Performance Enhancement and Best Practices

# Elevating Excellence of Kong Gateway

Wenchao Xiang
Senior Software Engineer, Kong

CONF42CAST

## Recent Performance Improvements

Plugins and core components

## How We Do Performance at Kong

Methodology


## Tuning the Gateway

Best practices


Recent Improvements

# Efficiency and Performance

Speed Up

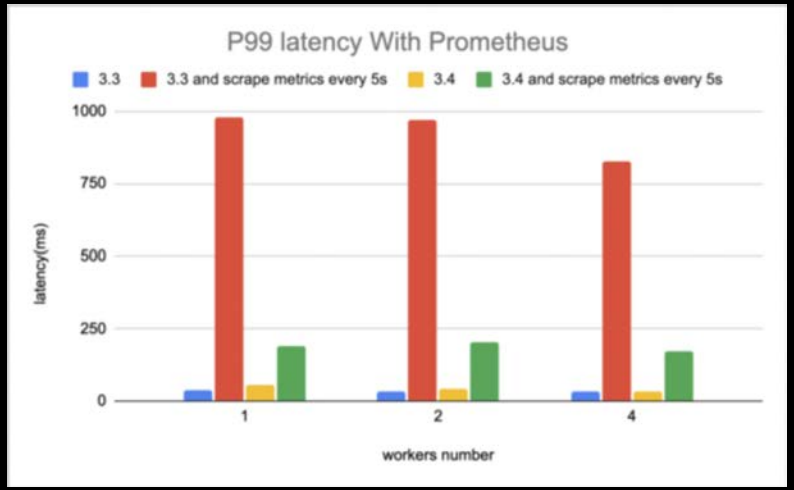
# Prometheus Scaping

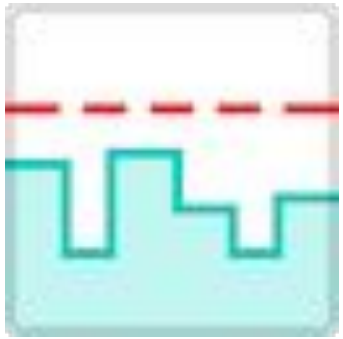
- Improve P99 during scaping
- Reduce high cardinality metrics

## Prometheus Plugin Improvements

1. Avoid excessive creation of temporary tables
2. Yield in long loop in upstream iteration
3. Fix NYI in full_metrics_name function, and reduce gsub function call

Release in 3.4





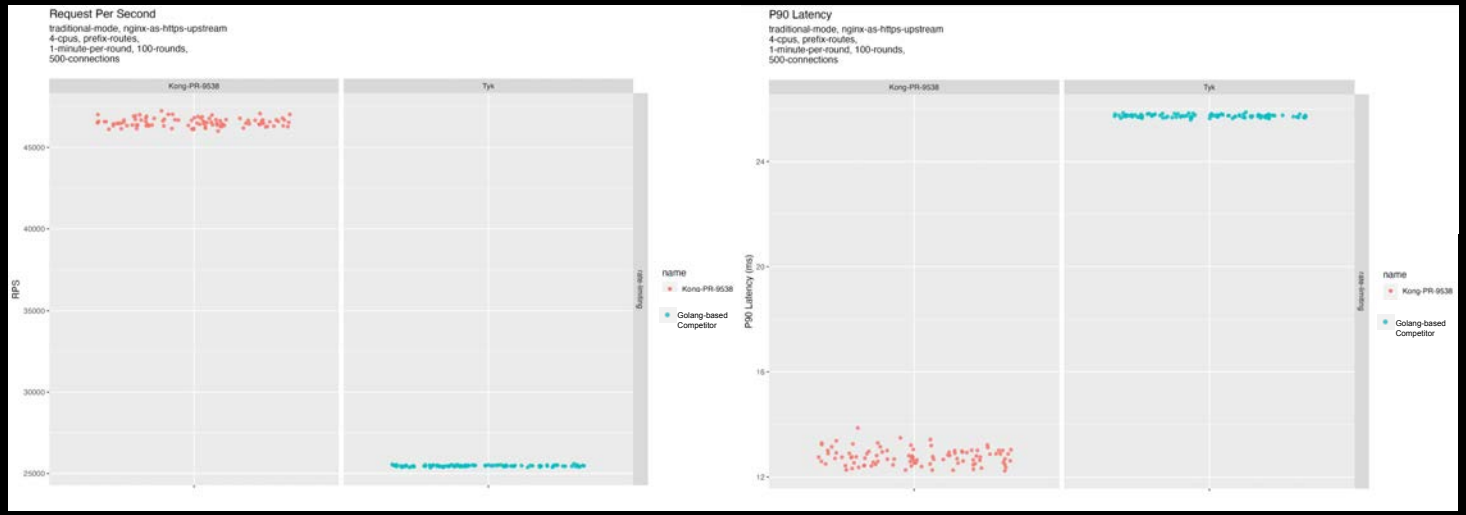
Speed Up

# Rate-limiting Plugin

- Increase P99 and RPS
- Batching with Redis

## Rate-Limiting Plugin Improvements

- Adds a new configuration `sync_rate` to the `redis` policy, which synchronizes metrics to redis periodically instead of on every request.



Speed Up

# The Router

- A DSL-based approach
- Rewritten in Rust

## Router Improvements

1. Optimize router rebuild to take less CPU
    a. Worst case latency from 20 secs to 5 secs (~75% improvement in wall clock time, tested with 10,000 routes)
    b. Released in 2.8

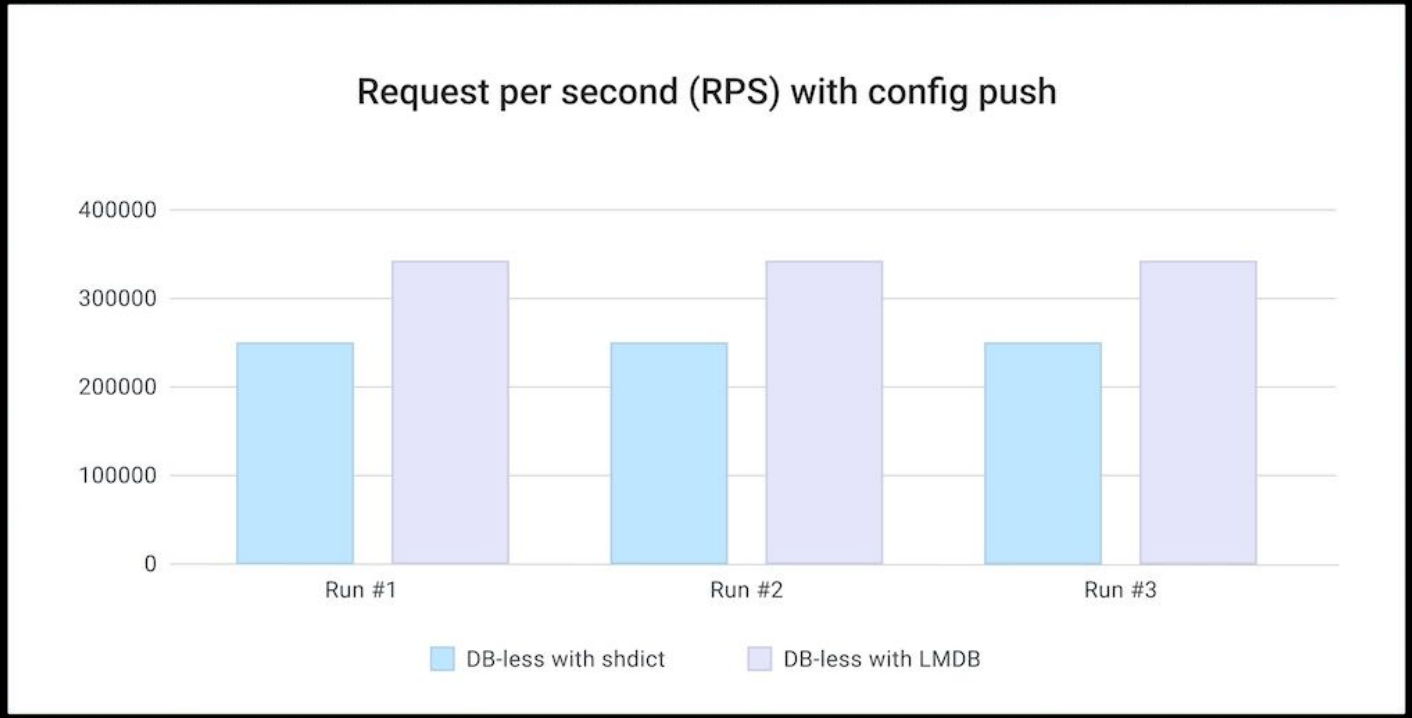
2. Conditional rebuild of Router
    a. Only rebuild router for if routes are impacted
    b. Release in 3.0


New Storage Engine for Hybrid and DB-less

# Lightning Memory-Mapped Database Manager (LMDB)

- Less memory usage
- Transactional embedded key/value store

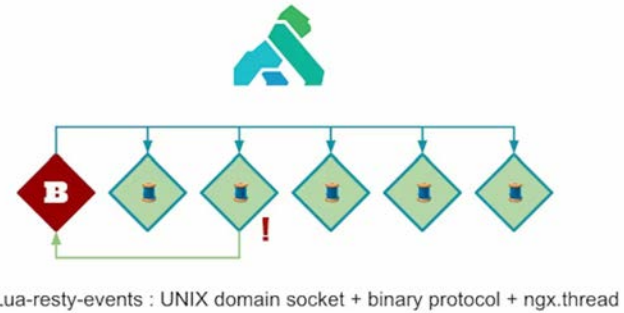
# Benchmark


Request per second (RPS) with config push
400000
300000
200000
100000
0
Run #1
Run #2
Run #3
DB-less with shdict
DB-less with LMDB

## Improvements available

LMDB as backend to increase config read throughput during rebuild.

a. 50%~70% additional drop in rebuild time, particularly when number of workers is large
b. Release in 3.0




Lua-resty-events : UNIX domain socket + binary protocol + ngx.thread

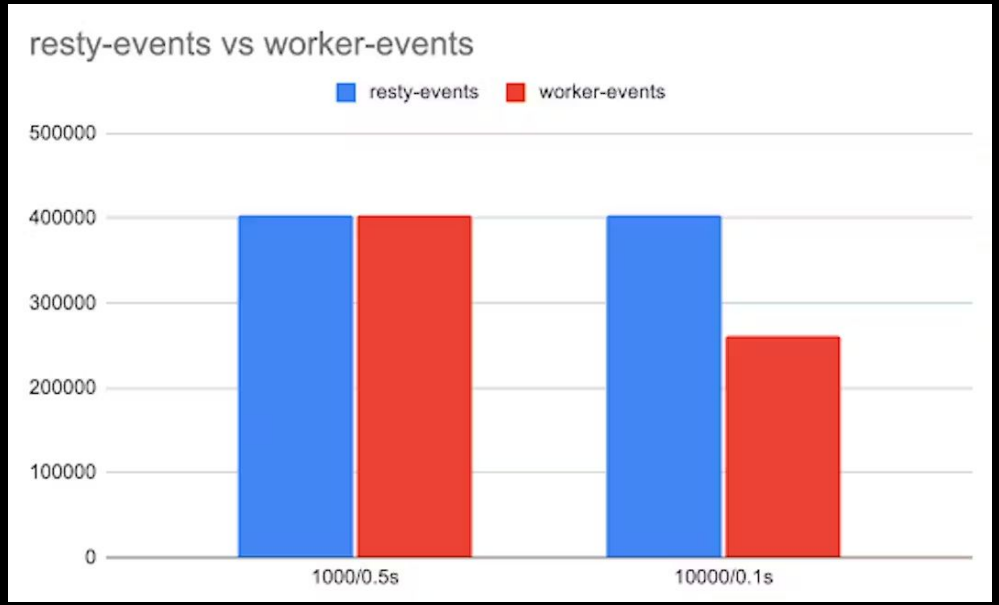
## New Event System

# Inter process Pub/Sub pattern for Nginx worker processes

- Event-broker
- Event-publisher
- Event-subscriber

- Eliminate the cost of lock and poll

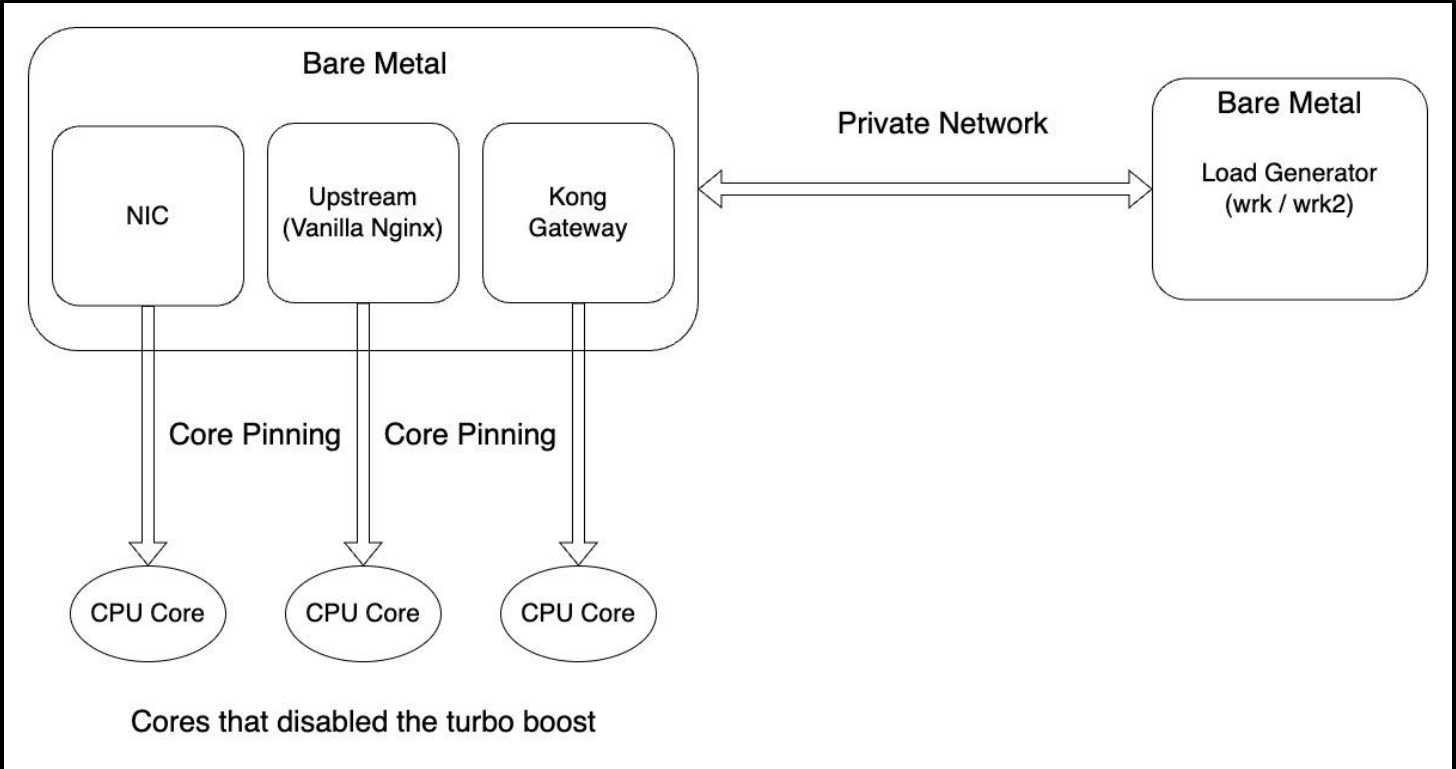
# Benchmark


resty-events vs worker-events
resty-events
worker-events
500000
400000
300000
200000
100000
0
1000/0.5s
10000/0.1s


Practices

# How we do performance at Kong?

# Infrastructure



# Infrastructure

Everything is for benchmark

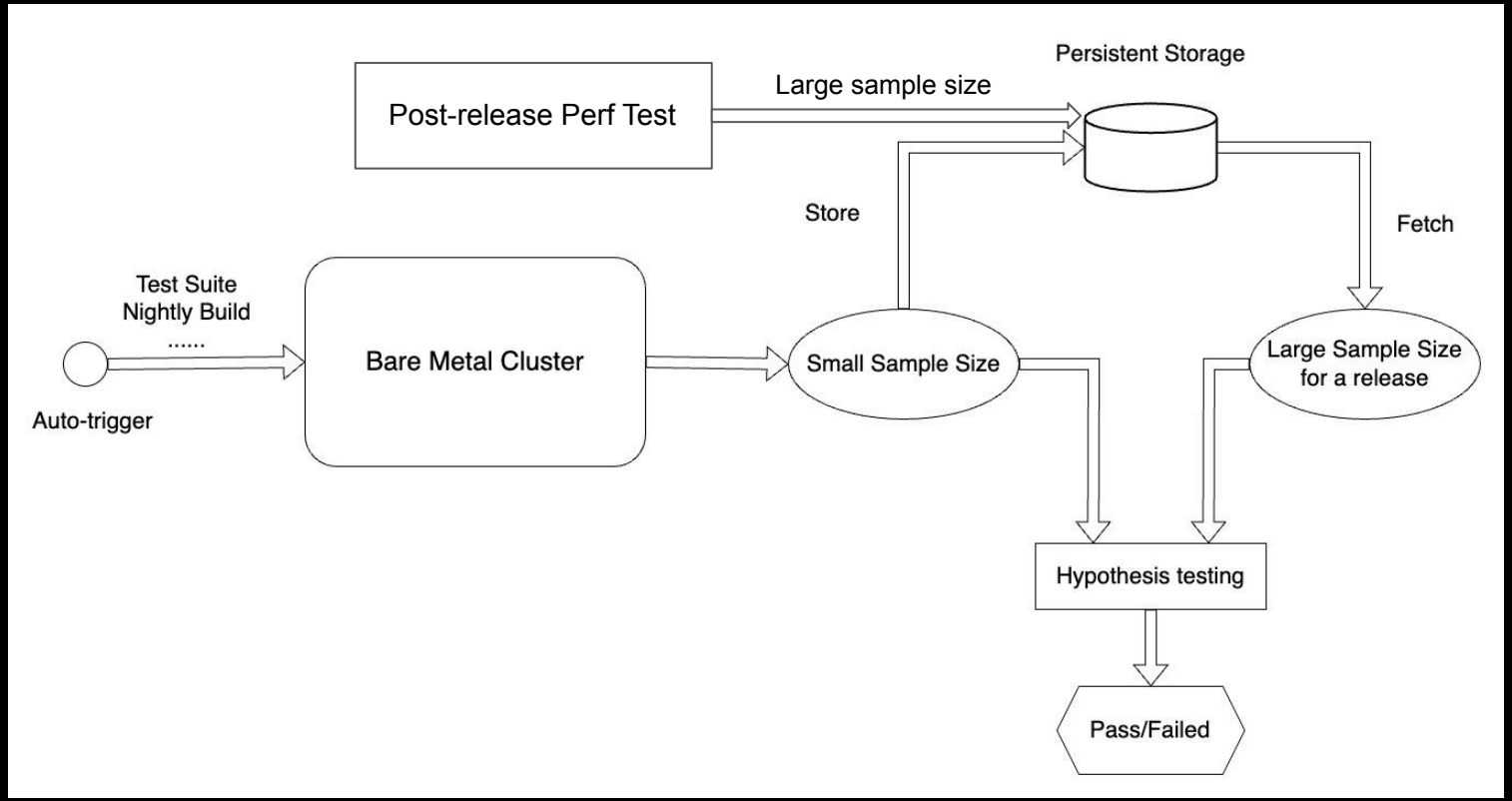
- Bare metal makes our environment more reproducible.
- Put upstream and Kong Gateway on the loopback to give wrk more bandwidth.
- Core pinning to avoid the competition of upstream, Kong Gateway, and NIC interrupt.
- Put Kong Gateway and wrk into the same private network to get rid of some uncontrollable factors.
- Disable turbo boost on CPU makes data more stable.

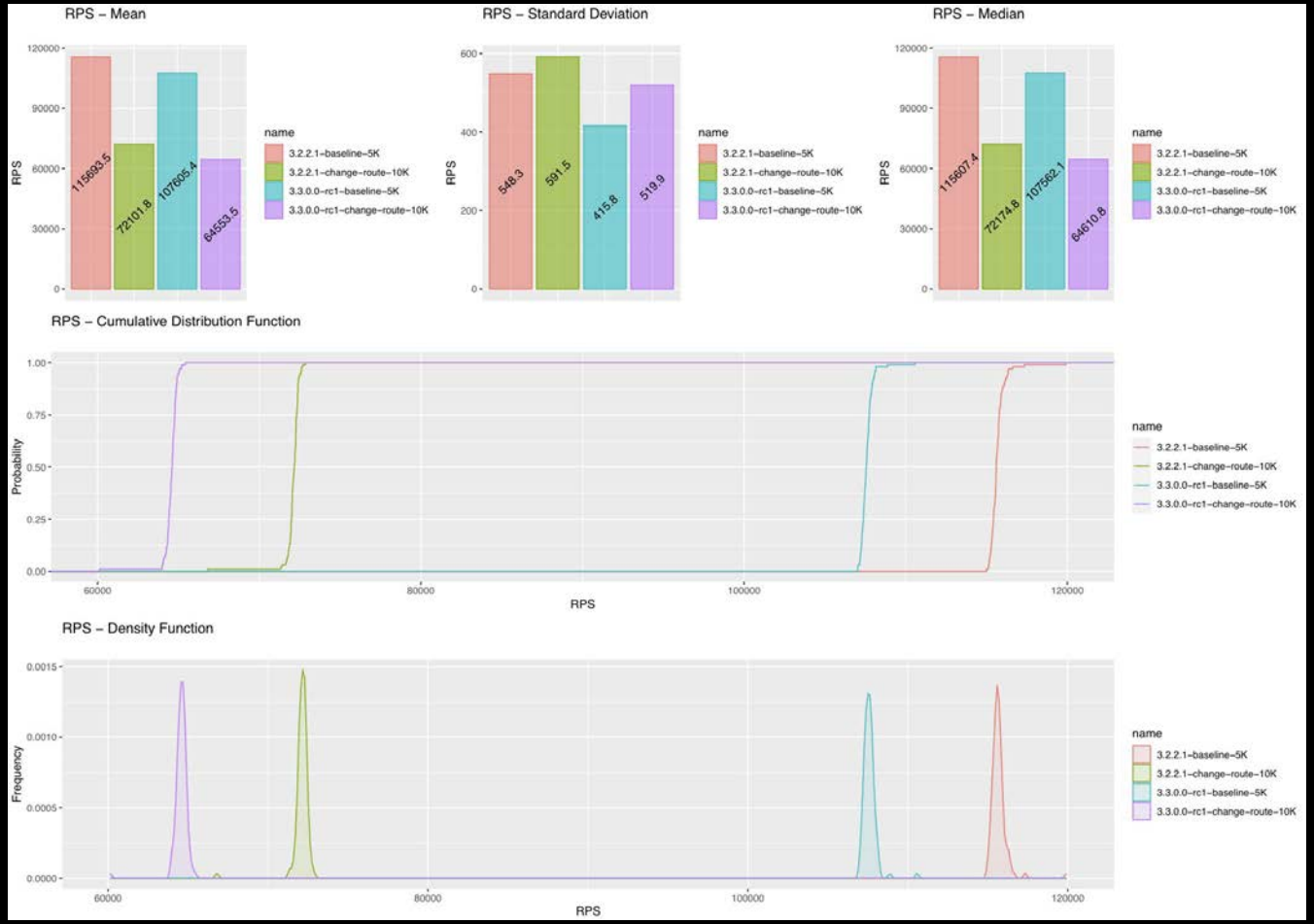
# Statistics

Check RPS, Latency, Memory By:

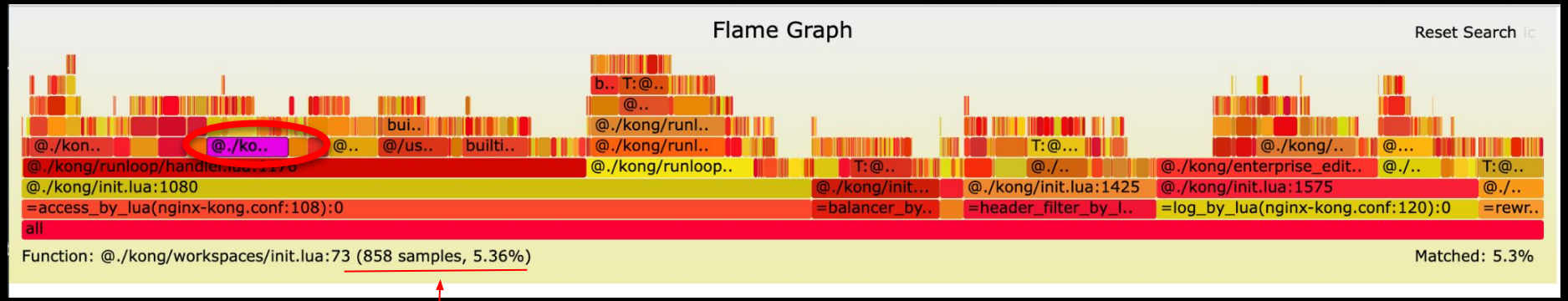
- Mean, median, standard deviation of each metric – By Bar Chart
- Distribution of each metric – By PDF and CDF
- CDF: Cumulative Distribution Function
- PDF: Probability Density Function
- Large sample size (>= 100) per test suite for each release.

# Nightly Performance Test




RPS – Mean
RPS
115693.5
72101.8
107605.4
64553.5
name
3.2.2.1–baseline–5K
3.2.2.1–change–route–10K
3.3.0.0–rc1–baseline–5K
3.3.0.0–rc1–change–route–10K
RPS – Standard Deviation
548.3
591.5
415.8
519.9
RPS – Median
115607.4
72174.8
107562.1
64610.8
RPS – Cumulative Distribution Function
Probability
RPS
RPS – Density Function
Frequency

# Infrastructure

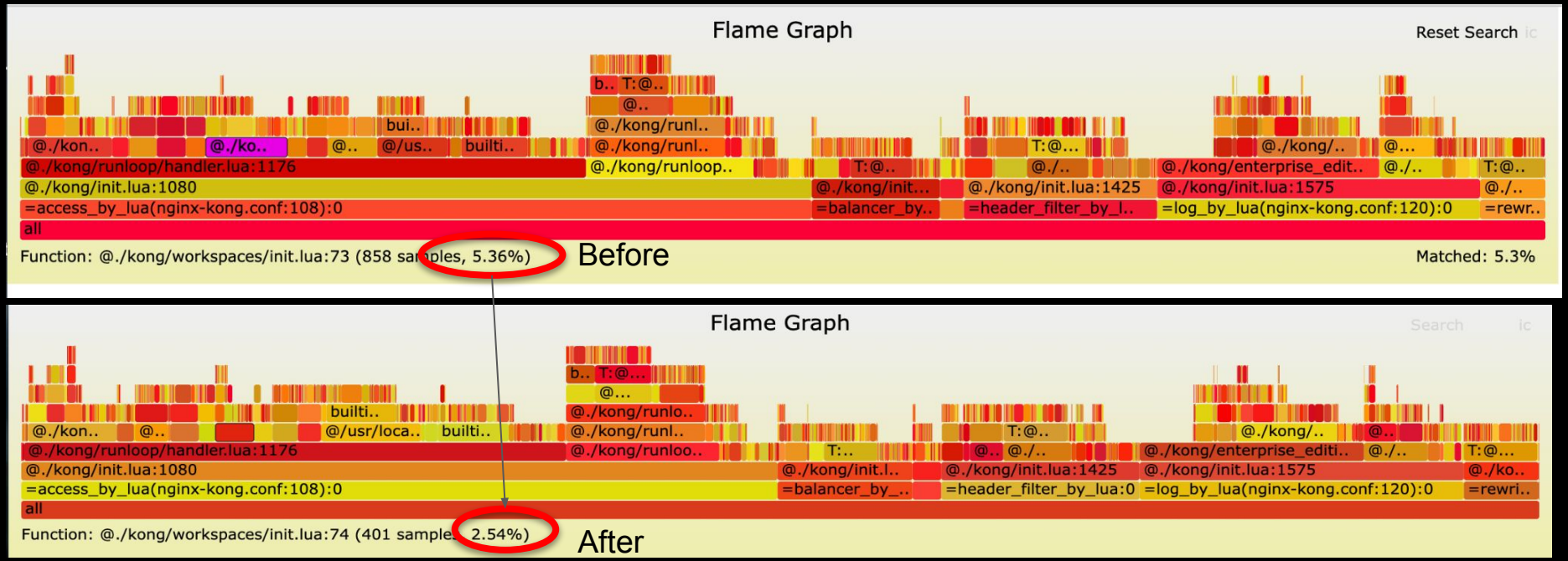


# Per request -> Per config init

Route <-> workspace mapping generating is expensive

```
     418 +  local workspace_names_init_cache = {}
355  419    if not kong.core_cache and db.strategy ~= "off" then
356  420      services_init_cache, err = build_services_init_cache(db)
357  421      if err then
358  422        services_init_cache = {}
359  423        log(WARN, "could not build services init cache: ", err)
360  424      end
     425 +    workspace_names_init_cache, err = build_workspace_names_init_cache(db)
     426 +    if err then
     427 +      workspace_names_init_cache = {}
     428 +      log(WARN, "could not build workspace names init cache: ", err)
     429 +    end
361  430    end
362  431
363  432    local detect_changes = db.strategy ~= "off" and kong.core_cache
         @@ -388,6 +457,13 @@ local function new_router(version)
388  457 +        return nil, err
389  458        end
390  459
     460 +      local ws_name, err = get_workspace_name_for_route(db, route, workspace_names_init_cache)
     461 +      if err then
     462 +        return nil, err
     463 +      end
     464 +
     465 +      route.ws_name = ws_name
     466 +
391  467        -- routes with no services are added to router
```

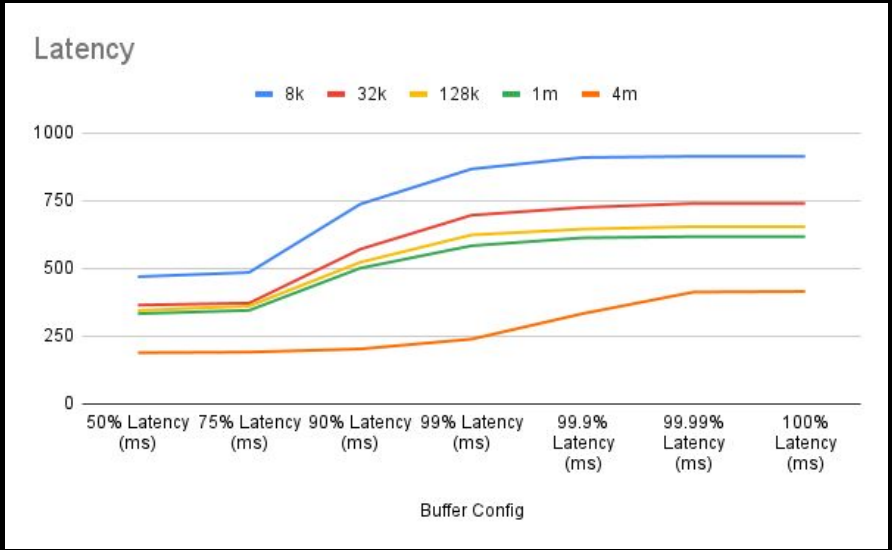
# Case Study

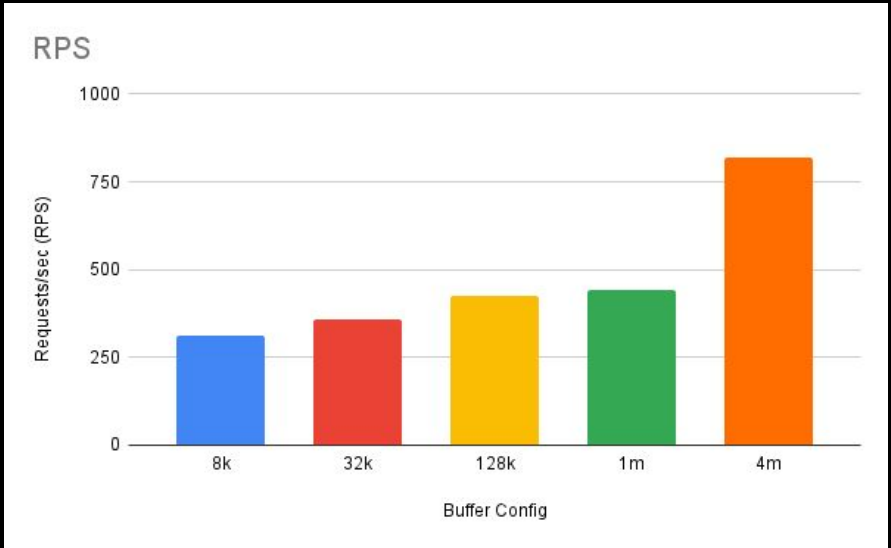

Flame Graph
Reset Search
@./kong/runloop/handler.lua:1176
@./kong/init.lua:1080
=access_by_lua(nginx-kong.conf:108):0
all
Function: @./kong/workspaces/init.lua:73 (858 samples, 5.36%)
Before
Matched: 5.3%
Flame Graph
@./kong/runloop/handler.lua:1176
@./kong/init.lua:1080
=access_by_lua(nginx-kong.conf:108):0
all
Function: @./kong/workspaces/init.lua:74 (401 samples, 2.54%)
After


Practices

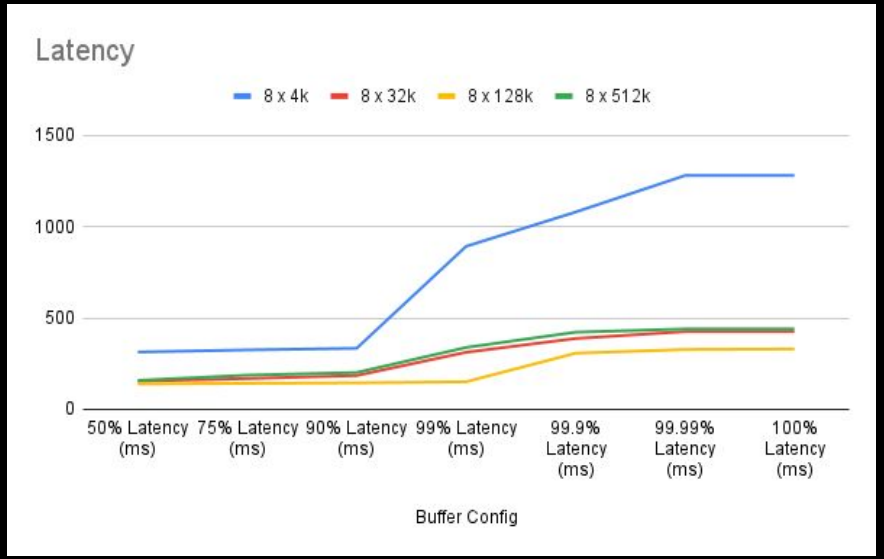
# Tuning the Gateway

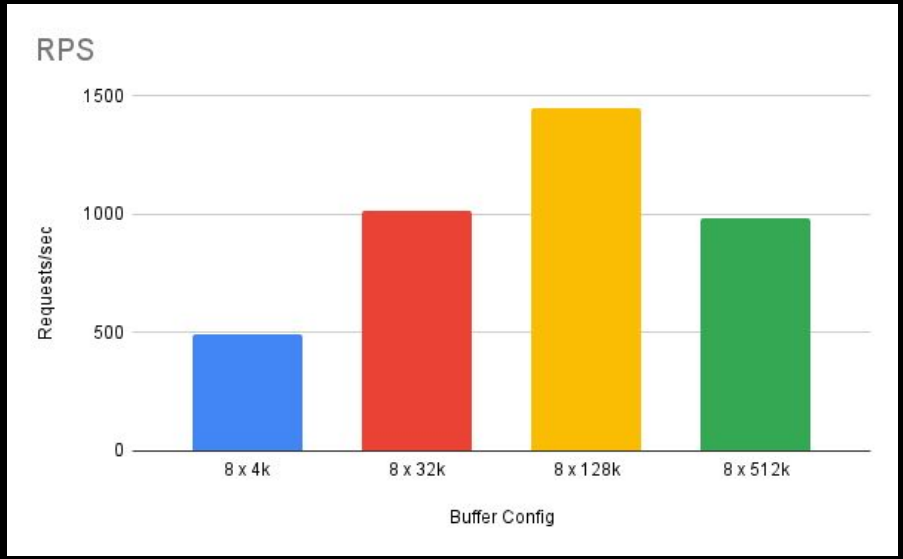
# Buffering - Client request body buffer



# Buffering - Upstream response buffer



## Scenarios not suitable for using buffers

- High-priority real-time scenarios
- Long-lasting connections/streamed transmission
- Line-by-line data processing
- Limited memory environments