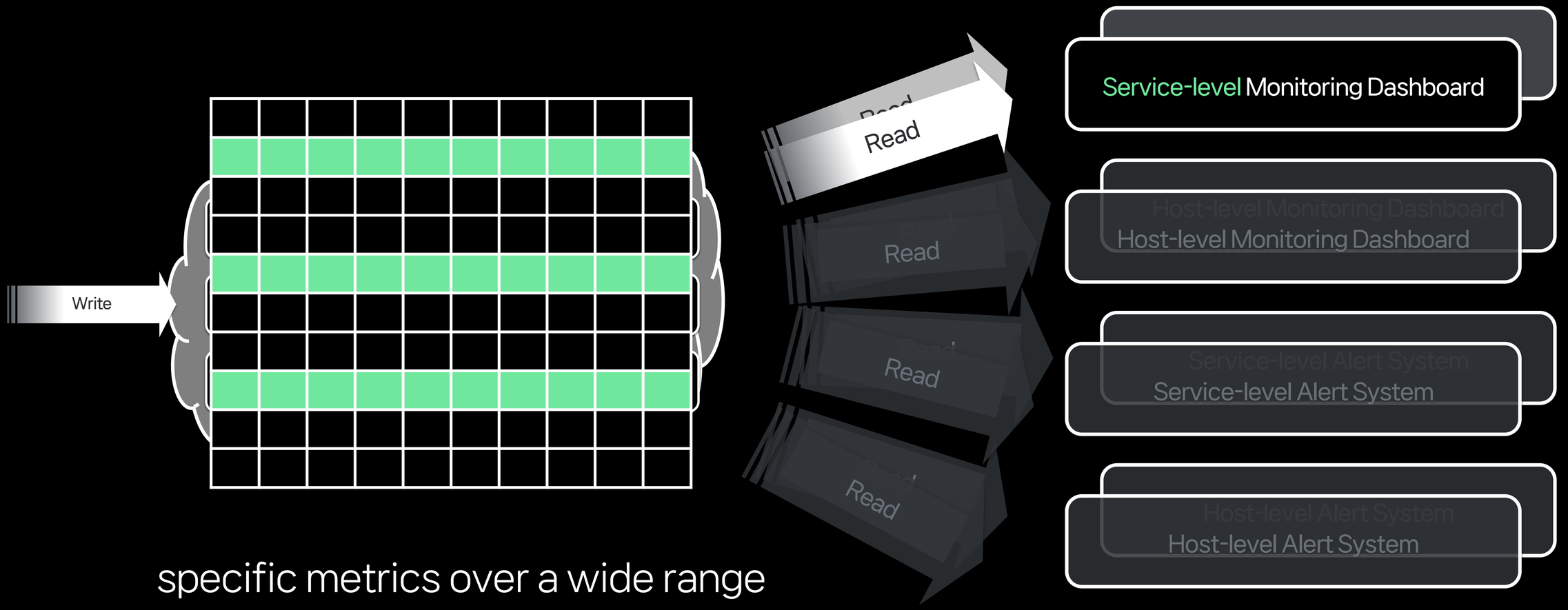
NAVER
DEVIEW
2023



Different data access patterns cause overload

3.2 The Rise of The Multiverse

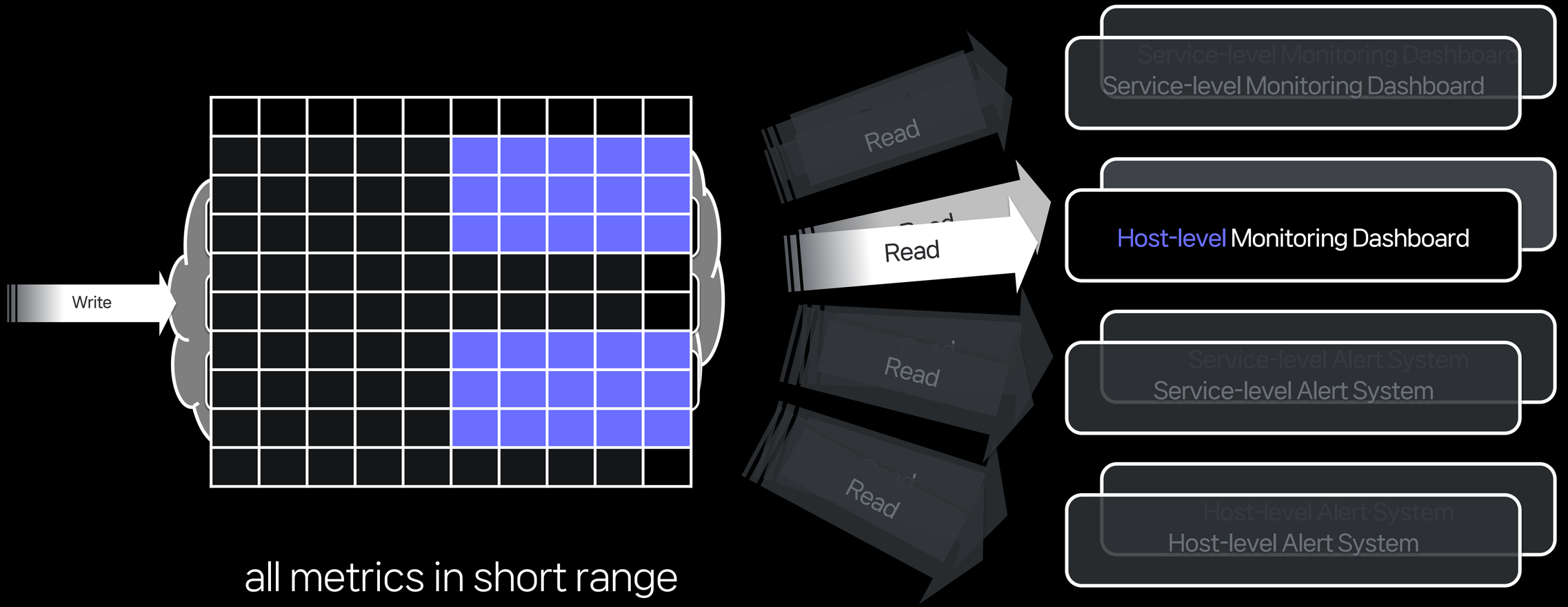
NAVER
DEVIEW
2023



specific metrics over a wide range

3.2 The Rise of The Multiverse

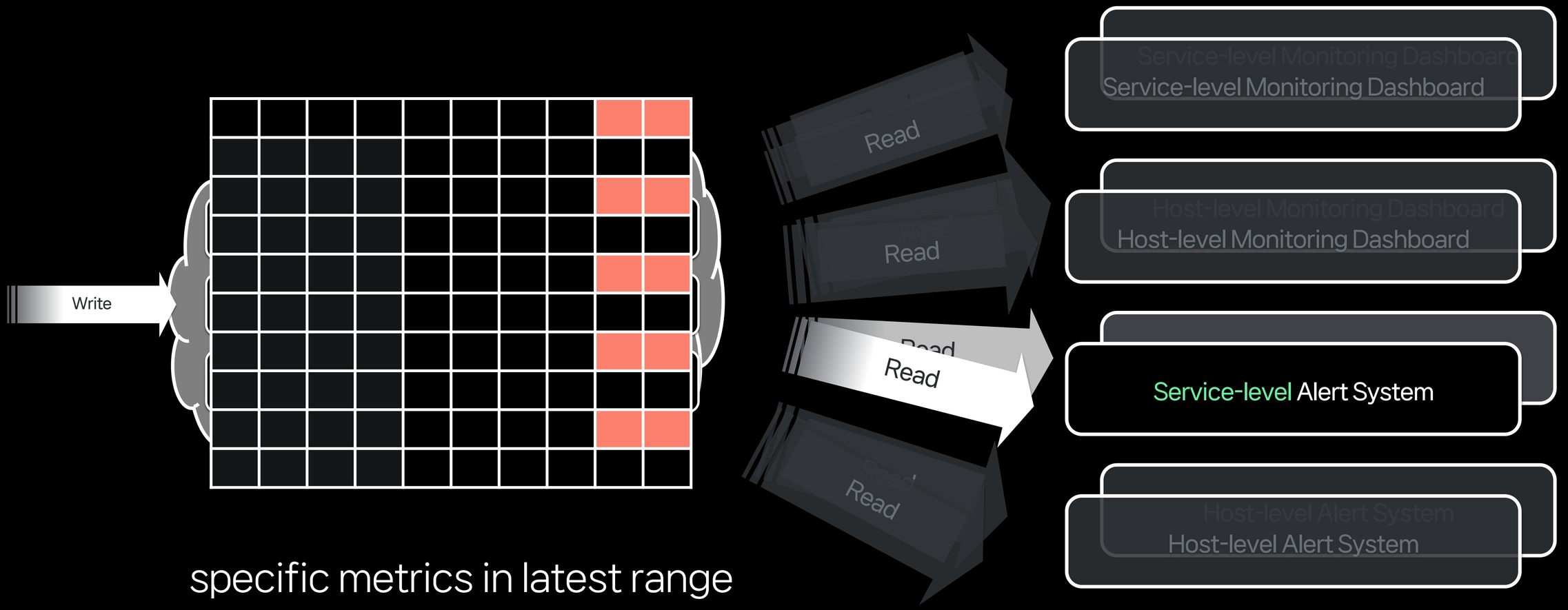
NAVER
DEVIEW
2023



all metrics in short range

3.2 The Rise of The Multiverse

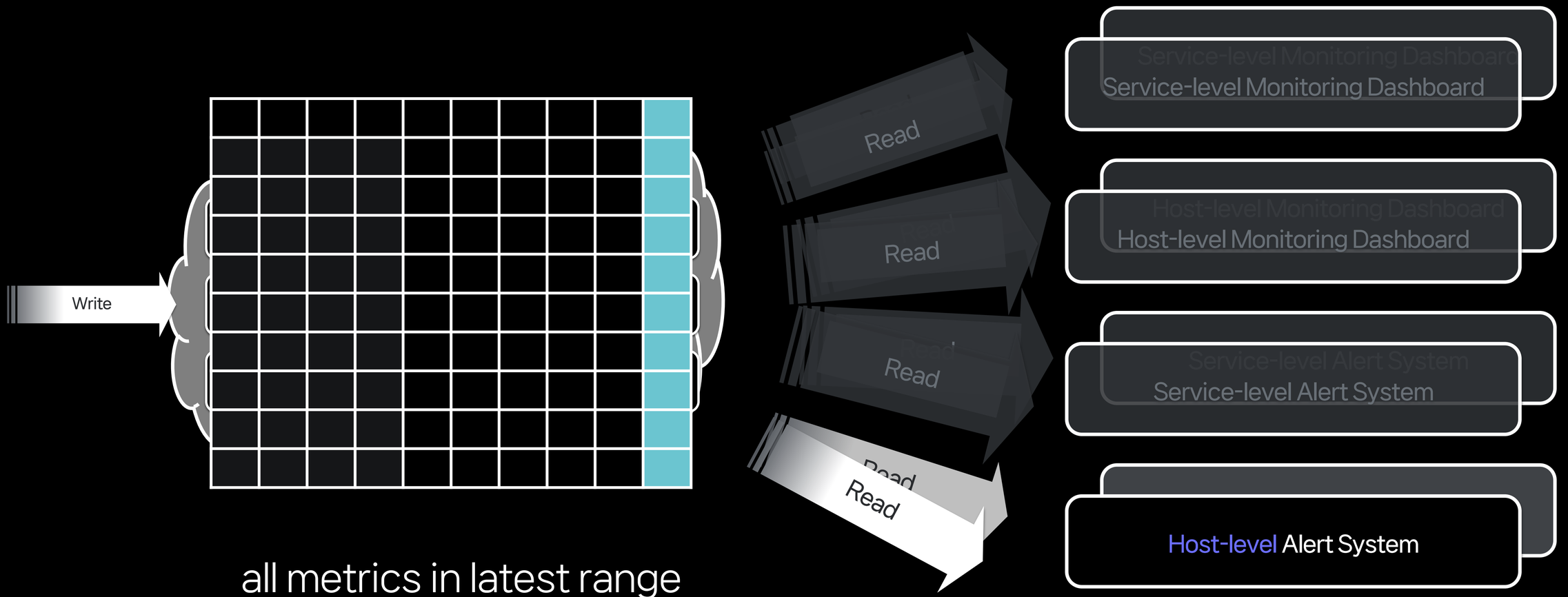
NAVER
DEVIEW
2023



specific metrics in latest range

3.2 The Rise of The Multiverse

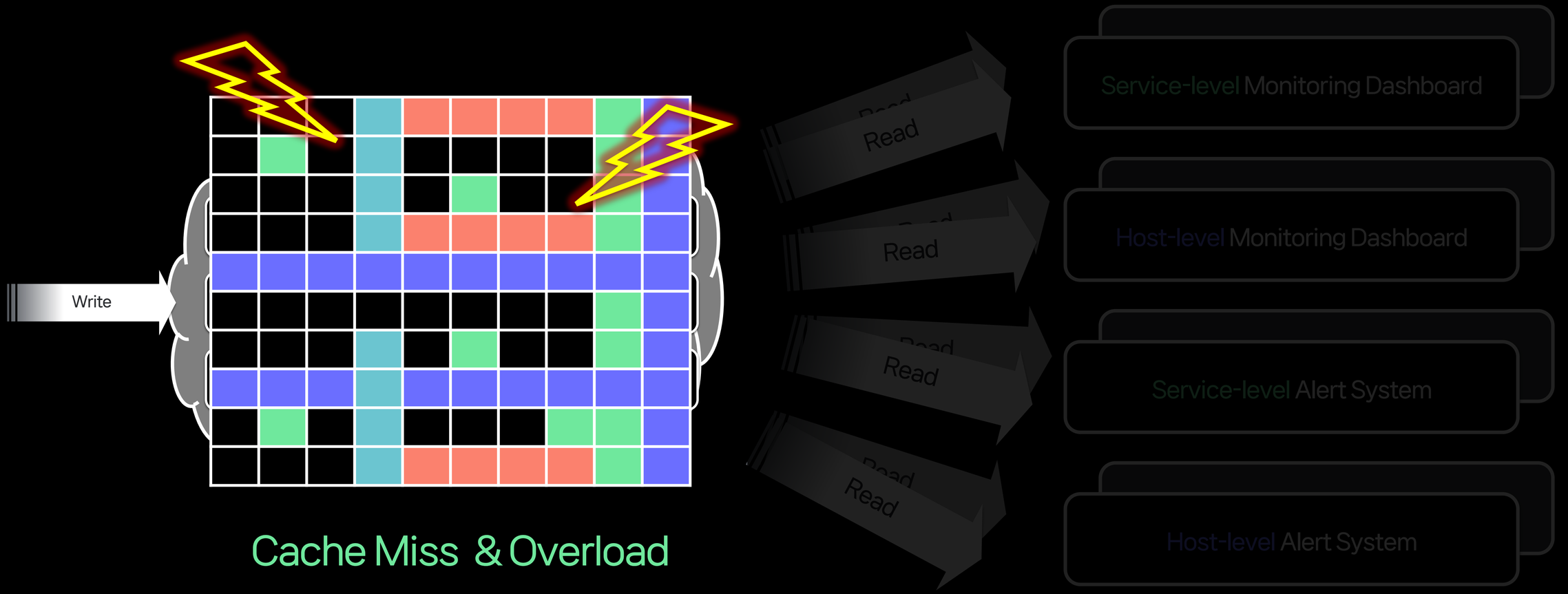
NAVER
DEVIEW
2023



all metrics in latest range

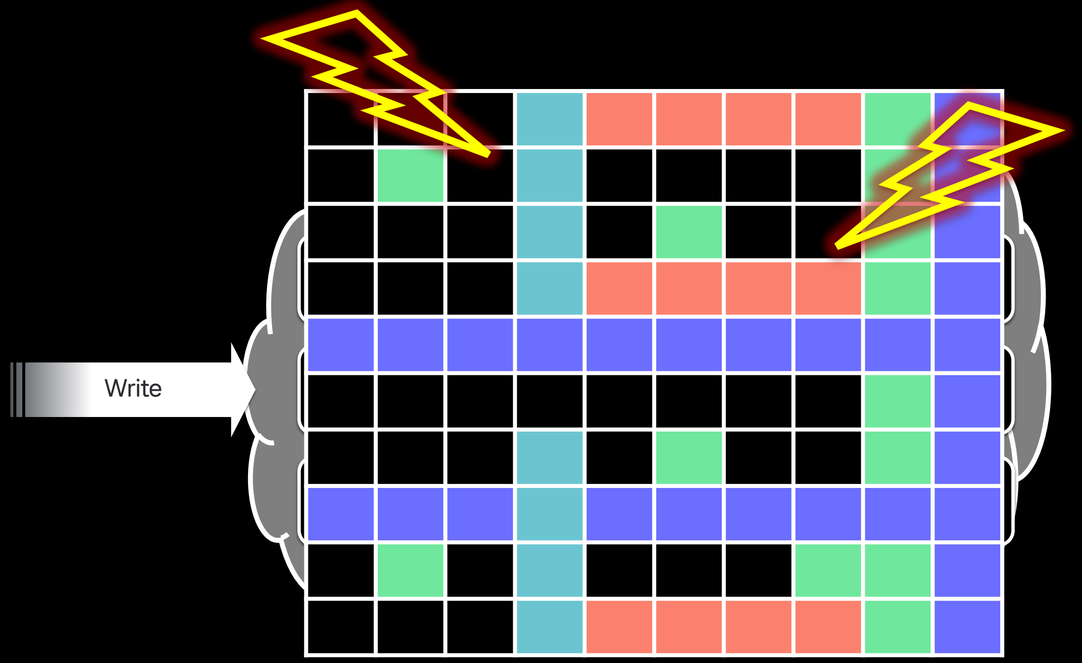
3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

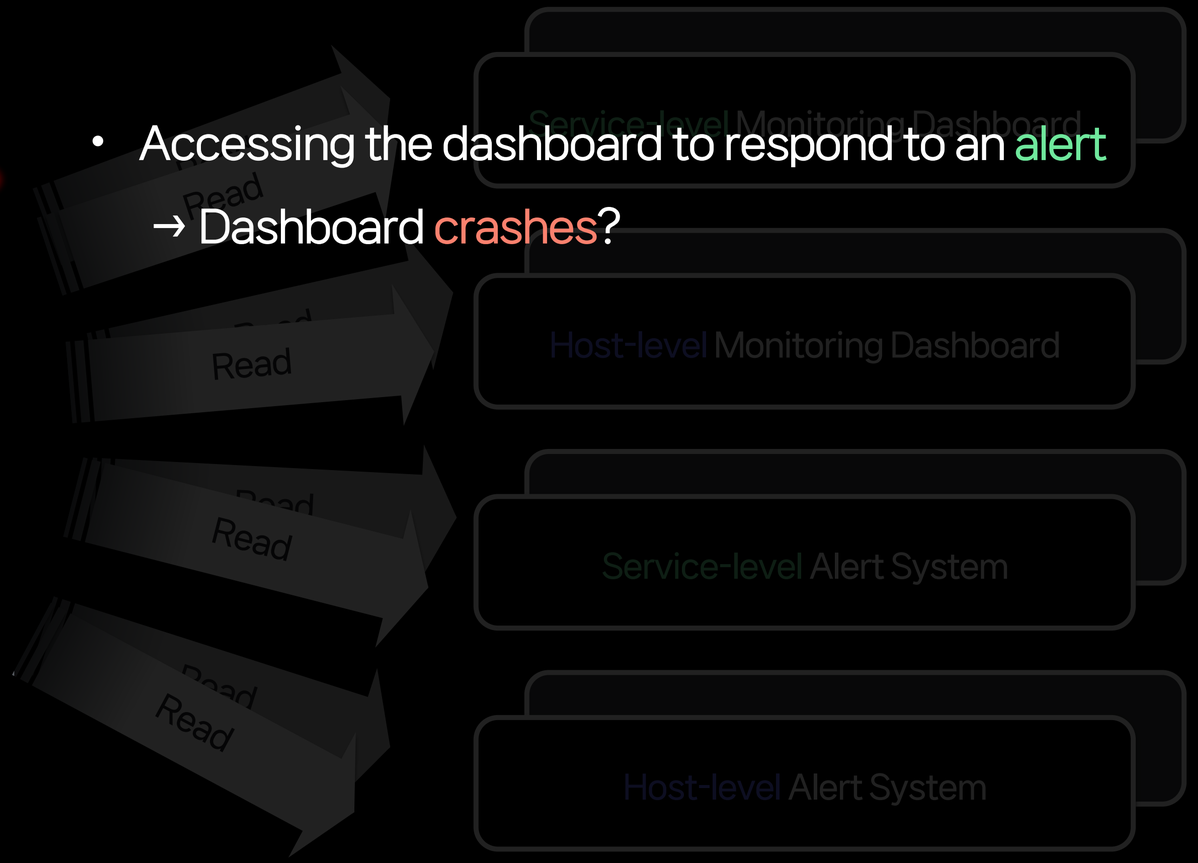


3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

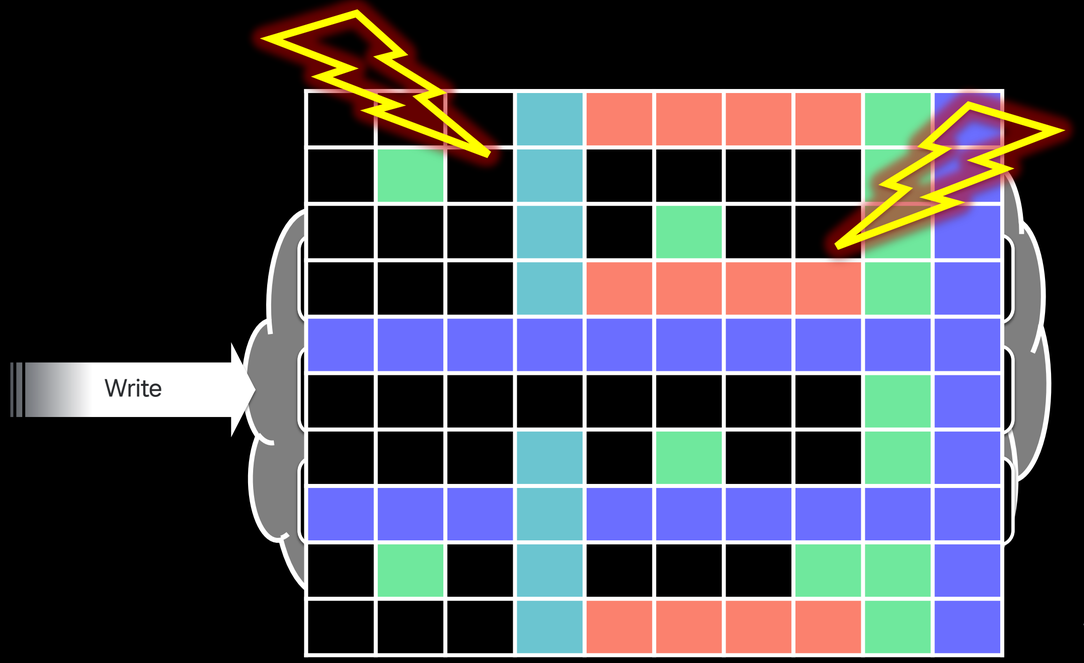


- Accessing the dashboard to respond to an alert
  → Dashboard crashes?

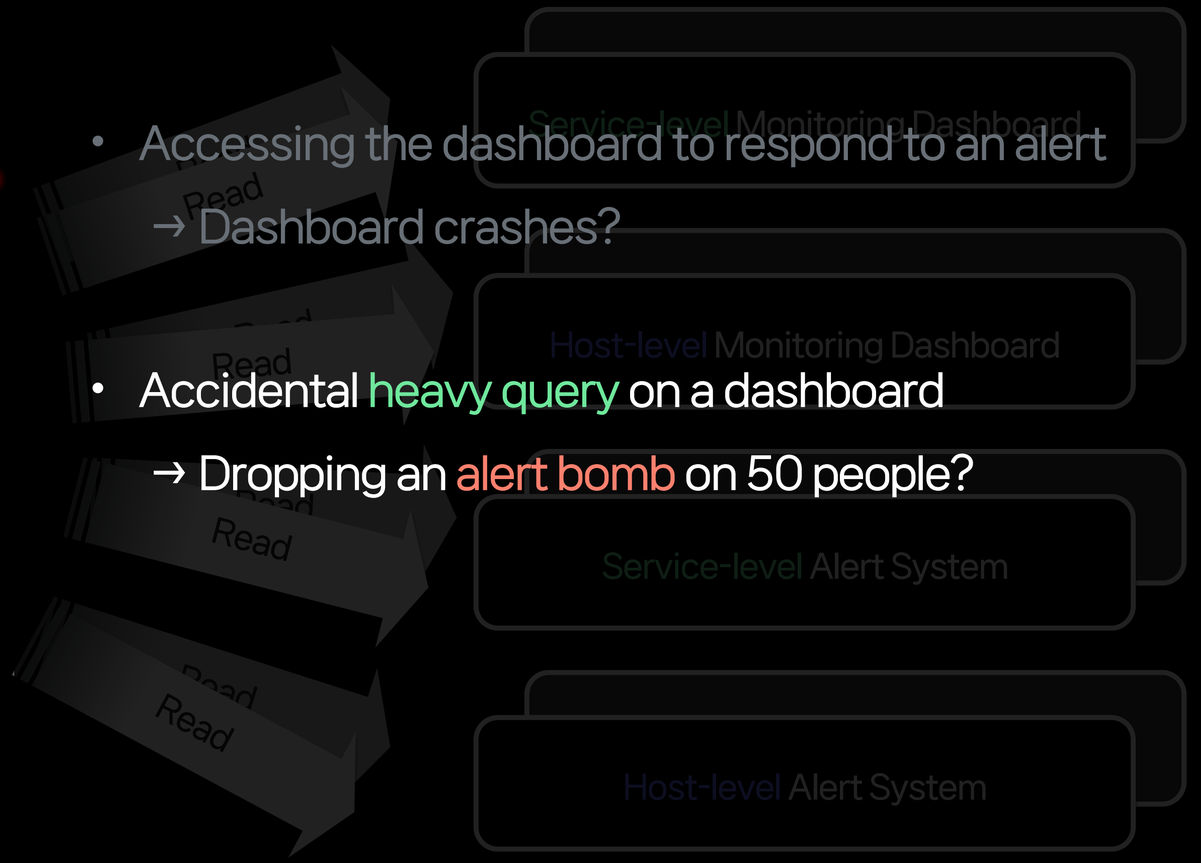


3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

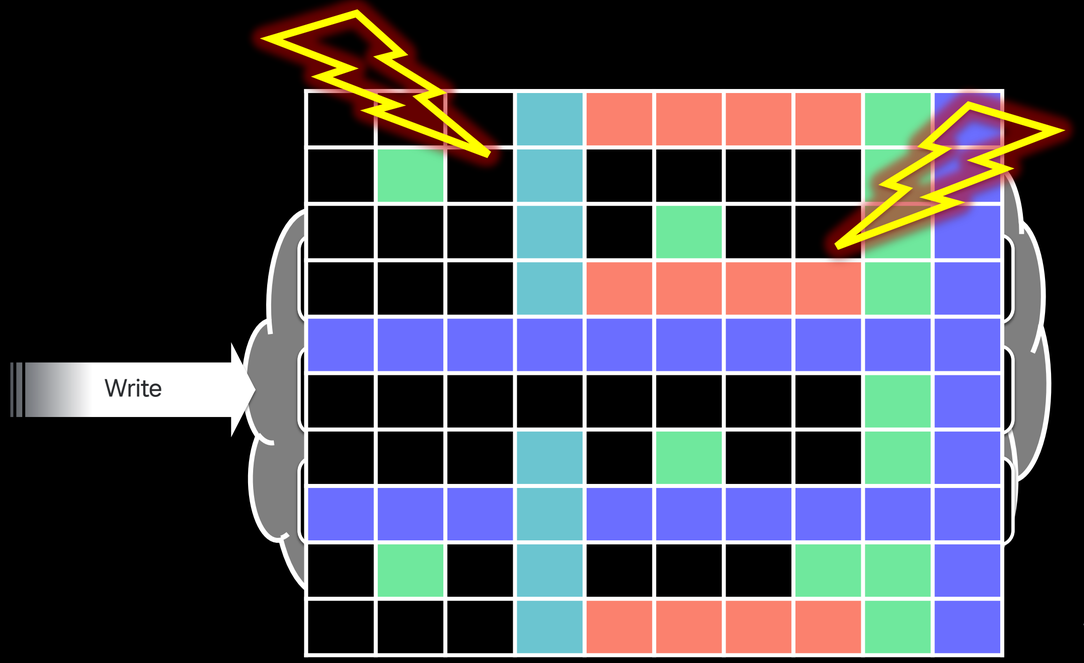


- Accessing the dashboard to respond to an alert
  → Dashboard crashes?
- Accidental heavy query on a dashboard
  → Dropping an alert bomb on 50 people?

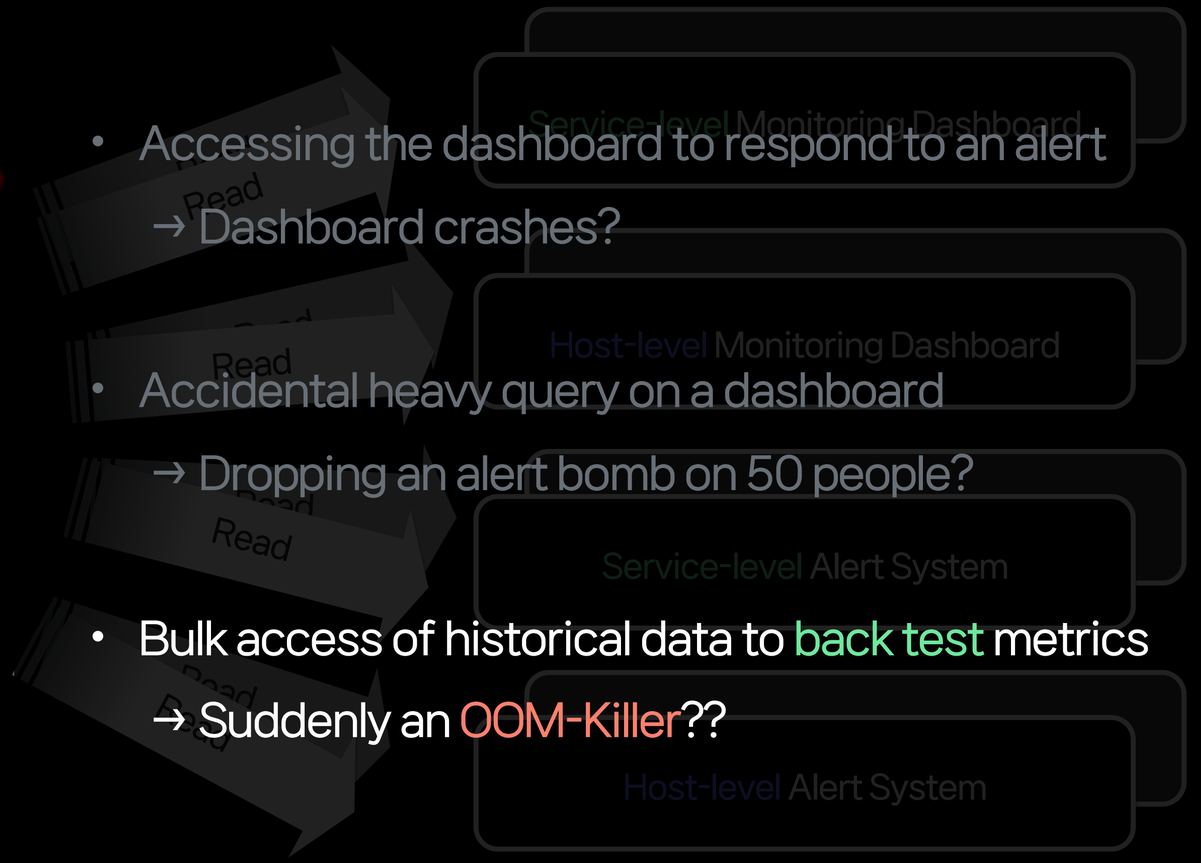


3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

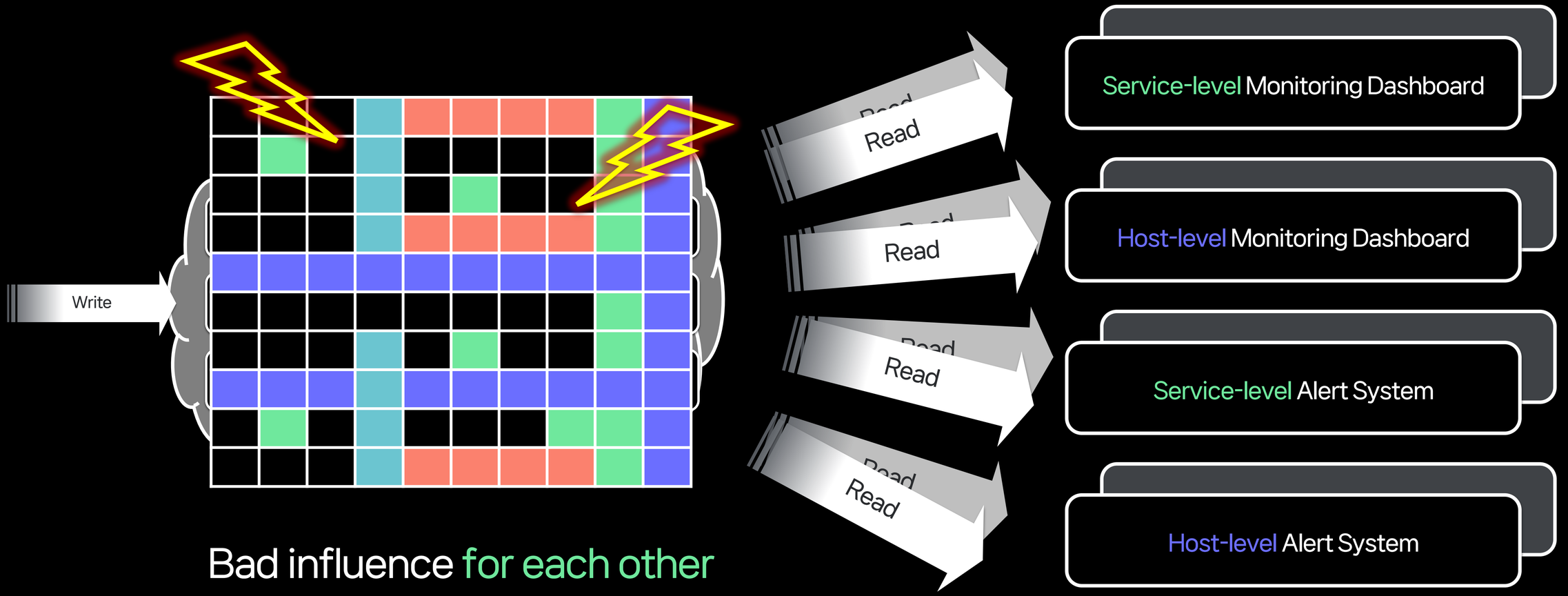


- Accessing the dashboard to respond to an alert
  → Dashboard crashes?
- Accidental heavy query on a dashboard
  → Dropping an alert bomb on 50 people?
- Bulk access of historical data to back test metrics
  → Suddenly an OOM-Killer??



3.2 The Rise of The Multiverse

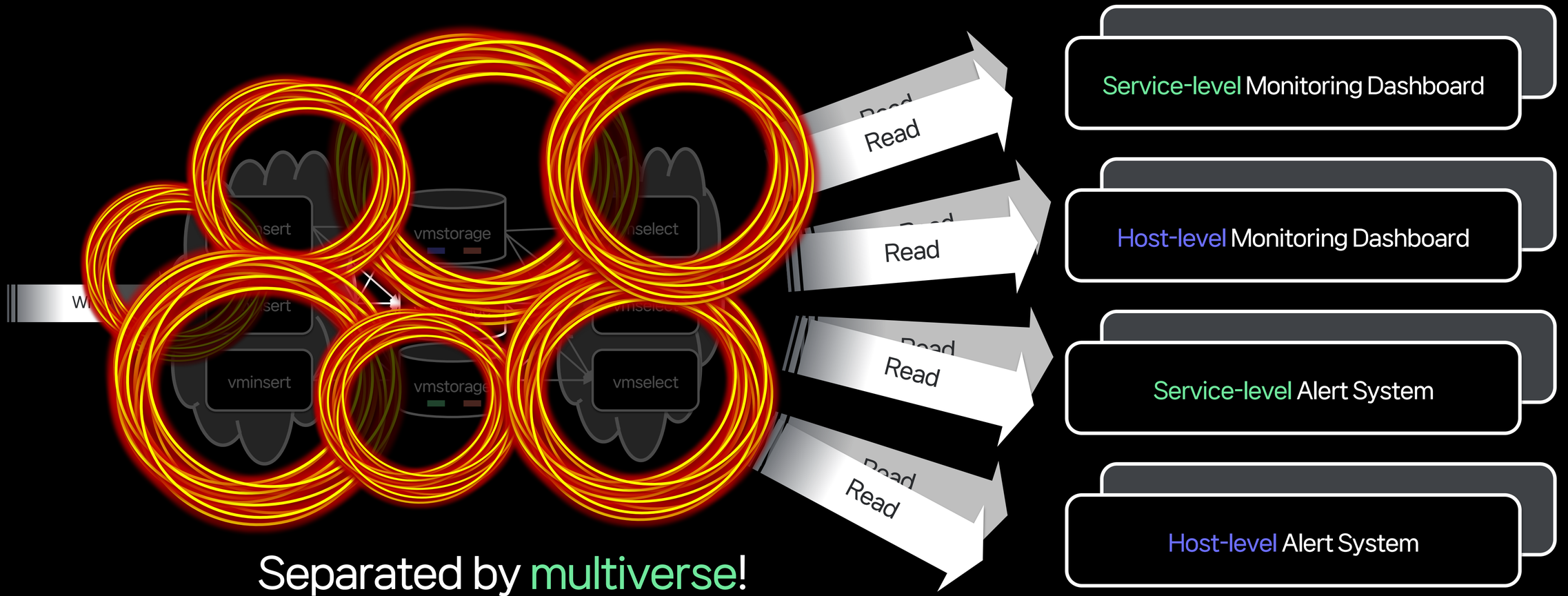
NAVER
DEVIEW
2023



Bad influence for each other

3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

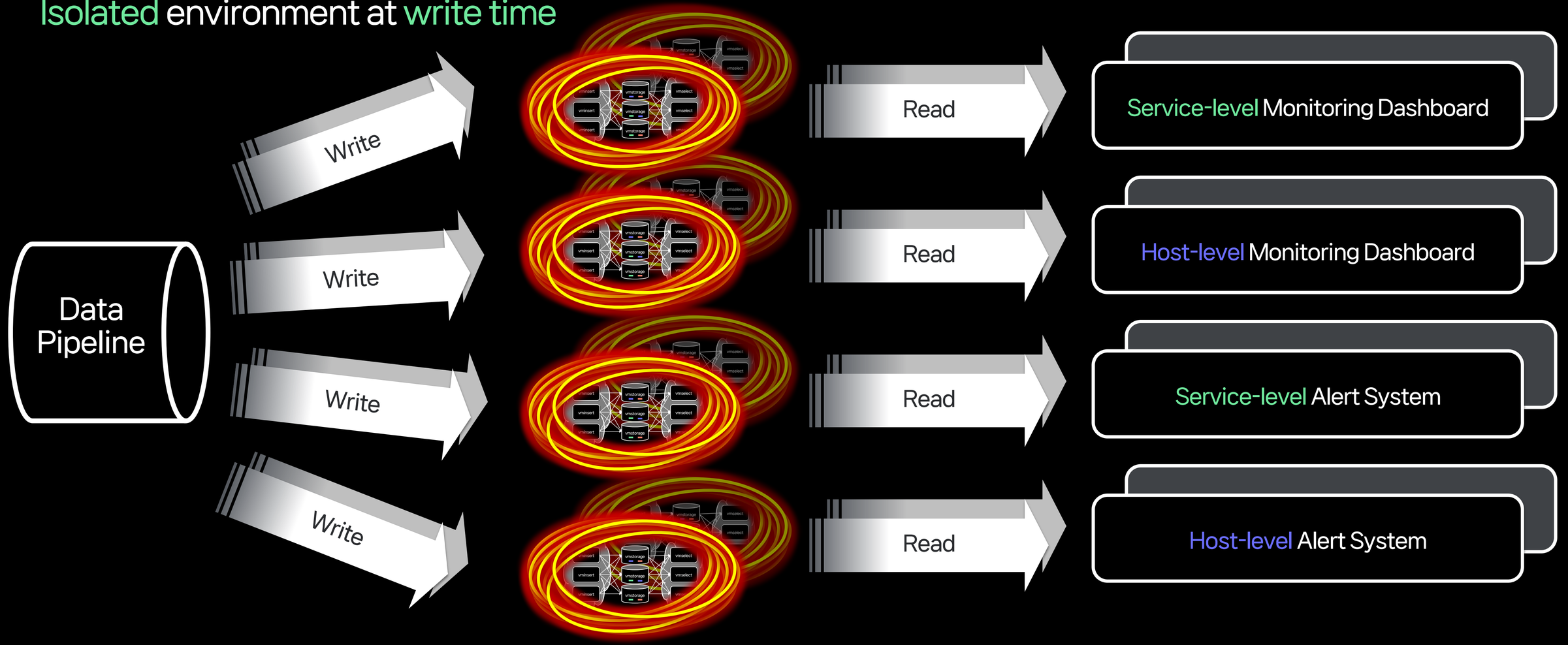


Separated by multiverse!

3.2 The Rise of The Multiverse

NAVER DEVIEW 2023

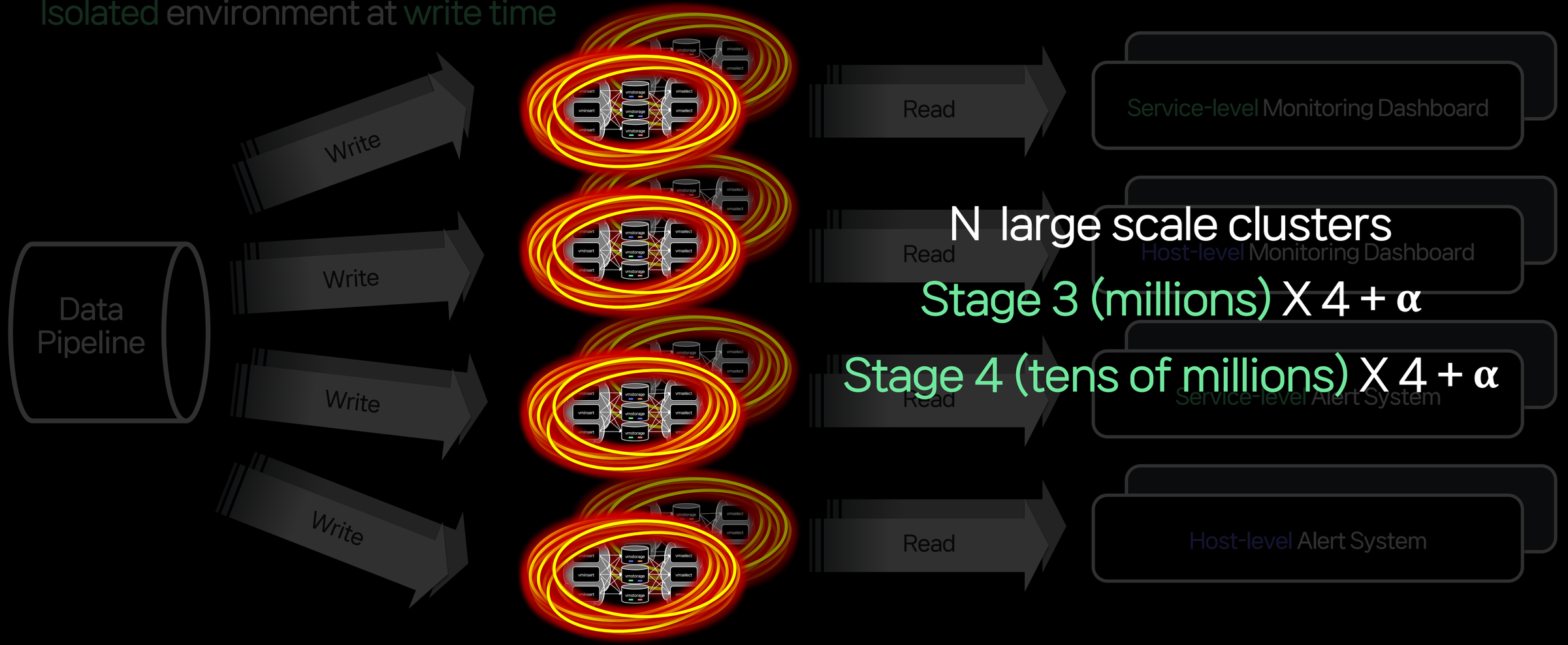
Isolated environment at write time



3.2 The Rise of The Multiverse

NAVER
DEVIEW
2023

Isolated environment at write time



3. Time series in the Multiverse of Madness

NAVER
DEVIEW
2023

Summary

- Using Cluster mode to resolve Single Point of Failure issues

3. Time series in the Multiverse of Madness

NAVER
DEVIEW
2023

Summary

- Using Cluster mode to resolve Single Point of Failure issues
- Different access patterns of various applications cause overload

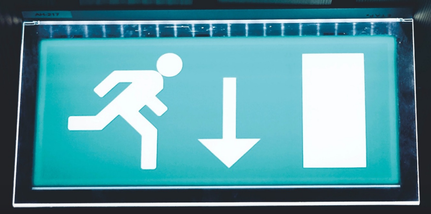
3. Time series in the Multiverse of Madness

NAVER
DEVIEW
2023

Summary

- Using Cluster mode to resolve Single Point of Failure issues
- Different access patterns of various applications cause overload
- Separate clusters into multiverses to prevent them from affecting each other

4. Lessons Learned



- Monitoring and alerting systems act as emergency lights.

- Monitoring and alerting systems act as emergency lights.
- The monitoring system should function normally even if the service fails

- Monitoring and alerting systems act as emergency lights.
- The monitoring system should function normally even if the service fails
- Shouldn't raise an alert if the service doesn't fail

NAVER
DEVIEW
2023

Why data loss and downtime should never happen in a time series DB

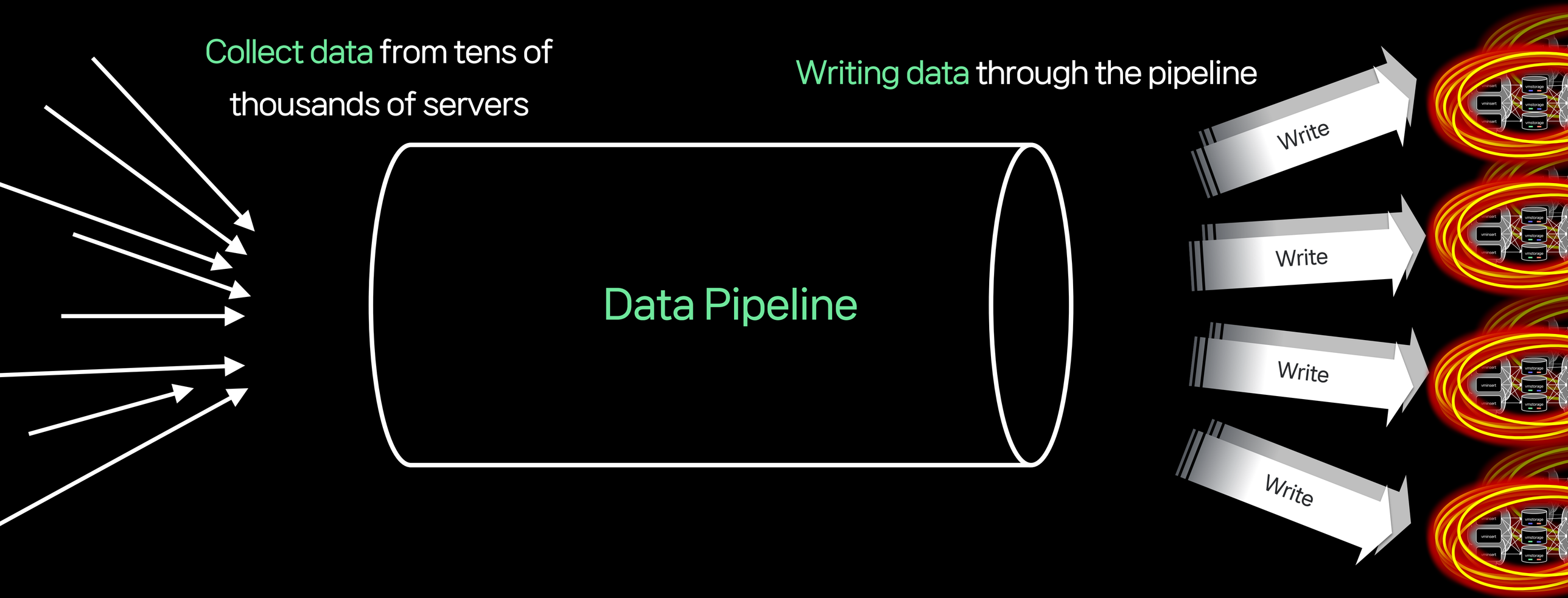
이미지 출처 : https://www.pexels.com/photo/lighted-running-signage-845265/

4.1 Write path for no data loss

NAVER
DEVIEW
2023

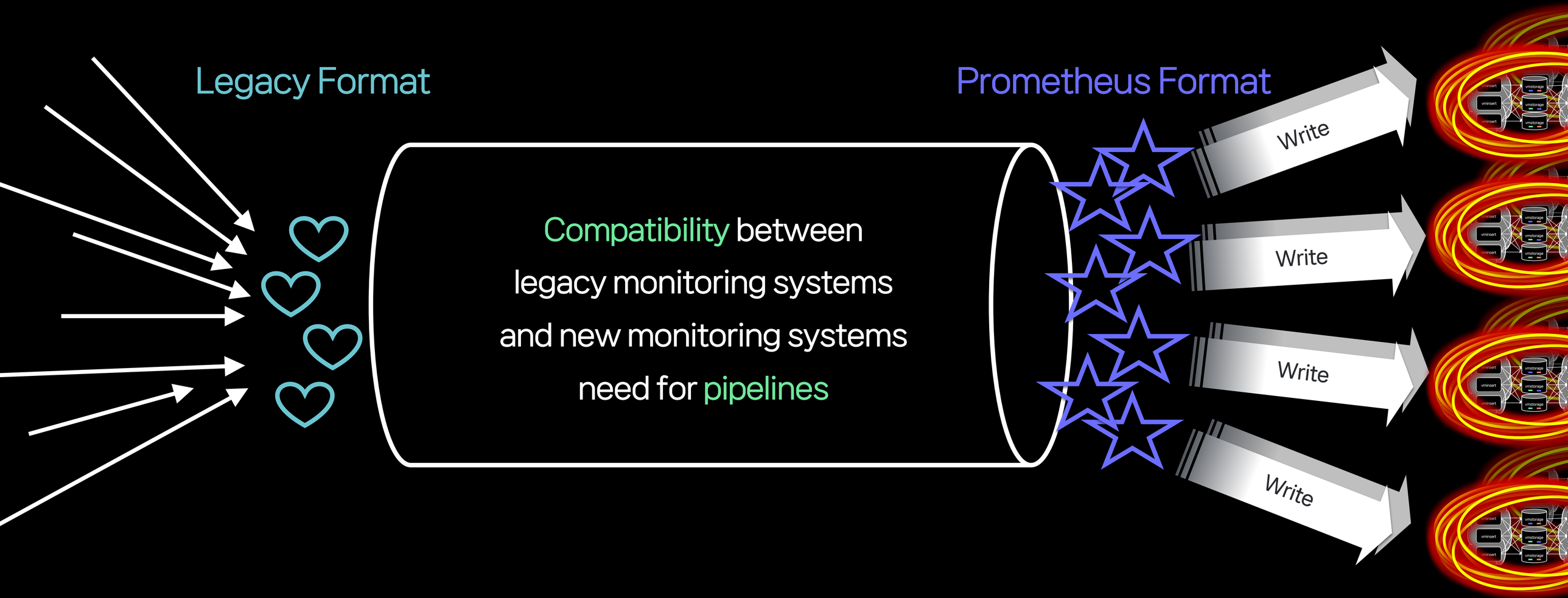
Collect data from tens of thousands of servers

Writing data through the pipeline



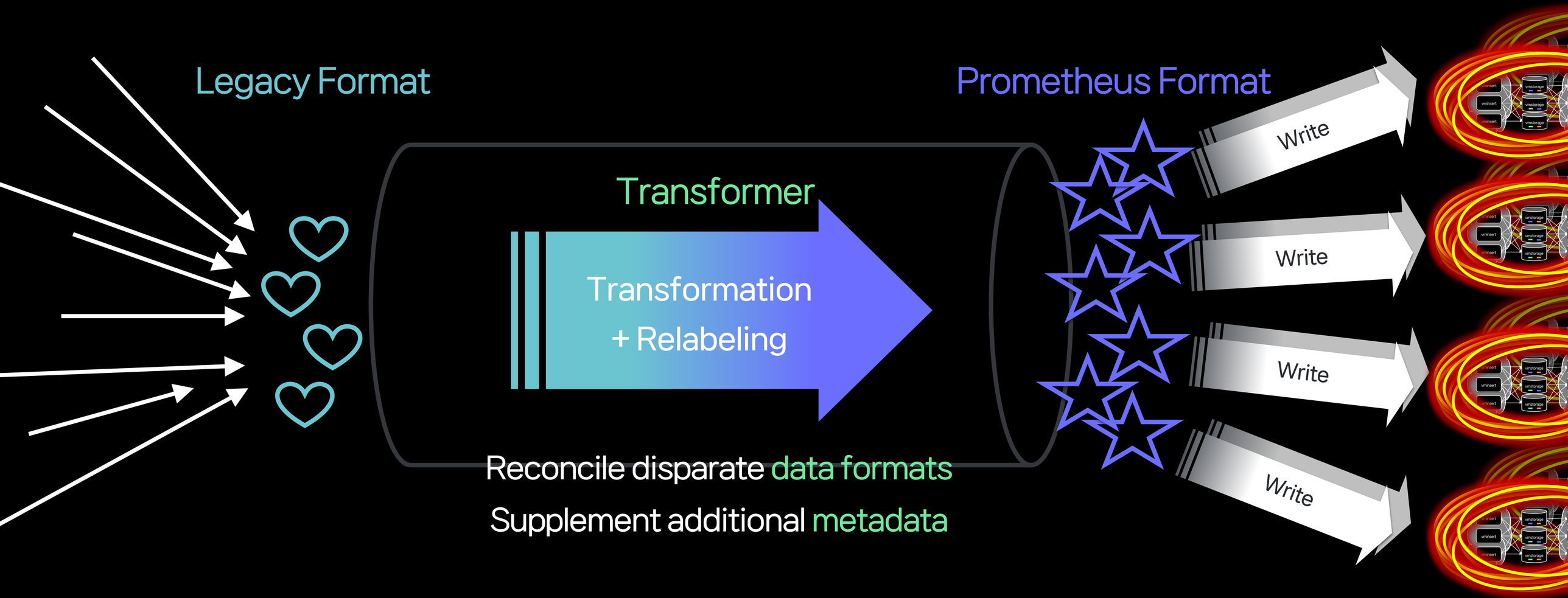
4.1 Write path for no data loss

NAVER
DEVIEW
2023



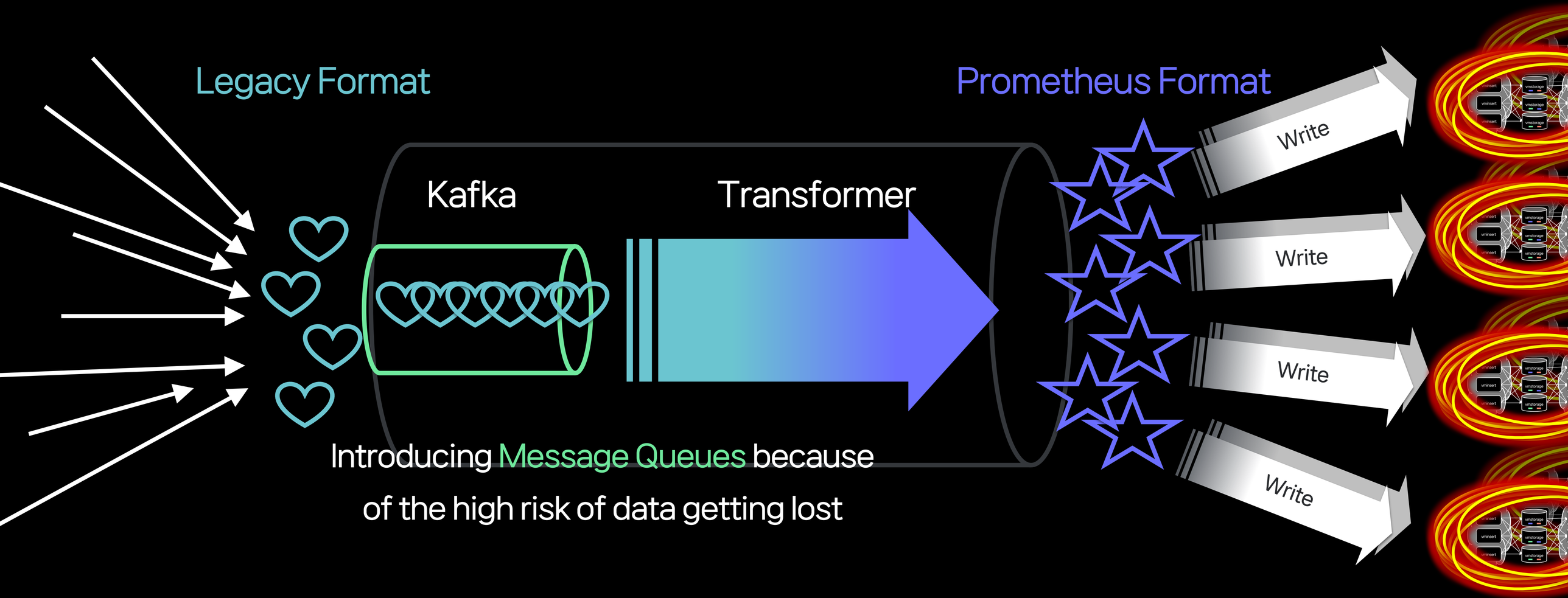
4.1 Write path for no data loss

NAVER
DEVIEW
2023



4.1 Write path for no data loss

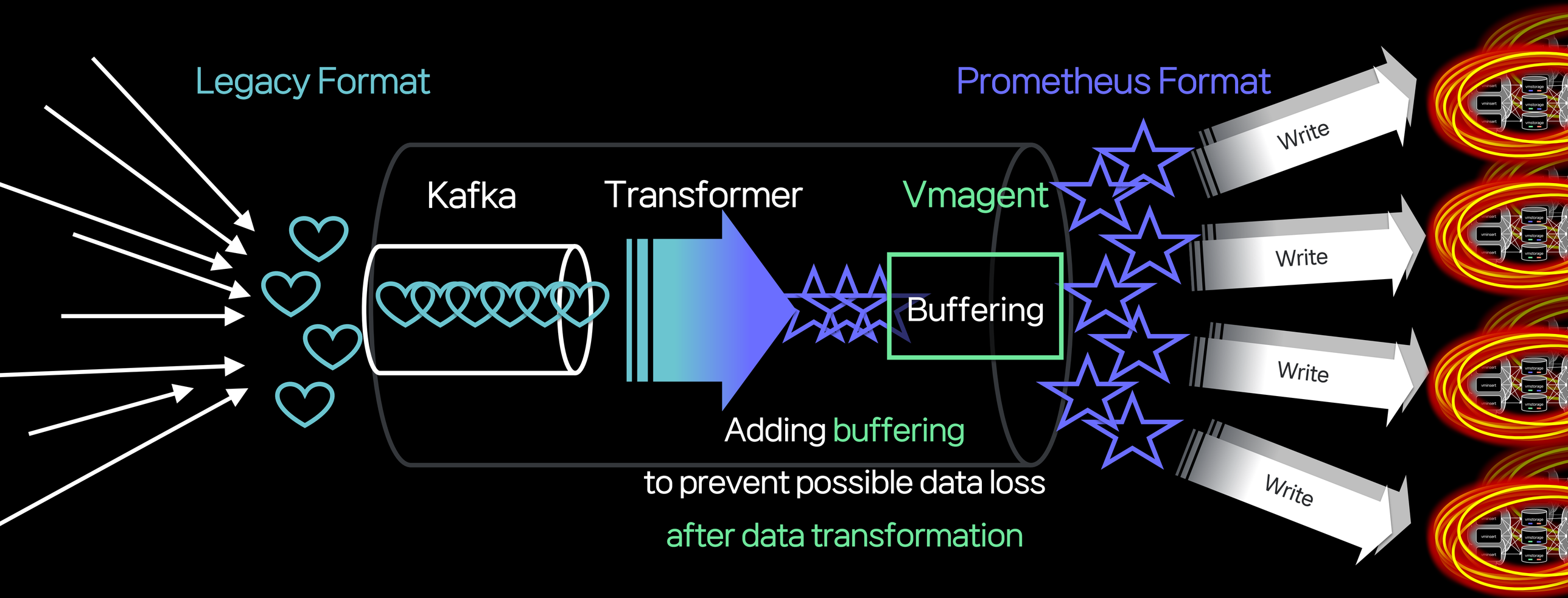
NAVER
DEVIEW
2023



Introducing Message Queues because
of the high risk of data getting lost

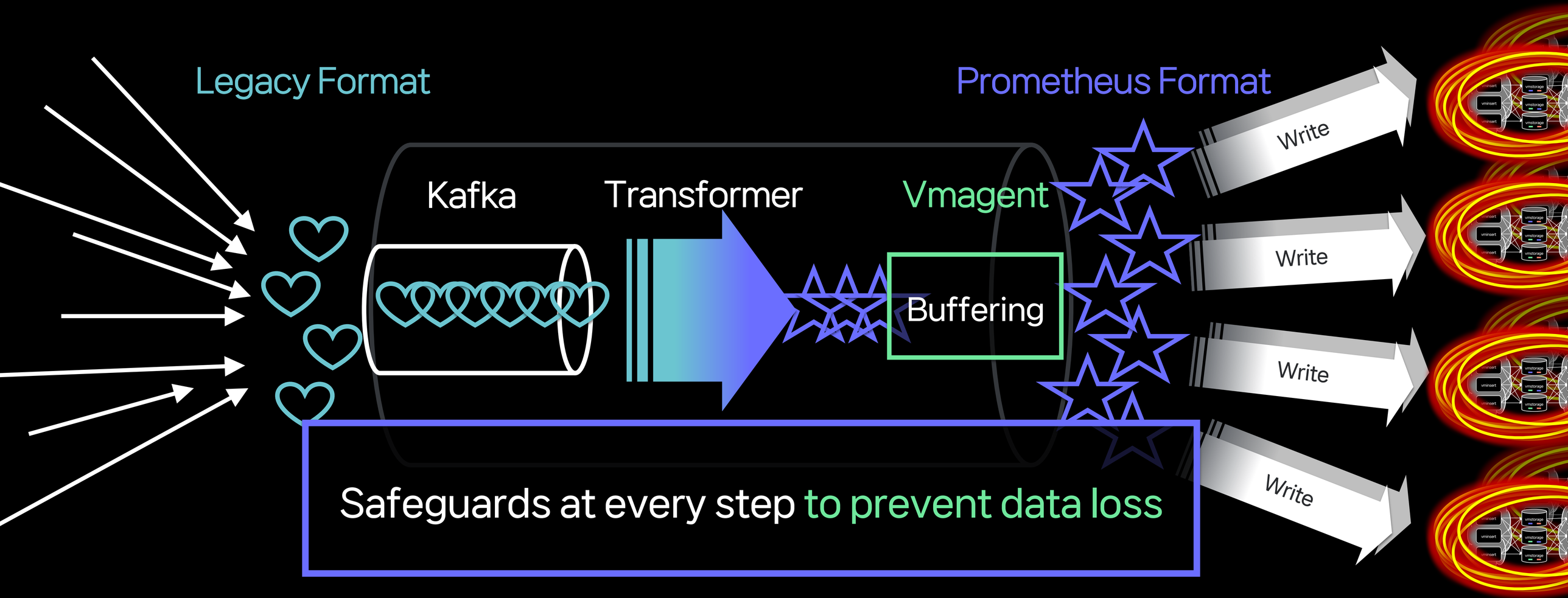
4.1 Write path for no data loss

NAVER
DEVIEW
2023



4.1 Write path for no data loss

NAVER
DEVIEW
2023

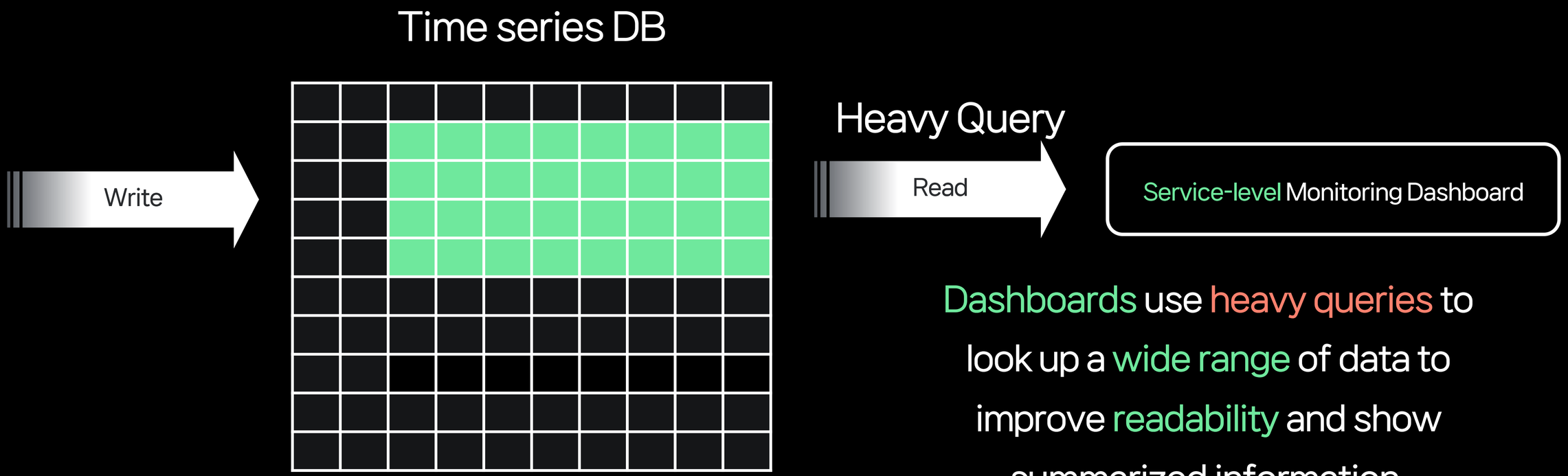


4.2 Read path for no downtime

NAVER
DEVIEW
2023

4.2 Read path for no downtime

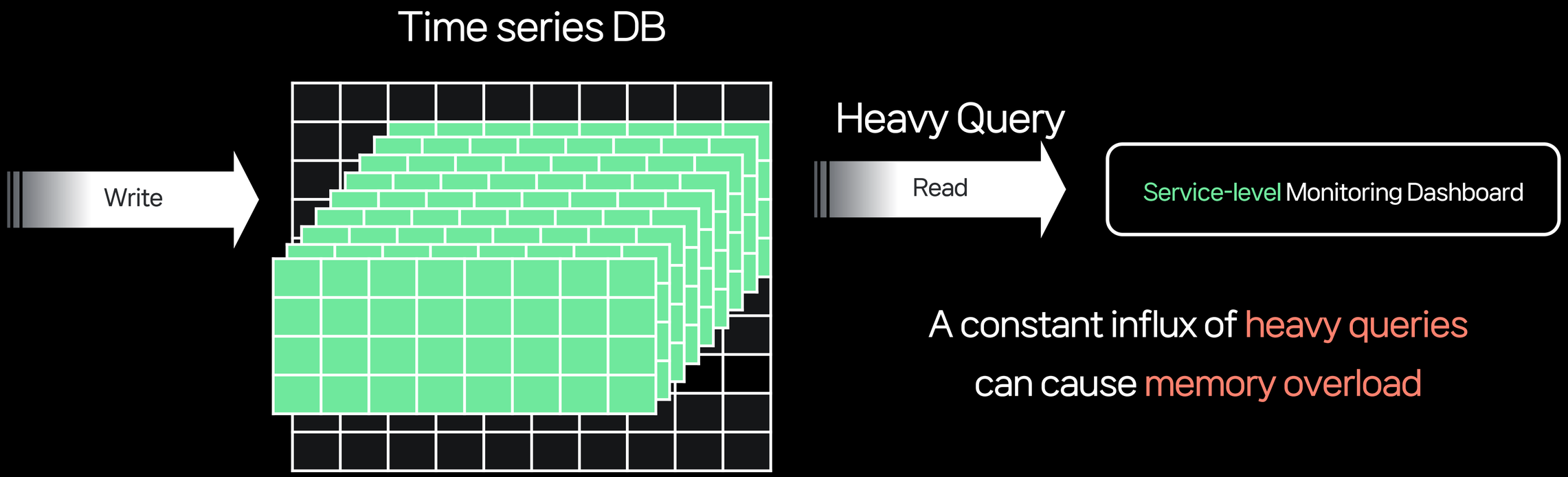
NAVER
DEVIEW
2023



Dashboards use heavy queries to look up a wide range of data to improve readability and show summarized information.

4.2 Read path for no downtime

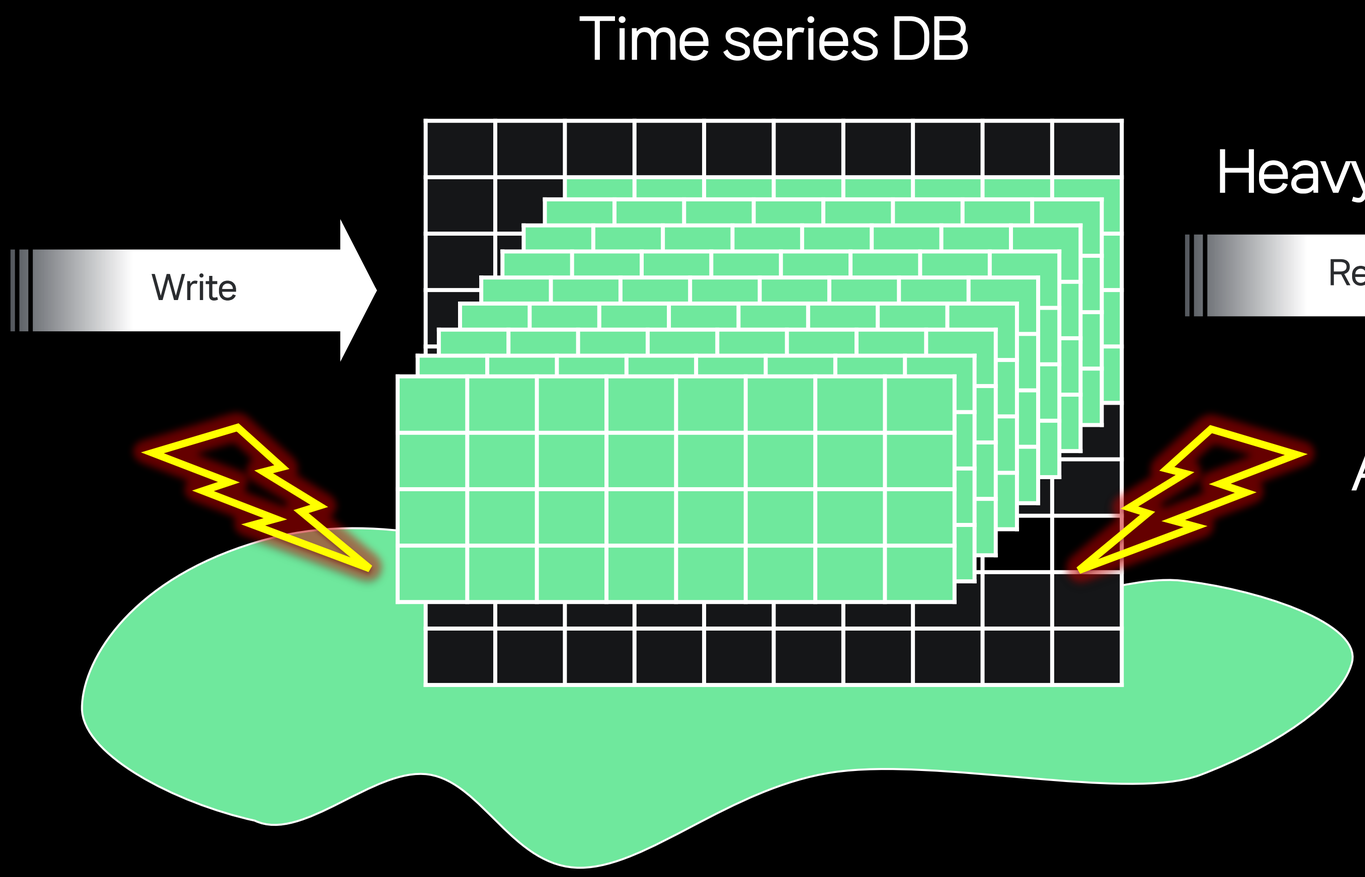
NAVER
DEVIEW
2023



A constant influx of heavy queries
can cause memory overload

4.2 Read path for no downtime

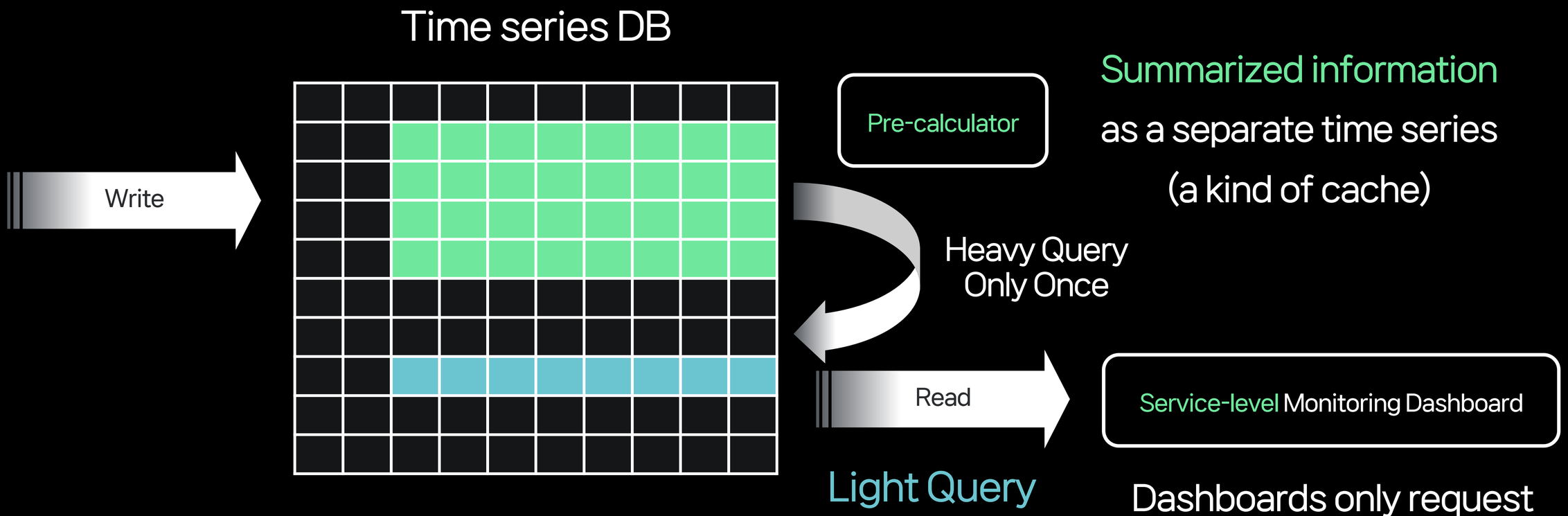
NAVER
DEVIEW
2023



A constant influx of heavy queries
can cause memory overload
→ May cause OOM-Kill!

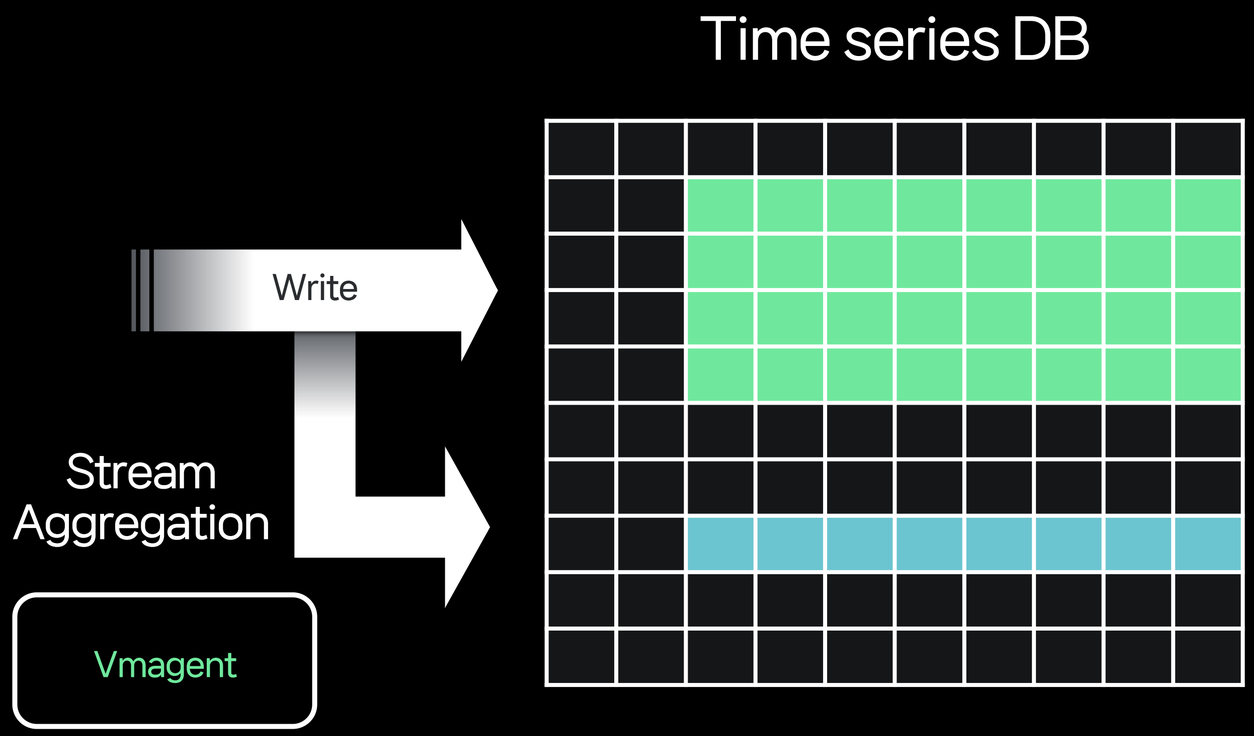
4.2 Read path for no downtime

NAVER
DEVIEW
2023



4.2 Read path for no downtime

NAVER
DEVIEW
2023



Summarized information from the beginning

Read

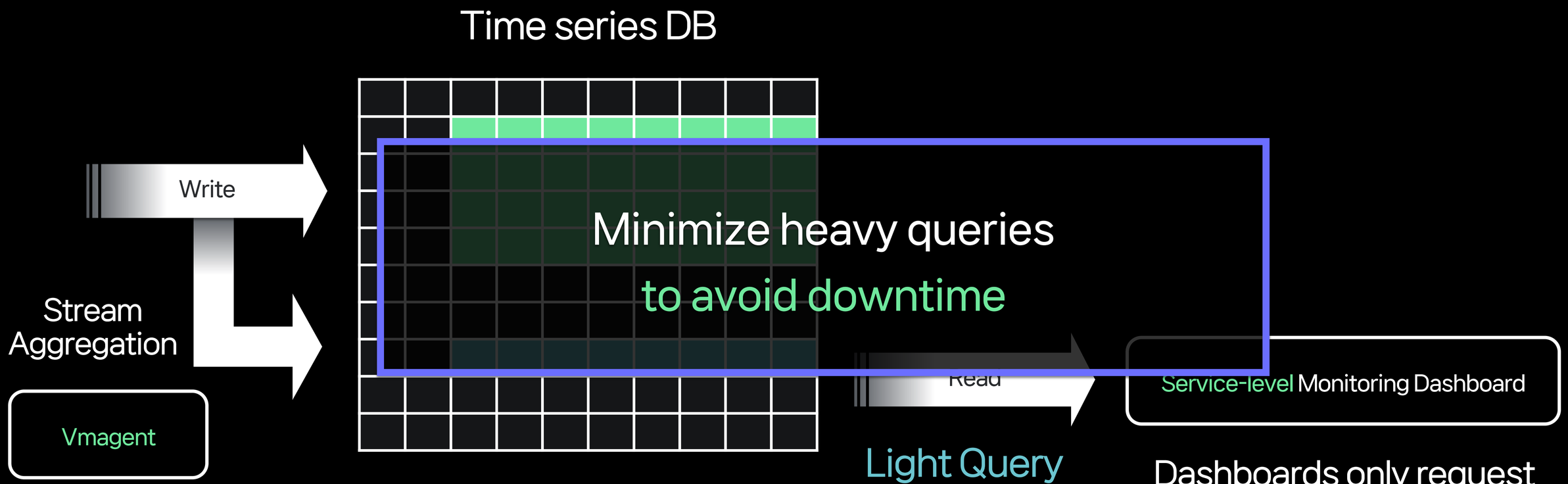
Light Query

Service-level Monitoring Dashboard

Dashboards only request
summarized information,
so use much lighter queries

4.2 Read path for no downtime

NAVER
DEVIEW
2023

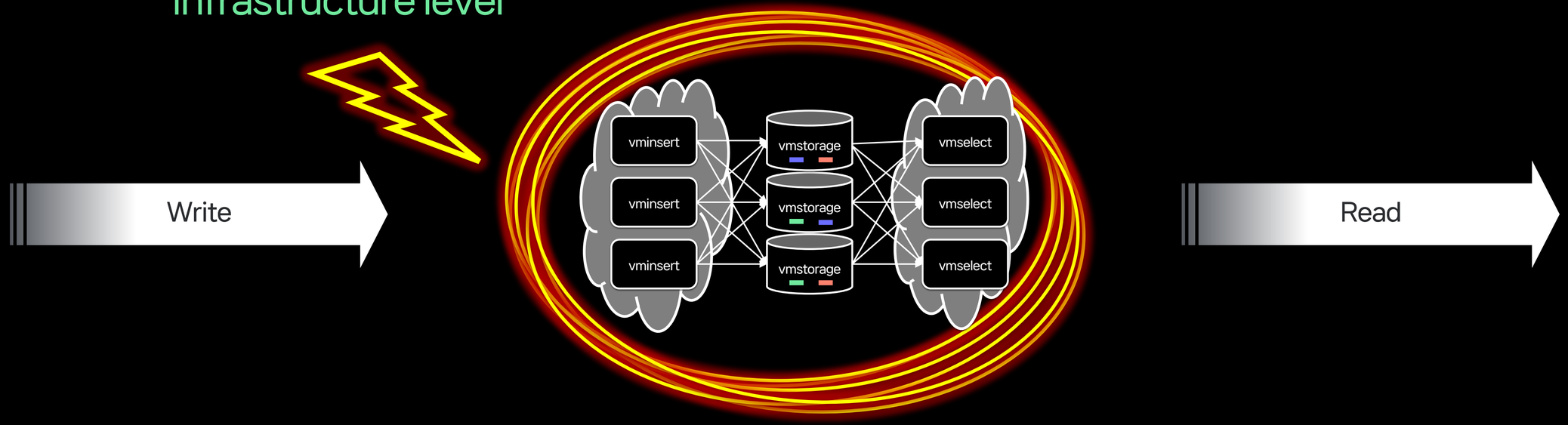


4.3 Multiverse Management

NAVER
DEVIEW
2023

The pain never ends

Various issues at the
infrastructure level



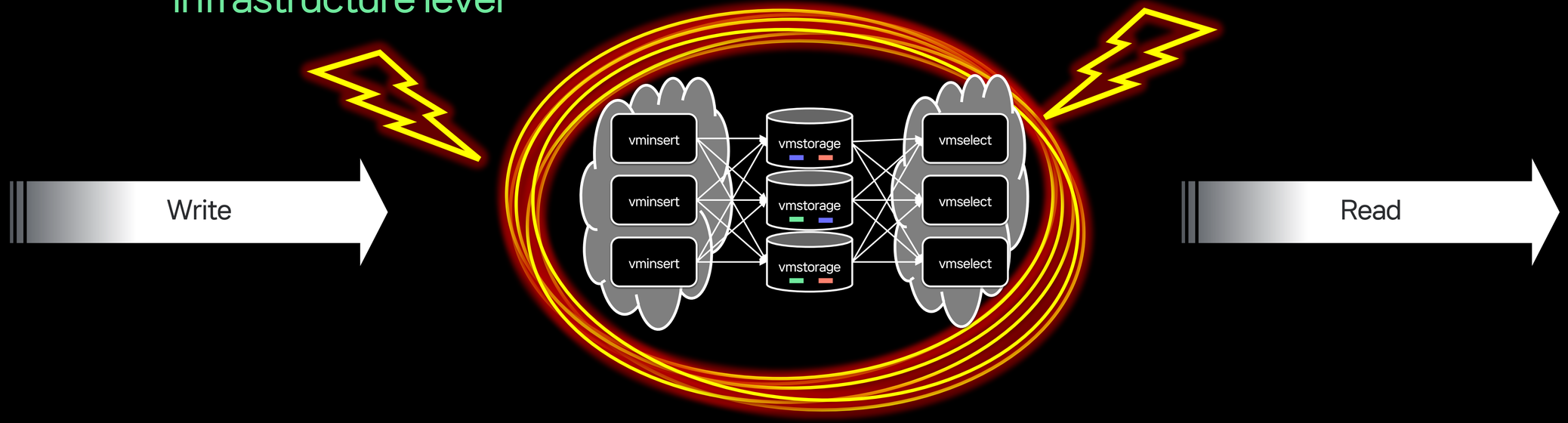
4.3 Multiverse Management

NAVER
DEVIEW
2023

The pain never ends

Various issues at the
infrastructure level

With momentary overloads
Write / Read Delay



4.3 Multiverse Management

NAVER
DEVIEW
2023

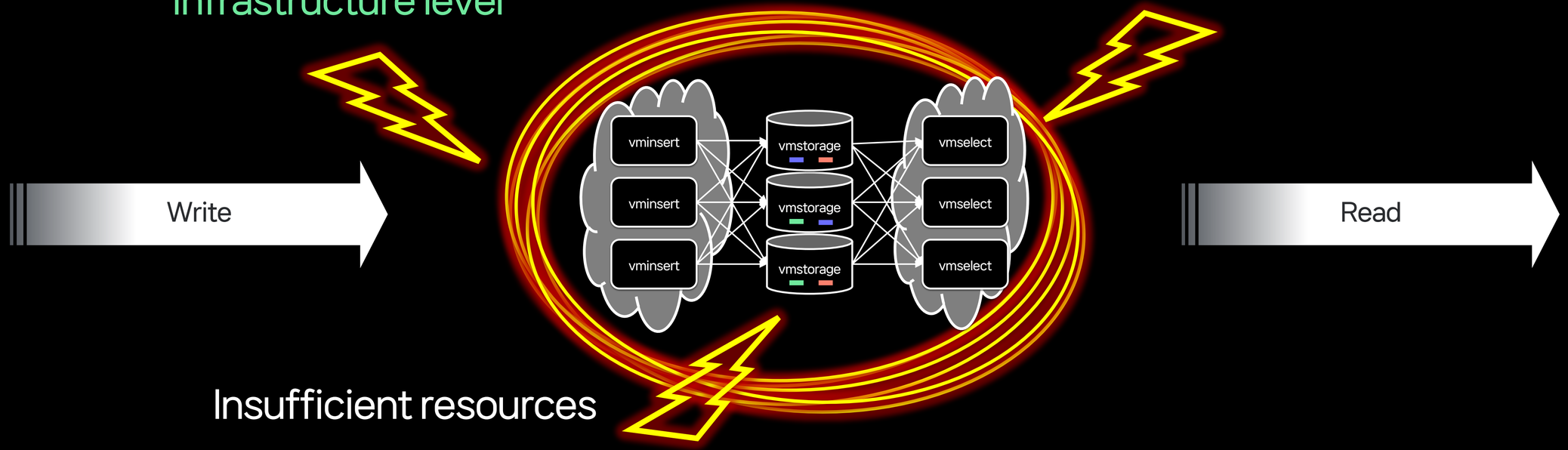
The pain never ends

With momentary overloads

Write / Read Delay

Various issues at the

infrastructure level



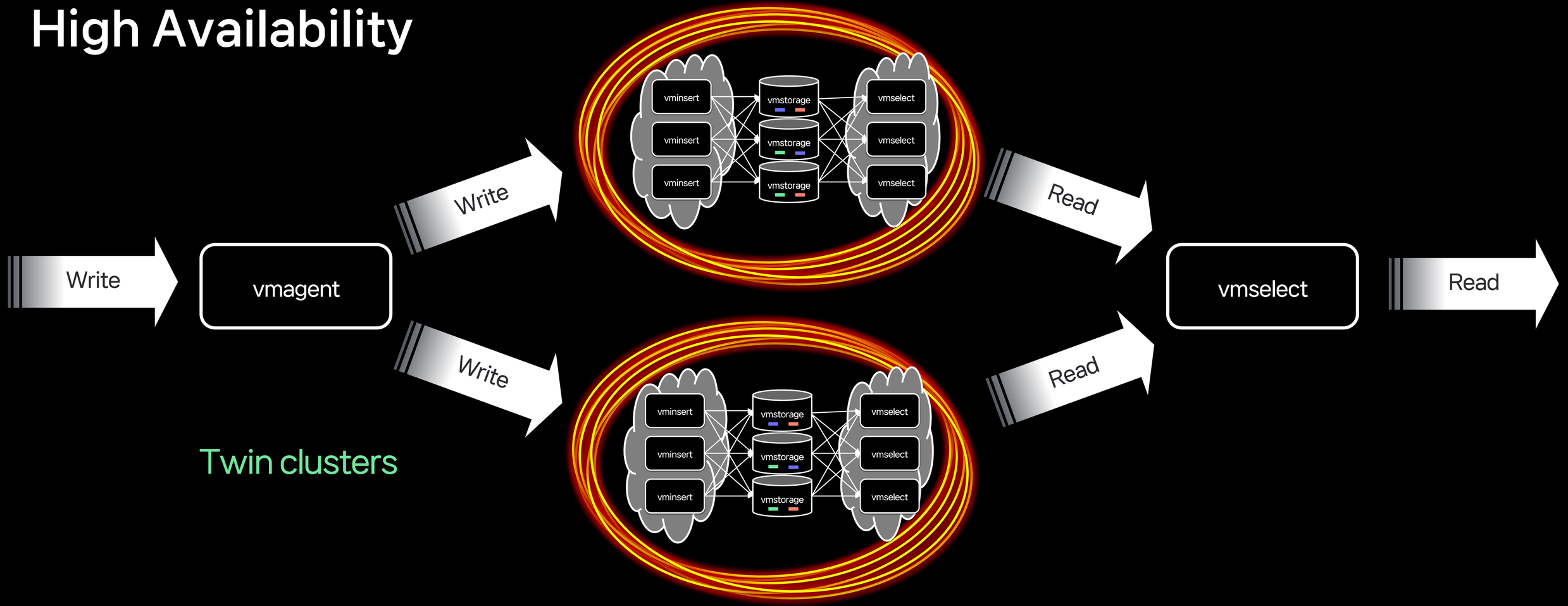
Insufficient resources

due to internal operations

4.3 Multiverse Management

NAVER
DEVIEW
2023

High Availability

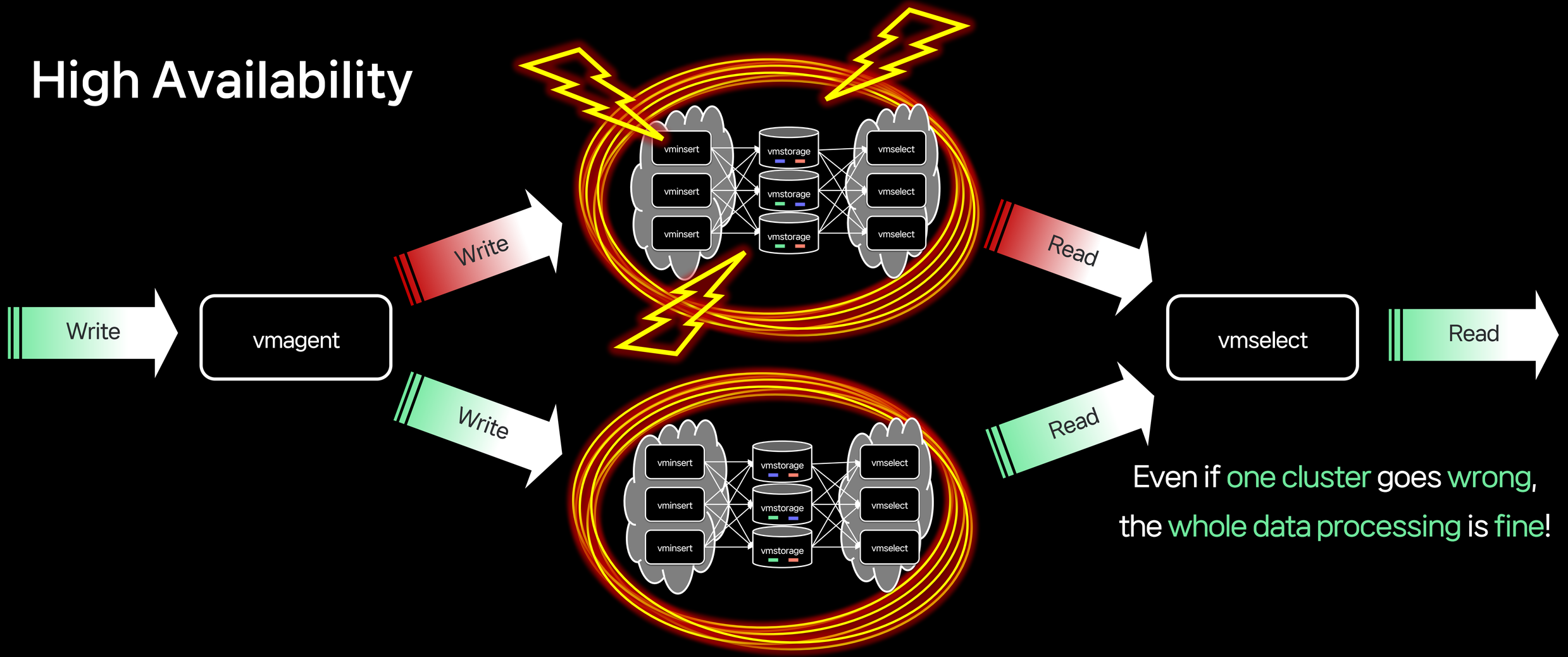


Twin clusters

4.3 Multiverse Management

NAVER
DEVIEW
2023

High Availability

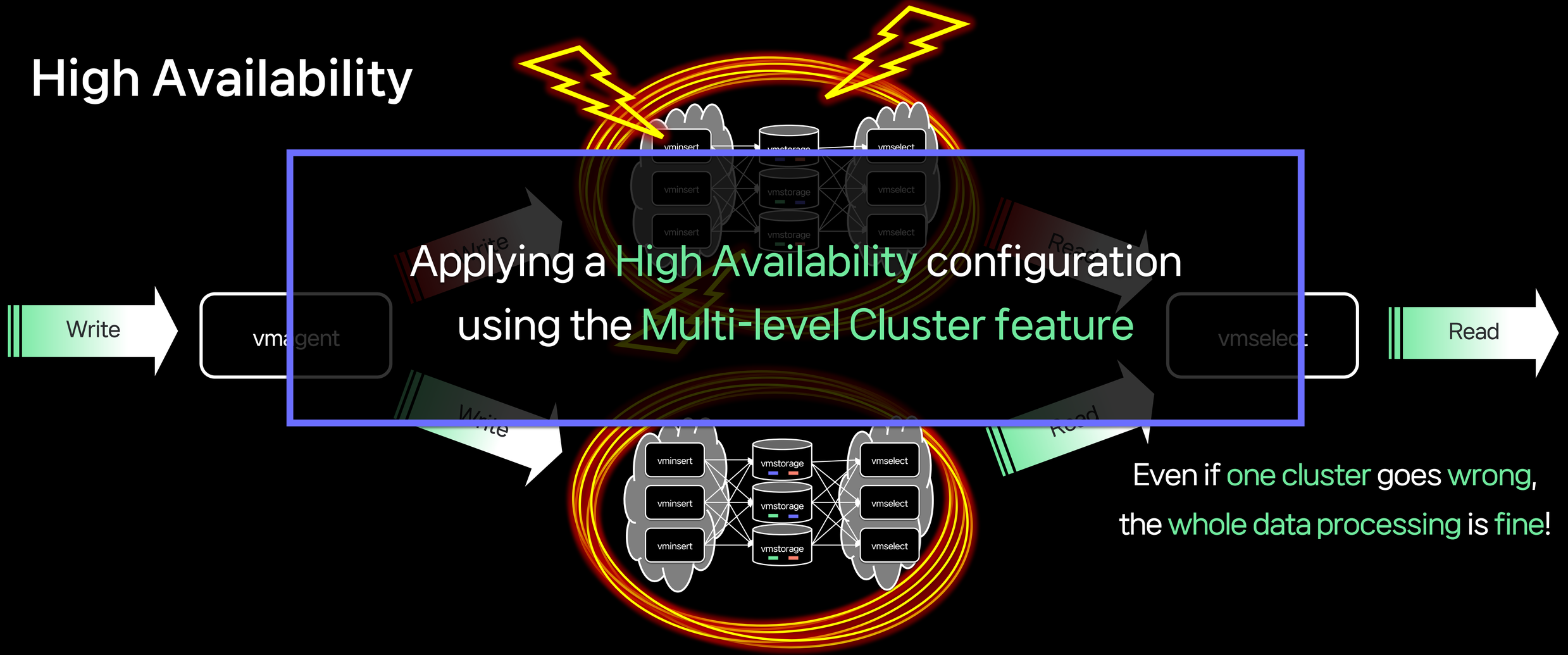


Even if one cluster goes wrong,
the whole data processing is fine!

4.3 Multiverse Management

NAVER
DEVIEW
2023

High Availability

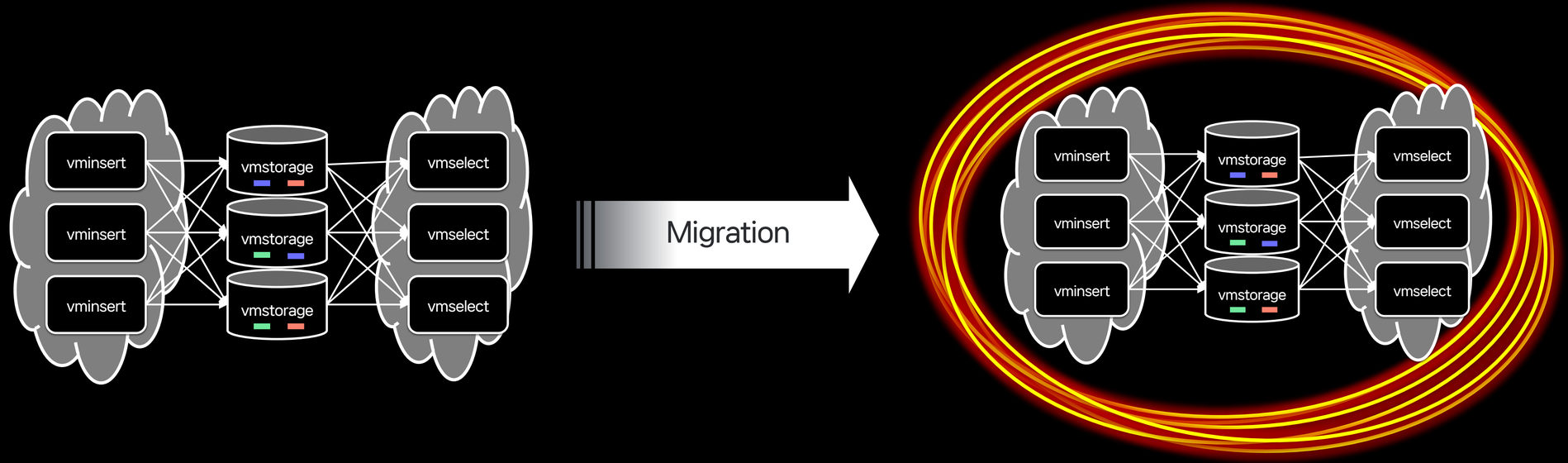


4.3 Multiverse Management

NAVER
DEVIEW
2023

Data Migration

- Data stored in one universe needs to be moved to another universe

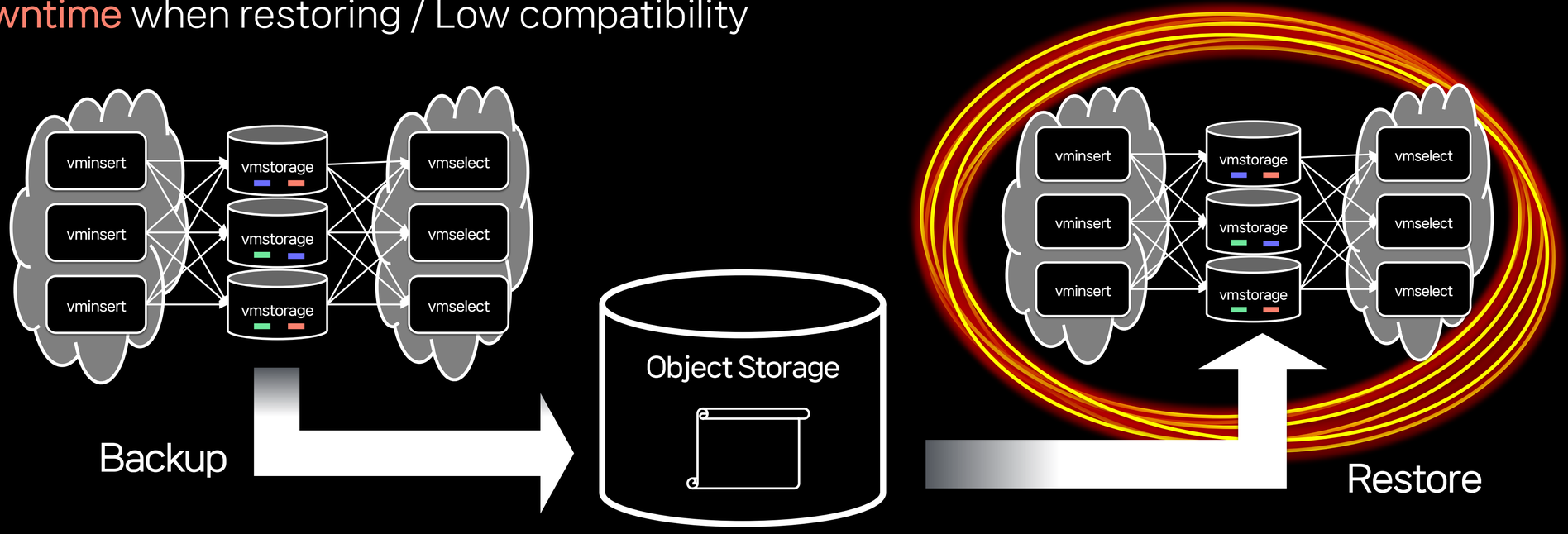


4.3 Multiverse Management

NAVER
DEVIEW
2023

Backup & Restore

- Dumped and pasted compressed data(storage to storage)
- Pros : Fast speed (can transfer as much as network bandwidth allows)
- Cons : Downtime when restoring / Low compatibility

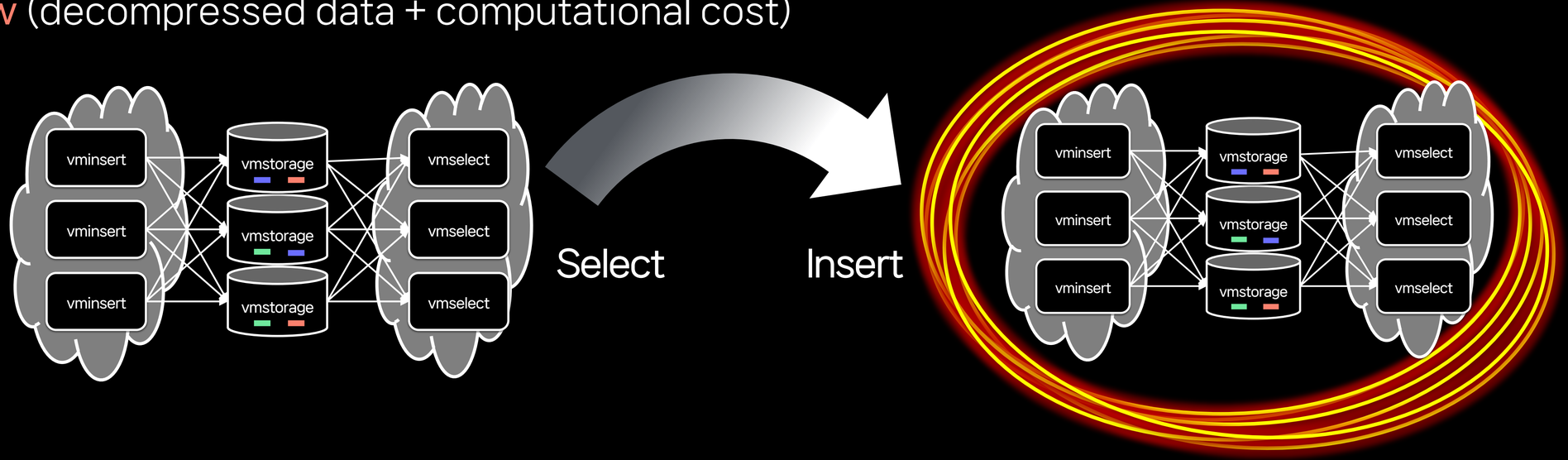


4.3 Multiverse Management

NAVER
DEVIEW
2023

Select & Insert

- Data is first selected from the source cluster and then inserted back into the destination cluster.
- Pros: Great compatibility, Zero downtime
- Cons: Slow (decompressed data + computational cost)



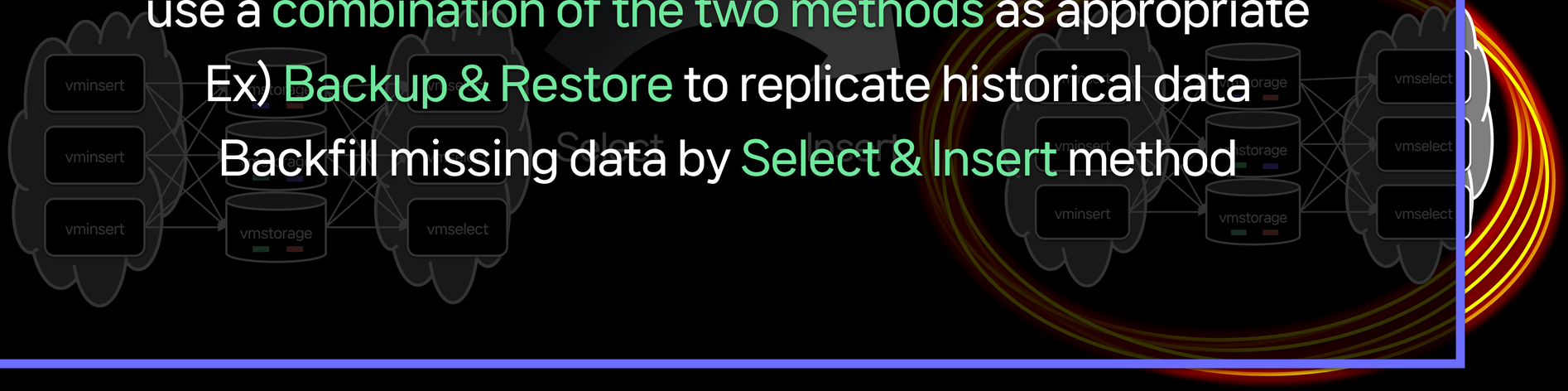
4.3 Multiverse Management

NAVER DEVIEW 2023

Select & Insert

- Data is first selected from the source cluster and then inserted back into the destination cluster.
- Pros: Great compatibility, Zero downtime
- Cons: Slow (decompress/process/compress/write)

There are clear pros and cons, so in practice
use a combination of the two methods as appropriate
Ex) Backup & Restore to replicate historical data
Backfill missing data by Select & Insert method



4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- Minimize metric loss and downtime to serve as a good emergency light

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- Minimize metric loss and downtime to serve as a good emergency light
- Introduce Message Queue and Data Buffering to prevent metrics loss

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- Minimize metric loss and downtime to serve as a good emergency light
- Introduce Message Queue and Data Buffering to prevent metrics loss
- Mitigate heavy queries with a pre-calculator to minimize downtime

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- Minimize metric loss and downtime to serve as a good emergency light
- Introduce Message Queue and Data Buffering to prevent metrics loss
- Mitigate heavy queries with a pre-calculator to minimize downtime
- Introduced Multi-level Cluster feature for high availability

4. Lessons Learned

NAVER
DEVIEW
2023

Summary

- Minimize metric loss and downtime to serve as a good emergency light
- Introduce Message Queue and Data Buffering to prevent metrics loss
- Mitigate heavy queries with a pre-calculator to minimize downtime
- Introduced Multi-level Cluster feature for high availability
- Utilize Data Migration feature for effective Multiverse operation

5. Takeaways

5. Takeaways

NAVER
DEVIEW
2023

Recall

Ref. : The cost of scale in Prometheus ecosystem

Stage	# of Time series	Environment
0	Less than a million	Legacy systems
1	1 million	Distributed systems
2	2 million	Poplular platforms
3	5 million	Custom applications
4	Tens of millions	Microservices
5	Billions	Kubernetes

5. Takeaways

NAVER
DEVIEW
2023

Recall

Ref. : The cost of scale in Prometheus ecosystem

Stage	# of Time series	Environment	Time series DB
0	Less than a million	Legacy systems	Prometheus
1	1 million	Distributed systems	
2	2 million	Poplular platforms	
3	5 million	Custom applications	
4	Tens of millions	Microservices	**VictoriaMetrics Single or Cluster**
5	Billions	Kubernetes	**VictoriaMetrics Cluster**

5. Takeaways

NAVER
DEVIEW
2023

No silver bullet

- VictoriaMetrics actually solves quite a few problems
- However, there is still a lot to think about and operational know-how to keep it running for a long time without downtime and metrics loss, and to handle various scenarios.

Let's think about this together

- Job description (in Korean)
- Contact Us

Q & A

Thank You

Appendix

NAVER
DEVIEW
2023

LSM Tree

- The Secret Sauce Behind NoSQL: LSM Tree
- Log Structured Merge Trees
- Scaling Write-Intensive Key-Value Stores

VictoriaMetrics

- VictoriaMetrics: scaling to 100 million metrics per second
- Specifics of Data Analysis in Time Series Databases
- The cost of scale in Prometheus ecosystem

Appendix

NAVER
DEVIEW
2023

Compare with other products

- VictoriaMetrics VS ScyllaDB
- VictoriaMetrics VS Prometheus
- VictoriaMetrics VS OpenTSDB
- VictoriaMetrics VS Mimir

Etc.

- VictoriaMetrics + Monarch
- VictoriaMetrics + Thanos
- VictoriaMetrics Articles