EMQ

EMQ X 4.3 trie improvements

EMQ X team· 2021-05

© 2020 EMQ Technologies Co., Ltd.

EMQ

What problem does "trie" solve

Given a set of strings, how to quickly find out if any input string exsits in the set.

Example set:

- to
- tea
- ted
- ten
- tn
- inn
- A

Example query:

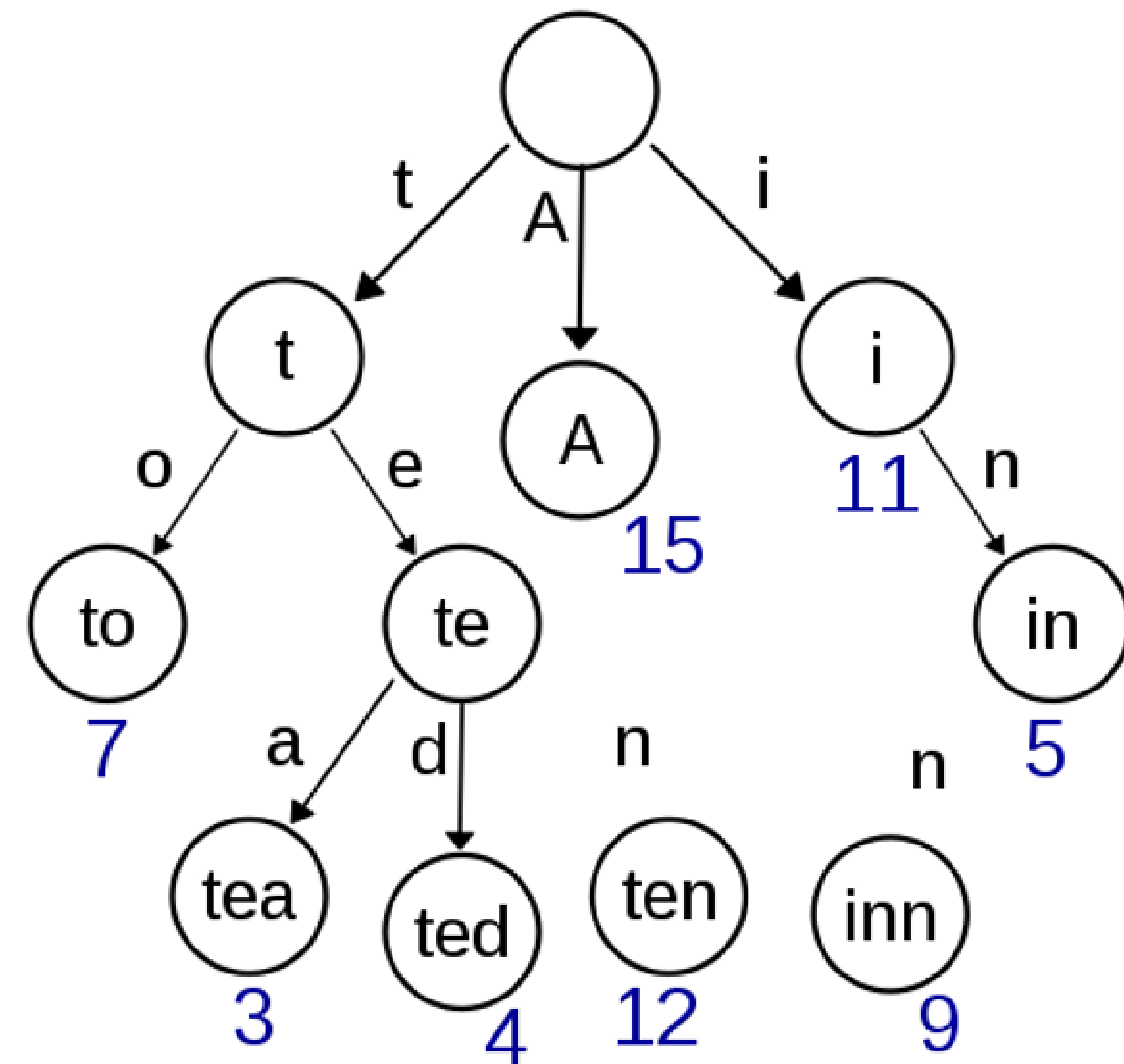
Is the word "trie" in this set?

© 2020 EMQ Technologies Co., Ltd.

What's trie

Also called **digital tree** or **prefix tree**, is a type of search tree, a tree data structure used for locating specific keys from within a set.

--- wikipedia



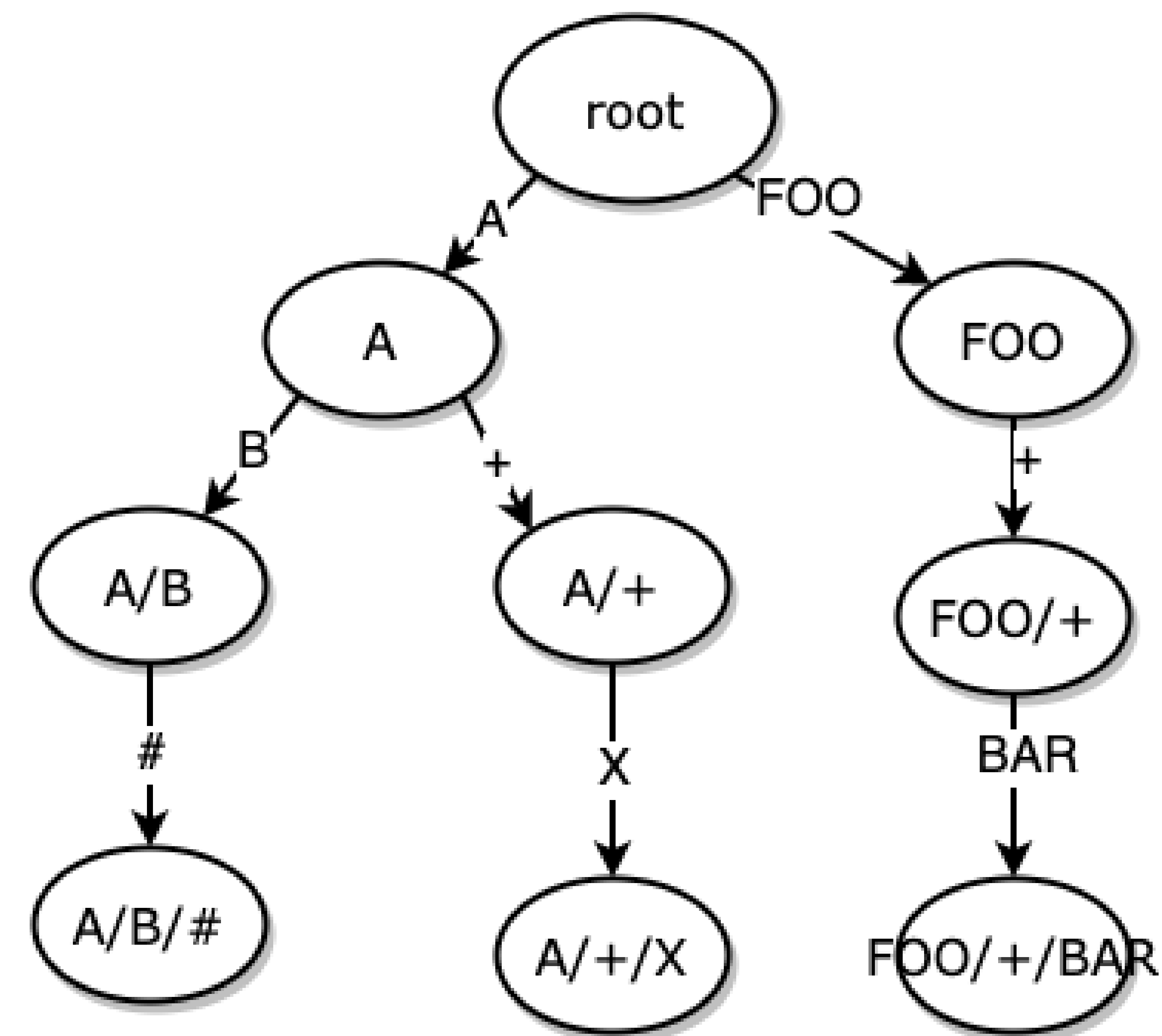
© 2020 EMQ Technologies Co., Ltd.

EMQ

MQTT topics trie

The main difference is that the nodes are not letters, but 'words' of MQTT topic name split by '/'

- A/B/#
- A/+/X
- FOO/+/BAR



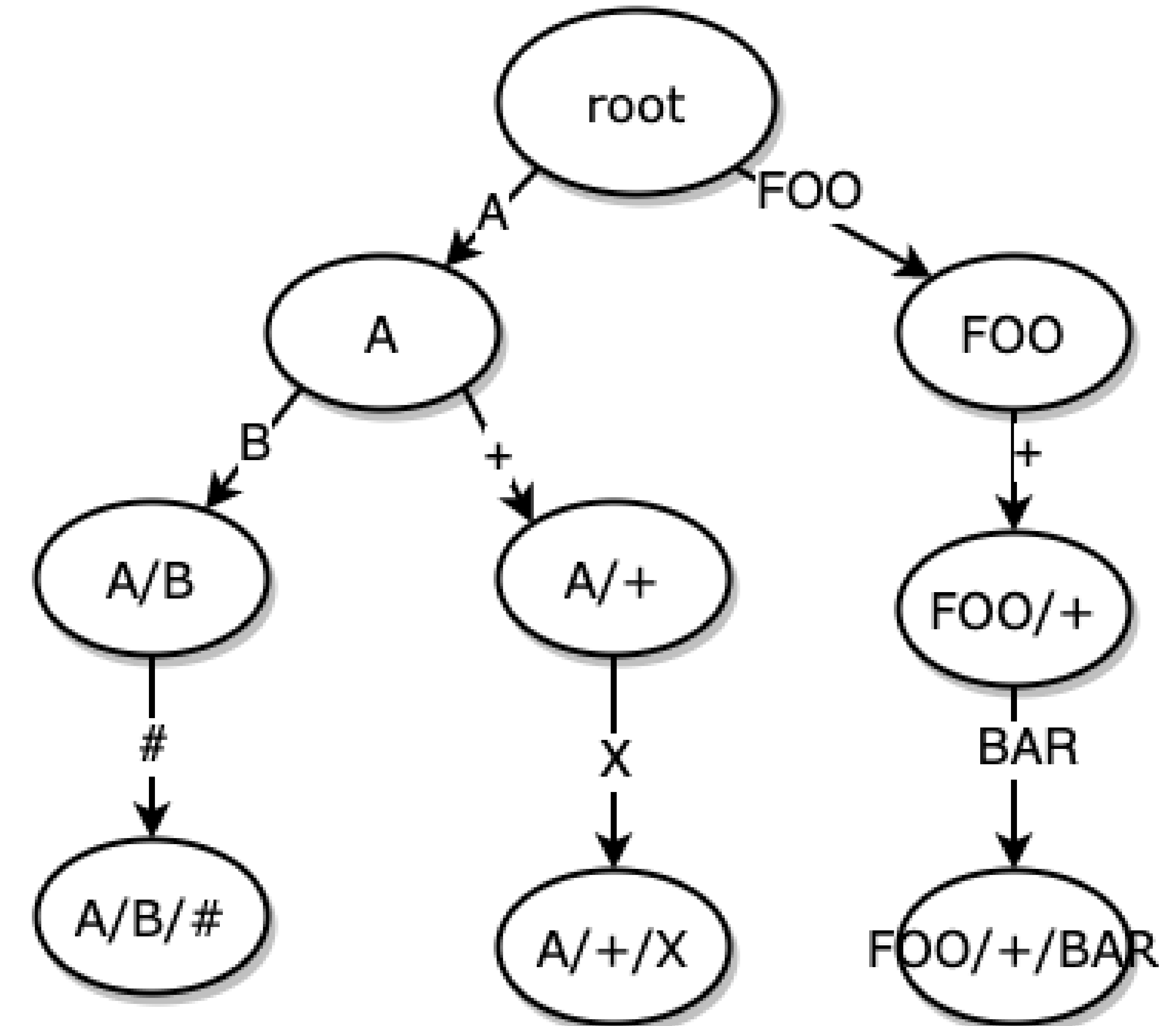
© 2020 EMQ Technologies Co., Ltd.

EMQ

Search procedure in EMQ X (before 4.3)

When searching for "FOO/something/BAR"

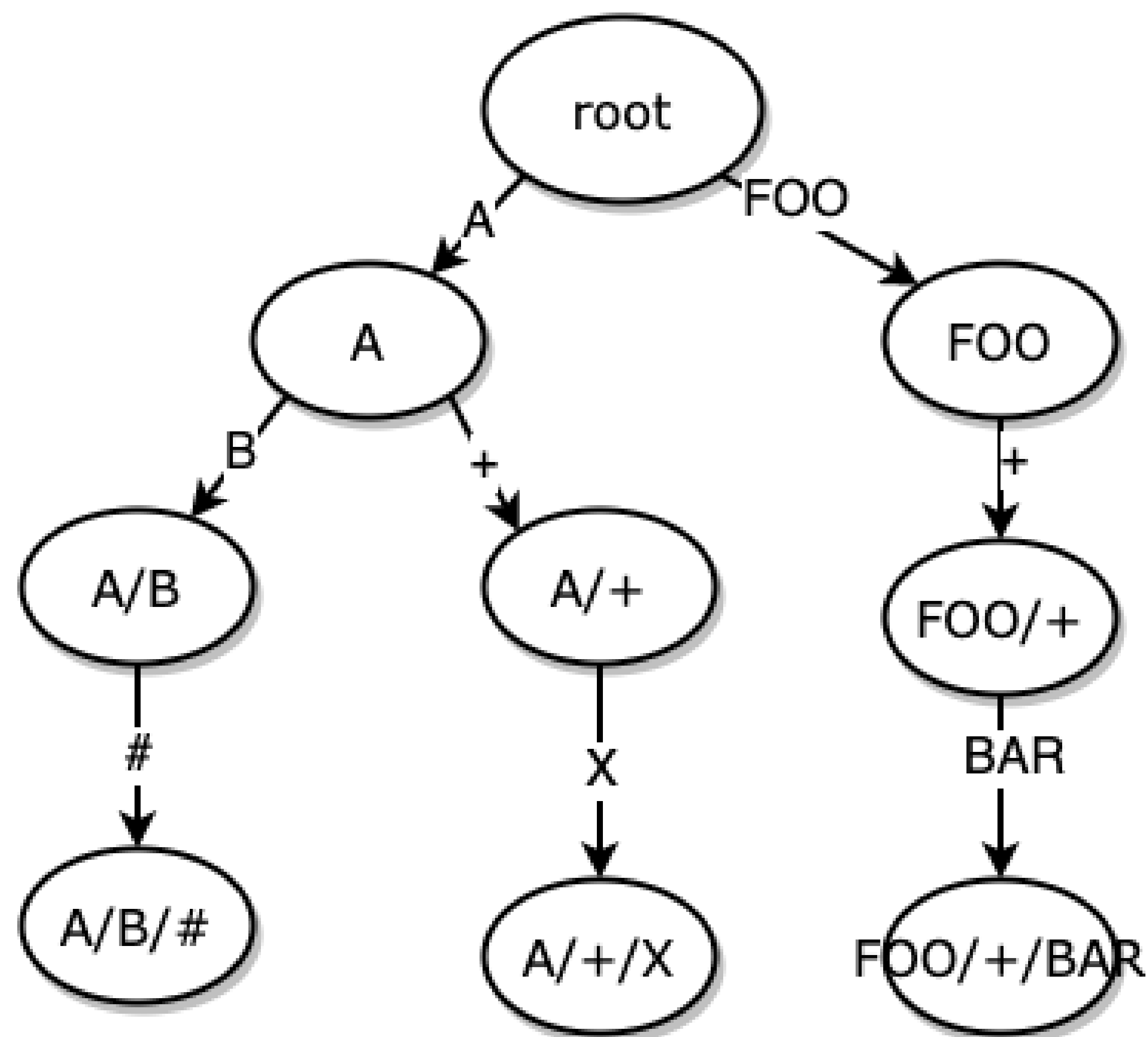
- node root ---> edge > 0
  - edge {root, #} --> not found
  - edge {root, +} --> not found
  - edge {root, FOO} ---> go to node FOO
    - node FOO ---> edge > 0
      - edge {FOO, #} ---> not found
      - edge {FOO, +} ---> to node FOO/+
        - node FOO/+ ---> edge > 0
          - edge {FOO/+, #} ---> not found
          - edge {FOO/+, +} ---> not found
          - edge {FOO/+, BAR} ---> to node FOO/+/BAR
            - **Find node FOO/+/BAR ---> found it!**
      - edge {FOO, something} --> not found

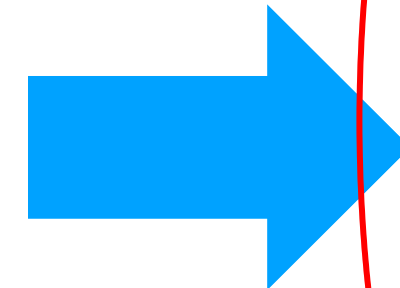


© 2020 EMQ Technologies Co., Ltd.

EMQ

Topics Trie in EMQ X (before 4.3)





Node Table

nodes	edges
root	2
A	2
A/B	1
A/B/#	0
A/+	1
A/+/X	0
FOO	1
FOO/+	1
FOO/+/BAR	0

Edge Table

From Node	With Edge	To Node
root	A	A
A	B	A/B
A/B	#	A/B/#
A	+	A/+
A+	X	A/+/X
root	FOO	FOO
FOO	+	FOO/+
FOO+	BAR	FOO/+/BAR

Think again: do we really need edges?

© 2020 EMQ Technologies Co., Ltd.

EMQ

Topics Trie in EMQ X (since 4.3) -- no compaction

Prefix	count
A	2
A/B	1
A/+	1
FOO	1
FOO/+	1

Topic
A/B/#
A/+/X
FOO/+/BAR

1. The virtual 'root' node is removed
2. No more edge information
3. Internally stored in one table with a tag

© 2020 EMQ Technologies Co., Ltd.

EMQ

Search procedure in EMQ X (since 4.3) -- no compaction

Edges are not stored when inserting, but computed while searching

When searching for "FOO/something/BAR"

- "#" --> not found
- "+" -> not found
- "FOO" --> prefix count > 0
  - "FOO/#" ---> not found
  - "FOO/+" ---> prefix count > 0
    - "FOO/+/#" ---> not found
    - "FOO/+/+" ---> not found
    - "FOO/+/BAR" --> found topic
  - "FOO/something" --> not found

Prefix	count
A	2
A/B	1
A/+	1
FOO	1
FOO/+	1

Topic
A/B/#
A/+/X
FOO/+/BAR

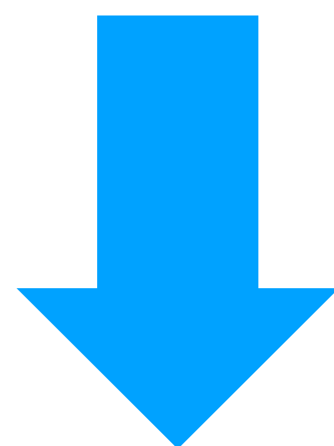
© 2020 EMQ Technologies Co., Ltd.

Topics Trie in EMQ X (since 4.3) -- with compaction

Prefix	count
A	2
A/B	1
A/+	1
FOO	1
FOO/+	1

Topic
A/B/#
A/+/X
FOO/+/BAR

Compaction: topic's non-wildcard levels can be merged



Prefix	count
A/+	1
FOO/+	1

Topic
A/B/#
A/+/X
FOO/+/BAR

© 2020 EMQ Technologies Co., Ltd.

EMQ

Search procedure in EMQ X (since 4.3) -- with compaction

When searching for "FOO/something/BAR"

- "#" --> not found
- "+" -> not found
- "FOO" --> not found
  - "FOO/#" ---> not found
  - "FOO/+" ---> prefix count > 0
    - "FOO/+/#" ---> not found
    - "FOO/+/+" ---> not found
    - "FOO/+/BAR" --> found topic
  - "FOO/something" --> not found
    - "FOO/something/#" -> not found
    - "FOO/something/+" -> not found
    - "FOO/something/BAR" -> not found

Prefix	count
A/+	1
FOO/+	1

Topic
A/B/#
A/+/X
FOO/+/BAR

© 2020 EMQ Technologies Co., Ltd.

EMQ

Disadvantage of compaction

Enumerate all possible prefixes when searching

When publishing to "1/2/3/4/5"

- #
- +
- 1/#
- 1/+
- 1/2/#
- 1/2/+
- 1/2/3/#
- 1/2/3/+
- 1/2/3/4/#
- 1/2/3/4/+
- 1/2/3/4/5
- 1/2/3/4/5/#

Lookups = Level * 2 + 2

© 2020 EMQ Technologies Co., Ltd.

EMQ

4.3-rc.4 vs 4.3.0 (100,000 wildcard topics)

* tested with set type ets

	4.3-rc.4	4.3.0 no-compaction	4.3.0 compaction
Trie tables RAM	68MB	29MB	14MB
Lookup latency (per-client avg)	11ns	20ns	126ns
Insert latency (per-client avg)	5.7ms	3.4ms	500ns

10 subscribers (simulated), 10,000 topics each
Insert Pattern: "device/{{id}}/+/{{num}}/#"
10 publishers (simulated), 100,000 lookups each
Search Pattern: "device/{{id}}/foo/{{num}}/bar"

© 2020 EMQ Technologies Co., Ltd.

EMQ

4.3-rc.4 vs 4.3.0 (80,000 wildcard topics)

* tested with set type ets
* CPU saturated

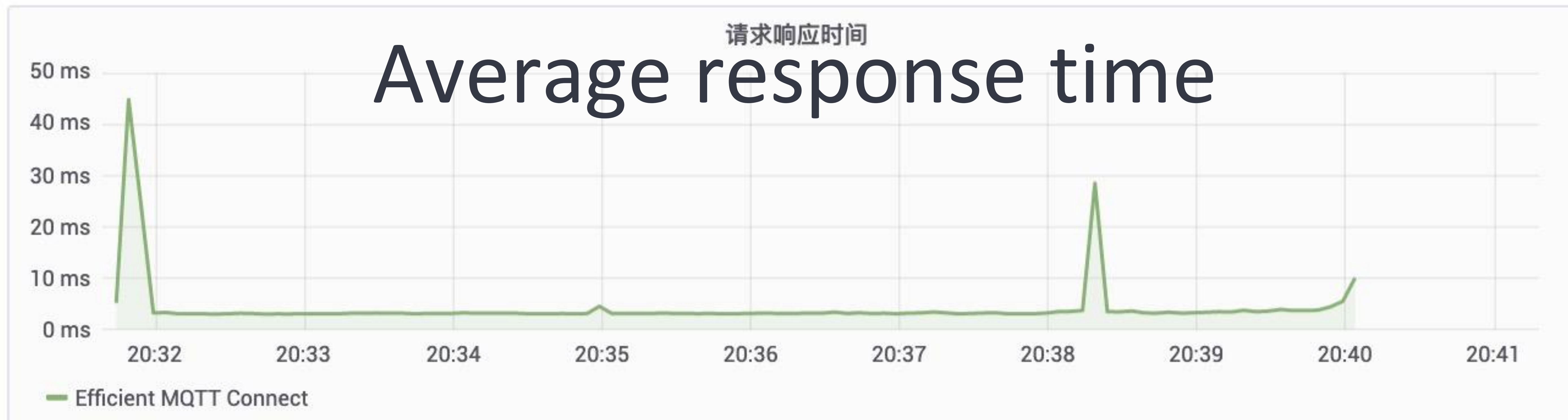
	4.3-rc.4	4.3.0 no-compaction	4.3.0 compaction
Trie tables RAM	109MB	46MB	23MB
Lookup latency (per-client avg)	40ms	10ms	44ms
Insert latency (per-client avg)	3.6s	1.3s	282ms

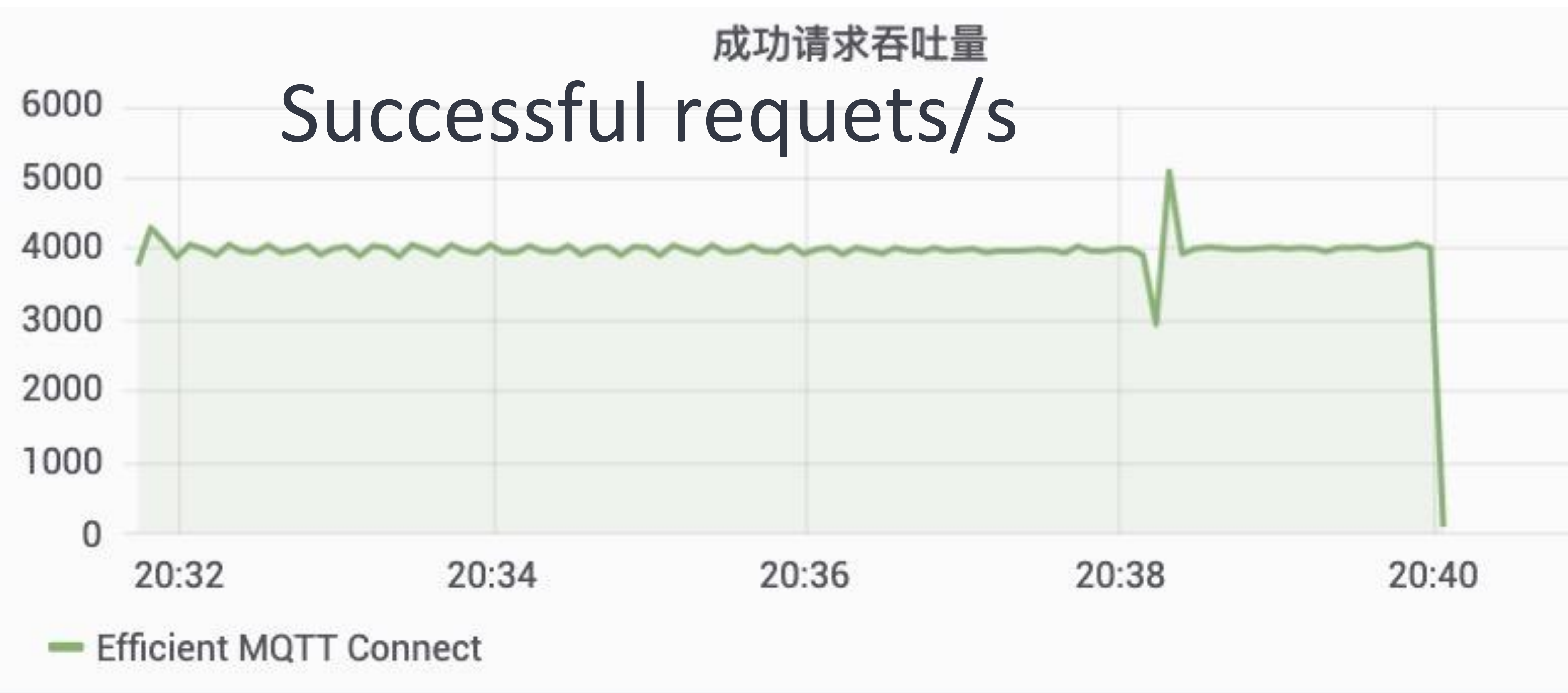
8,000 subscribers (simulated), 10 topics each
Insert Pattern: "device/{{id}}/+/{{num}}/#"
80,000 publishers (simulated), 100 lookups each
Search Pattern: "device/{{id}}/foo/{{num}}/bar"

© 2020 EMQ Technologies Co., Ltd.

EMQ

Cluster test (2 million subscribers)





3-Node Cluster

8 Cores
16 GB RAM
4000 Subscribe/s

© 2020 EMQ Technologies Co., Ltd.

EMQ

Best practices

- Avoid sharing prefixes between subscribers
  - Good example: foo/{{client_id}}/+/bar
  - Bad example: foo/+/{{client_id}}/bar

- Avoid using topics with too many levels
  - Good example: foo/{{client_id}}/my.application.namespace/#
  - Bad example: foo/{{client_id}}/my/application/namespace/#

© 2020 EMQ Technologies Co., Ltd.

EMQ

Summary

- Trie compaction is made default in 4.3.0
- Change config with:

  broker.perf.trie_compaction=false # in emqx.conf
  OR
  export EMQX_BROKER__PERF__TRIE_COMPACTION=false

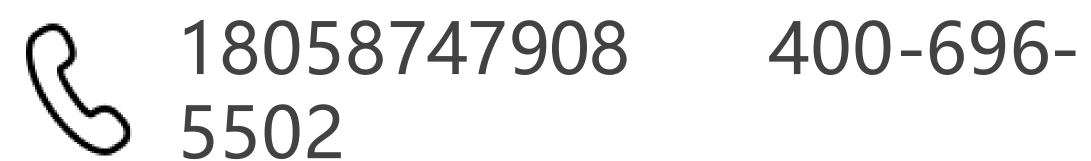
- Performance compare base (v4.3-rc.4) in this presentation includes other optimizations which are demoed in another session by William Yang

© 2020 EMQ Technologies Co., Ltd.

EMQ

Thank You

Kudos to William Yang for the trie compaction idea



18058747908 400-696-5502

contact@emqx.io

© 2020 EMQ Technologies Co., Ltd