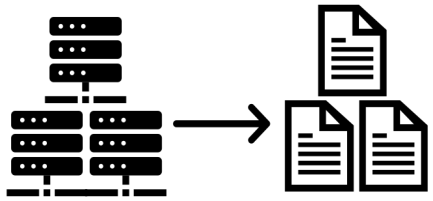
μSlope: High Compression and Fast Search on Semi-Structured Logs

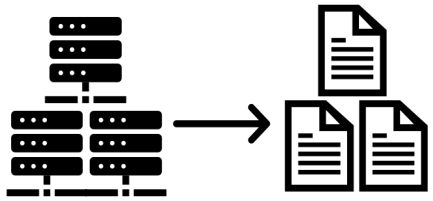
Rui Wang, Devin Gibson, Kirk Rodrigues, Yu Luo,
Yun Zhang, Kaibo Wang, Yupeng Fu, Ting Chen, Ding Yuan

YScope Uber University of Toronto



Software systems

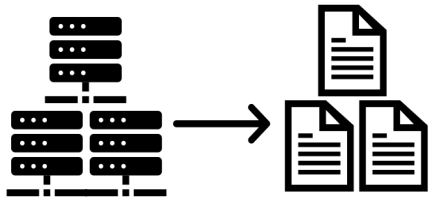
Logs



Software systems

Logs

- Error detection
- Performance debugging
- Operation management
- Usage analysis
- Security



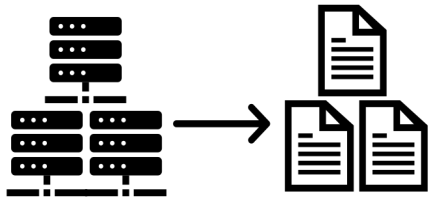
Software systems

Logs

- Error detection
- Performance debugging
- Operation management
- Usage analysis
- Security

```
2022-04-25T00:00:01.000 INFO Task task_12
assigned to container, operation took 0.335
seconds.
```

Unstructured log



Software systems

Logs

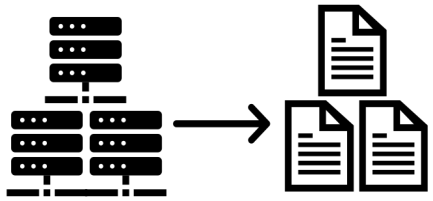
- Error detection
- Performance debugging
- Operation management
- Usage analysis
- Security

```
2022-04-25T00:00:01.000 INFO Task task_12
assigned to container, operation took 0.335
seconds.
```

Unstructured log

```
{
   "timestamp": "2022-04-25T00:00:01.000",
   "level": "INFO",
   "task": "task_12",
   "message": "Task assigned to container",
   "operationDuration": 0.335
}
```

Semi-structured log



Software systems

Logs

- Error detection
- Performance debugging
- Operation management
- Usage analysis
- Security

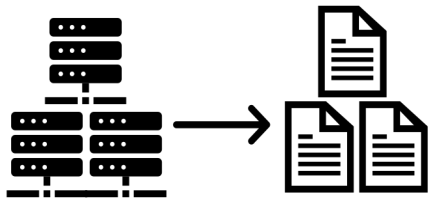
```
2022-04-25T00:00:01.000 INFO Task task_12
assigned to container, operation took 0.335
seconds.
```

Unstructured log

```
{
   "timestamp": "2022-04-25T00:00:01.000",
   "level": "INFO",
   "task": "task_12",
   "message": "Task assigned to container",
   "operationDuration": 0.335
}
```

Semi-structured log

10 PB/day,
or 60PB/week



Software systems

Logs

Error detection

Performance debugging

Operation management

Usage analysis

Security

```
2022-04-25T00:00:01.000 INFO Task task_12
assigned to container, operation took 0.335
seconds.
```

Unstructured log

```
{
   "timestamp": "2022-04-25T00:00:01.000",
   "level": "INFO",
   "task": "task_12",
   "message": "Task assigned to container",
   "operationDuration": 0.335
}
```

Semi-structured log

$0.0125 per GB per month
1 PB logs result in annual storage cost at $35,019,817

10 PB/day,
or 60PB/week

Dynamic schemas

- Semi-structured logs can have different schemas within a dataset

Dynamic schemas

- Semi-structured logs can have different schemas within a dataset

```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}
```

```
{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Dynamic schemas

- Semi-structured logs can have different schemas within a dataset

```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}
```

```
{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Dynamic schemas

- Semi-structured logs can have different schemas within a dataset

```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}
```

```
{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Traditional RDBMS cannot handle semi-structured data

Traditional RDBMS cannot handle semi-structured data

level	message	serviceA.traceId	request	timestamp	serviceB.traceId	serviceB.error
"warn"	"Could not fetch cell for flow 1."	"abc"	"vehicle_compliance"	"2022-04-14T07:58:02.368Z"	NULL	NULL
"error"	"Error handling inbound request"	NULL	NULL	"2022-04-14T07:58:02.372Z"	"xyz"	"application_error"

…

Traditional RDBMS cannot handle semi-structured data

level	message	serviceA .traceId	request	timestamp	serviceB .traceId	serviceB .error
"warn"	"Could not fetch cell for flow 1."	"abc"	"vehicle_compliance"	"2022-04-14T07:58:02.368Z"	NULL	NULL
"error"	"Error handling inbound request"	NULL	NULL	"2022-04-14T07:58:02.372Z"	"xyz"	"application_error"

…

❌ Sparse table

Traditional RDBMS cannot handle semi-structured data

level	message	serviceA .traceId	request	timestamp	serviceB .traceId	serviceB .error
"warn"	"Could not fetch cell for flow 1."	"abc"	"vehicle_compliance"	"2022-04-14T07:58:02.368Z"	NULL	NULL
"error"	"Error handling inbound request"	NULL	NULL	"2022-04-14T07:58:02.372Z"	"xyz"	"application_error"

…

- ❌ Sparse table
- ❌ Predefined schema for each field

Traditional RDBMS cannot handle semi-structured data

level	message	serviceA .traceId	request	timestamp	serviceB .traceId	serviceB .error
"warn"	"Could not fetch cell for flow 1."	"abc"	"vehicle_compliance"	"2022-04-14T07:58:02.368Z"	NULL	NULL
"error"	"Error handling inbound request"	NULL	NULL	"2022-04-14T07:58:02.372Z"	"xyz"	"application_error"

…

- ✗ Sparse table
- ✗ Predefined schema for each field
- ✗ Polymorphism limitation

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

```
{
    "hello": "world"
}
```

JSON log example

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

```
{
    "hello": "world"
}
```

JSON log example

```
\x16\x00\x00\x00           // size (32-bit): 22 bytes
\x02                       // 0x02 = value type String
hello\x00                  // key name
\x06\x00\x00\x00world\x00  // size of value (6 bytes), value
\x00                       // 0x00 = 'end of object'
```

BSON

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

```
{
    "hello": "world"
}
```

JSON log example

```
\x16\x00\x00\x00            // size (32-bit): 22 bytes
\x02                        // 0x02 = value type String
hello\x00                   // key name
\x06\x00\x00\x00world\x00   // size of value (6 bytes), value
\x00                        // 0x00 = 'end of object'
```

BSON

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

```
{
    "hello": "world"
}
```

JSON log example

```
\x16\x00\x00\x00            // size (32-bit): 22 bytes
\x02                        // 0x02 = value type String
hello\x00                   // key name
\x06\x00\x00\x00world\x00   // size of value (6 bytes), value
\x00                        // 0x00 = 'end of object'
```

BSON

- Extra metadata

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

```
{
    "hello": "world"
}
```

JSON log example

```
\x16\x00\x00\x00            // size (32-bit): 22 bytes
\x02                        // 0x02 = value type String
hello\x00                   // key name
\x06\x00\x00\x00world\x00   // size of value (6 bytes), value
\x00                        // 0x00 = 'end of object'
```

BSON

- Extra metadata
- Row-oriented format

Limitations of systems with native JSON support

- E.g. BSON from MongoDB, jsonb from PostgreSQL, and OSON from Oracle

```
{
    "hello": "world"
}
```

JSON log example

```
\x16\x00\x00\x00            // size (32-bit): 22 bytes
\x02                        // 0x02 = value type String
hello\x00                   // key name
\x06\x00\x00\x00world\x00   // size of value (6 bytes), value
\x00                        // 0x00 = 'end of object'
```

BSON

❌ Low compression ratio

- Extra metadata
- Row-oriented format

Index-based search engines: Inefficiencies in ingestion

Index-based search engines: Inefficiencies in ingestion

- They use inverted indexes

Index-based search engines: Inefficiencies in ingestion

splunk> elastic

- They use inverted indexes
- The index size is at the same order of magnitude as the raw data

Index-based search engines: Inefficiencies in ingestion

splunk> elastic

- They use inverted indexes
- The index size is at the same order of magnitude as the raw data

❌ Low compression ratio

Index-based search engines: Inefficiencies in ingestion

splunk> elastic

- They use inverted indexes
- The index size is at the same order of magnitude as the raw data
- Ingestion involves complex processing

❌ Low compression ratio

Index-based search engines: Inefficiencies in ingestion

splunk>

elastic

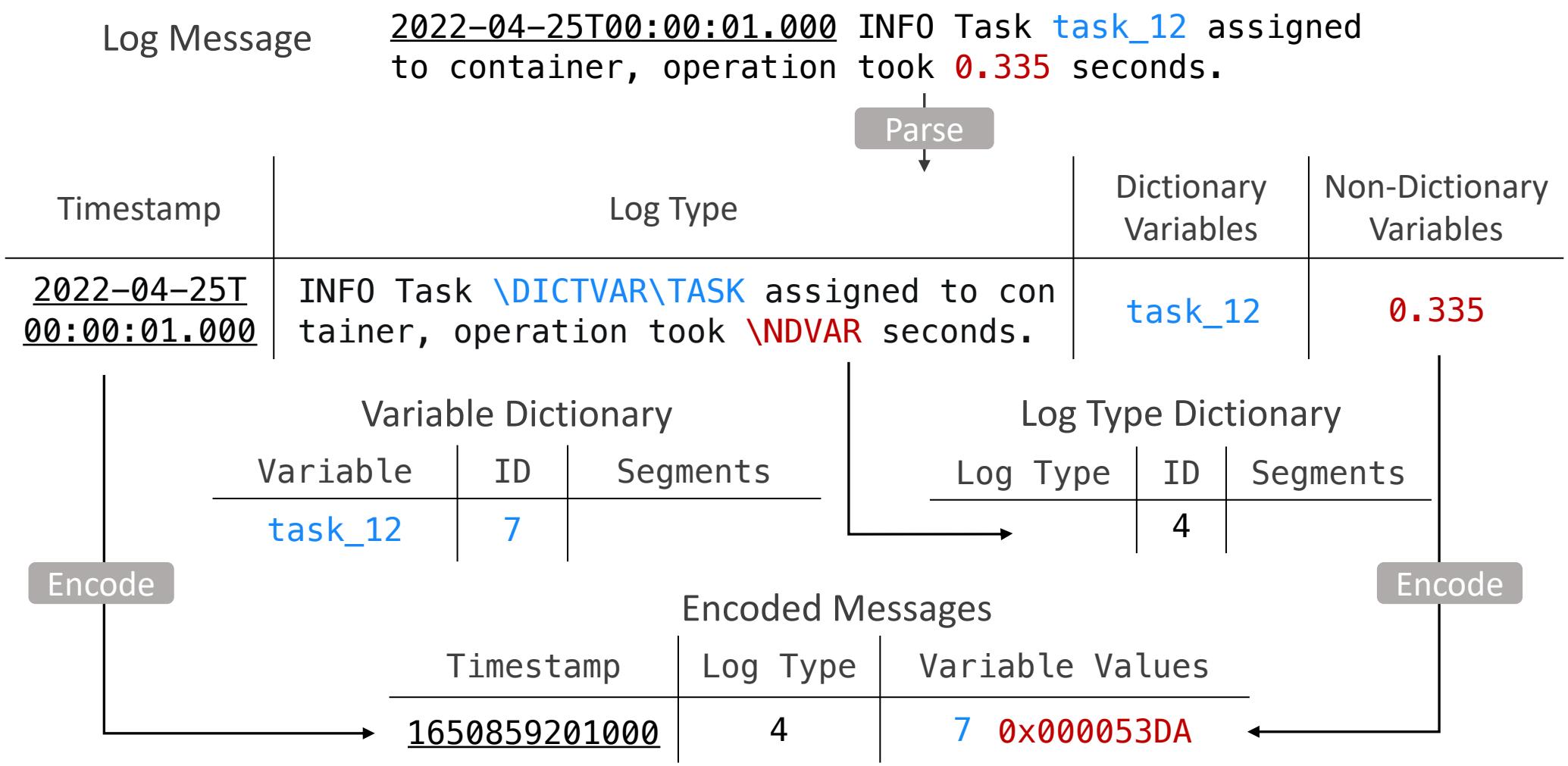
- They use inverted indexes
- The index size is at the same order of magnitude as the raw data
- Ingestion involves complex processing

❌ Low compression ratio

❌ Low ingestion speed

❌ High resource usage

CLP[1]: Deisgned for *unstructured* logs

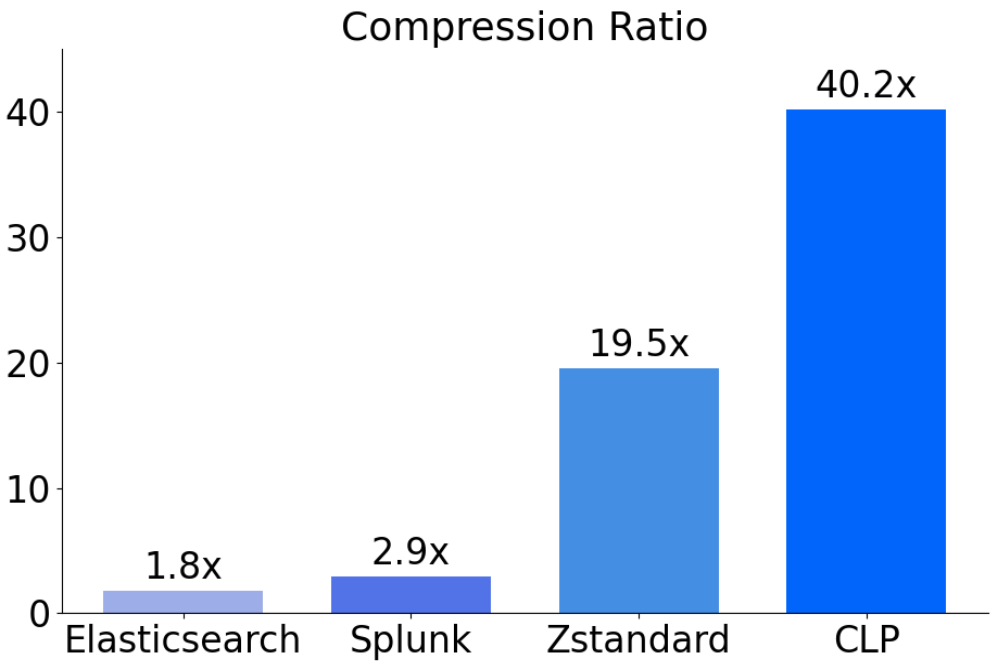


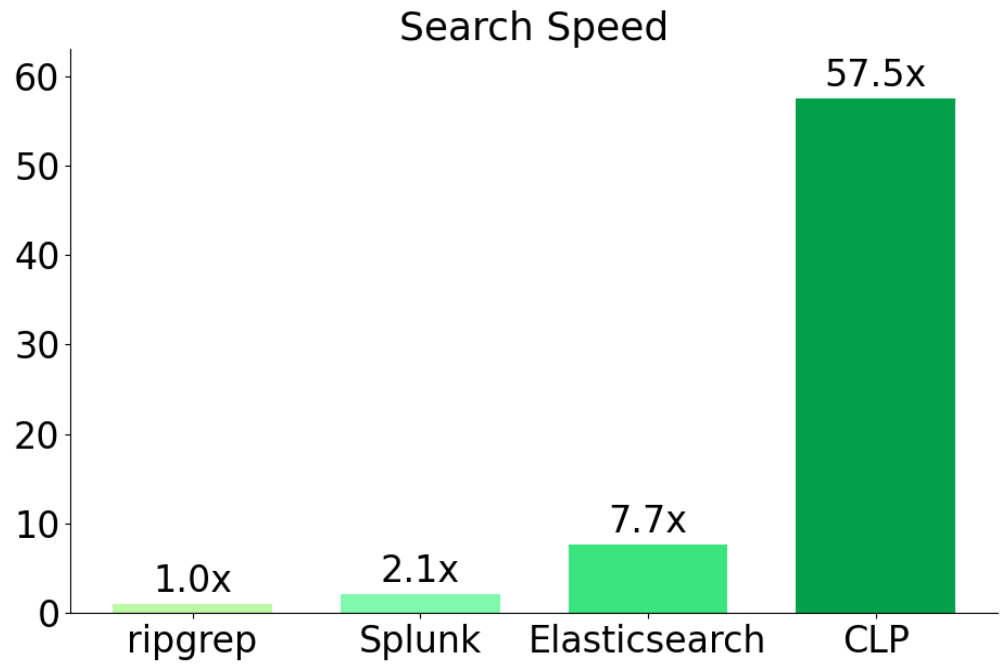
1. Kirk Rodrigues, Yu Luo, and Ding Yuan. “CLP: Efficient and Scalable Search on Compressed Text Logs”. In: 15th USENIX Symposium on Operating Systems Design and Implementation. OSDI ’21. USENIX, 2021, pp. 183–198.

CLP[1]: Deisgned for *unstructured* logs

Log Message

```
2022-04-25T00:00:01.000 INFO Task task_12 assigned
to container, operation took 0.335 seconds.
```





Timestamp	Log Type	Variable Values
1650859201000	4	7 0x000053DA

1. Kirk Rodrigues, Yu Luo, and Ding Yuan. “CLP: Efficient and Scalable Search on Compressed Text Logs”. In: 15th USENIX Symposium on Operating Systems Design and Implementation. OSDI ’21. USENIX, 2021, pp. 183–198.

Challenges of using CLP with semi-structured logs

Challenges of using CLP with semi-structured logs

Search taking lot of time using clg binary #154

Closed **bb-rajakarthik** opened this issue on Aug 29, 2023 · 4 comments

bb-rajakarthik commented on Aug 29, 2023

Bug

We are using CLP for compressing logs generated by our Kubernetes cluster which are in JSON format. A sample log is given below:


```
{
"log_time": "2023-08-29T13:55:09.477456Z",
"stream": "stdout",
"time": "2023-08-29T13:55:09.477456564Z",
"@timestamp": "2023-08-29T19:25:09.477+05:30",
"@Version": "1",
"message": " Method: POST;Root=1-64edf8bd-5c762a676349ee71616bb687 , Request Body : {"orders":
[{"order_type":"normal","external_reference_id":"69426","items":
[{"offset_in_minutes":"721","quantity":"1","external_product_id":"225090"}]}]}",
"level": "INFO",
"level_value": 20000,
"request_id": "6f3f3651-a22b-42a0-b5fe-412d2167c5ca",
"kubernetes_docker_id": "caa9102a169a1495e5790cb2c17cb21d0a279ffc50d802d413938870ba59c7c0",
}
```

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

request: 1_vehicle_compliance

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

```
request: 1_vehicle_compliance
```

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

request: 1_vehicle_compliance

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

```
*.traceId: abc AND request: 1*
```

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

```
*.traceId: abc AND request: 1*
```

```
*{*"traceID"*:*"abc"*}*"request"*:*"1*"*
*"request"*:*"1*"*{*"traceID"*:*"abc"*}*
```

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

```
*.traceId: abc AND request: 1*
```

```
*{*"traceID"*:*"abc"*}*"request"*:*"1*"*
*"request"*:*"1*"*{*"traceID"*:*"abc"*}*
```

NOT, OR operators and numeric comparison ?

Challenges of using CLP with semi-structured logs

ID	Log Type
0	{"level": "warn","message": "Could not fetch cell for flow \DICTVAR","serviceA": {"traceID": "\DICTVAR"},"request": "\DICTVAR","timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}"
1	{"level": "error","message": "Error handling inbound request.","serviceB": {"traceID": "xyz","error": "application_error"},"timestamp": "\DICTVAR:\NDVAR:\DICTVAR"}

Log Type Dictionary

ID	Variable Value
0	1.
1	abc
2	1_vehicle_compliance
3	2022-04-14T07
4	02.368Z
5	02.372Z

Variable Dictionary

Timestamp	Logtype	Variable values
NULL	0	0 1 2 3 58 4
NULL	1	3 58 5

Encoded Messages

`*.traceId: abc AND request: 1*`

`*{*"traceID"*:*"abc"*}*"request"*:*"1*"*`
`*"request"*:*"1*"*{*"traceID"*:*"abc"*}*`

- ❌ Incompatible query interface
- ❌ Poor query performance

NOT, OR operators and numeric comparison ?

μSlope: High compression and *index-less* search

- Automatic schema inference
- Queries on different fields connected by logical operators
- High compression ratio and fast search

Semi-structured **logs** are highly repetitive.

Characterization study

Characterization study

We analyzed

- 21 log datasets from Uber core services and public software
  - Each contains 1 million log records
- 7665 unique top queries performed on Uber logs

Characterization study

We analyzed

- 21 log datasets from Uber core services and public software
  - Each contains 1 million log records
- 7665 unique top queries performed on Uber logs

Key takeaways

- Schemas are dynamic (up to 6176 unique schemas), but also ***repetitive***.
- 71% of the values are variables (single-word strings) and they are highly repetitive.
- 29% of the queries can be completed only by querying the schema structure.

Characterization study

We analyzed

- 21 log datasets from Uber core services and public software
  - Each contains 1 million log records
- 7665 unique top queries performed on Uber logs

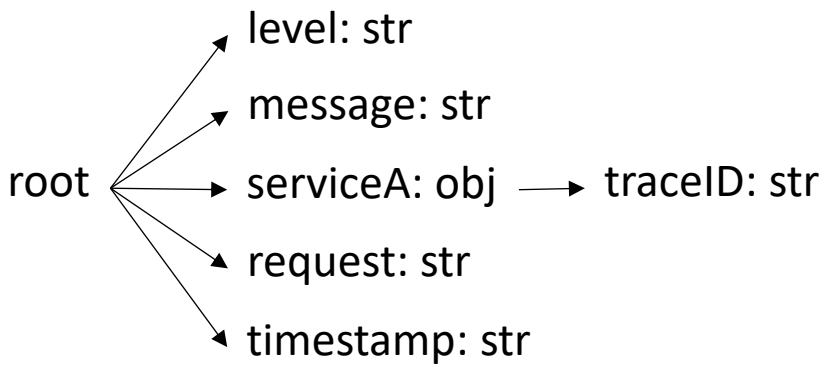
Key takeaways

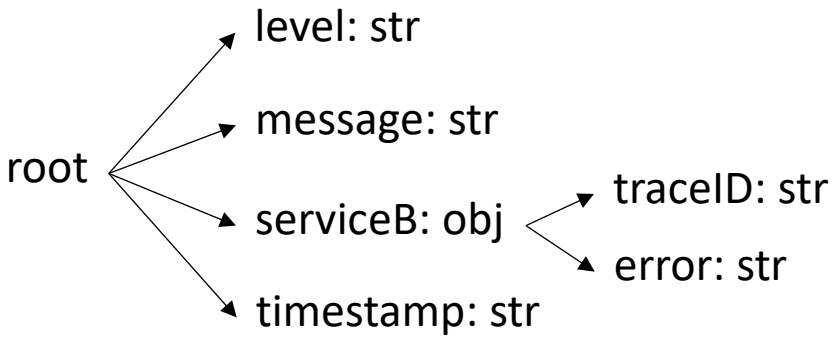
- Schemas are dynamic (up to 6176 unique schemas), but also ***repetitive***.
- 71% of the values are variables (single-word strings) and they are highly repetitive.
- 29% of the queries can be completed only by querying the schema structure.

Find more details in the paper!

Schema Tree and Schema Map

Schema Tree and Schema Map

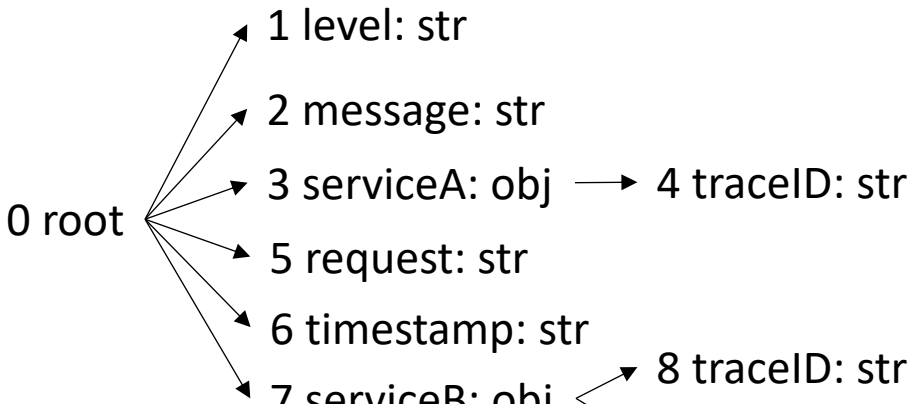





```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}
```

```
{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Schema Tree and Schema Map

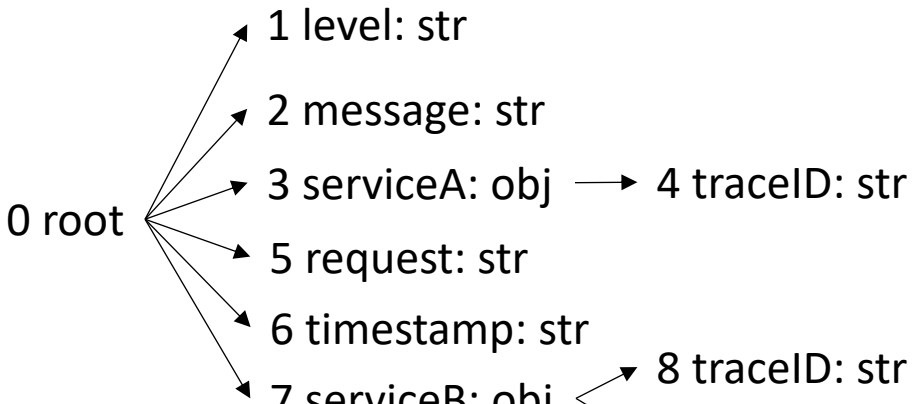


Merged Parsed Tree (MPT)


```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}

{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Schema Tree and Schema Map



Merged Parsed Tree (MPT)

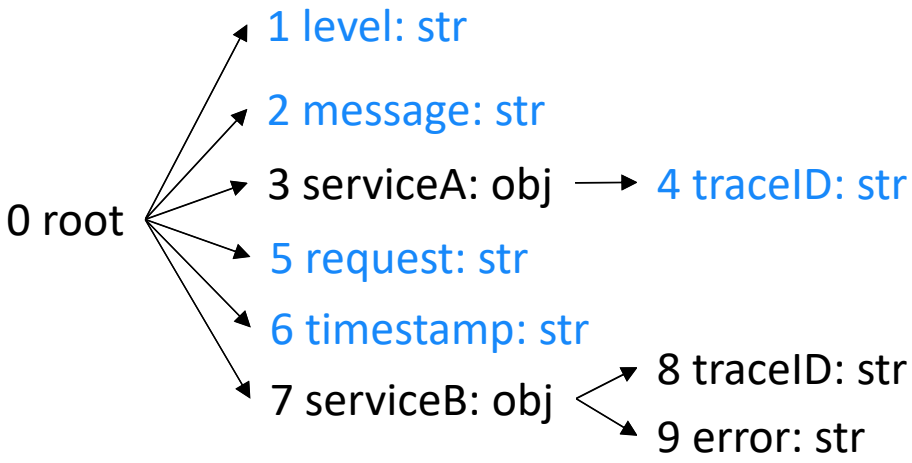
Node IDs	Schema ID

Schema Map

```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}

{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Schema Tree and Schema Map



Merged Parsed Tree (MPT)

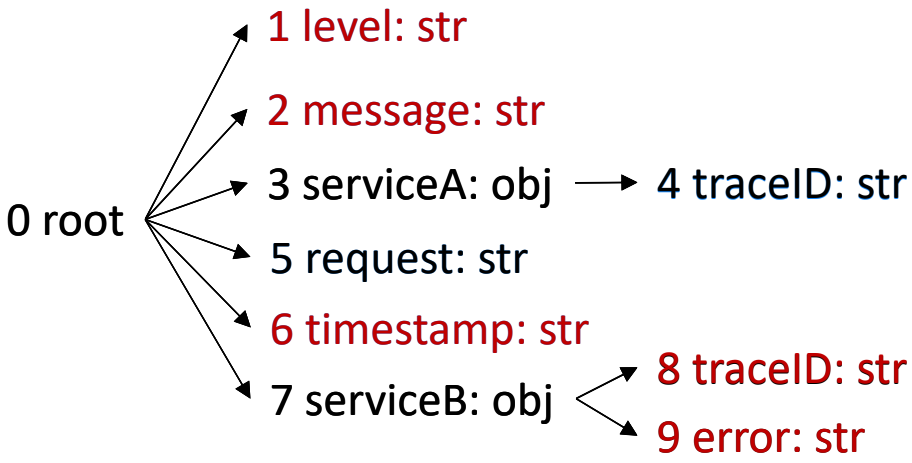
Node IDs	Schema ID
1 2 4 5 6	0

Schema Map

```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}

{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Schema Tree and Schema Map



Merged Parsed Tree (MPT)

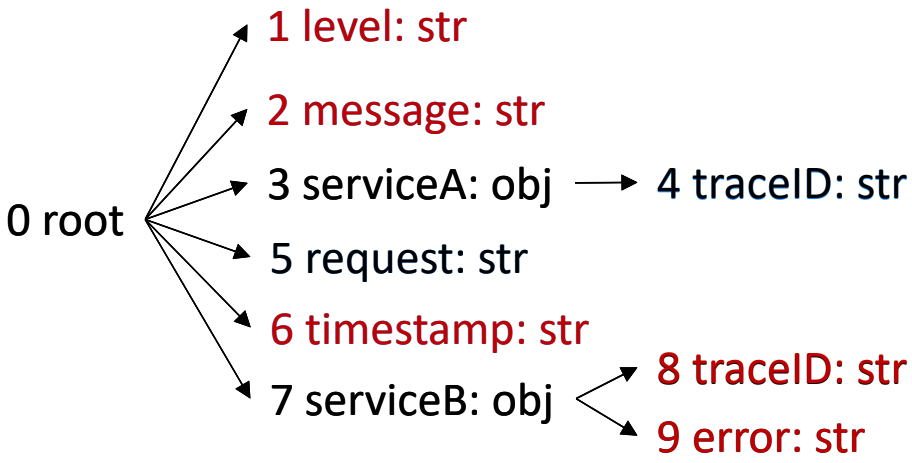
Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

Schema Map


```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}

{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```

Schema Tree and Schema Map



Merged Parsed Tree (MPT)

Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

Schema Map


```
{
    "level": "warn",
    "message": "Could not fetch cell for flow 1.",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "2022-04-14T07:58:02.368Z"
}

{
    "level": "error",
    "message": "Error handling inbound request.",
    "serviceB": {
        "traceID": "xyz",
        "error": "application_error"
    },
    "timestamp": "2022-04-14T07:58:02.372Z"
}
```


- MPT and Schema Map takes up less than 0.001% of the total compressed data size

Data encoding and storage

Data encoding and storage

- Log records are stored in tables partitioned by schemas.

Data encoding and storage

- Log records are stored in tables partitioned by schemas.
- Values are encoded based on their types

Data encoding and storage

- Log records are stored in tables partitioned by schemas.
- Values are encoded based on their types

```
{
    "level": "warn",
    "message": "Could not…",
    "serviceA": {
        "traceID": "abc"
    },
    "request": "1_vehicle_compliance",
    "timestamp": "…368Z"
}
```

```
{
    "level": "error",
    "message": "Error handling…",
    "serviceB": {
        "traceID": "xyz"
        "error": "application_error"
    },
    "timestamp": "…372Z"
}
```

Node ID	1	2	4	5	6
Encoded Vars	warn	Could not…	abc	1_vehicle_compliance	…368Z

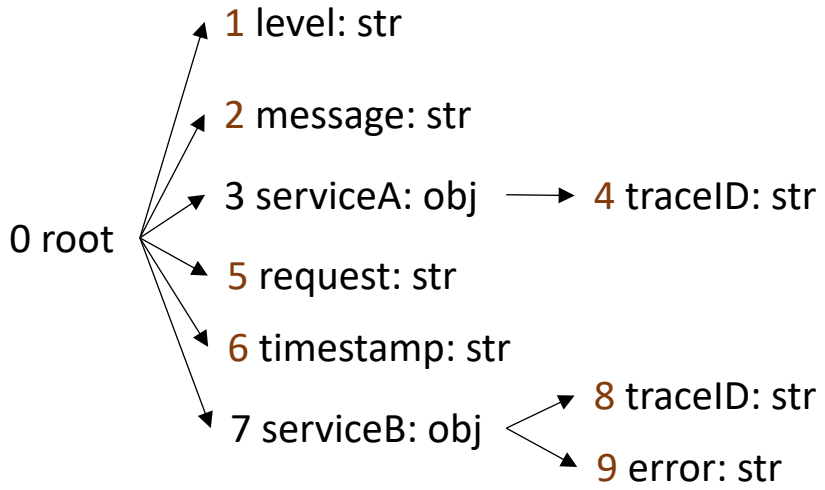
schema[0] Encoded Record Table

Node ID	1	2	6	8	9
Encoded Vars	error	Error…	xyz	application_error	…372Z

schema[1] Encoded Record Table

Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

Schema Map



Merged Parsed Tree

Data encoding and storage

ID	Variable Value
V0	warn
V1	abc
V2	1_vehicle_compliance
V3	error
V4	xyz
V5	application_error

Variable Dictionary

Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

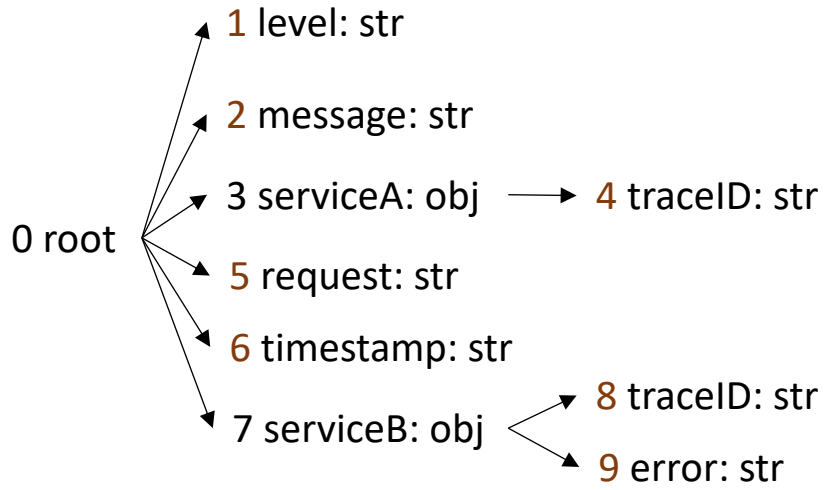
Schema Map

Node ID	1	2	4	5	6
Encoded Vars	V0	Could not…	V1	V2	…368Z

schema[0] Encoded Record Table

Node ID	1	2	6	8	9
Encoded Vars	V3	Error…	V4	V5	…372Z

schema[1] Encoded Record Table



Merged Parsed Tree

Data encoding and storage

ID	Variable Value
V0	warn
V1	abc
V2	1_vehicle_compliance
V3	error
V4	xyz
V5	application_error

Variable Dictionary

ID	Log Type
L0	Could not fetch cell for flow \INT.
L1	Error handling inbound request.

Log Type Dictionary

Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

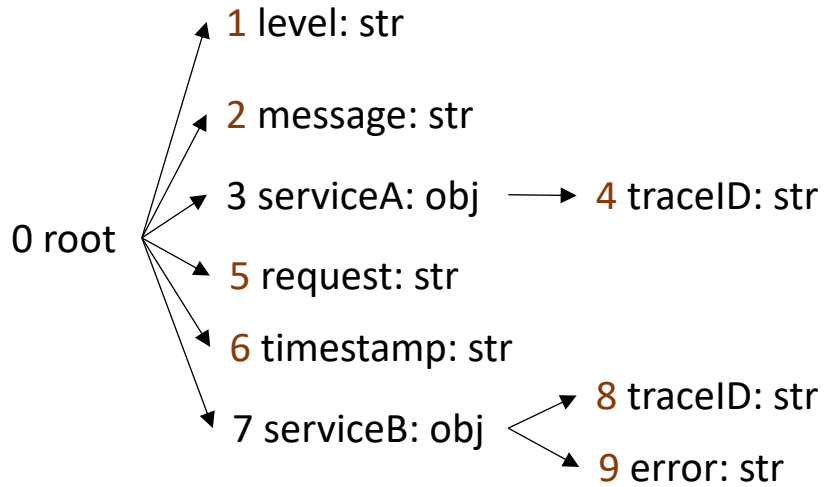
Schema Map

Node ID	1	2		4	5	6
Encoded Vars	V0	L0	1	V1	V2	…368Z

schema[0] Encoded Record Table

Node ID	1	2		6	8	9
Encoded Vars	V3	L1		V4	V5	…372Z

schema[1] Encoded Record Table



Merged Parsed Tree

Data encoding and storage

ID	Variable Value
V0	warn
V1	abc
V2	1_vehicle_compliance
V3	error
V4	xyz
V5	application_error

Variable Dictionary

ID	Log Type
L0	Could not fetch cell for flow \INT.
L1	Error handling inbound request.

Log Type Dictionary

ID	Format
T0	yyyy-MM-dd'T'HH:mm:ss'.'SSS'Z'

Timestamp Dictionary

Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

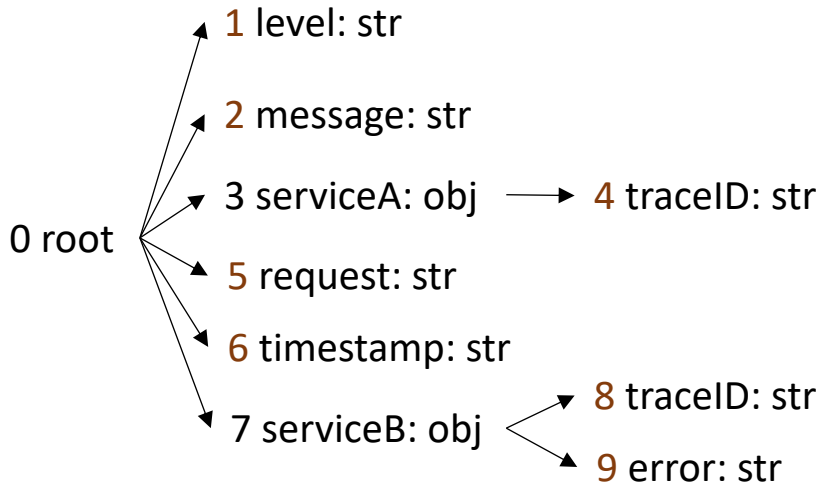
Schema Map

Node ID	1	2		4	5	6	
Encoded Vars	V0	L0	1	V1	V2	T0	…368

schema[0] Encoded Record Table

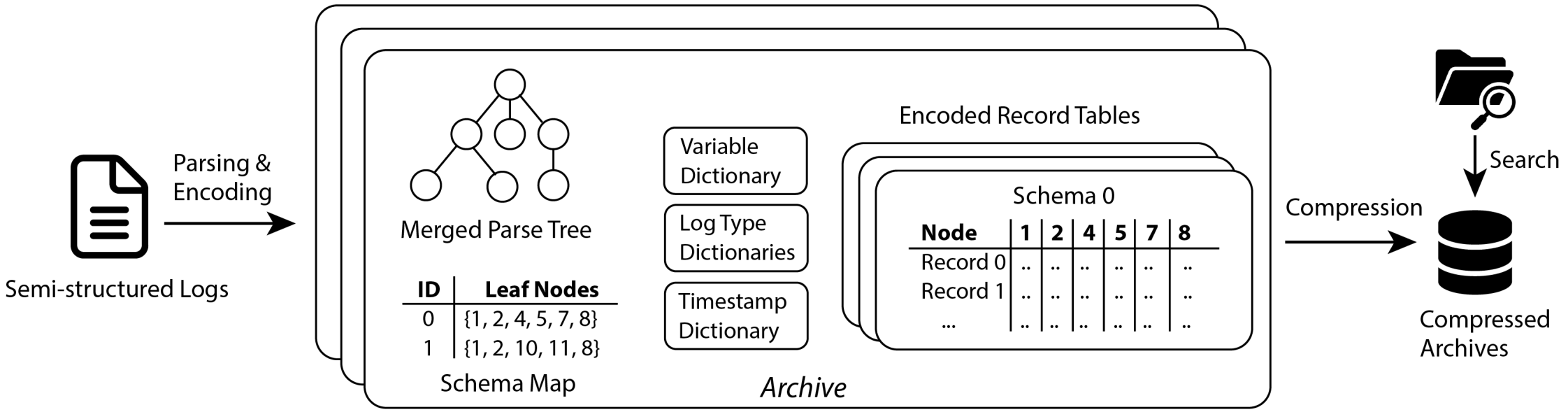
Node ID	1	2		6	8	9	
Encoded Vars	V3	L1		V4	V5	T0	…372

schema[1] Encoded Record Table

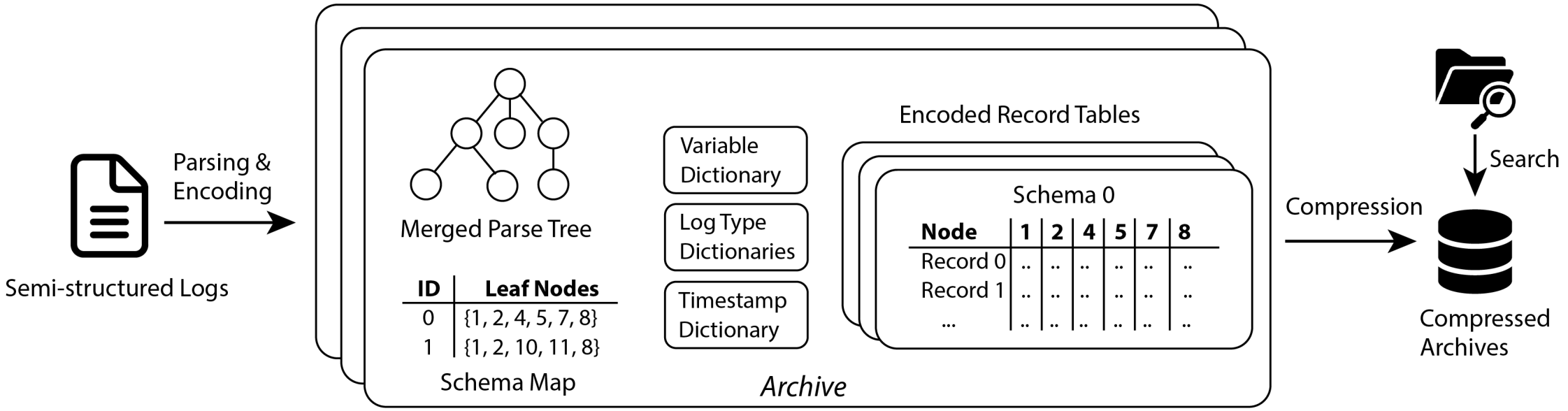


Merged Parsed Tree

μSlope architecture

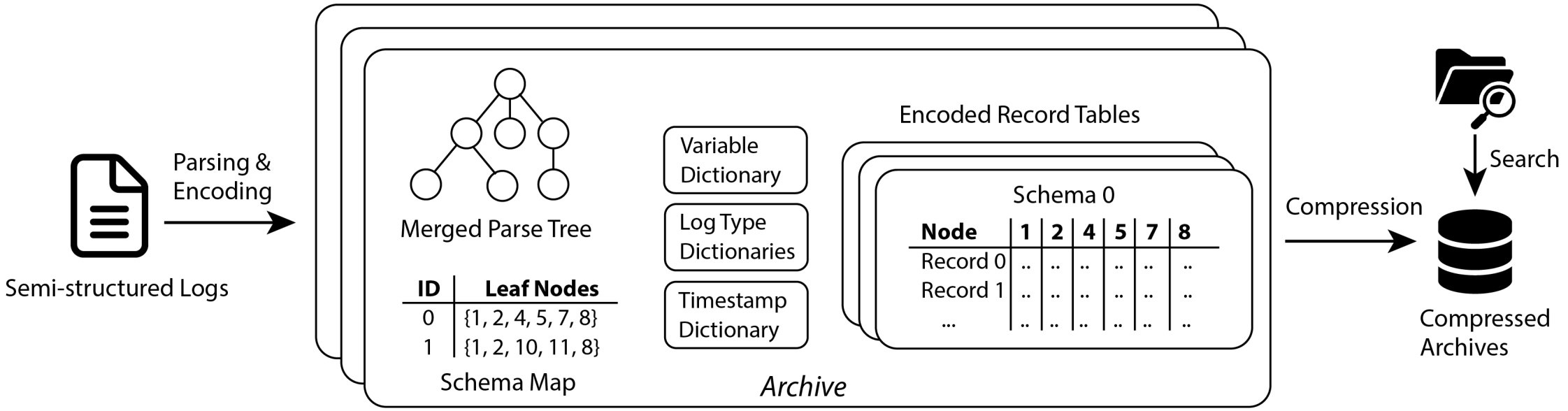


μSlope architecture



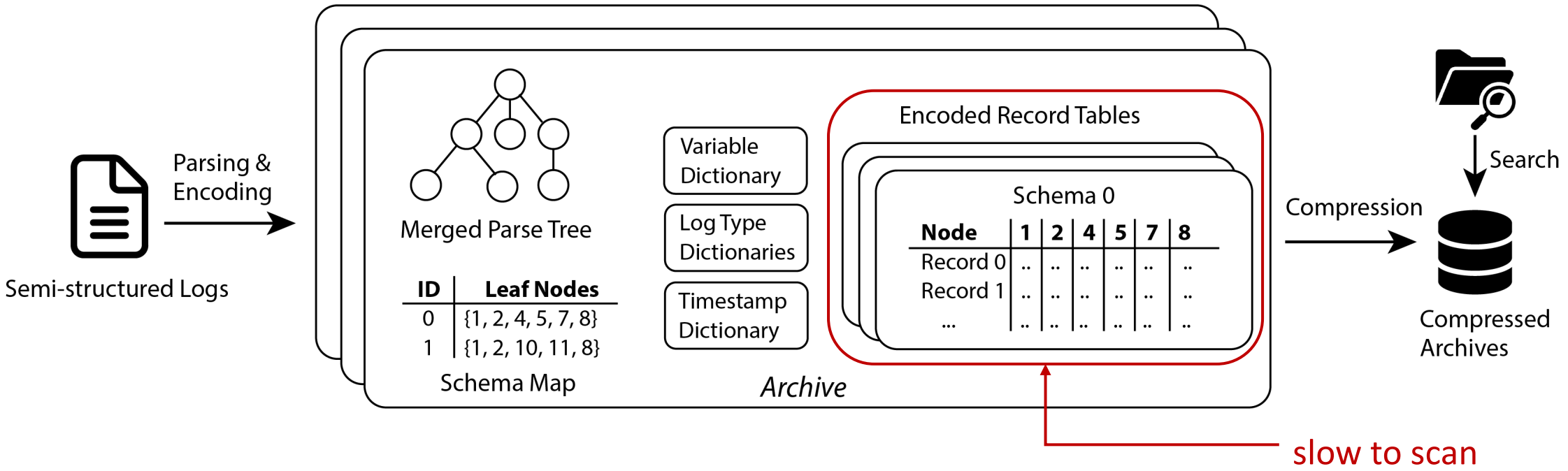
- Data is compressed into archives

μSlope architecture



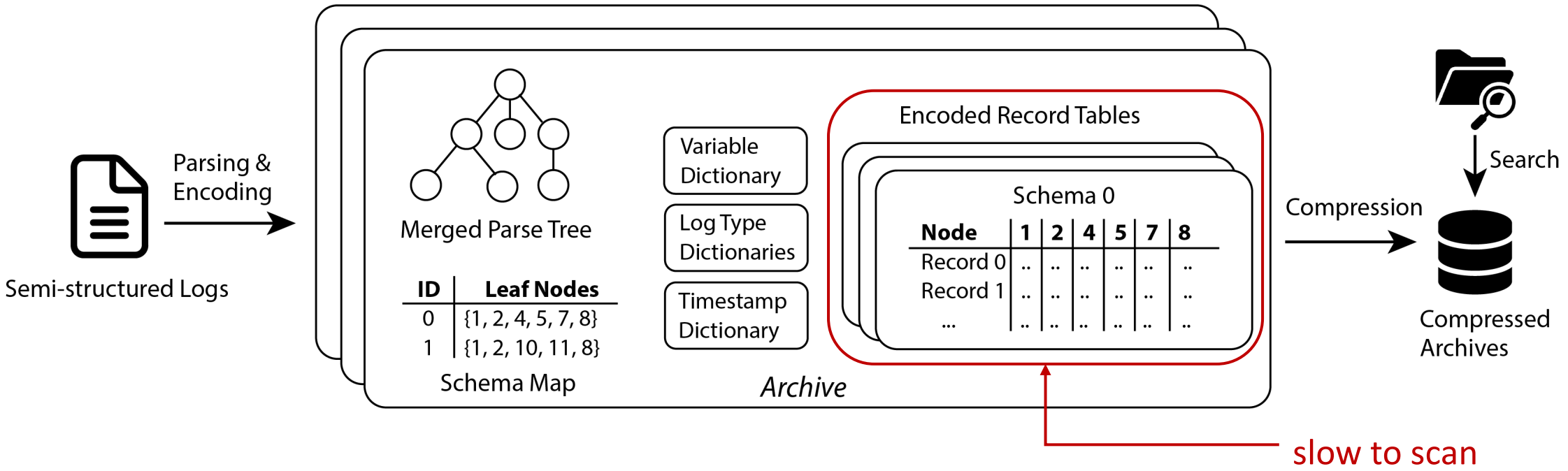
- Data is compressed into archives
- Archives are independently searched and compressed

μSlope architecture



- Data is compressed into archives
- Archives are independently searched and compressed

μSlope architecture



- Data is compressed into archives
- Archives are independently searched and compressed
- Schema metadata and dictionary lookup speed up search

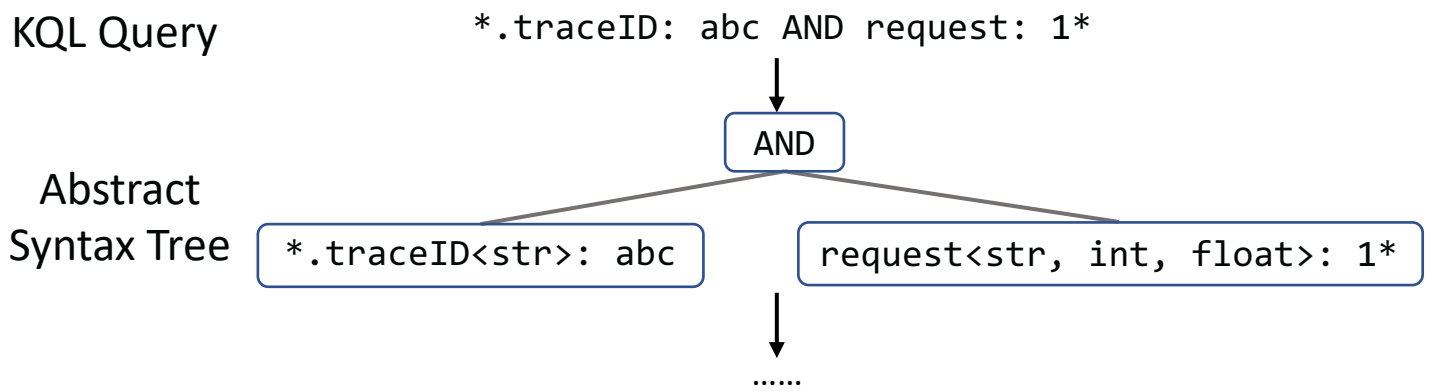
Search

Search

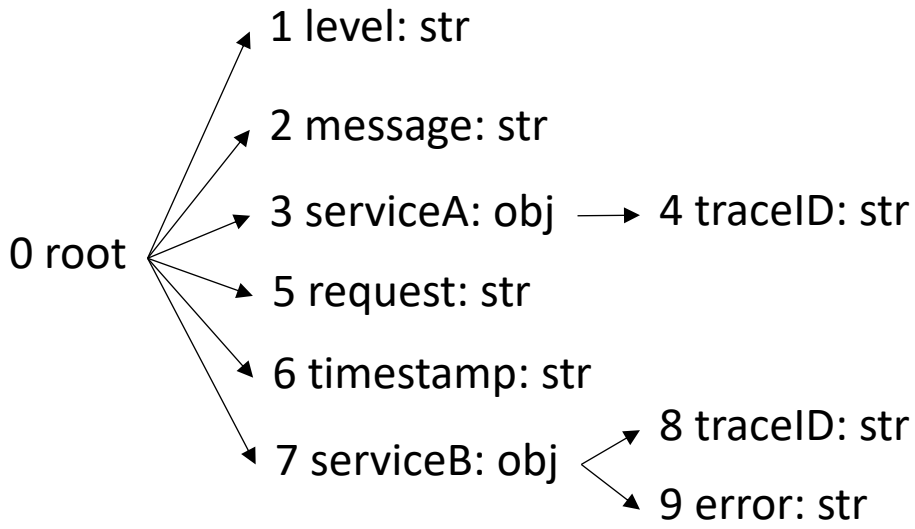
KQL Query

```
*.traceID: abc AND request: 1*
```

Search


KQL Query
.traceID: abc AND request: 1
Abstract
Syntax Tree
AND
*.traceID<str>: abc
request<str, int, float>: 1*
……

Search



Merged Parsed Tree

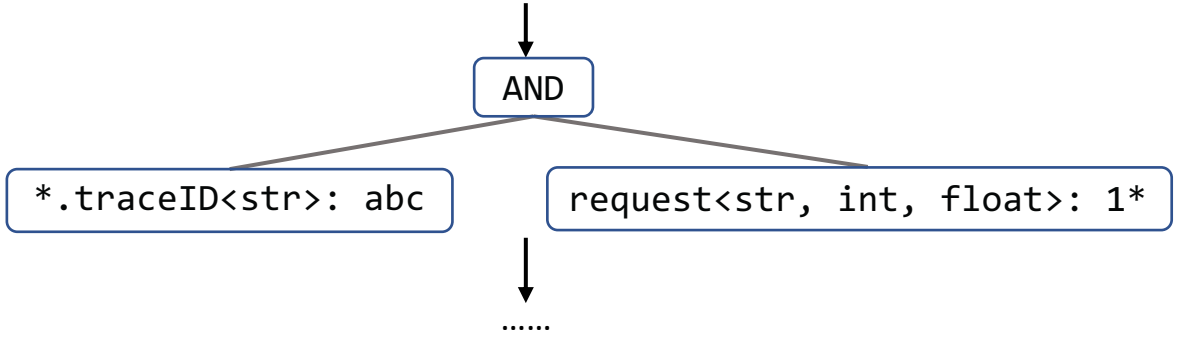
Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

Schema Map

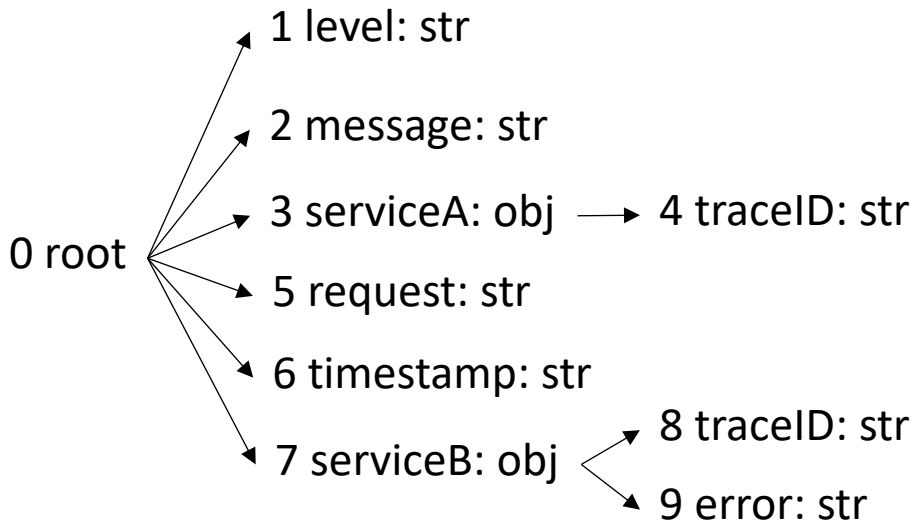
KQL Query

```
*.traceID: abc AND request: 1*
```

Abstract Syntax Tree



Search



Merged Parsed Tree

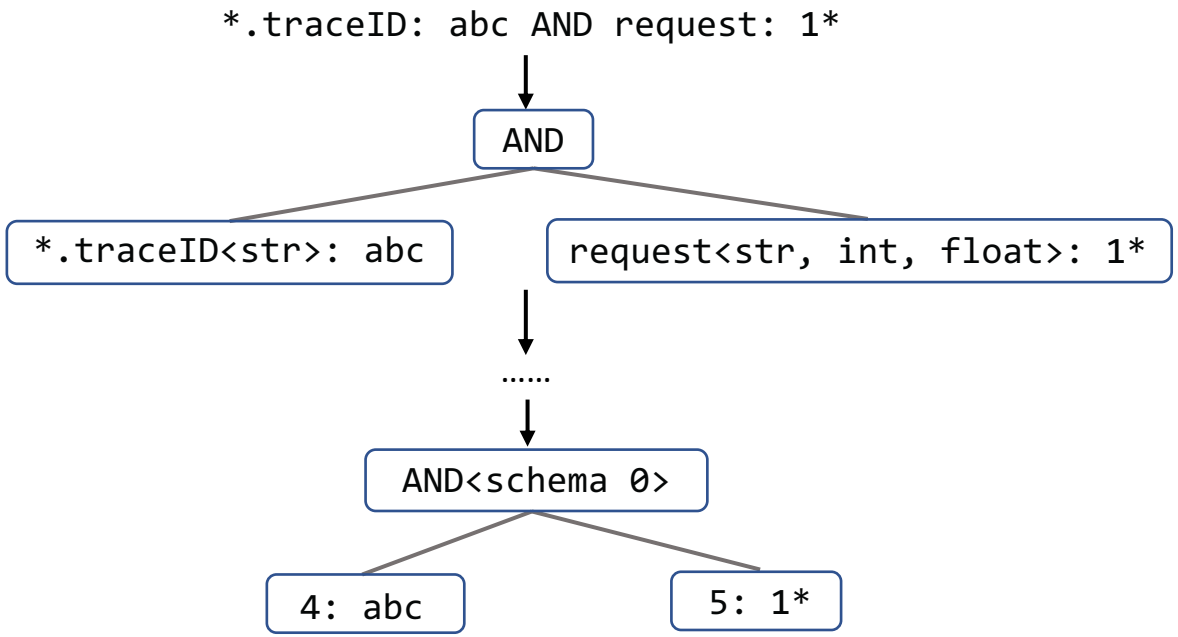
Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

Schema Map

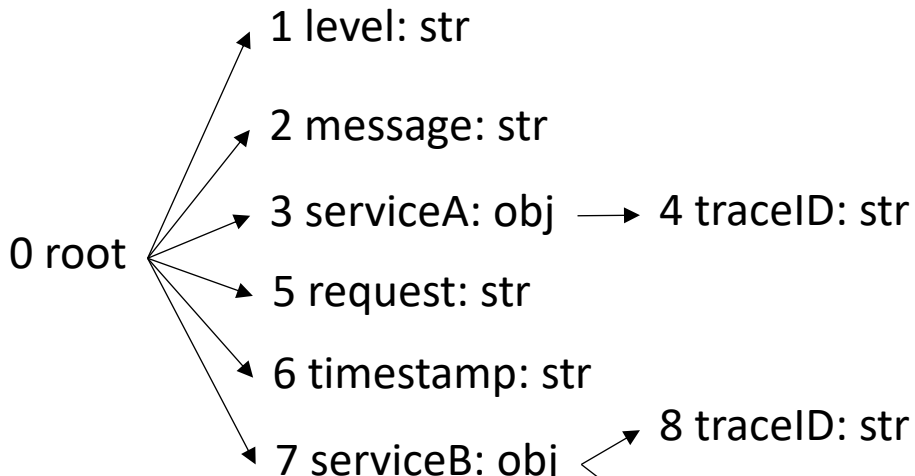
KQL Query

Abstract Syntax Tree

Query Plan Tree



Search



Merged Parsed Tree

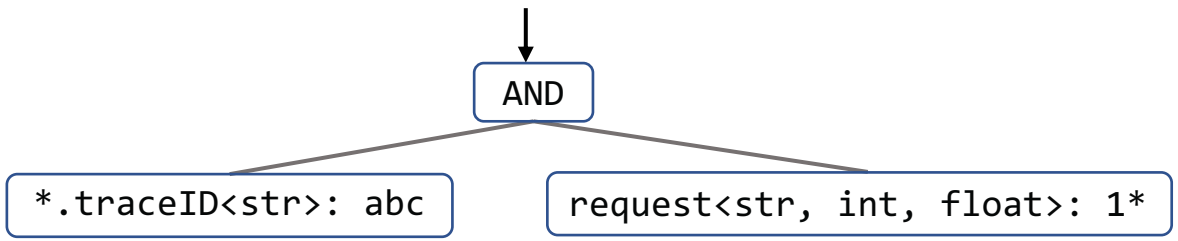
Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

Schema Map

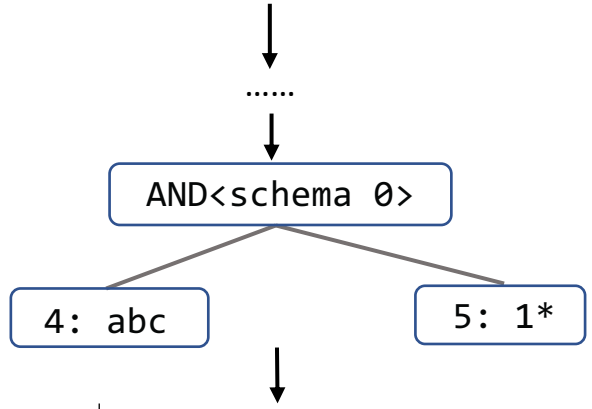
KQL Query

.traceID: abc AND request: 1

Abstract Syntax Tree



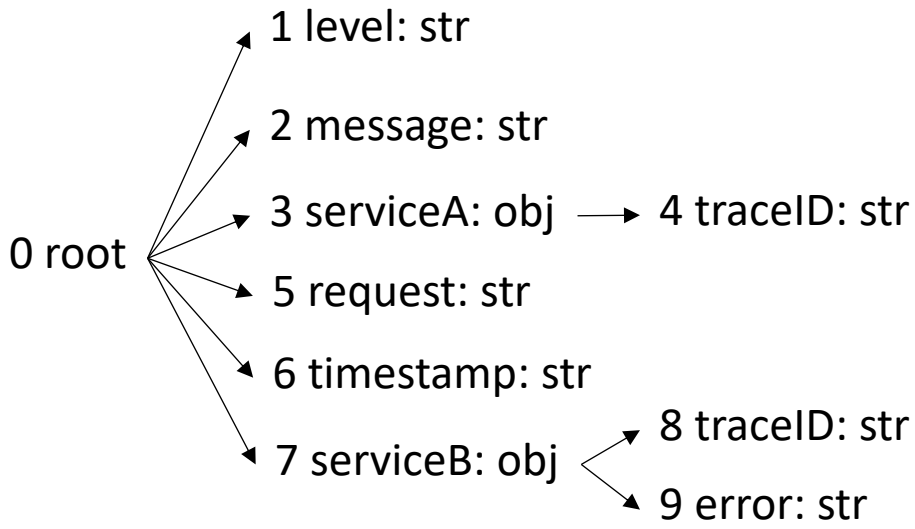
Query Plan Tree



Search on Strings

ID	Variable Value
…	…
V1	abc
V2	1_vehicle_compliance
…	…

Search



Merged Parsed Tree

Node IDs	Schema ID
1 2 4 5 6	0
1 2 6 8 9	1

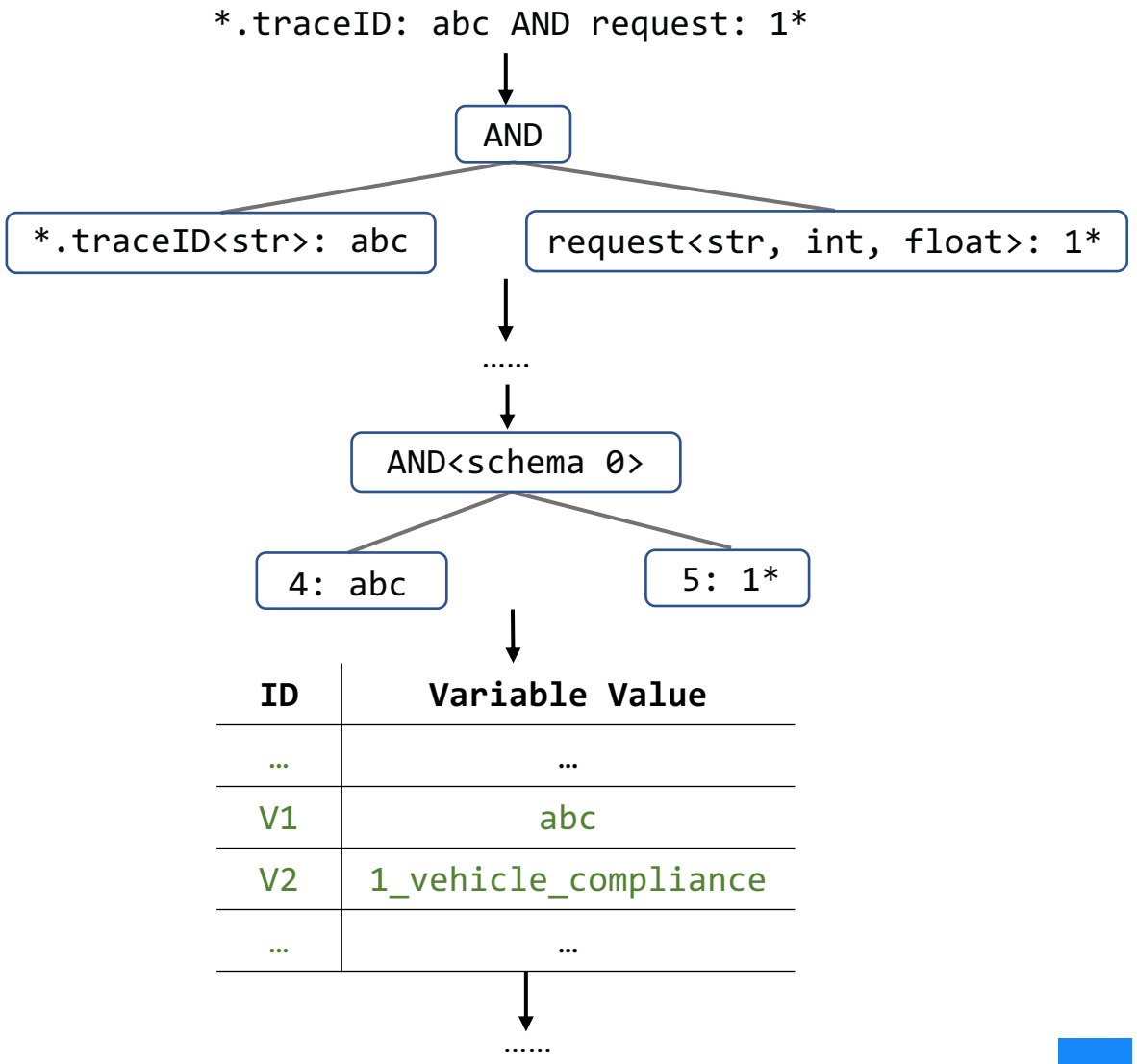
Schema Map

KQL Query

Abstract Syntax Tree

Query Plan Tree

Search on Strings



Evaluation

Evaluation

compression ratio

search performance

Evaluation

compression ratio

search performance

- In single-threaded setup and parallel setup

Evaluation

compression ratio

search performance

- In single-threaded setup and parallel setup
- 21 JSON datasets for compression
  - 16 from Uber (30.0GB to 102.9GB)
  - 5 from public software (392.8MB to 64.8GB)

	Name	Uncompressed Size	Number of Records
Uber Logs	LogA	30.0GB	22,996,492
	LogB	47.1GB	16,606,964
	LogC	60.4GB	15,306,125
	LogD	50.7GB	58,309,754
	LogE	91.8GB	22,345,071
	LogF	102.9GB	17,251,752
	LogG	30.9GB	3,046,845
	LogH	30.8GB	11,461,221
	LogI	39.7GB	27,209,375
	LogJ	36.0GB	13,605,274
	LogK	30.2GB	57,919,224
	LogL	37.1GB	45,827,554
	LogM	36.5GB	42,206,452
	LogN	38.0GB	22,307,407
	LogO	38.6GB	4,438,786
	LogP	38.3GB	34,840,347
Public Logs	Spark	2.0GB	1,011,651
	MongoDB	64.8GB	186,287,600
	CockroachDB	9.8GB	16,520,377
	elasticsearch	8.0GB	140,012,234
	PostgreSQL	392.8MB	1,000,000

Evaluation

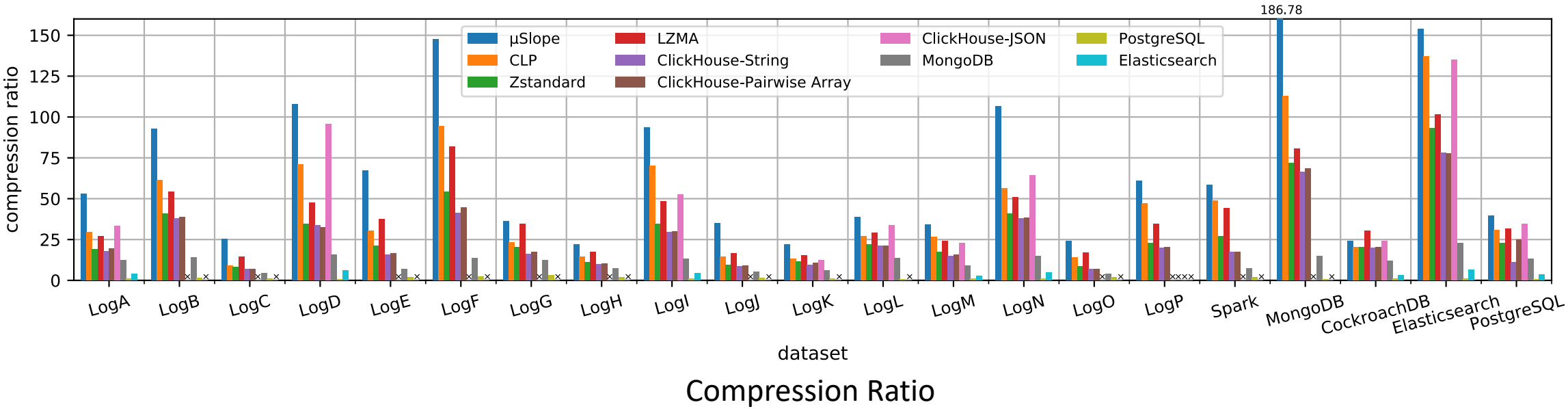
compression ratio

search performance

- In single-threaded setup and parallel setup
- 21 JSON datasets for compression
  - 16 from Uber (30.0GB to 102.9GB)
  - 5 from public software (392.8MB to 64.8GB)
- 15 queries on 3 datasets

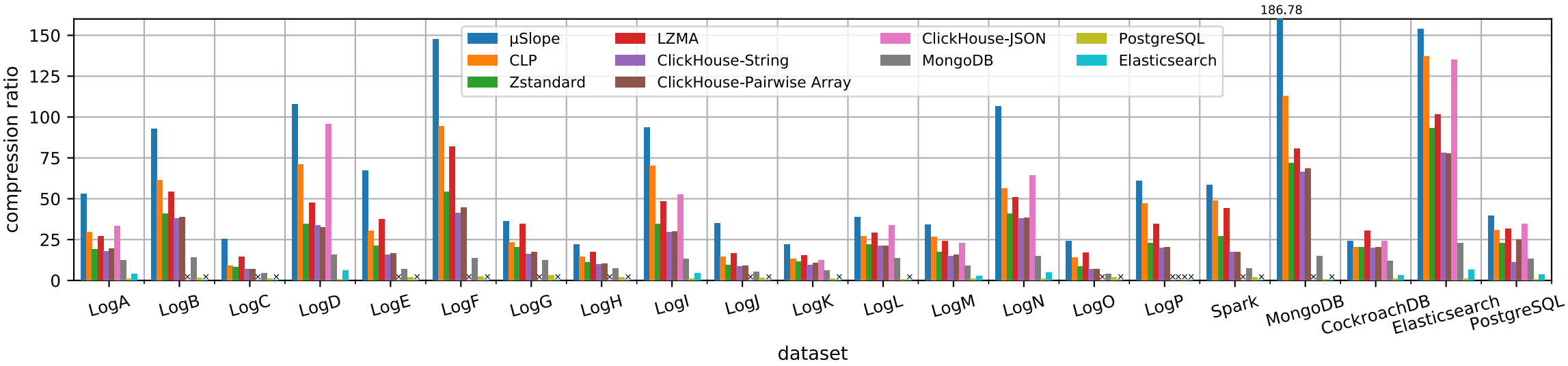
	Name	Uncompressed Size	Number of Records
Uber Logs	LogA	30.0GB	22,996,492
	LogB	47.1GB	16,606,964
	LogC	60.4GB	15,306,125
	LogD	50.7GB	58,309,754
	LogE	91.8GB	22,345,071
	LogF	102.9GB	17,251,752
	LogG	30.9GB	3,046,845
	LogH	30.8GB	11,461,221
	LogI	39.7GB	27,209,375
	LogJ	36.0GB	13,605,274
	LogK	30.2GB	57,919,224
	LogL	37.1GB	45,827,554
	LogM	36.5GB	42,206,452
	LogN	38.0GB	22,307,407
	LogO	38.6GB	4,438,786
	LogP	38.3GB	34,840,347
Public Logs	Spark	2.0GB	1,011,651
	MongoDB	64.8GB	186,287,600
	CockroachDB	9.8GB	16,520,377
	elasticsearch	8.0GB	140,012,234
	PostgreSQL	392.8MB	1,000,000

Compression performance



Compression Ratio

Compression performance



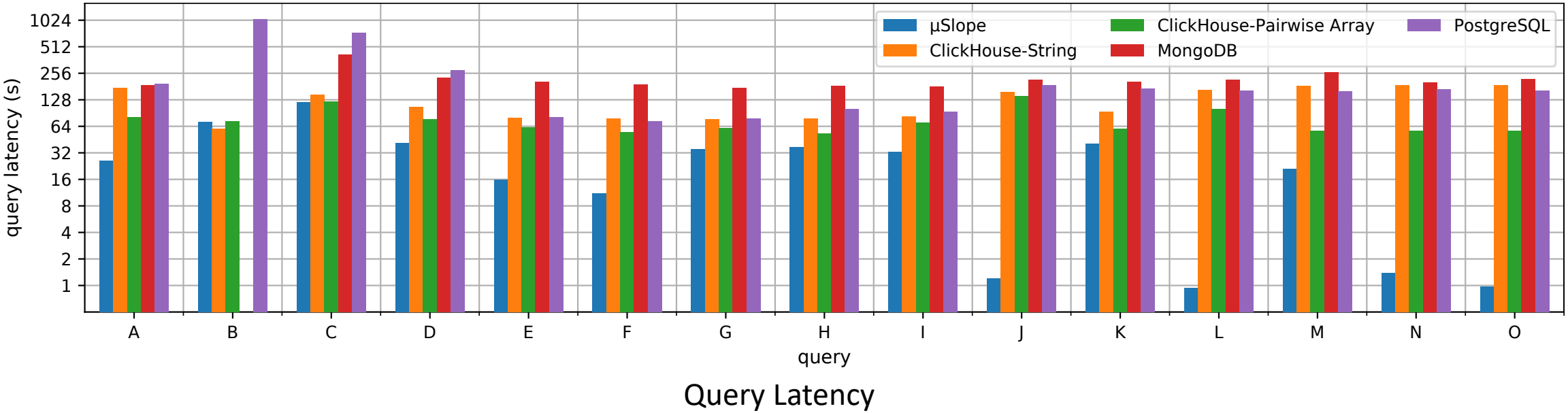
Compression Ratio

The average compression ratio of μSlope is **68.1:1**

- 1.5x of CLP's
- 1.7x of LZMA's
- 2.3x of Zstandard's
- 6.1x of MongoDB's
- 15.7x of Elasticsearch's

Query performance

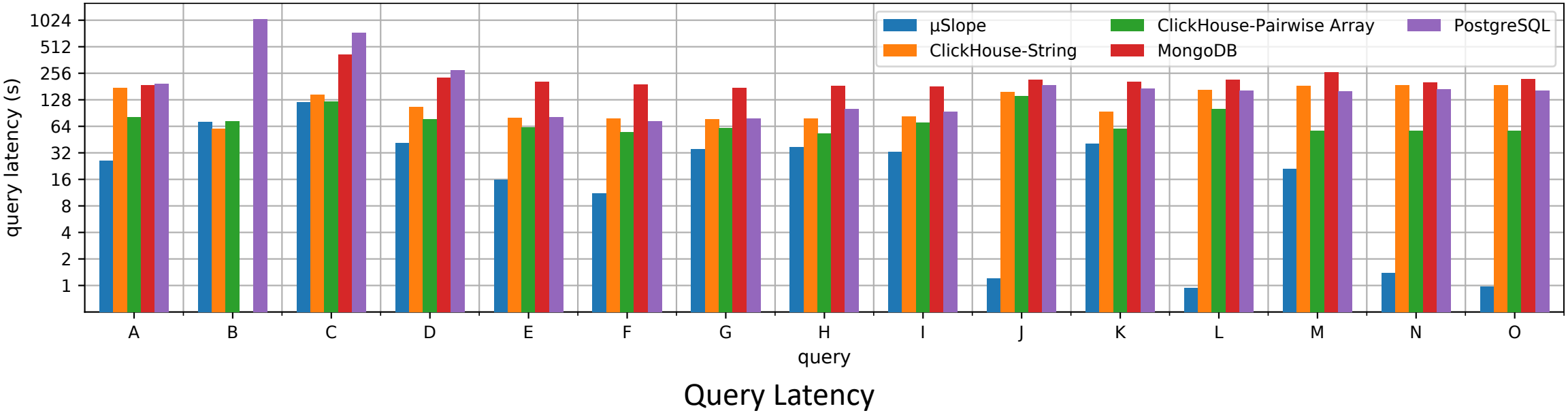
- 9 queries from Uber and 6 from public software



Query Latency

Query performance

- 9 queries from Uber and 6 from public software



Query Latency

μSlope is
- 2.5x faster than the fastest setup of ClickHouse
- 6.7x faster than MongoDB
- 8.1x faster than PostgreSQL

Conclusion

μSlope: An efficient semi-structured log management system that

- Handles dynamic schema structures
- Achieves unprecedented compression ratio
- Allows search without full decompression

- Open-sourced at *https://github.com/y-scope/clp*!
- Our email: info@yscope.com

YScope Uber UNIVERSITY OF TORONTO