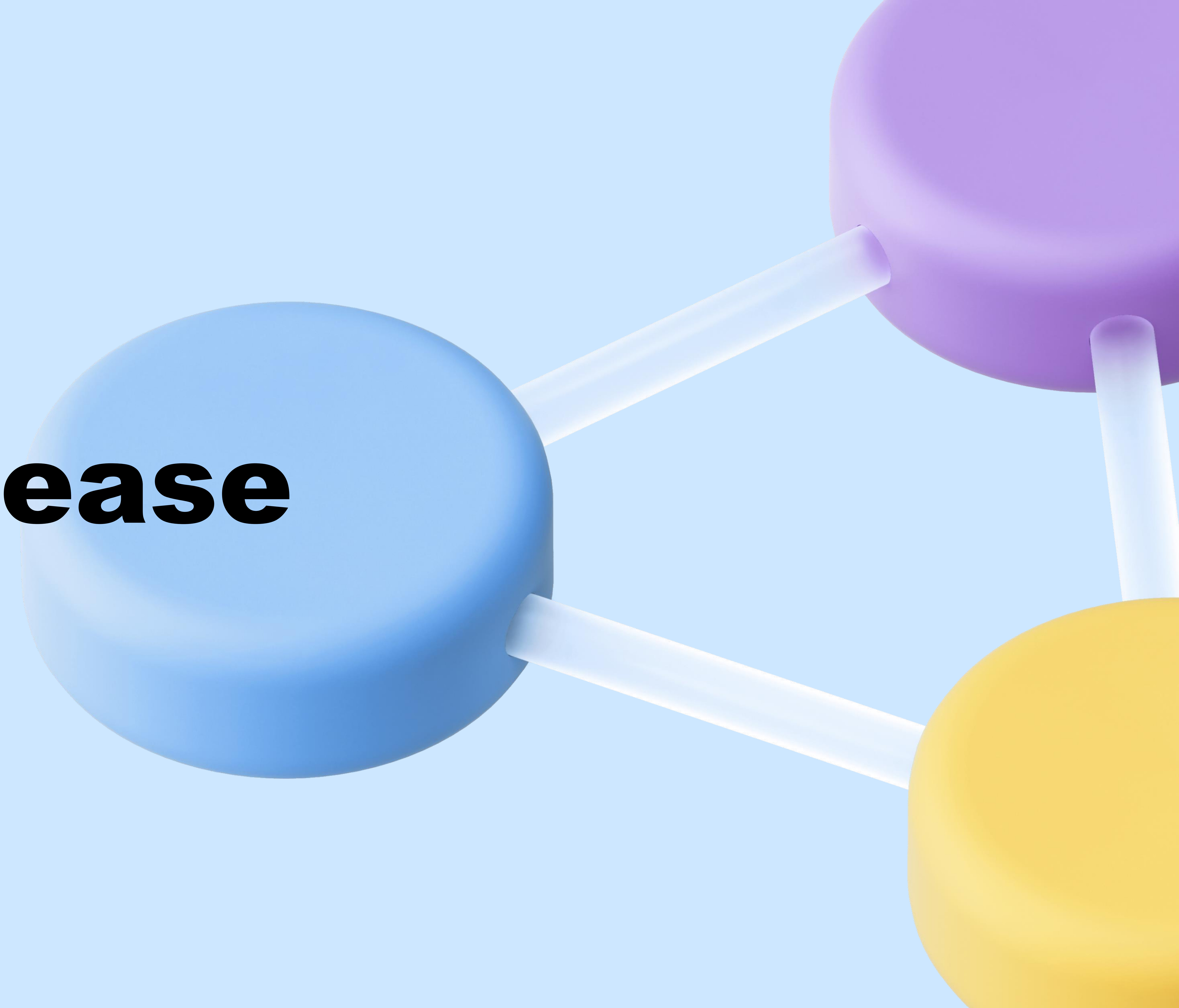
YDB

YDB Release v23.1



Release 23.1 Webinar

1

What's new in YDB v23.1

- 3 new features
- 5 performance improvements
- 5 issues fixed

2

Q&A session

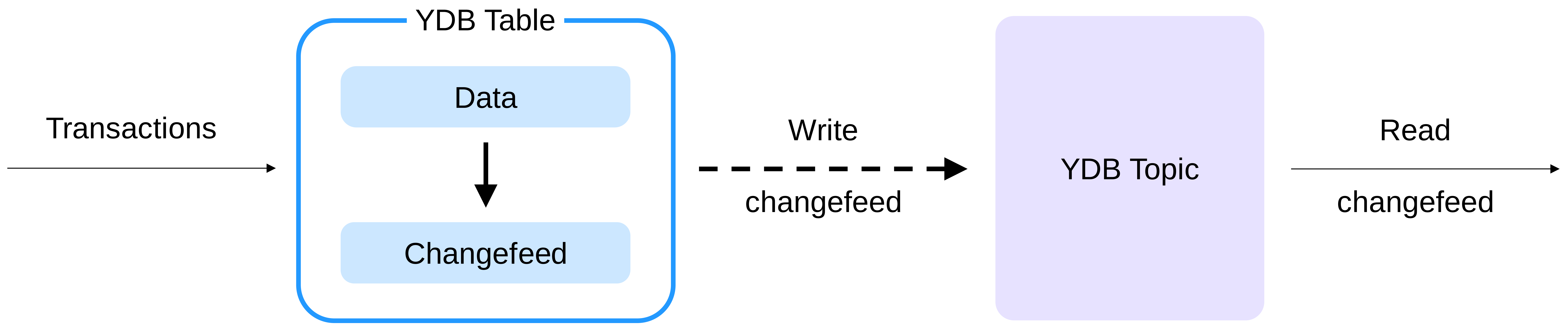
Upgrade instructions: https://ydb.tech/en/docs/administration/upgrade

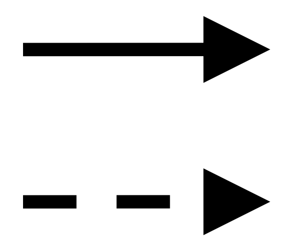
Initial table scan on CDC changefeed creation

By Ilnaz Nizametdinov

- By default CDC sends only new changes
- There are scenarios when changes need to be applied on top of the current (initial) state
- Initial table scan solves the problem of obtaining the initial state

How CDC works





Replication using CDC

- **Enable CDC**

  Add changefeed

- **Fill replica with initial data**

  SELECT

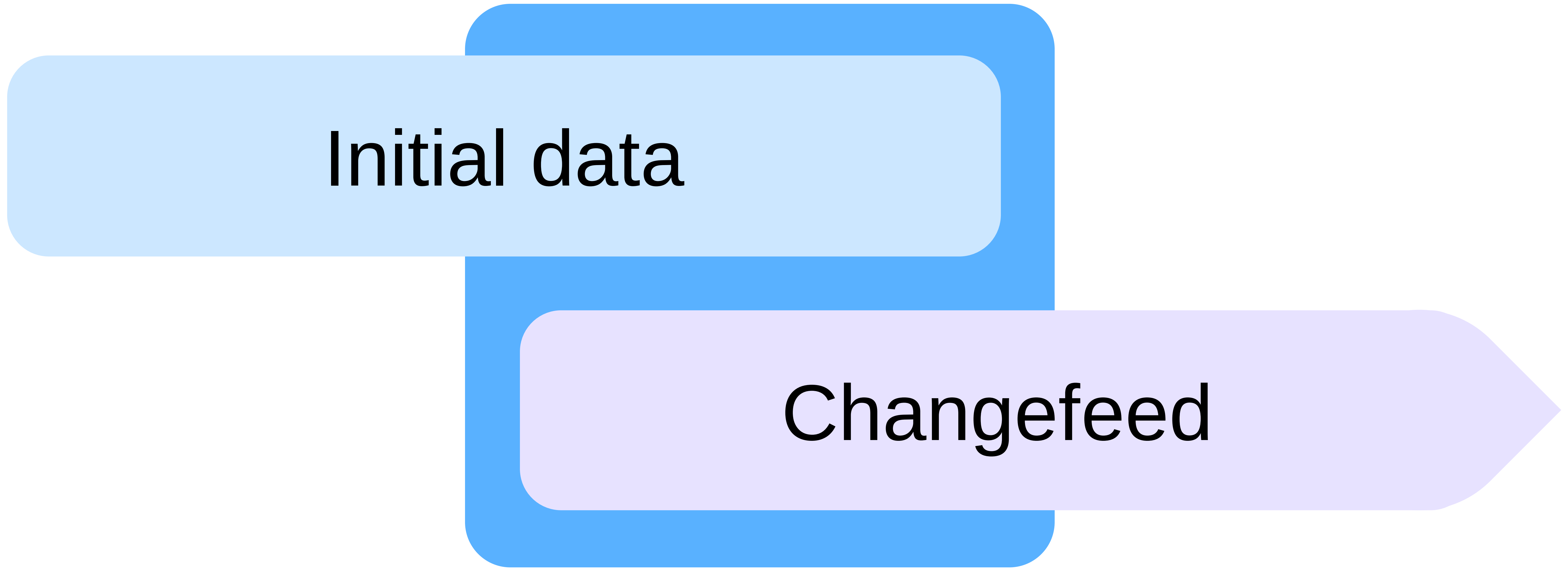
  ReadTable

  Using a backup

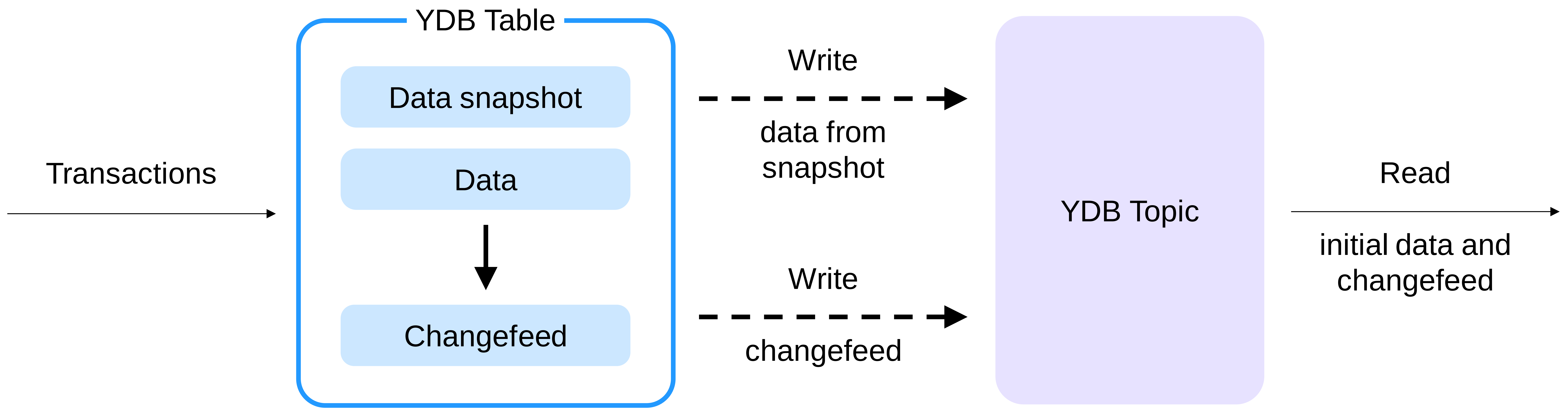
- **Apply changefeed**

Problems

- Switching between different ways of working with data
- It is difficult to apply changefeed on top of the initial state



How CDC Initial Scan works



→ Synchronous

- - ▶ Asynchronous

Replication using CDC Initial Scan

Enable CDC (add changefeed) with option

INITIAL_SCAN = TRUE

Apply changefeed

- Includes initial data
- Order guarantee: initial state of the row will arrive first, and changes will come after it

Audit log

By Andrei Rykov

The audit log is a specialized tool for monitoring key actions and events within the YDB product

The audit log empowers users

- Monitor system interactions
- Detect unauthorized activities
- Assist in incident investigations

Based on the configuration settings, YDB has the capability to direct audit log data to

- The standard error output (stderr)
- An individual file located on each node within the YDB cluster

Audit log

By Andrei Rykov

In this update, logging of changes to YDB schema objects was added: databases, directories, tables, topics. Additionally, it logs changes in the number of partitions, backup and restore operations, as well as modifications to access and more

An audit log record includes

- Date and time of the event
- The user/account that initiated the operation
- Description of operations on YDB objects
- The result of the request

```
2023-03-13T19:59:27.614731Z:
{
  "tx_id":"562949953426315",
  "subject":"user",
  "remote_address":"ipv6:[xxxx:xxx:xxx:xxx:x:xxxx:xxx:xxxx]:xxxxx",
  "component":"schemeshard",
  "operation":"CREATE TABLE",
  "paths":"[/my_dir/db1/my_table]",
  "database":"/my_dir/db1",
  "status":"SUCCESS",
  "detailed_status":"StatusAccepted"
}
```

Automatic actor system pools configuration

By Aleksander Kryukov

Main changes

Dynamic change of number of threads in a pool depending on system load

Once a second the system checks load and either gives or takes away a thread

The number of CPUs used is less than or equal to the number allocated to the process

Simplified actor system configuration

You can now show only node kind and total number of cores for the process

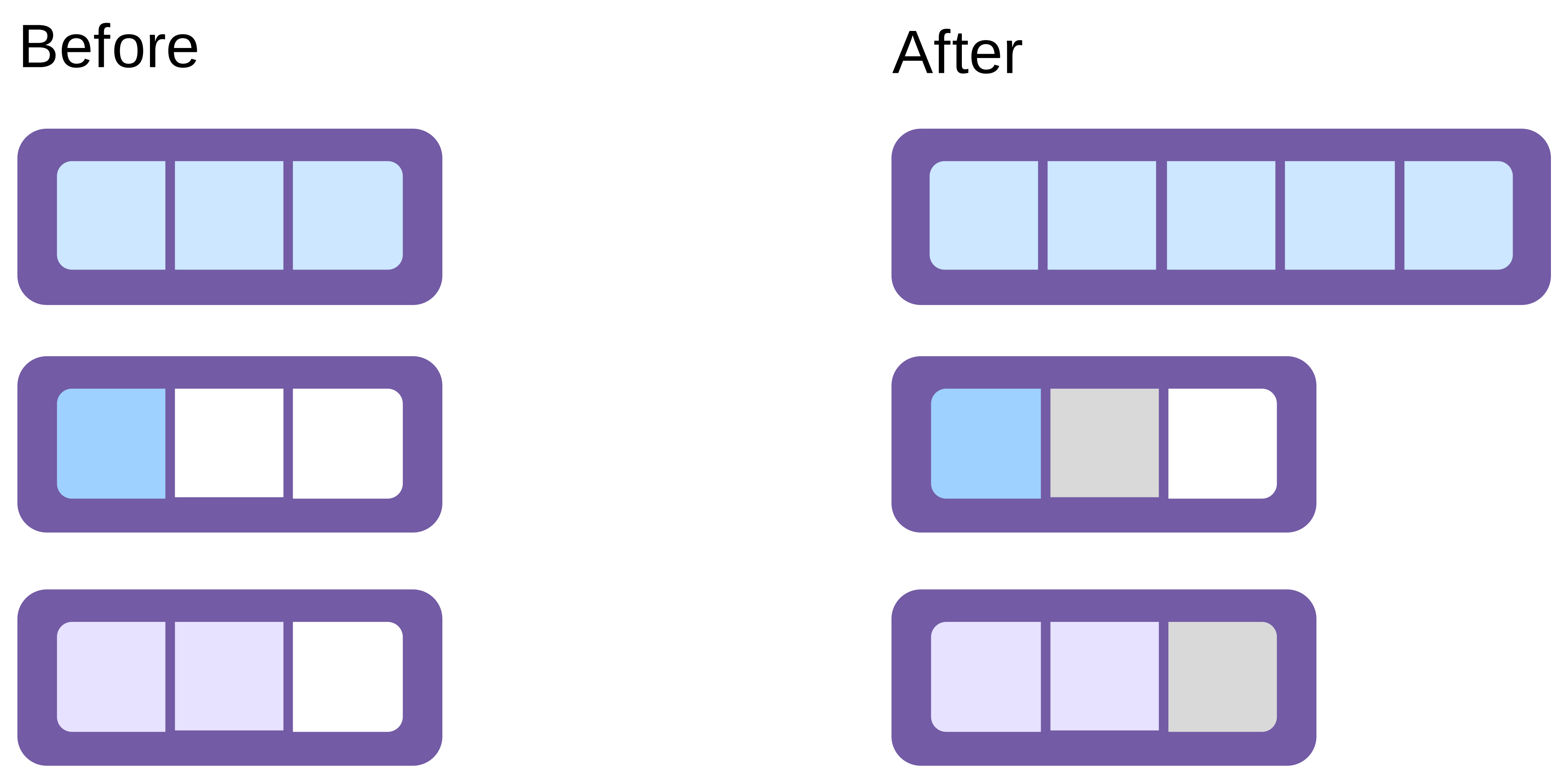
By default it'll take the total number of available cores (affinity/physical)

Automatic actor system pools configuration

Dynamic change of number of threads in a pool depending on system load

Once a second the system checks load and either gives or takes away a thread

The number of CPUs used is less than or equal to the number allocated to the process



Automatic actor system pools configuration

Simplified actor system configuration

You can now specify only node kind and total number of cores for the process

By default it'll take the total number of available cores (affinity/physical)

Before

```
actor_system_config:
 executor:
 - name: System
   spin_threshold: 0
   threads: 2
   type: BASIC
 - name: User
   spin_threshold: 0
   threads: 3
   type: BASIC
 - name: Batch
   spin_threshold: 0
   threads: 2
   type: BASIC
 - name: IO
   threads: 1
   time_per_mailbox_micro_secs: 100
   type: IO
 - name: IC
   spin_threshold: 10
   threads: 1
   time_per_mailbox_micro_secs: 100
   type: BASIC
 scheduler:
   progress_threshold: 10000
   resolution: 256
   spin_threshold: 0
```

After

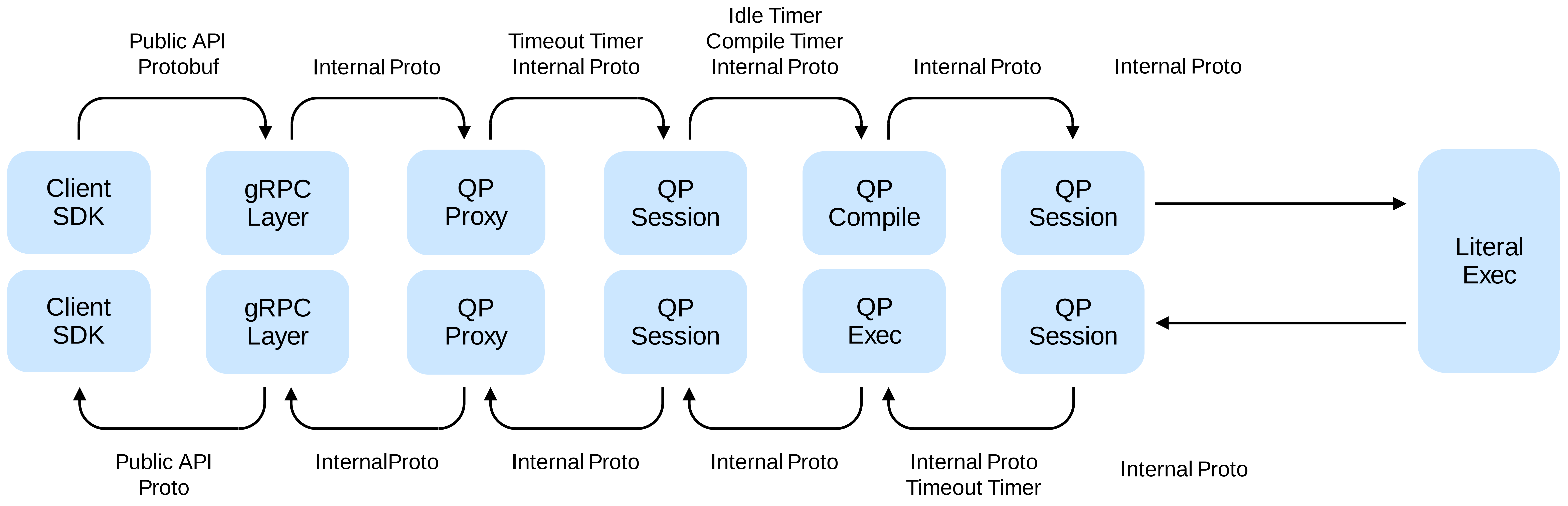
```
actor_system_config
  use_auto_config: true
  cpu_count: 9
  node_type: COMPUTE
```

Improved data transfer formats between query execution stages. Before

By Vitalii Gridnev

```
DECLARE $lines as List<Struct<id:UInt64,value:Utf8>>;

UPSERT INTO `table` SELECT * FROM AS_TABLE($lines);
```

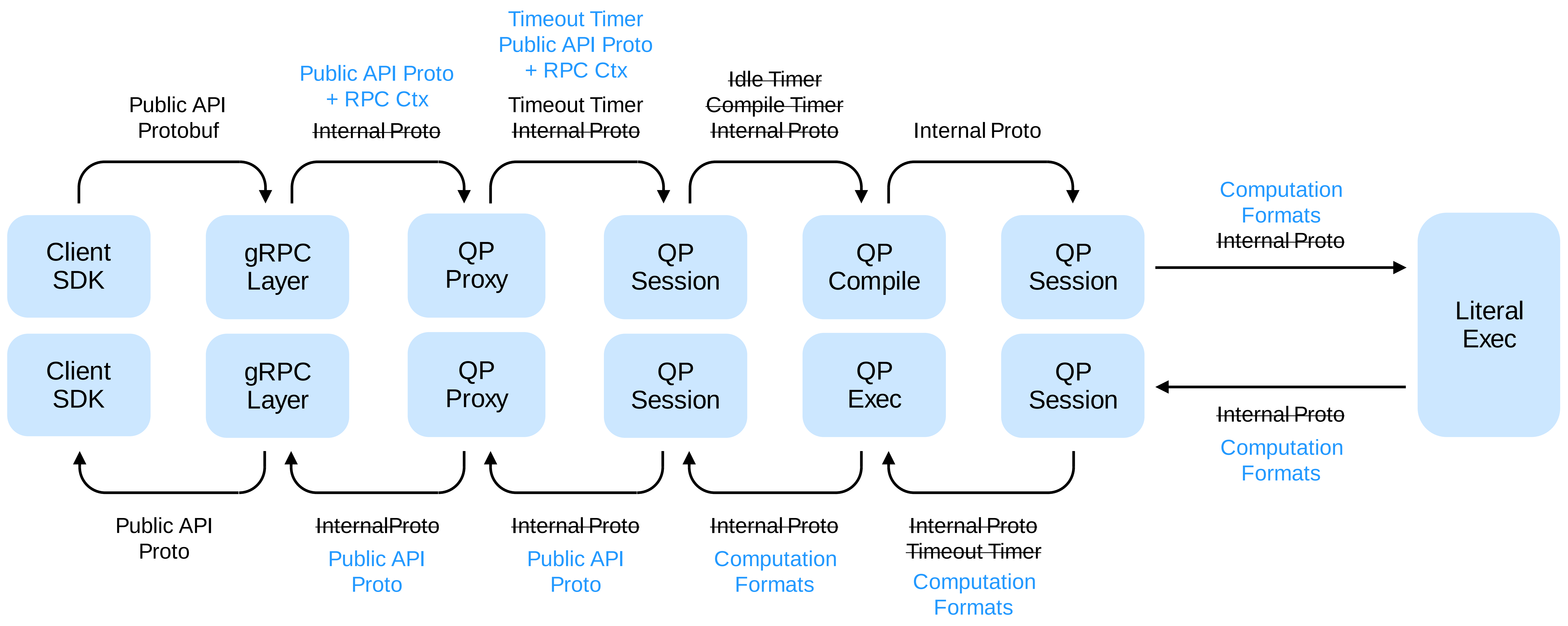


Improved data transfer formats between query execution stages. After

By Vitalii Gridnev

```
DECLARE $lines as List<Struct<id:UInt64,value:Utf8>>;

UPSERT INTO `table` SELECT * FROM AS_TABLE($lines);
```



Improved data transfer formats between query execution stages

By Vitalii Gridnev

Improved interprocess data formats in Query Processing

Avoid Protocol Buffers as much as possible and use data formats which are native for computation

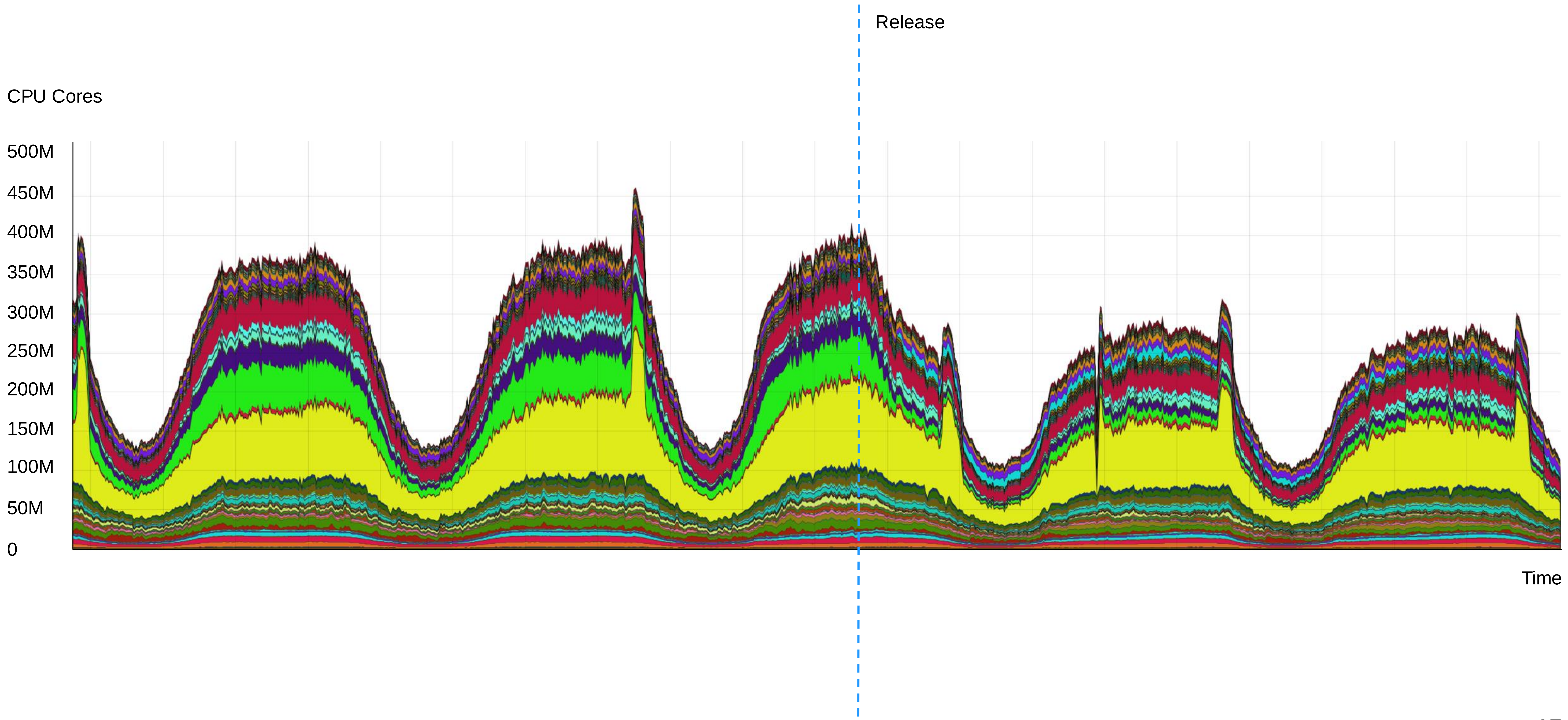
Actor system usage refactoring

Literal execution is now inlined into session (to avoid heavy thread wakeup)

Session idle timers are refactored, removed duplication of query timeout timers

RPC context passthrough, avoid useless copies of protocol buffers

Improved data transfer formats between query execution stages



Computation graph caching

By Vlad Kuznetsov

Each query goes through an execution stage

Overall path:

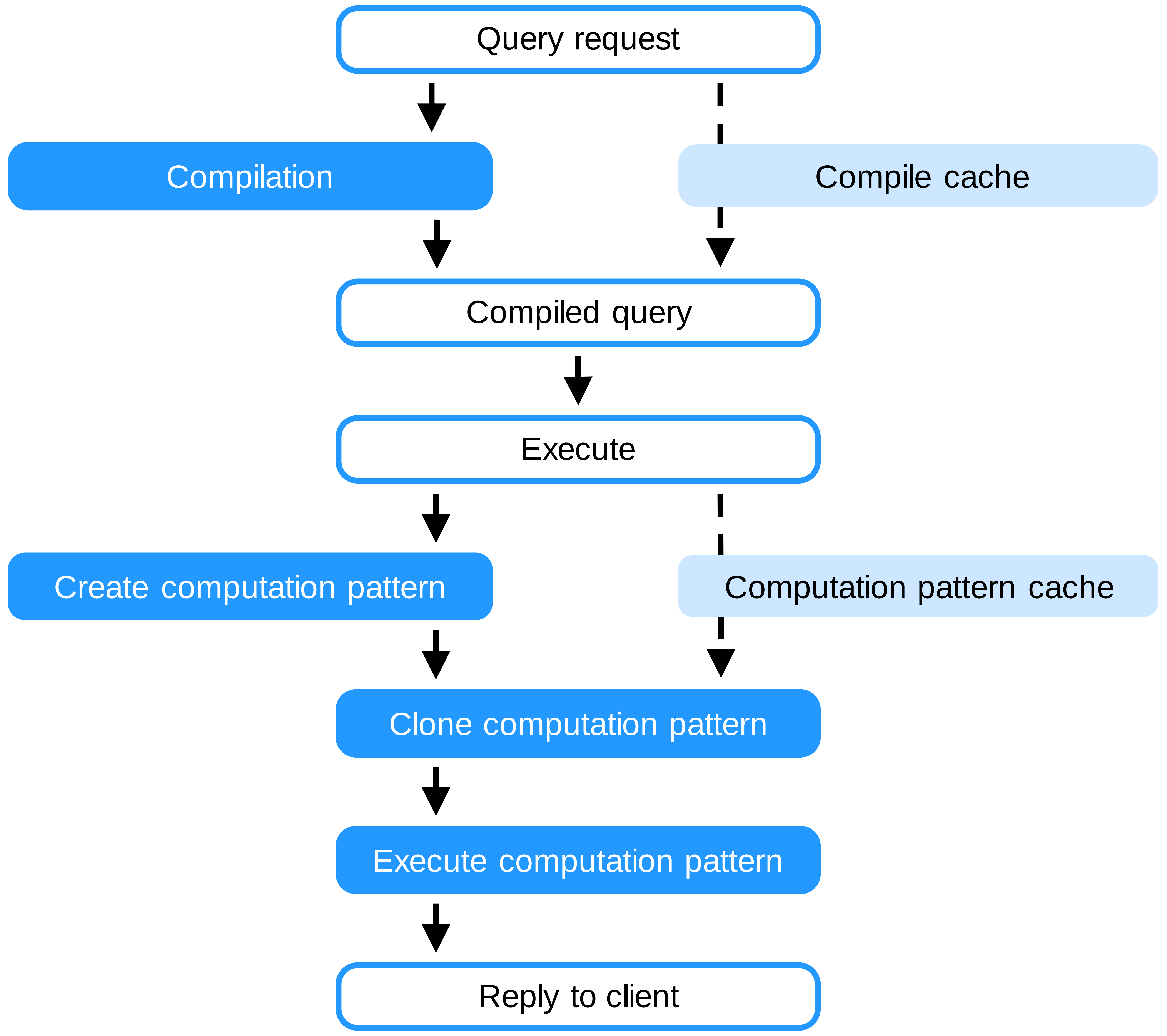
1. Query compilation
2. Building computation graph pattern
3. Execution of computation graph

In OLTP scenario the first two stages can easily be more expensive than the third. The obvious solution is to cache them

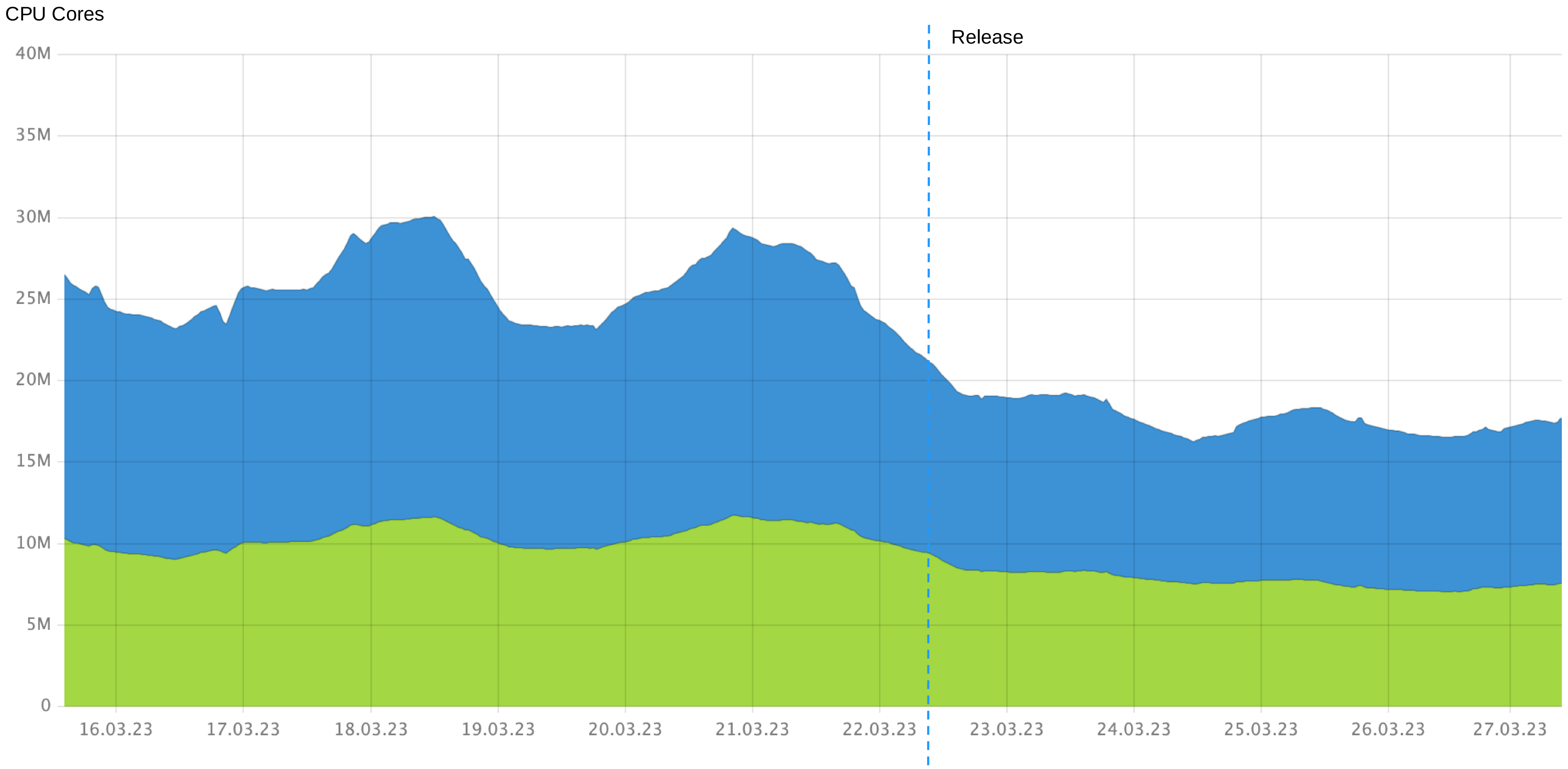
YDB had query compilation cache for a while now, and in 23.1 there‘s a new cache for the second stage — computation graph patterns

On a cache hit the pattern is cloned, enriched with temporary values and it is ready for execution

Computation graph caching


Query request
Compilation
Compile cache
Compiled query
Execute
Create computation pattern
Computation pattern cache
Clone computation pattern
Execute computation pattern
Reply to client

Computation graph caching



Atomic secondary index replacement

By Daniil Cherednik

- In production environment you may need to change a secondary index, but without modification of application working with the table
- The most common case — adding COVER columns
- To make this possible atomic secondary index replace feature was added
- When replace happens, compiled queries are invalidated and their re-compilation starts using the new one
- To replace an index you need to:
  1. Prepare an index with a new name
  2. Replace index via CLI (or SDK)

Example

```
ydb table index rename goods --index-name price_index_new --
to price_index --replace
```

Secondary indices overview

By Yulia Sidorina

There are two query types

Data Query
OLTP transations

Scan Query
Analytical ad-hoc queries

Use of secondary indices in Data Query

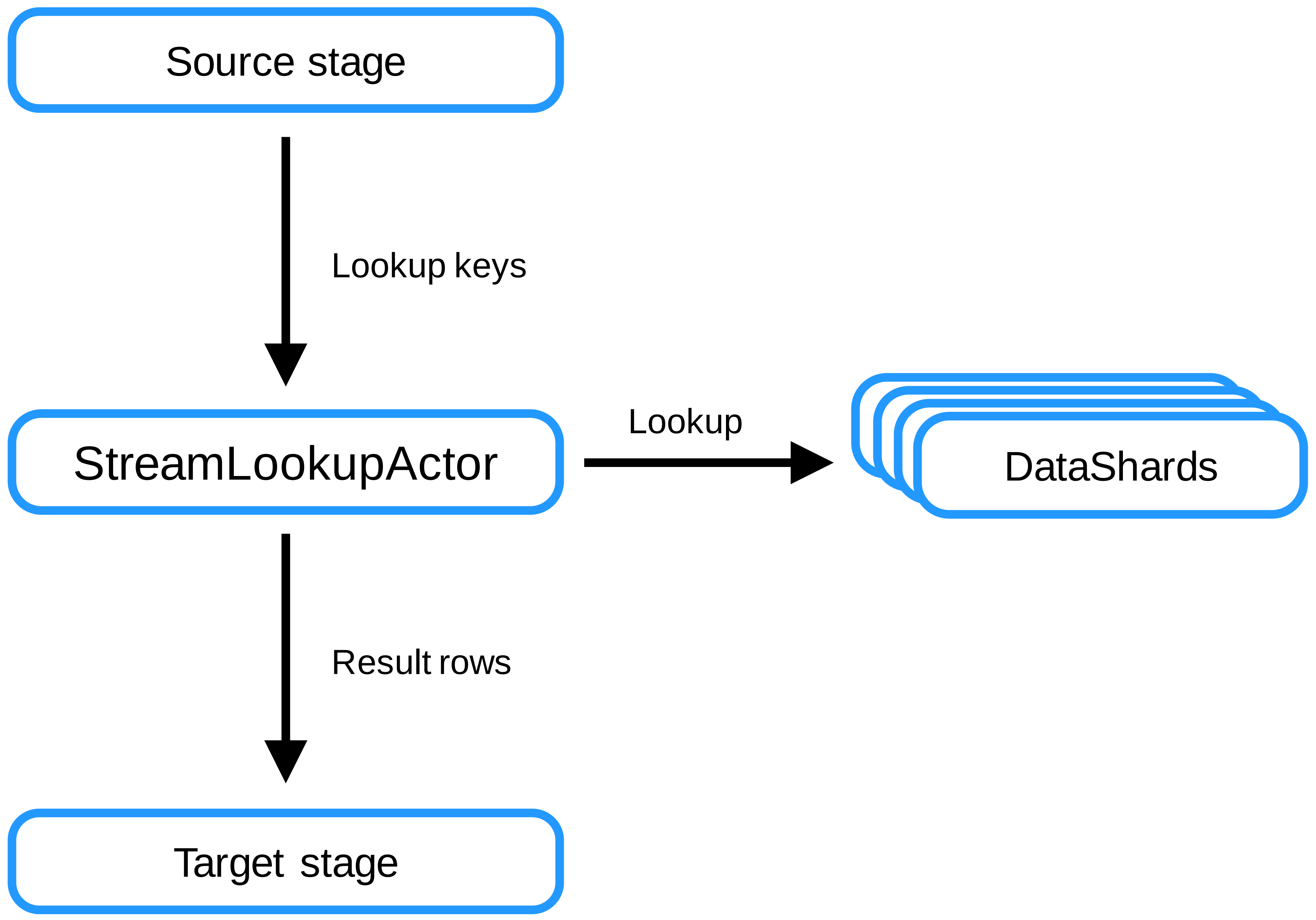
```
SELECT Column1, Column2 FROM Table VIEW SecondaryIndex WHERE Fk = 'SomeValue'
```

Secondary indices in scan queries

By Yulia Sidorina

- When main table retrieval is not needed (only index or cover columns are used) not much changes were needed, we just work with the index table
- Otherwise it works via a new StreamLookupActor

Enable EnableKqpScanQueryStreamLookup flag in your YDB configuration



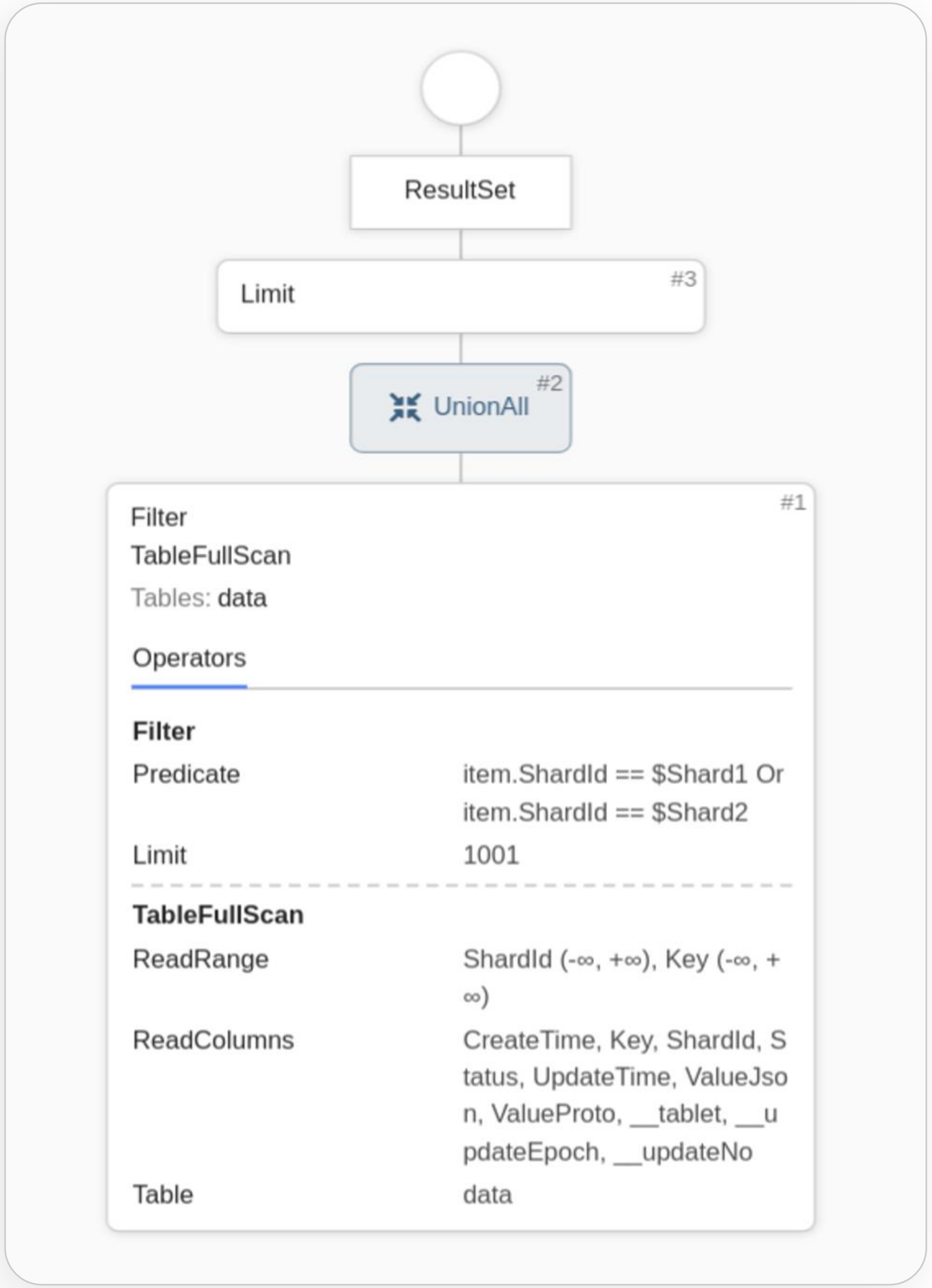
Improved predicate pushdown for table reads

By Mikhail Surin

Made possible to use dynamic read ranges

```
DECLARE $Shard1 AS Uint32;
DECLARE $Shard2 AS Uint32;

SELECT *
FROM `/eu/ugc/prod/ugcdb/data`
where ShardId = $Shard1 or ShardId = $Shard2;
```



Filter
TableFullScan
Tables: data

Operators

Filter

Predicate	item.ShardId == $Shard1 Or item.ShardId == $Shard2
Limit	1001

TableFullScan

ReadRange	ShardId (-∞, +∞), Key (-∞, +∞)
ReadColumns	CreateTime, Key, ShardId, Status, UpdateTime, ValueJson, ValueProto, __tablet, __updateEpoch, __updateNo
Table	data

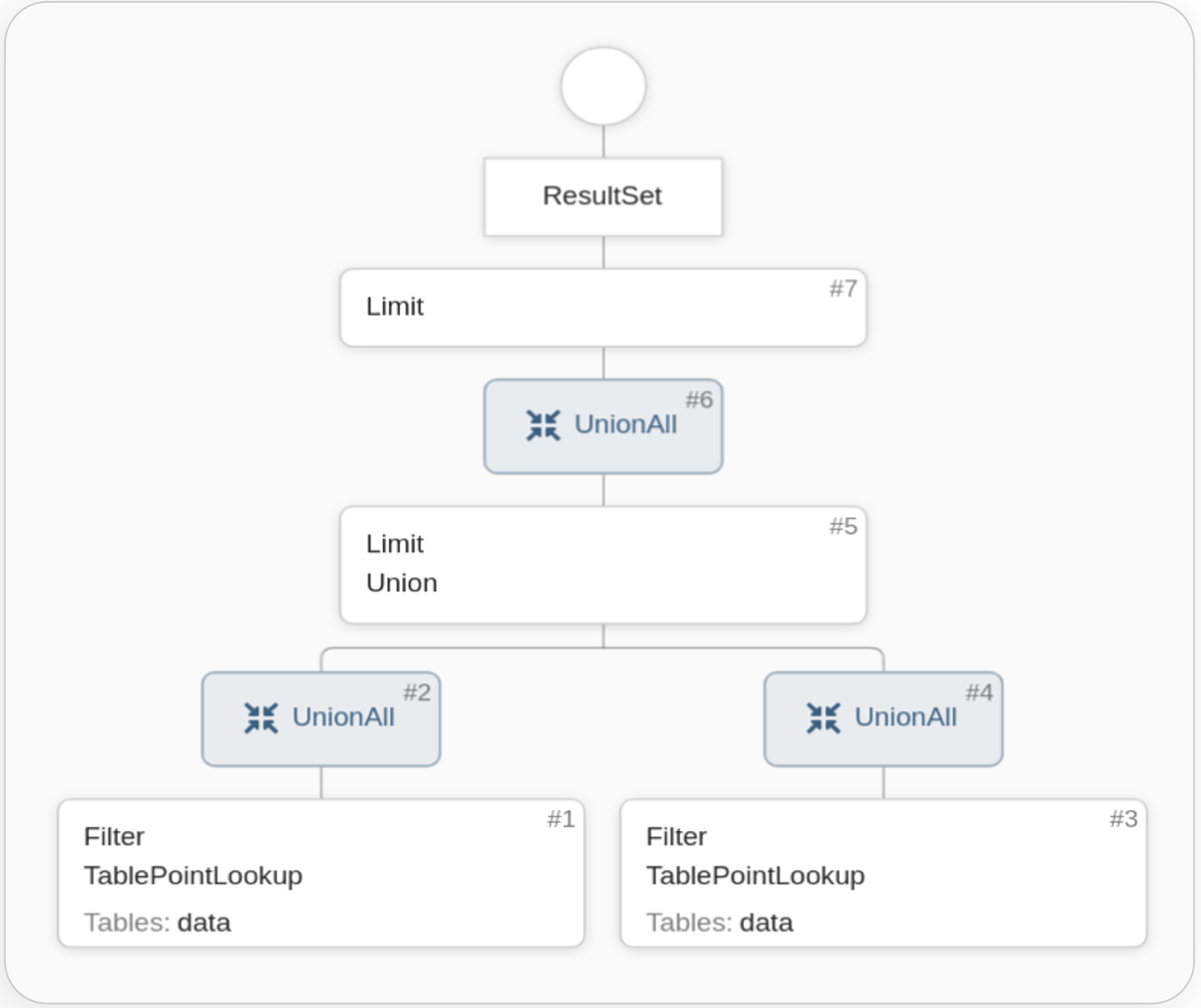
Improved predicate pushdown for table reads

By Mikhail Surin

Possible workaround

```
DECLARE $Shard1 AS Uint32;
DECLARE $Shard2 AS Uint32;

SELECT *
 FROM `/eu/ugc/prod/ugcdb/data` where ShardId = $Shard1
UNION ALL
SELECT *
 FROM `/eu/ugc/prod/ugcdb/data` where ShardId = $Shard2
```



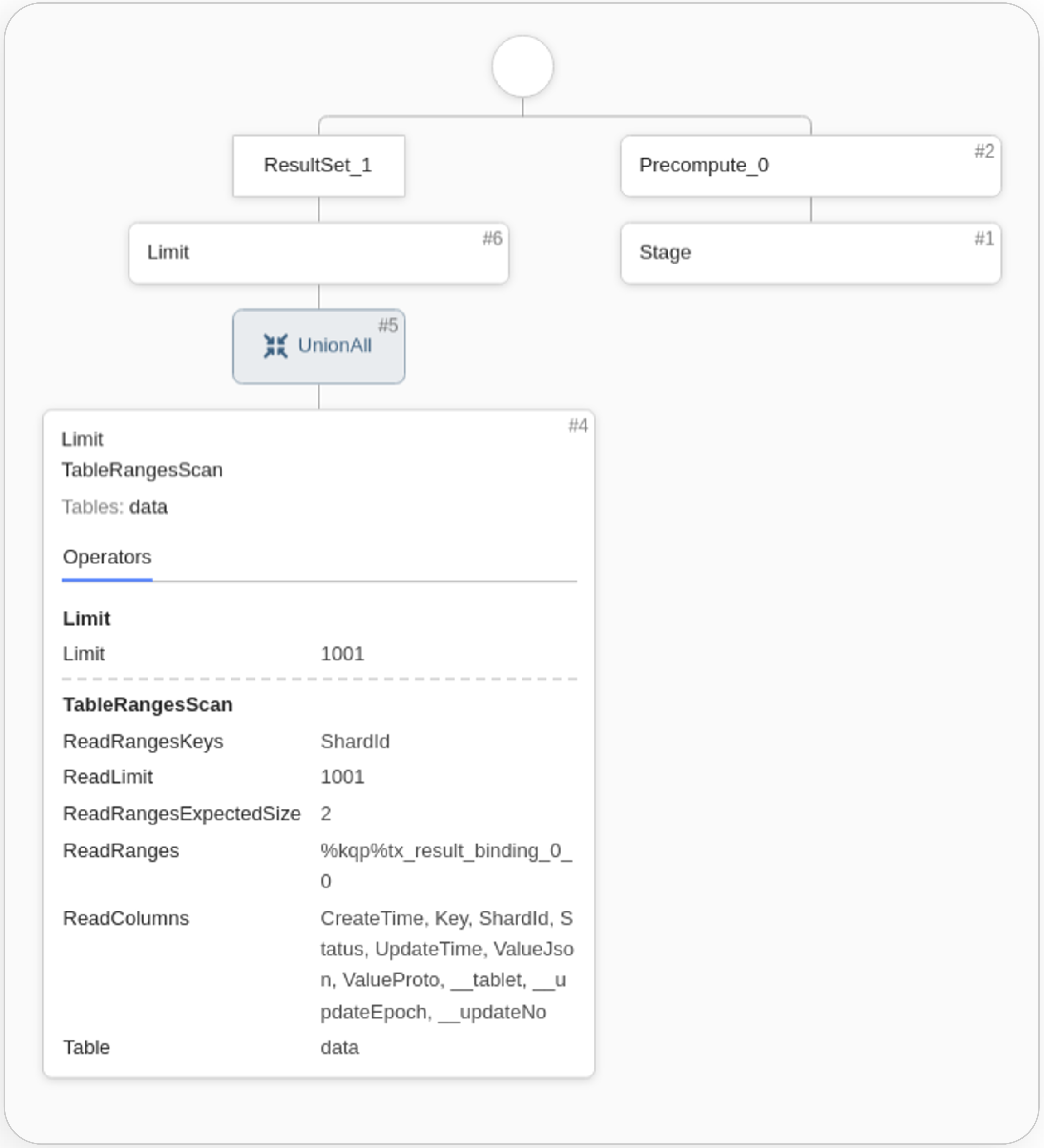
Improved predicate pushdown for table reads

By Mikhail Surin

Precompute stage
in 23-1

```
DECLARE $Shard1 AS Uint32;
DECLARE $Shard2 AS Uint32;

SELECT *
  FROM `/eu/ugc/prod/ugcdb/data`
where ShardId = $Shard1 OR ShardId = $Shard2
```



#4
Limit
TableRangesScan
Tables: data

Operators

Limit

Limit	1001

TableRangesScan

ReadRangesKeys	ShardId
ReadLimit	1001
ReadRangesExpectedSize	2
ReadRanges	%kqp%tx_result_binding_0_0
ReadColumns	CreateTime, Key, ShardId, Status, UpdateTime, ValueJson, ValueProto, __tablet, __updateEpoch, __updateNo
Table	data

Join YDB community

t.me/ydb_ru

Star YDB on GitHub

clck.ru/rdFQw