

From MySQL to Redis

or “Welcome in the 10s”

Raphaël Vinot

Chaos Computer Club Trier

Table of contents

1

Context

- Me, Myself and I
- BGP Ranking
- MySQL

2

Redis

- What is it?
- Benchmark

3

Migration

- Backend
- Access to the information

Content

1 Context

- Me, Myself and I
- BGP Ranking
- MySQL

2 Redis

- What is it?
- Benchmark

3 Migration

- Backend
- Access to the information

Me, Myself and I

- ① truly **hate** *SQL
 - yes, no objectivity
- ② lazy
 - but you knew it
- ③ like Active Record (elixir in Python)
- ④ KISS
 - hate write documentation, see 2.

Content

1 Context

- Me, Myself and I
- BGP Ranking
- MySQL

2 Redis

- What is it?
- Benchmark

3 Migration

- Backend
- Access to the information

BGP Ranking

- Goal: rank the Autonomous Systems (AS)
- Sources:
 - many free and non-free datasets
 - some big datasets
- Requirements:
 - fast
 - use “few” memory
 - 32b system, 4Gb max/process
 - 64b system, “no limits” :)

BGP Ranking

- Goal: rank the Autonomous Systems (AS)
- Sources:
 - many free and non-free datasets
 - some big datasets
- Requirements:
 - fast
 - use “few” memory
 - 32b system, 4Gb max/process
 - 64b system, “no limits” :)

Content

1 Context

- Me, Myself and I
- BGP Ranking
- MySQL

2 Redis

- What is it?
- Benchmark

3 Migration

- Backend
- Access to the information

MySQL

- uptime: 3 month
- size: 10Gb, ~20 Millions records, biggest table: 17 Millions
- It works well...
- ... until I found a bug and had to modify an index.
- Consequence: unusable
- Solution:
 - add more logic in the usage of MySQL
 - Or... change the backend

MySQL

- uptime: 3 month
- size: 10Gb, ~20 Millions records, biggest table: 17 Millions
- It works well...
- ... until I found a bug and had to modify an index.
- Consequence: unusable
- Solution:
 - add more logic in the usage of MySQL
 - Or... change the backend

Content

1

Context

- Me, Myself and I
- BGP Ranking
- MySQL

2

Redis

- What is it?
- Benchmark

3

Migration

- Backend
- Access to the information

Redis is...

- Key-value **store** database
- Data in memory and/or on the disk
- License: BSD
- Usage:
 - shared memory
 - saving data

Types

- strings
- lists
- sets/zsets
- hash

One ~~Two~~ Three more things...

- Pipeline
 - at least 3x faster
 - non-transactional mode
- Slave instances
- Soon: disk storage (no more swap)

Content

1

Context

- Me, Myself and I
- BGP Ranking
- MySQL

2

Redis

- What is it?
- **Benchmark**

3

Migration

- Backend
- Access to the information

Benchmark

- The test was done with 50 simultaneous clients performing 100000 requests.
- The value SET and GET is a 256 bytes string.
- The Linux box is running Linux 2.6, it's Xeon X3320 2.5Ghz.
- Text executed using the loopback interface (127.0.0.1).
 - **Results:** about 110000 SETs per second, about 81000 GETs per second.
- See on the Official Website
- Well, marketing ?

Benchmark

- The test was done with 50 simultaneous clients performing 100000 requests.
- The value SET and GET is a 256 bytes string.
- The Linux box is running Linux 2.6, it's Xeon X3320 2.5Ghz.
- Text executed using the loopback interface (127.0.0.1).
 - **Results:** about 110000 SETs per second, about 81000 GETs per second.
- See on the Official Website
- Well, marketing ?

Benchmark

- MySQL
 - 1 Million IPs: >4 Hours
 - 70.000 Ranks computed in >20 min
- Redis:
 - 1 Million IPs: 30 min
 - 70.000 Ranks: 1 min

Content

- 1 Context
 - Me, Myself and I
 - BGP Ranking
 - MySQL
- 2 Redis
 - What is it?
 - Benchmark
- 3 Migration
 - Backend
 - Access to the information

Architecture: MySQL

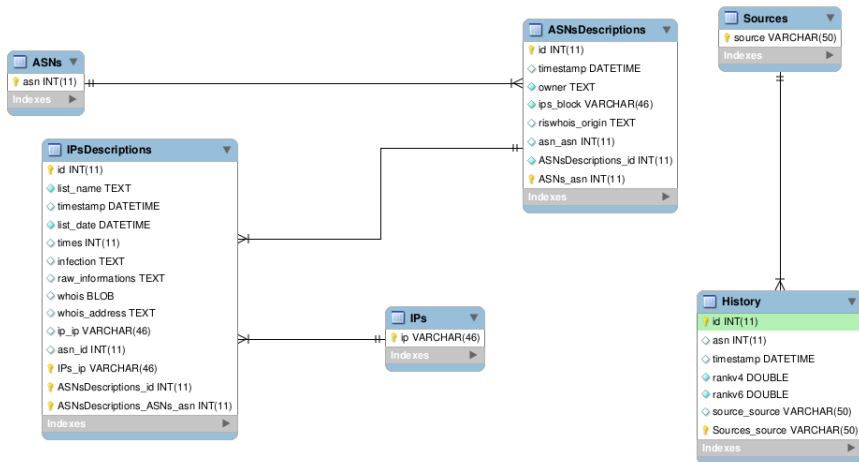


Figure: MySQL Schema

Architecture: Redis - Instances

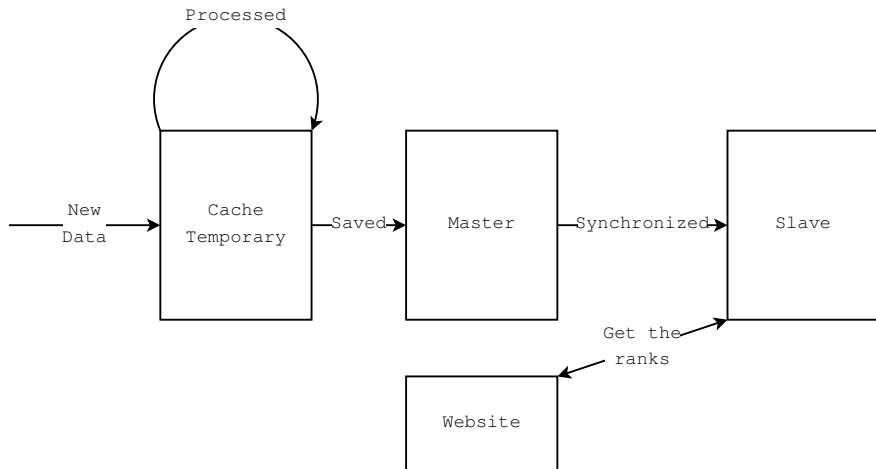


Figure: Instances

Architecture: Redis - Content

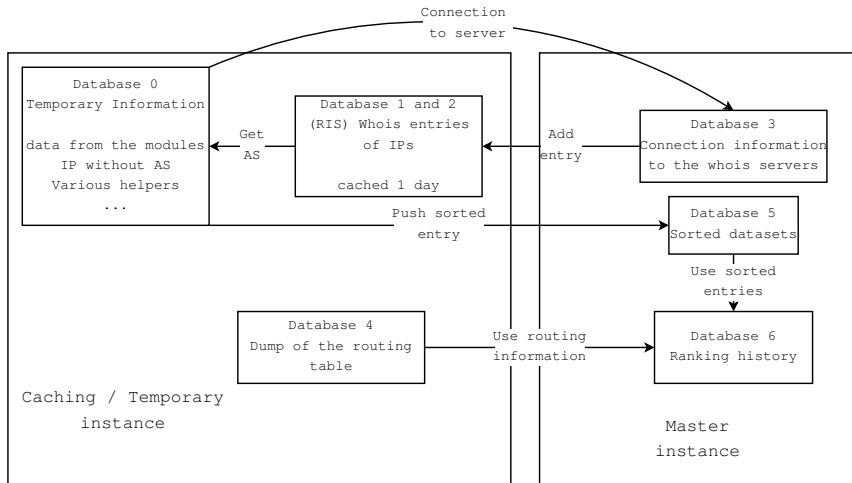


Figure: Redis schema - simple

Content

- 1 Context
 - Me, Myself and I
 - BGP Ranking
 - MySQL
- 2 Redis
 - What is it?
 - Benchmark
- 3 Migration
 - Backend
 - Access to the information

Motto

Worst case: $M * O(\log(n))$

M: Number of queries to get a result

n: Number of entries in the database

- queries
- ram
- cpu

Good key

- easy to remember
 - order of the fields is important
- contains information
 - KISS
- **never** depend directly on the number of entries in the database
 - Bad example: ip|YYYY-MM-DD|source
- contains (at least) **all** the information you need
 - having a bit more is not a big deal

Examples 1

- 1 YYYY-MM-DD|sources
 - list of available sources
- 2 YYYY-MM-DD|ListOfTheBadGuys|asns
 - ASNs found in the ListOfTheBadGuys dataset, for the day
- 3 YYYY-MM-DD|ListOfTheBadGuys|asns__details
 - the same but for the subnets

Examples 2

- ① `asn|timestamp|YYYY-MM-DD|ListOfTheBadGuys`
 - IPs associated with the subnet
 - ② `asn|YYYY-MM-DD|ListOfTheBadGuys|v4`
 - rank of the AS
 - ③ `asn|YYYY-MM-DD|ListOfTheBadGuys|v4|details`
 - ranks of all the subnets, in a sorted set :)
- Note: it also works in IPv6, I just don't have any dataset :)

Examples 2

- ① `asn|timestamp|YYYY-MM-DD|ListOfTheBadGuys`
 - IPs associated with the subnet
 - ② `asn|YYYY-MM-DD|ListOfTheBadGuys|v4`
 - rank of the AS
 - ③ `asn|YYYY-MM-DD|ListOfTheBadGuys|v4|details`
 - ranks of all the subnets, in a sorted set :)
- Note: it also works in IPv6, I just don't have any dataset :)

Fragen ?

Bonus: shared memory and multiprocessing

Easy way

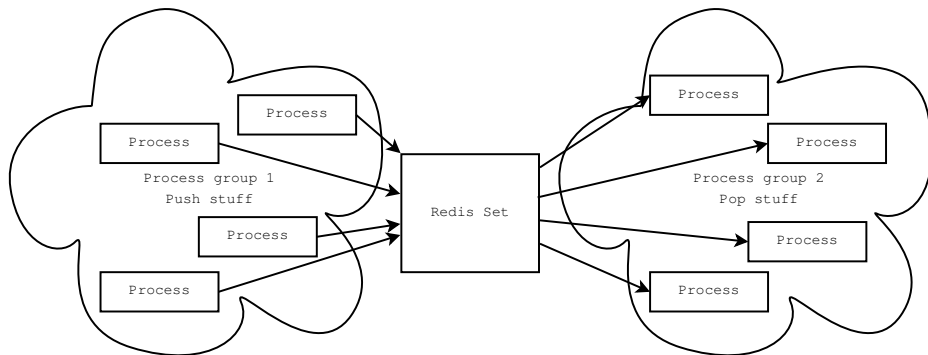


Figure: Multiprocessing Simple

Bonus: shared memory and multiprocessing

Tricky way

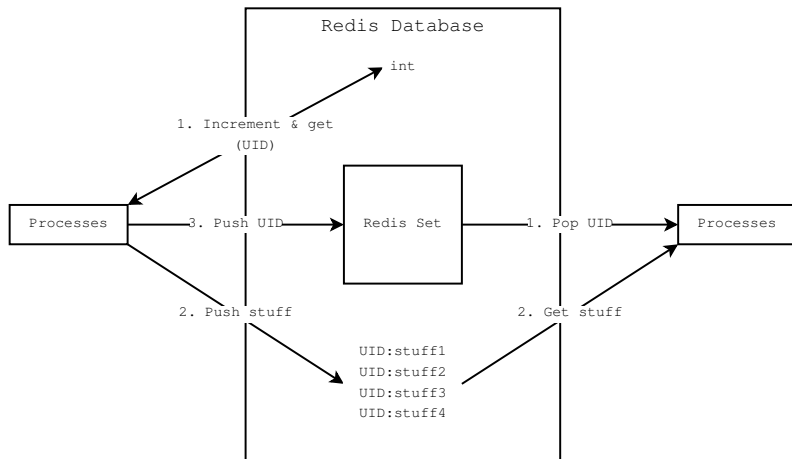


Figure: Multiprocessing tricky