Filtering 100M objects
What can go wrong?

Alexey Ragozin
alexey.ragozin@gmail.com
Dec 2013

Problem description

- 100M object (50M was tested)
- ~ 100 fields per object
- ~ 1kb per object (ProtoBuf binary format)
- Simple queries

  `select … where … order by … [limit N]`
- Expected query result set – 200k
- Max query result set – 50% of all data

Problem description

- 100M object (50M was tested)
- ~100 fields per object
- ~1kb per object (ProtoBuf binary format)

Object size is Ok

- Simple queries

```
select … where … order by … [limit N]
```

Inline with Coherence filters

- Expected query result set – 200k
- Max query result set – 50% of all data

Problem description

- 100M object (50M was tested)
- ~ 100 fields per object
- ~ 1kb per object (ProtoBuf binary format)
- Simple queries

  `select … where … order by … [limit N]`

- Expected query result set – 200k
- Max query result set – 50% of all data

Challenge

Problem description

- 100M object (50M was tested)
- ~ 100 fields per object
- ~ 1kb per object (ProtoBuf binary format)
- Simple queries

  ```
  select … where … order by … [limit N]
  ```

- Expected query result set – 200k
- Max query result set – 50% of all data

Real challenge

Big result set problem

Calling **`NamedCache`** method

- Single TCMP message to each participating member
- Processing of on remote member
- Single TCMP result message from each member
- Aggregation all results in caller JVM

Return form method

Huge TCMP message = **cluster crash**

Naive strategy

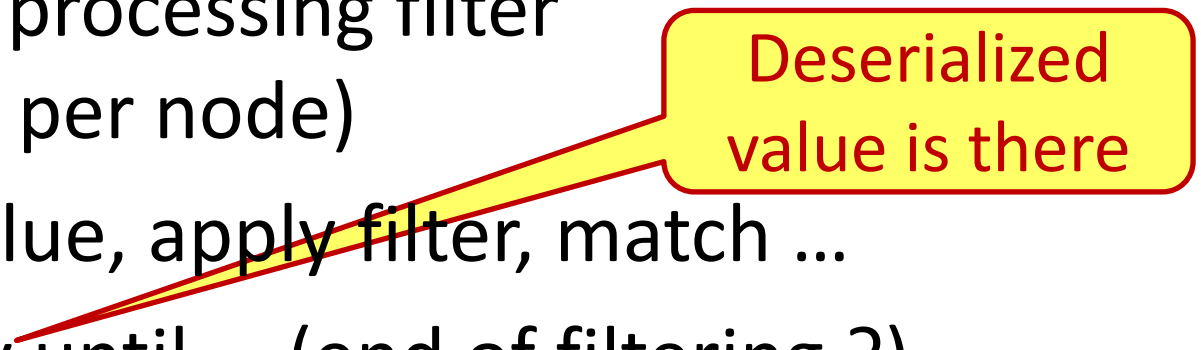
Processing of query

- Send aggregator with filter, retrieve
  - ✓ Keys
  - ✓ Field for sorting
- Sort whole result set (keys + few fields)
- Apply limit
- Retrieve and send objects in fixed batches

Testing …

OutOfMemoryError on storage node

- Storage node processing filter (600K objects per node)
  - Deserialize value, apply filter, match …
  - … retain entry until … (end of filtering ?)
- Filter processing may take few seconds
- There could be few concurrent queries



Solving …

Using indexes

- Index only filter does not deserialize object
- We cannot index everything
- Single unindexed predicate would call deserialization

Special filter to cut deserialized object reference

- We do not need object (aggregator extracts from binary)
- Desirialized object now collected in young space
- Synthetic wrapper object + messing with serialization

Testing …

Very high memory usage on service node

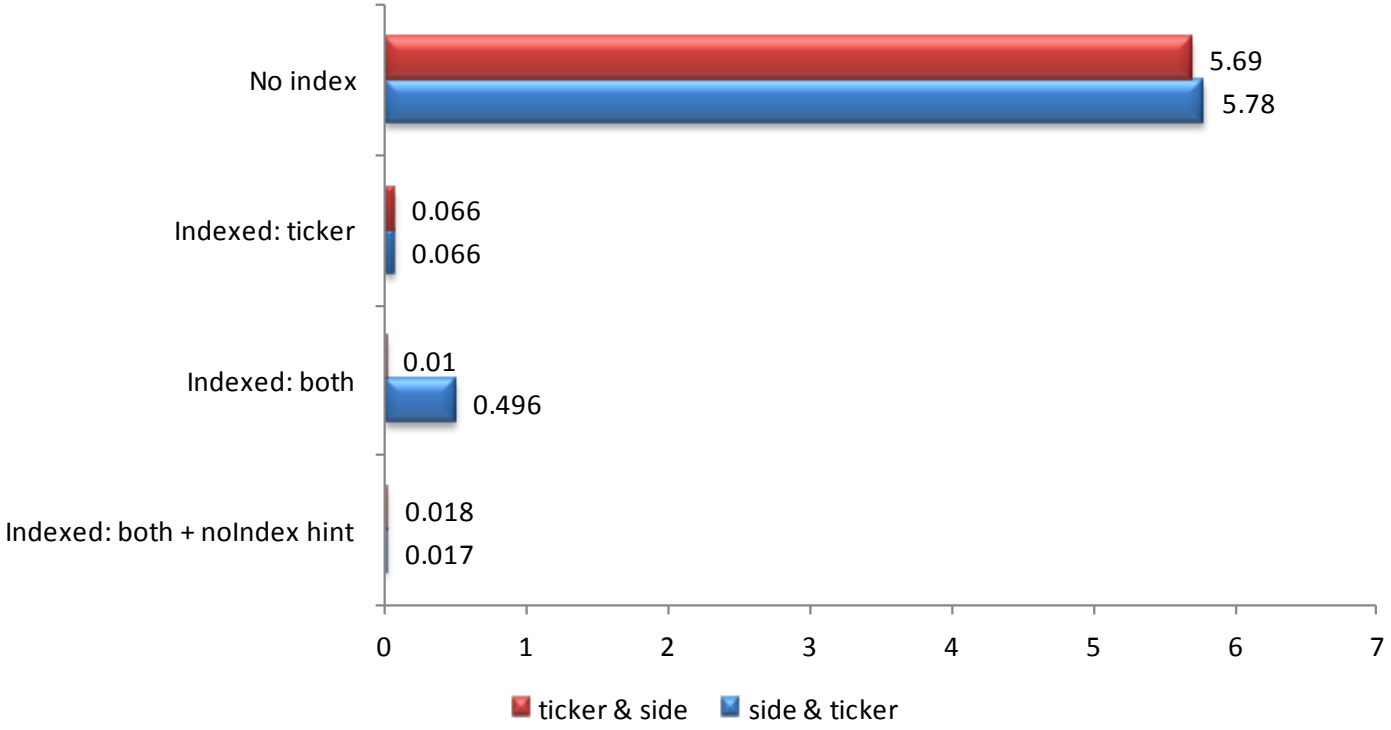
- Collecting and sorting large result set
  - Have to use huge young space (8Gib)
  - Query concurrency in limited by memory

Single threaded sorting

- It is very fast tough

Indexes and attribute cardinality

"Status" attribute – 90% of objects are OPEN



http://blog.ragozin.info/2013/07/coherence-101-filters-performance-and.html

Indexes and attribute cardinality

Possible strategies to remedy

- Transform query
- Wrap “bad” predicates into `NoIndexFilter`
- Fix filter execution “planner”

Indexes and attribute cardinality

Can we go without indexes?

- Full scan 50M – 80 cores, 3 servers
- 30 seconds
- Too slow!

Naive strategy

Almost good enough

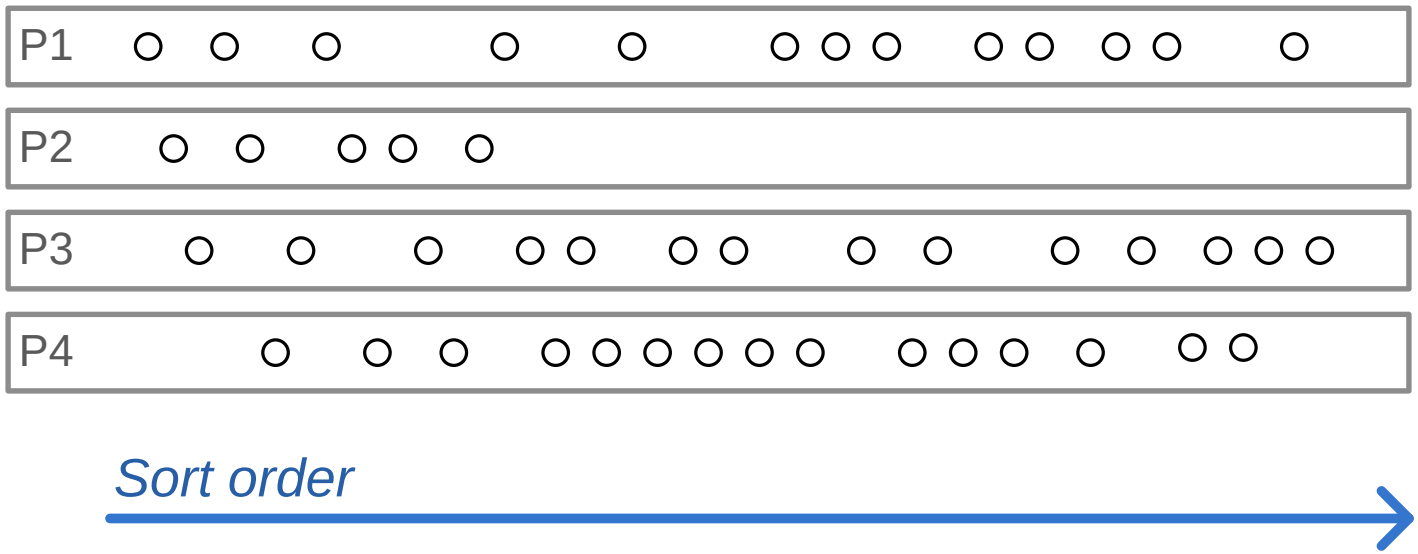
Problems with naive strategy

- Big memory problems on *service* process
- Max result set size is limited
- No control on max TCMP packet size
- Indexes may be inefficient

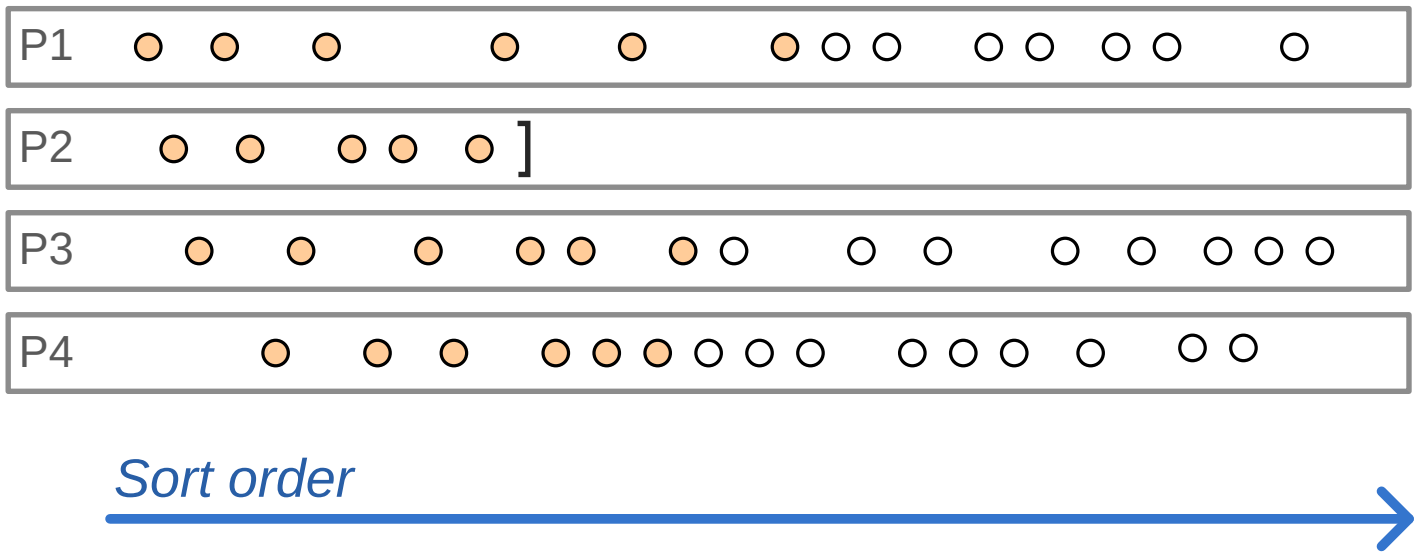
Incremental retrieval

- Result set is always sorted
- Primary key is always last sort attribute
- Aggregator on invocation
  - ✓ Sort its partial result set
  - ✓Selects first N
  - ✓ Return N references (key + sort attribute)
  - ✓ Return remaining size of each partition

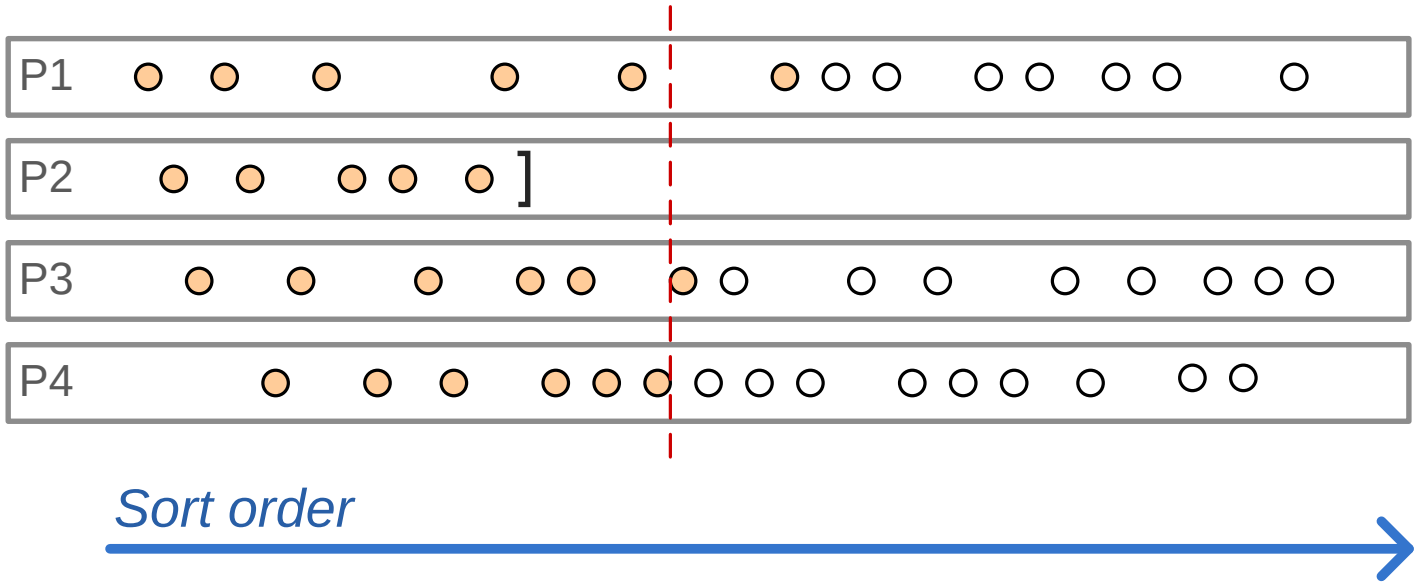
Incremental retrieval



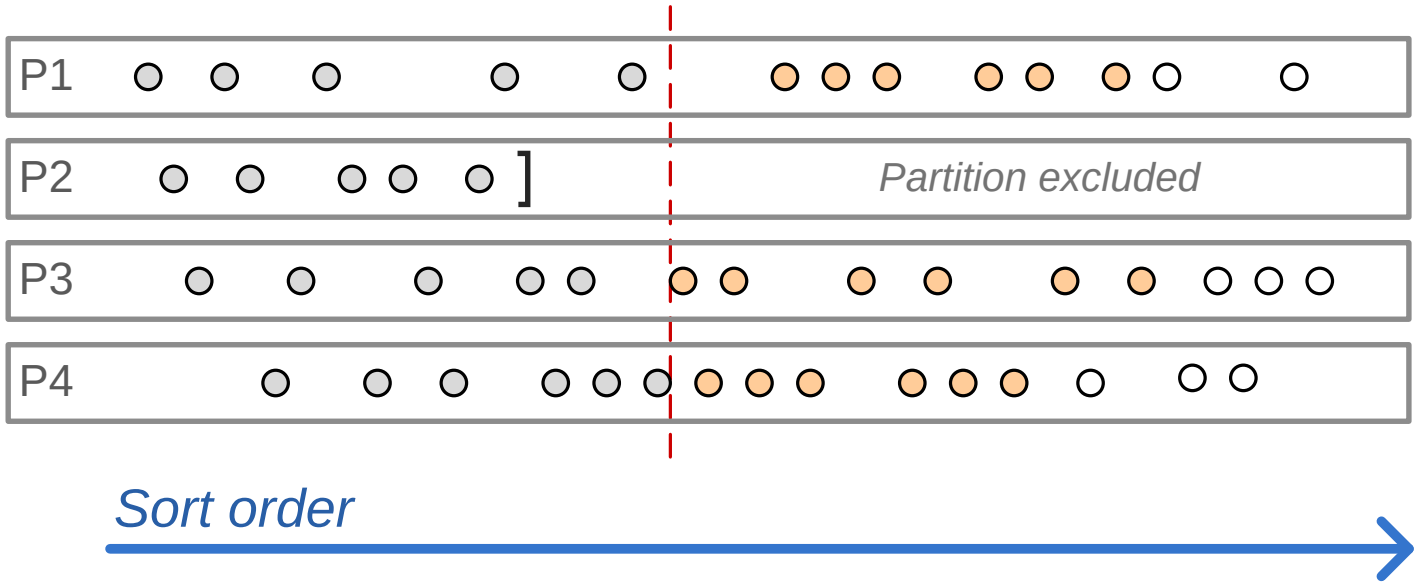
Incremental retrieval



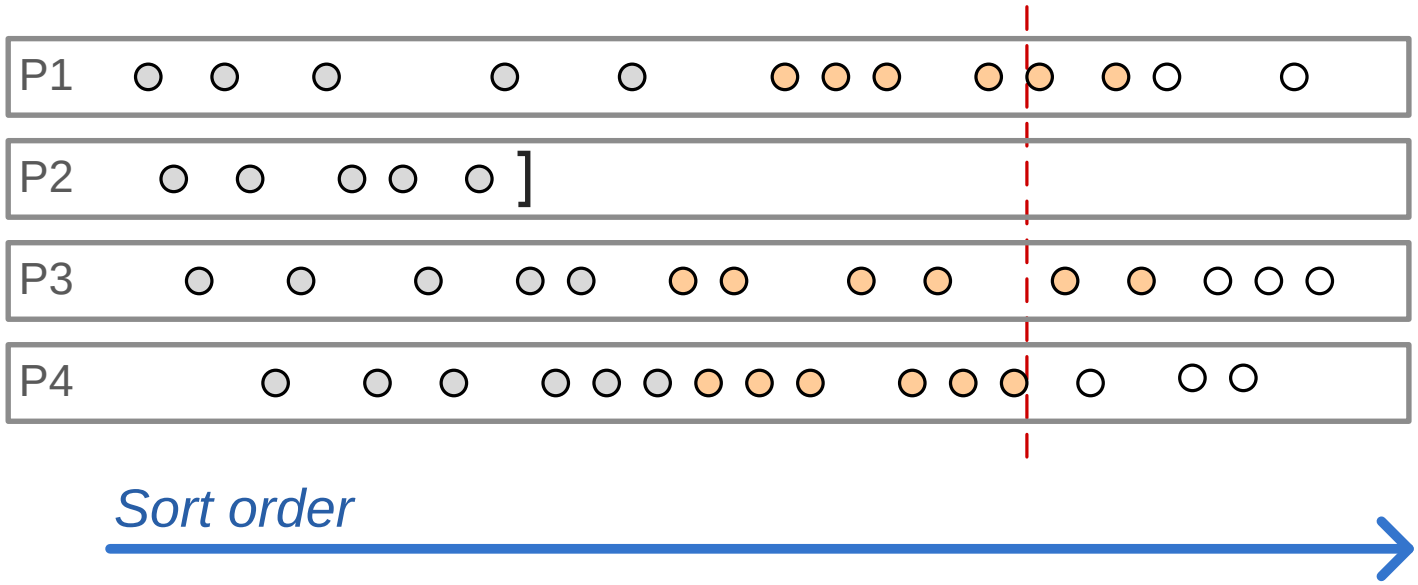
Incremental retrieval



Incremental retrieval



Incremental retrieval



Incremental retrieval

Advantages

- Size of TCMP packets is under control
- Reduced traffic for LIMIT queries
- Fixed memory requirements for service node

Partial retrieval limit

- Target result set – 200k
- 80 nodes
- Best performance with ~1500 limit

Incremental retrieval

A little nuance …

- Filter based aggregator is executed by one thread
- How many times aggregate(…) method would be called?

❑ Once
❑ Twice
❑ Once per partition
❑ Other

Coherence limits amount of data passed to aggregate(…) based on binary side of data.

Incremental retrieval

What about snapshot consistency?

- There were no consistency to begin with
- No consistency between nodes
- Index updates are not transactional

But we need result set of query to be consistent!

- Hand made MVCC
- If you REALLY, REALLY, REALLY need it

Hand made MVCC

Synthetic key to have multiple versions in cache

Data affinity to exploit partition level consistency

Timestamp based surface – consistent snapshot

if timestamp is a part of key

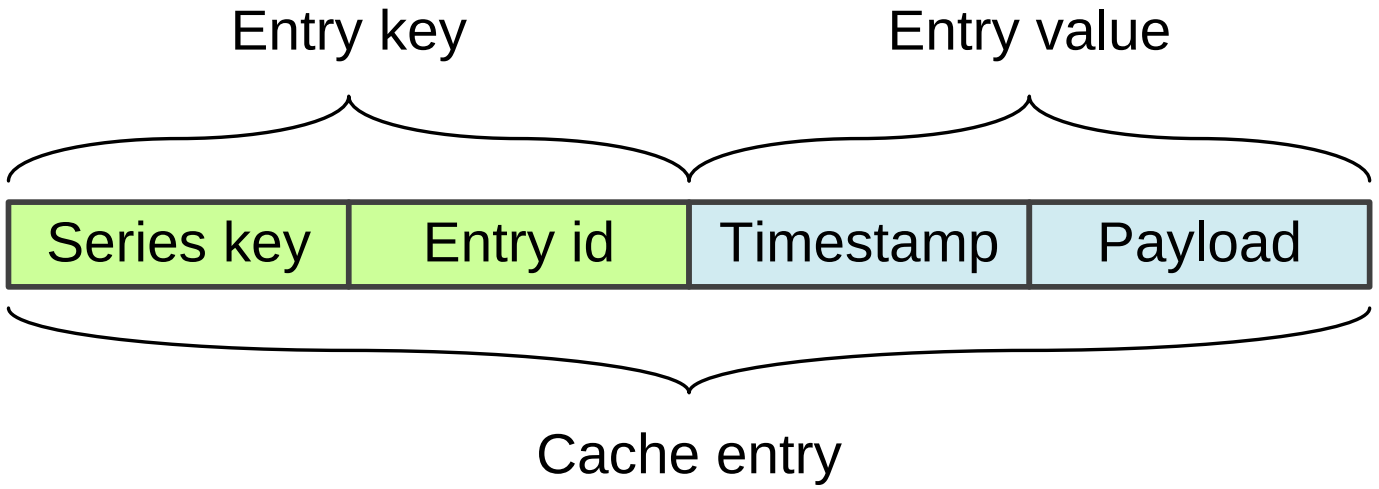
`IndexAwareFilter` can be used (without an index)

otherwise

TimeSeriesIndex - *https://github.com/gridkit/coherence-search-timeseries*

Time series index

Special index for managing versioned data

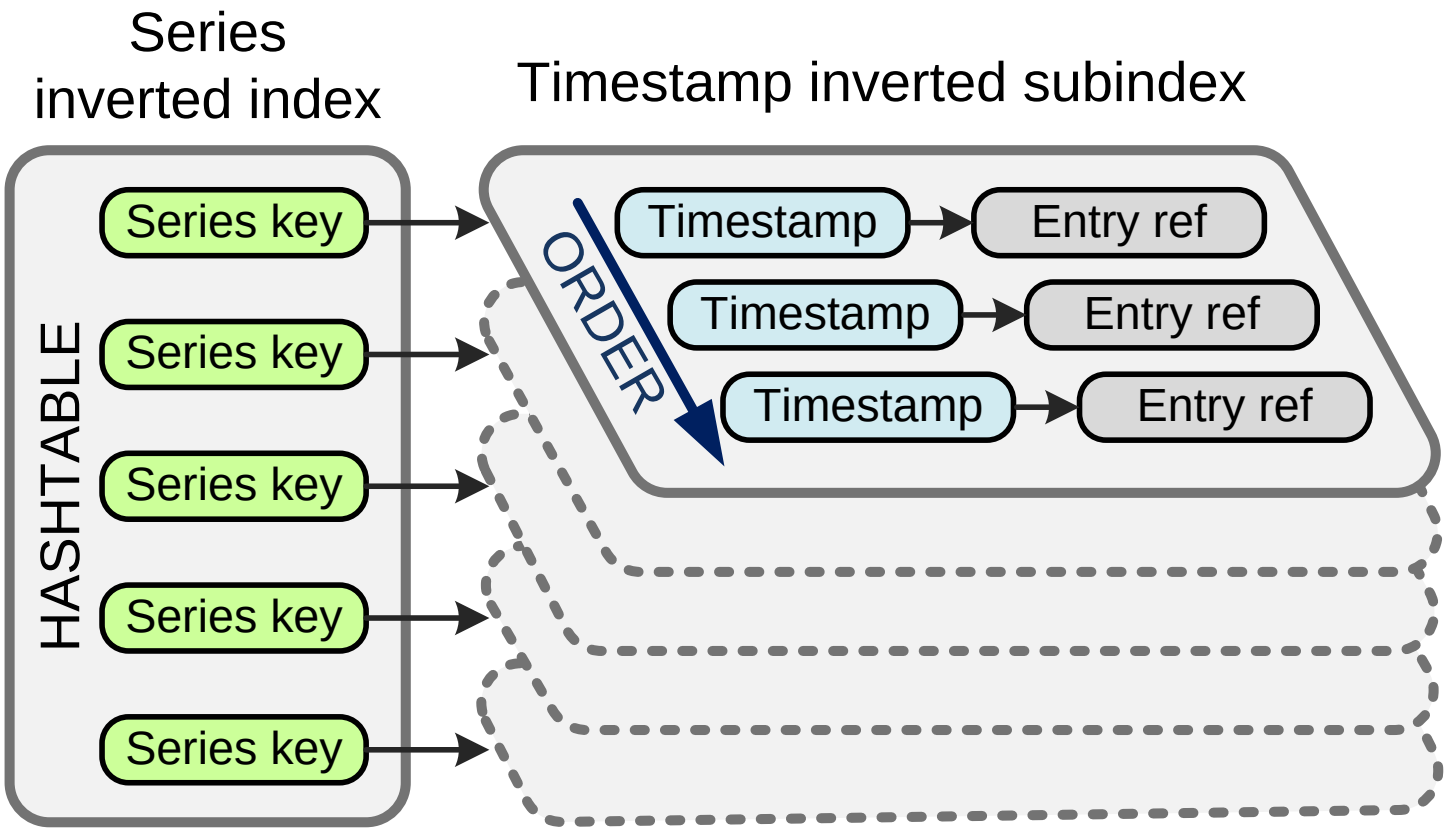


Getting last version for series *k*

```
select * from versions where series=k and version =
   (select max(version) from versions where key=k)
```

https://github.com/gridkit/coherence-search-timeseries

Time series index



https://github.com/gridkit/coherence-search-timeseries

TCMP vs TCP

TCP

- WAN networks
- Slow start
- Sliding window
- Timeout packet loss detection

Fair network sharing

TCMP

- Single switch networks
- Fast NACKs
- Loss detection by packet order
- Per packet resend

Low latency communications

Bandwidth maximization

TCMP vs TCP

In bandwidth completion
TCP doesn't have a chance against TCMP

Having TCP and TCMP in one network

- Normally TCMP is limited by proxy speaking TCP
- Traffic amplification effects (TCMP traffic >> TCP traffic)
- Bandwidth strangled TCP becomes unstable
  - ✓ Hanging for few seconds (retransmit timeouts)
  - ✓ Spurious connection resets

Keep TCMP in separate switch if possible!

http://blog.ragozin.info/2013/09/coherence-101-entryprocessor-traffic.html

Bonus: ProtoBuf extractor

Inspired by POF extractor

- Extracts fields for binary data
- Does not require generated classes or IDL
- Use field IDs to navigate data
- **XPath like expressiveness** (i.e. extract from map by key)
- **Can processes any number of extractors in single pass**
- Apache 2.0 licensed

https://github.com/gridkit/binary-extractors

Bonus: SJK diagnostic tool

SJK – CLI tool exploiting JVM diagnostic interfaces

- Connect to JVM by PID
- Display thread CPU usage in real time (like top)
- Display per thread memory allocation rate
- Dead objects histogram

… and more

https://github.com/aragozin/jvm-tools

Thank you

http://blog.ragozin.info
- my articles
http://code.google.com/p/gridkit
http://github.com/gridkit
- my open source code
http://aragozin.timepad.ru
- tech meetups in Moscow

Alexey Ragozin
alexey.ragozin@gmail.com