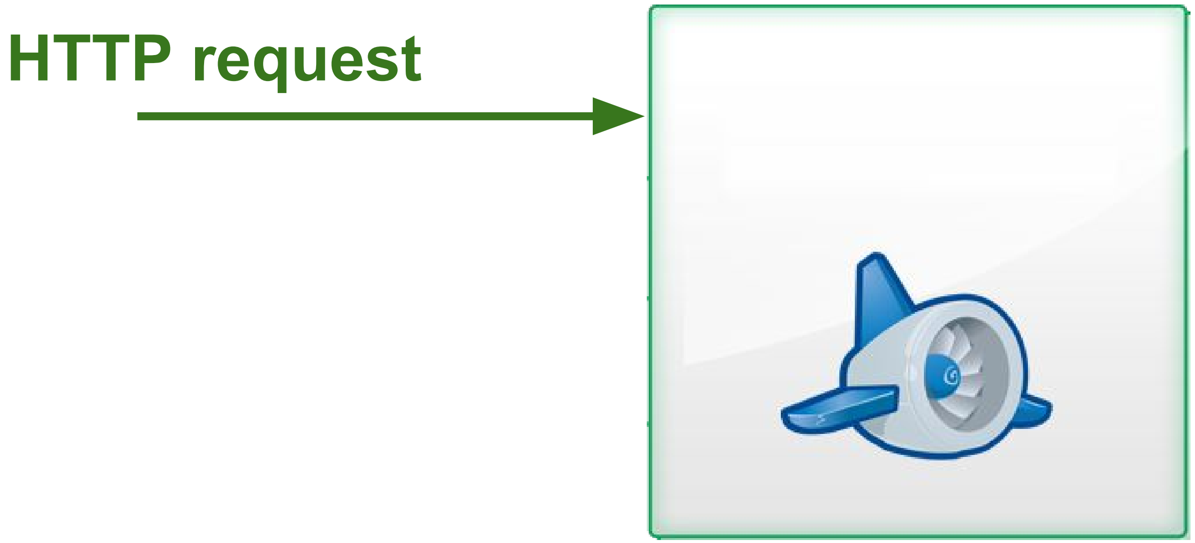
# go appengine()

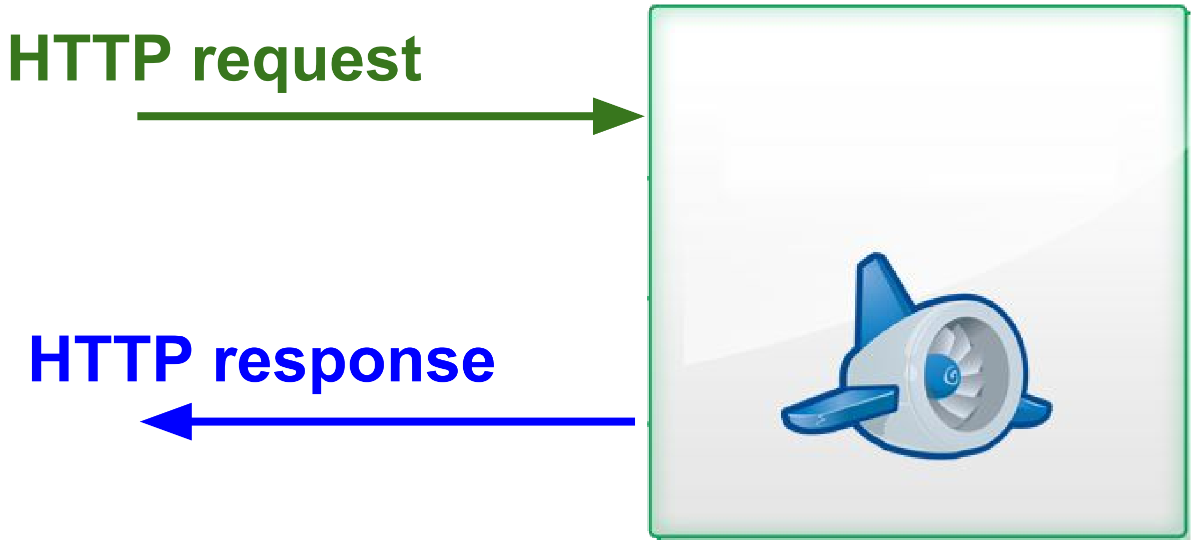
by Valentin Deleplace

## Server instance

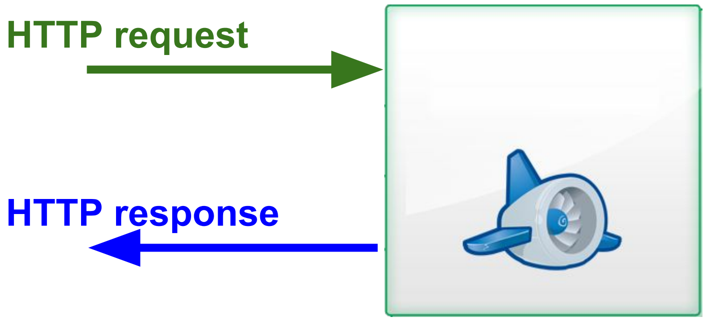
## Server instance



# Server instance



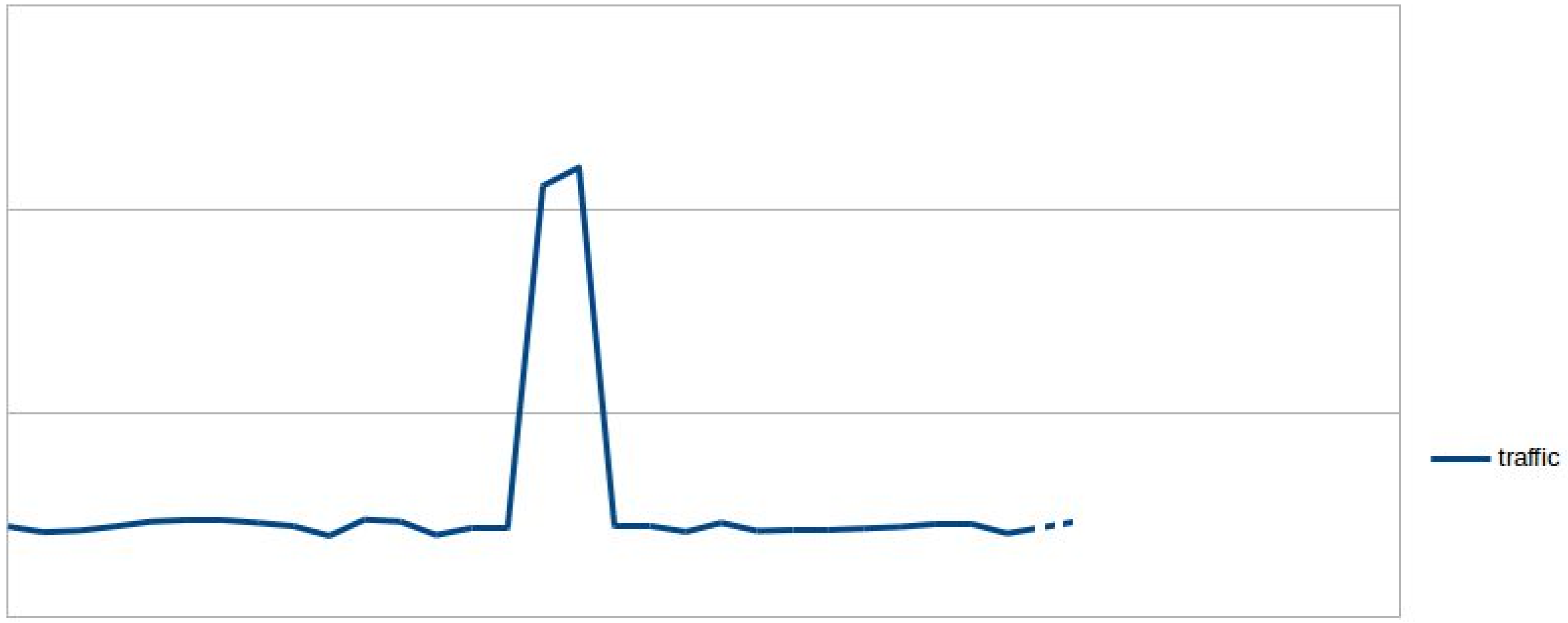
# Server instance



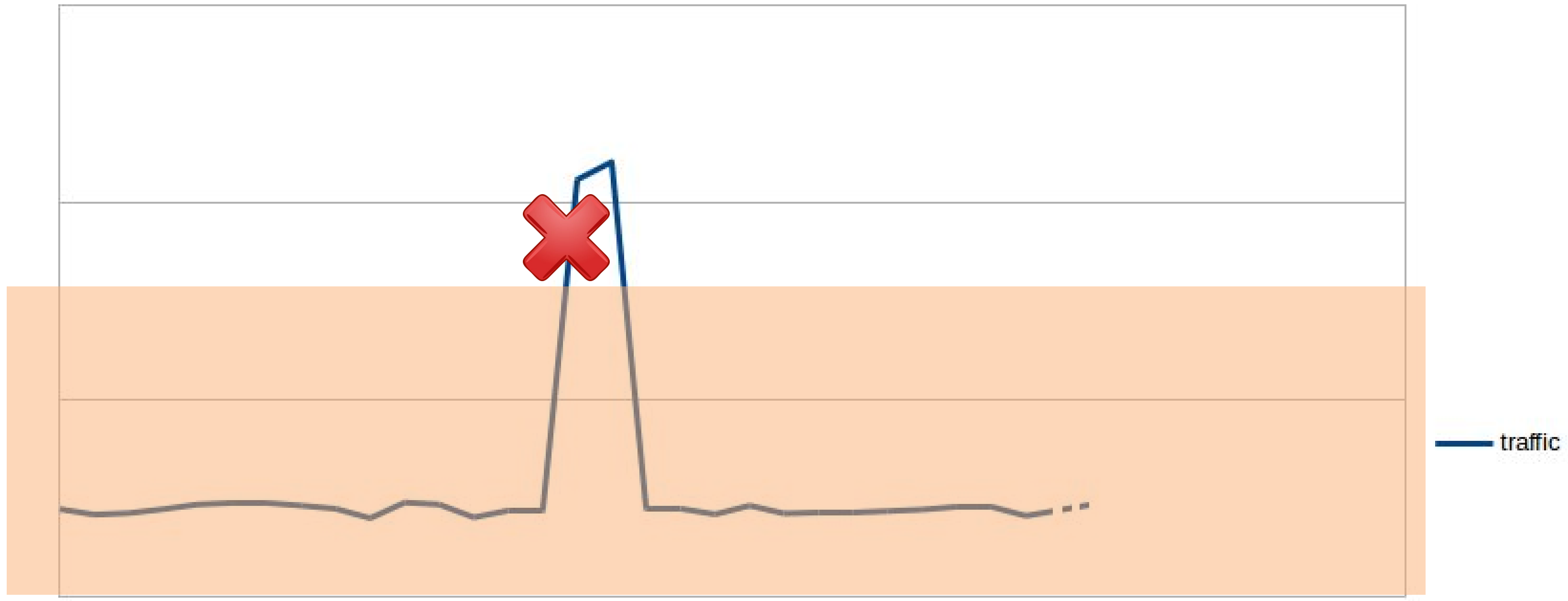
*...written in...*

| | |
|---|---|
| **Go** | **Java** |
| **Php** | **Python** |

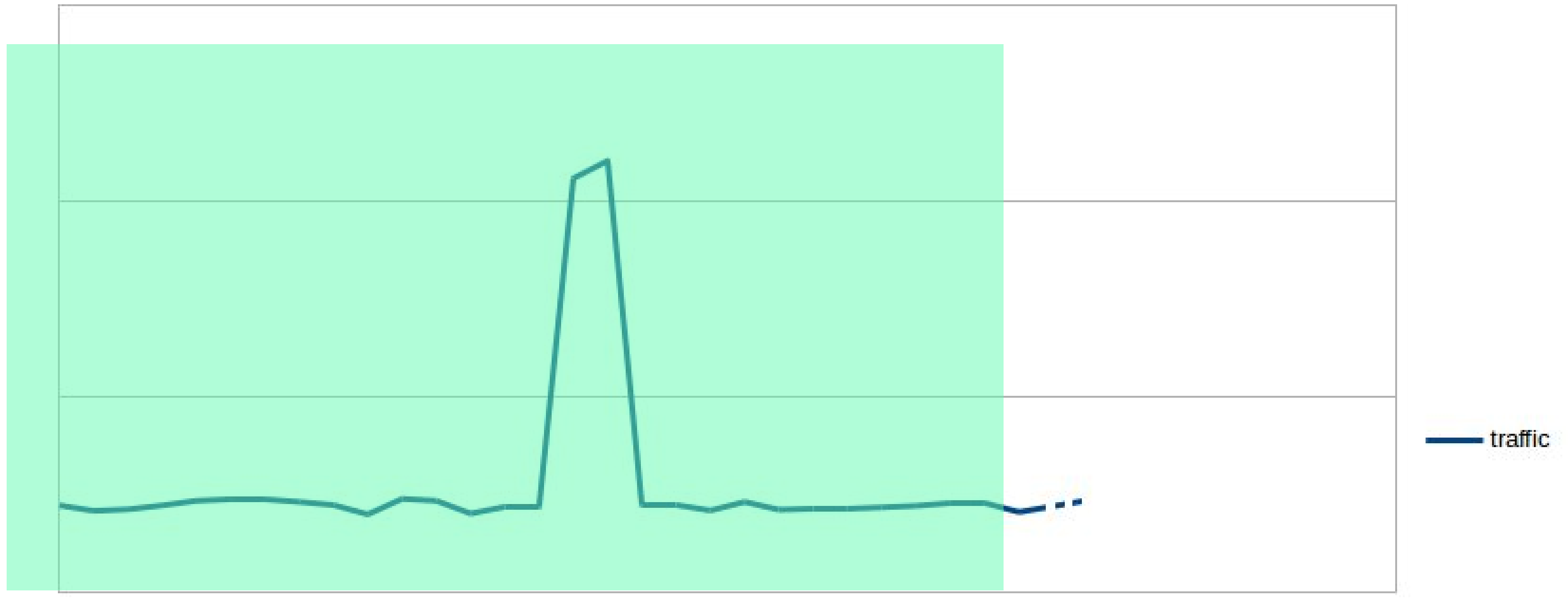
## Provisioning instances



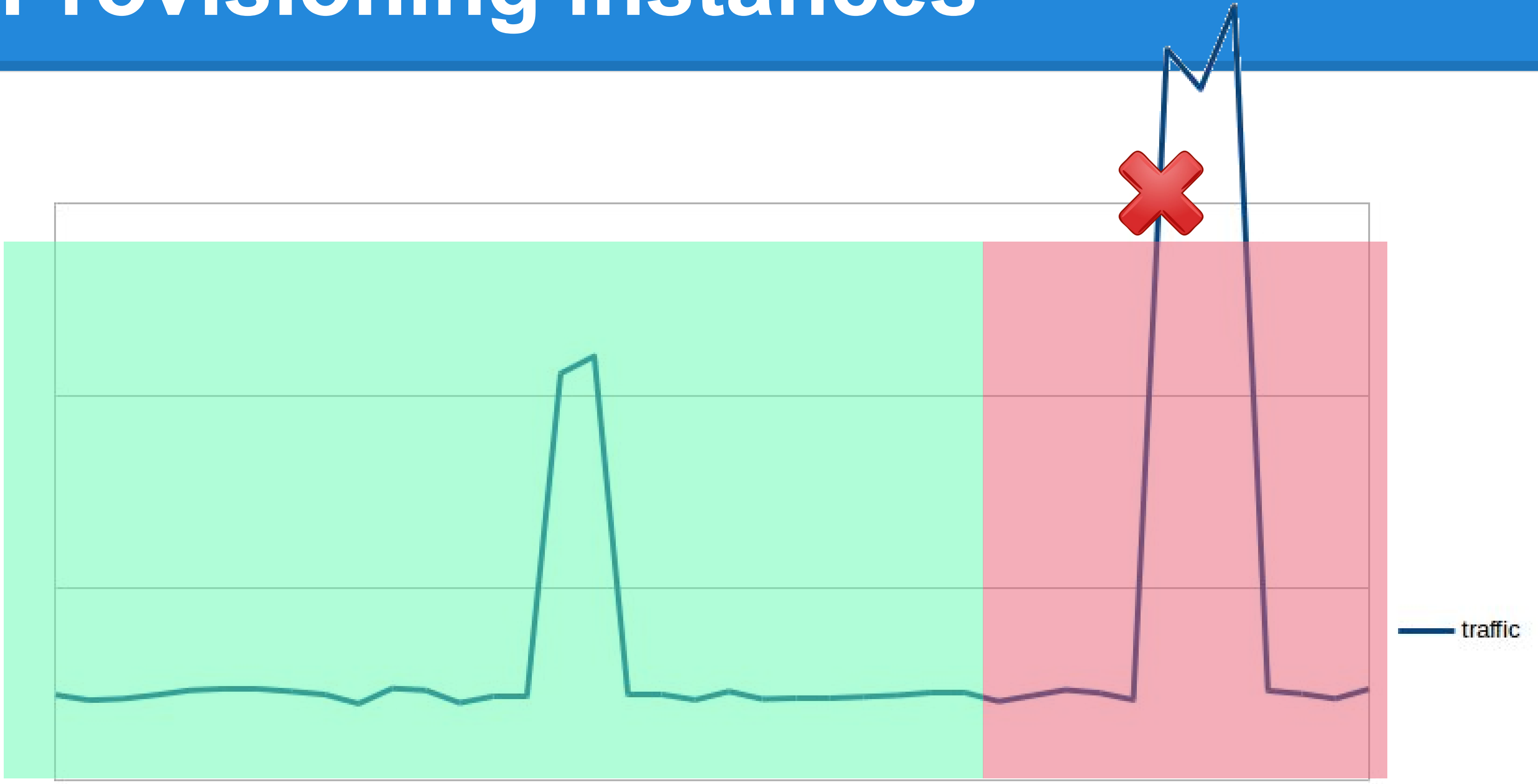
# Provisioning instances



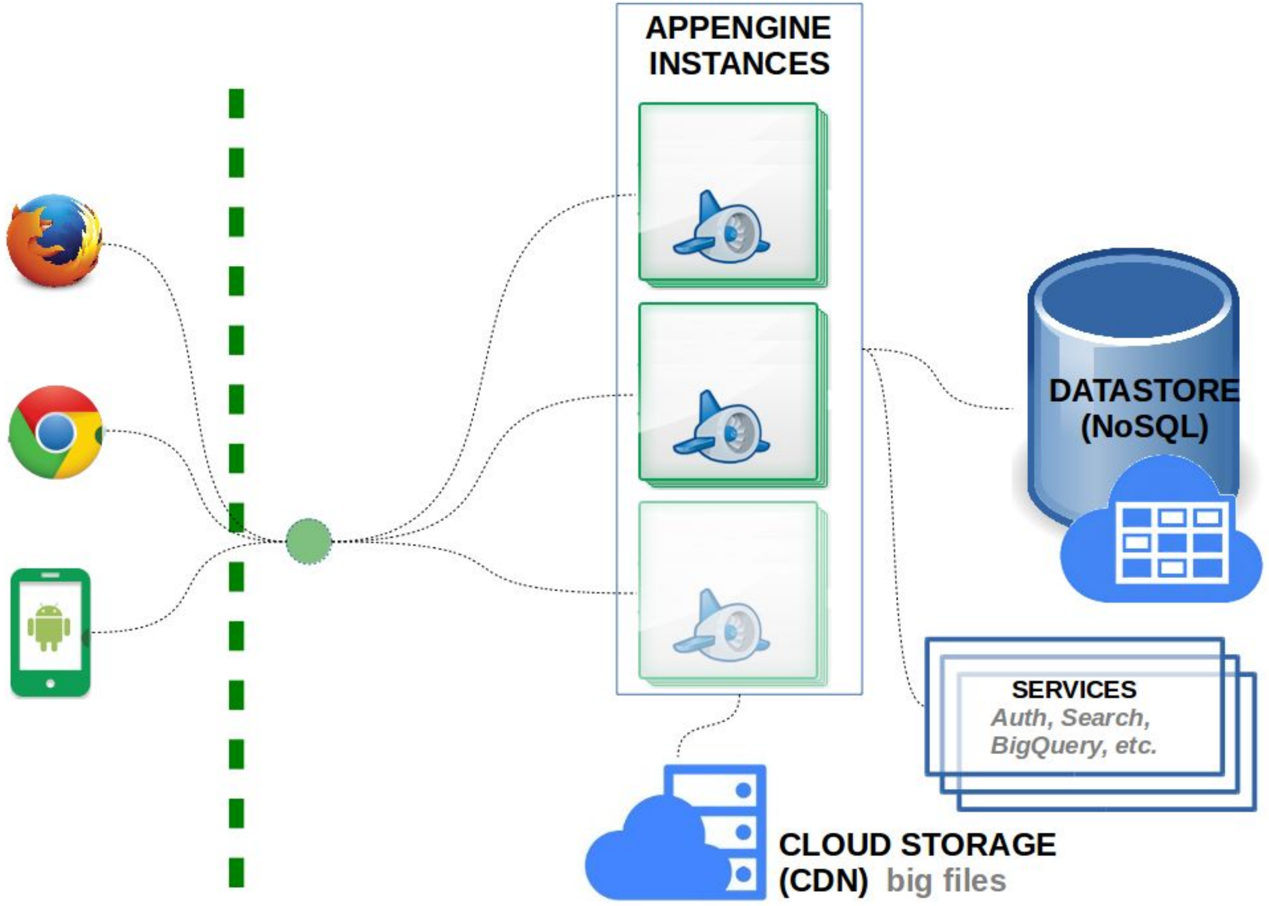
## Provisioning instances



## Provisioning instances

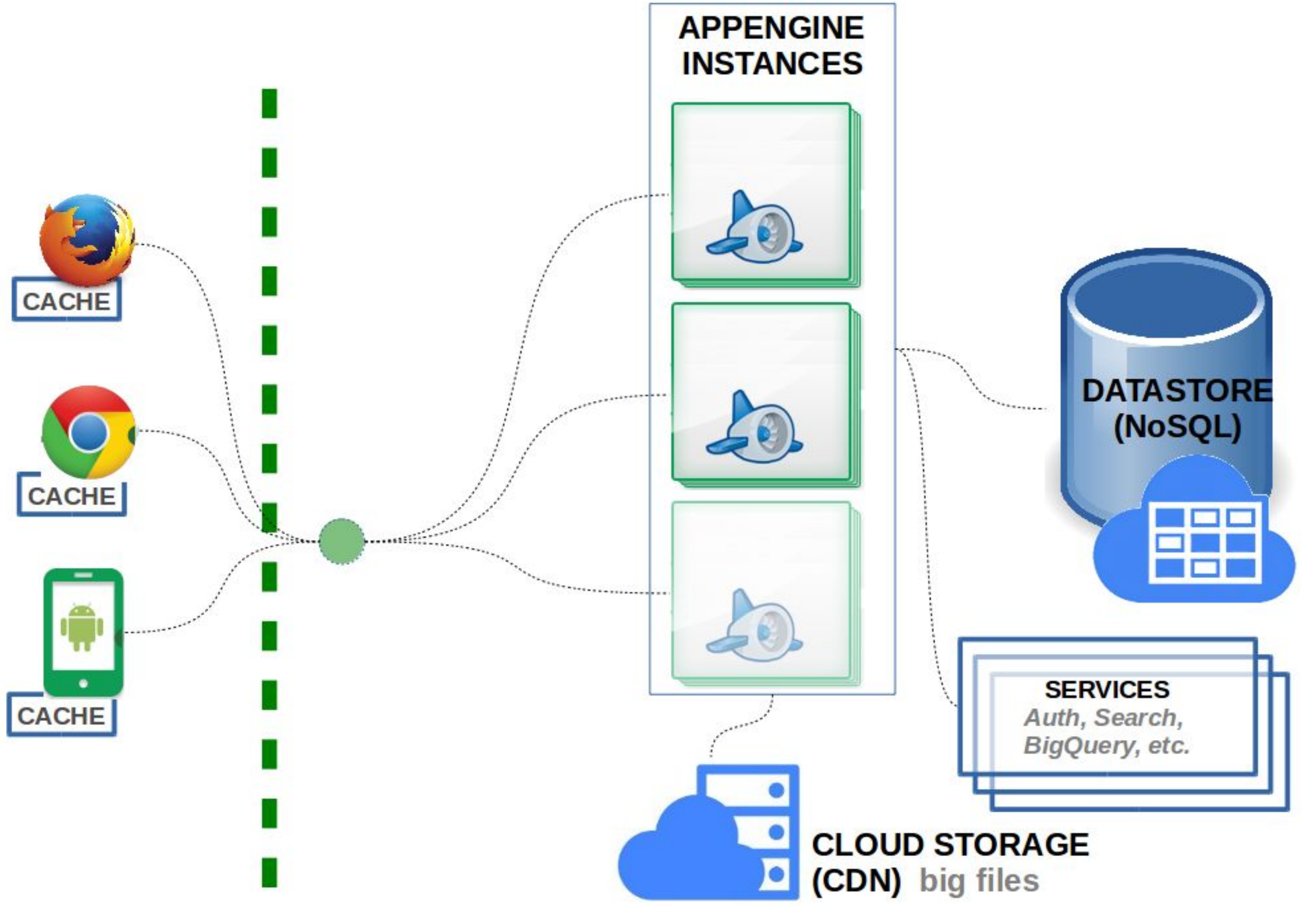


## GAE



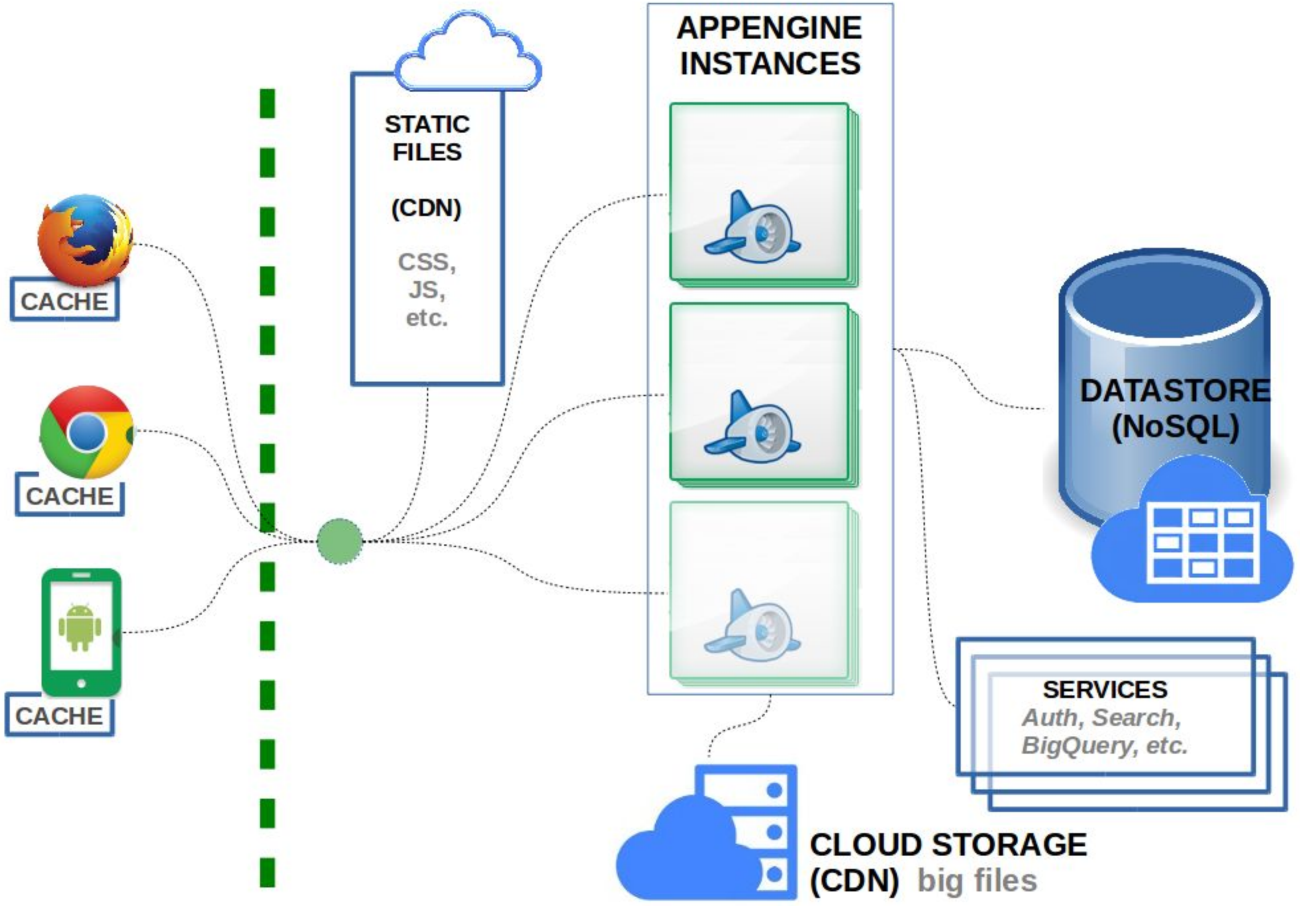
*Google infrastructure*

## GAE



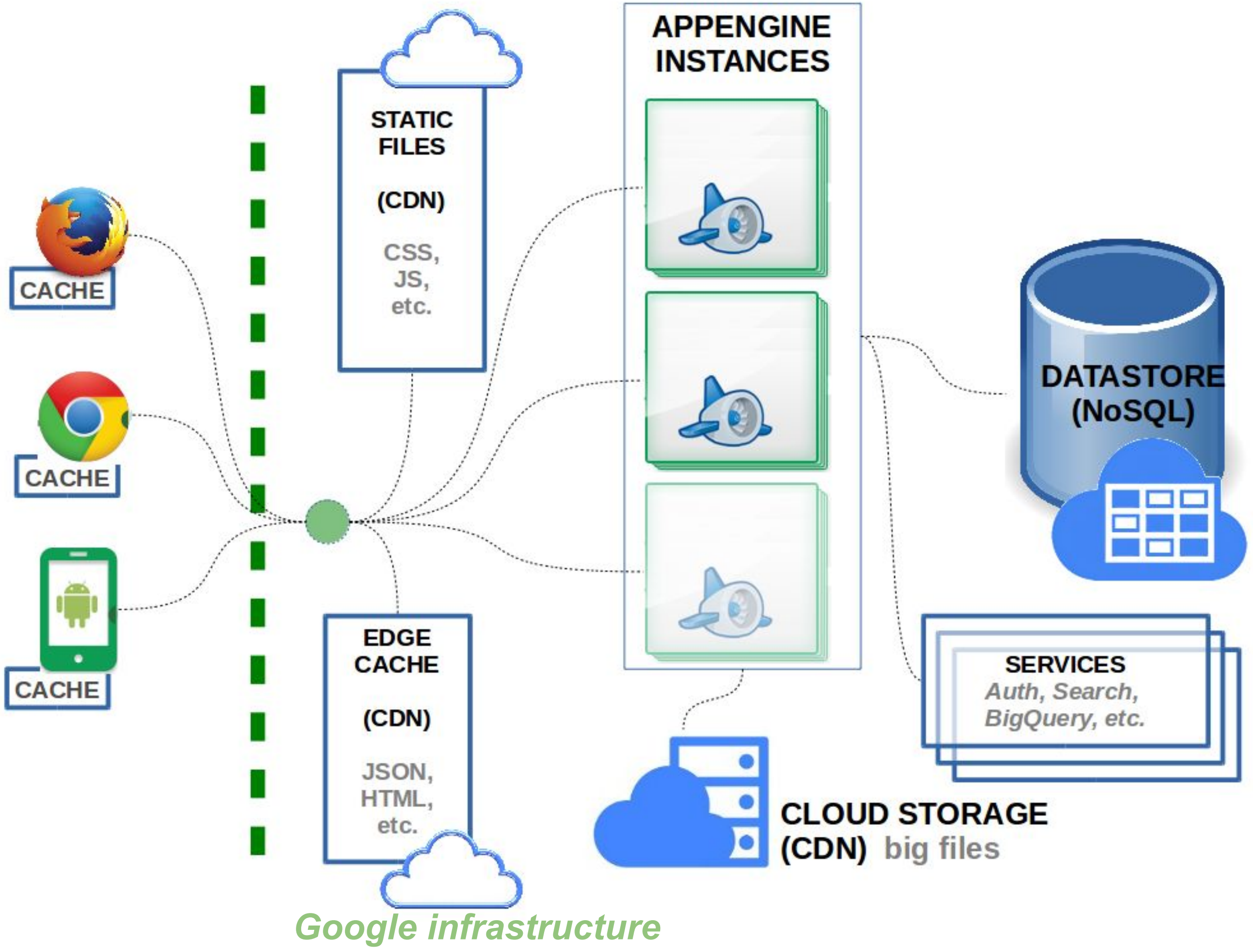
*Google infrastructure*

# GAE

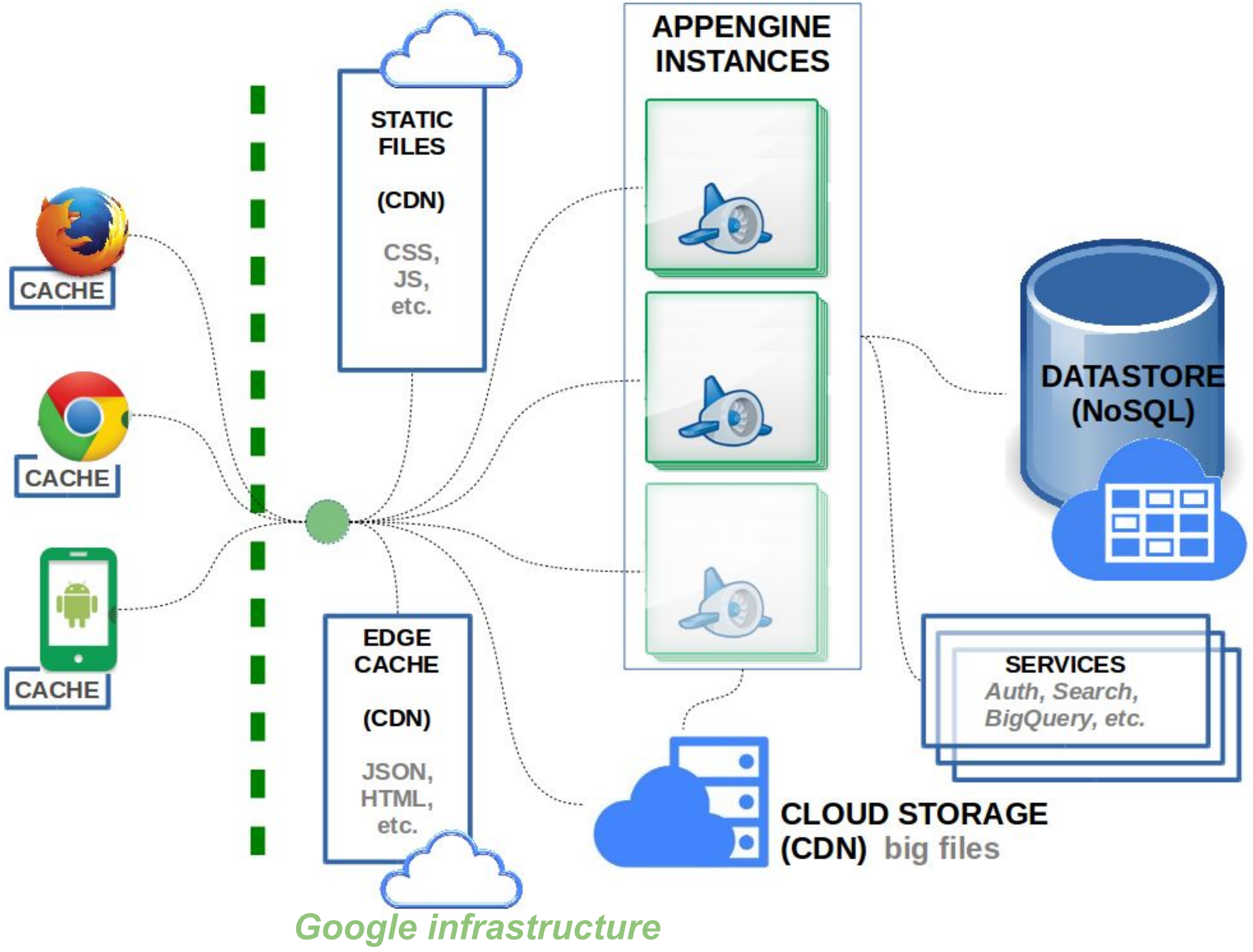


*Google infrastructure*

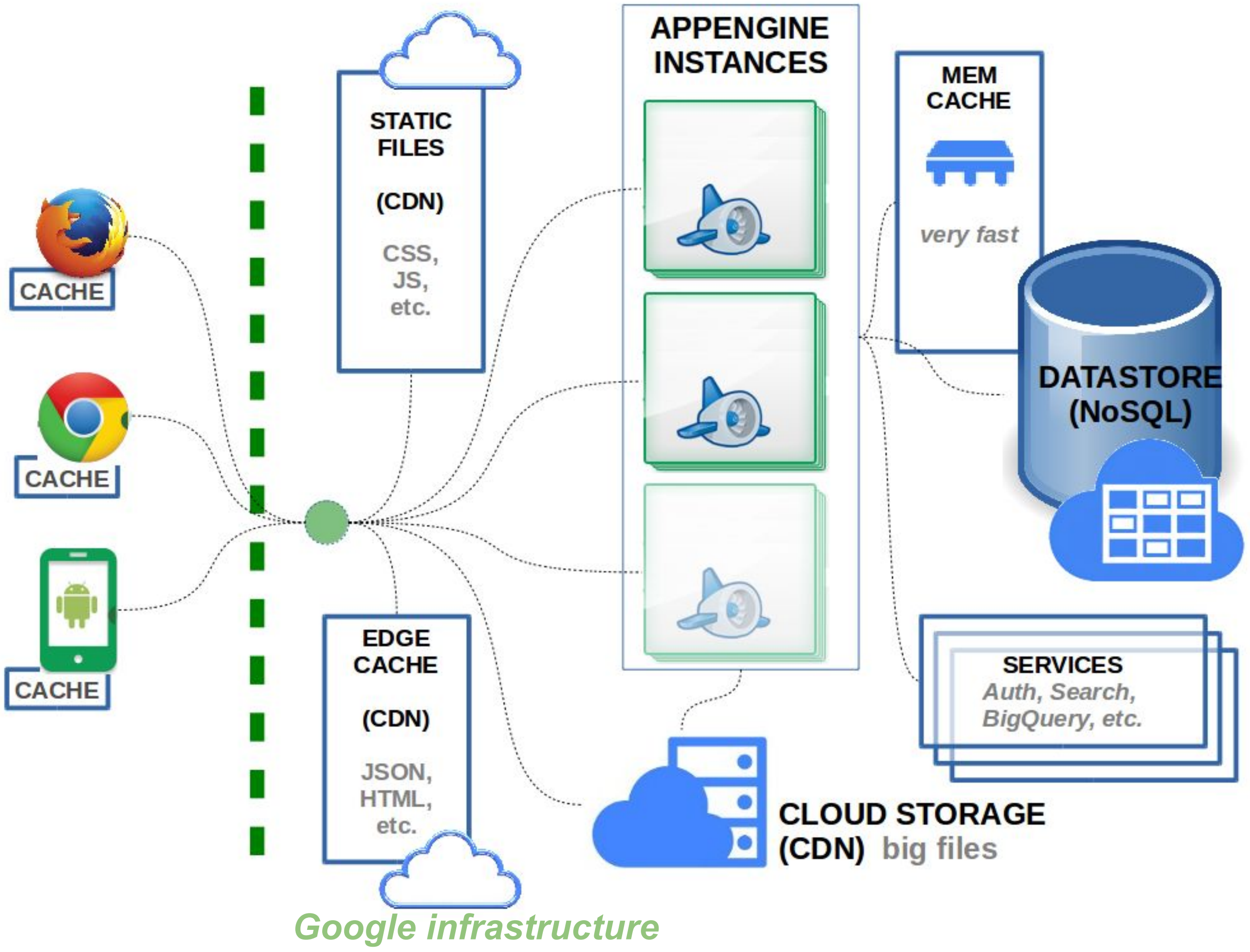
# GAE


CACHE
CACHE
CACHE
STATIC FILES
(CDN)
CSS, JS, etc.
EDGE CACHE
(CDN)
JSON, HTML, etc.
APPENGINE INSTANCES
DATASTORE (NoSQL)
SERVICES
Auth, Search, BigQuery, etc.
CLOUD STORAGE (CDN) big files
Google infrastructure

# GAE


CACHE
CACHE
CACHE
STATIC FILES
(CDN)
CSS, JS, etc.
EDGE CACHE
(CDN)
JSON, HTML, etc.
APPENGINE INSTANCES
DATASTORE (NoSQL)
SERVICES
Auth, Search, BigQuery, etc.
CLOUD STORAGE
(CDN) big files
Google infrastructure

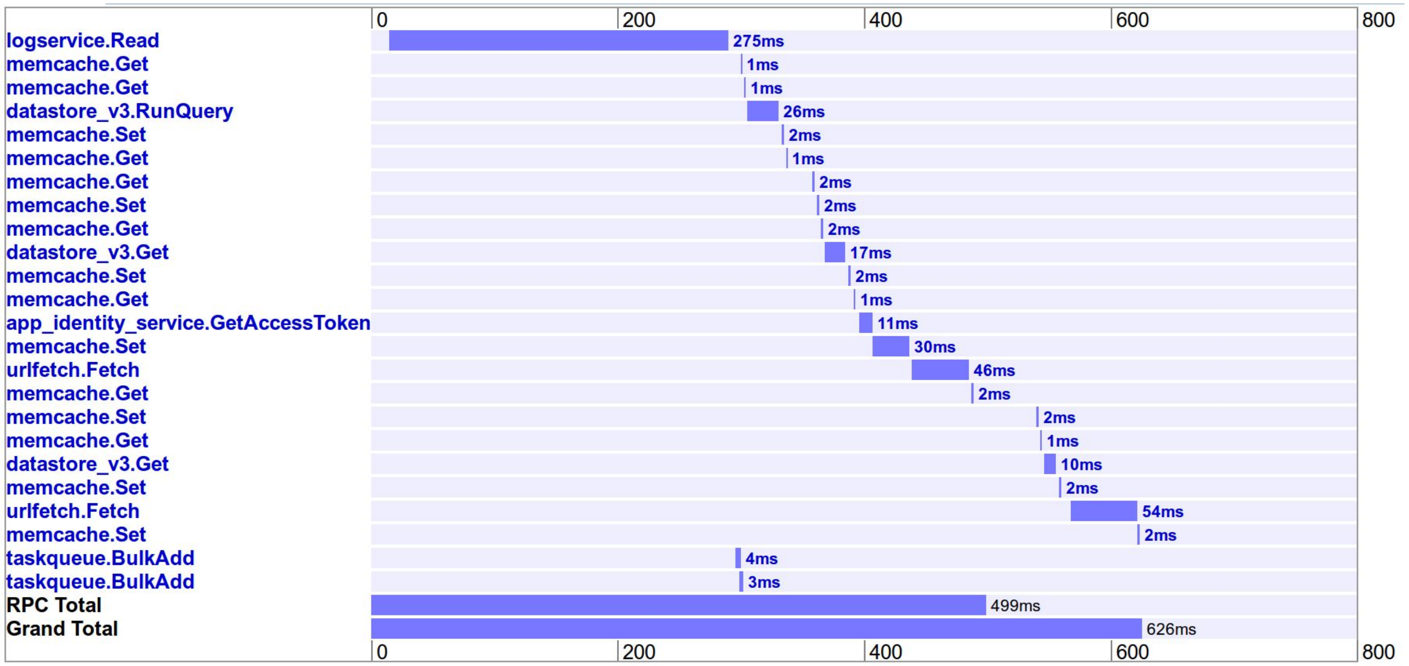
# GAE


CACHE
CACHE
CACHE
STATIC FILES
(CDN)
CSS, JS, etc.
EDGE CACHE
(CDN)
JSON, HTML, etc.
APPENGINE INSTANCES
MEM CACHE
very fast
DATASTORE (NoSQL)
SERVICES
Auth, Search, BigQuery, etc.
CLOUD STORAGE (CDN) big files
Google infrastructure

# RPC calls waterfall

## Appstats

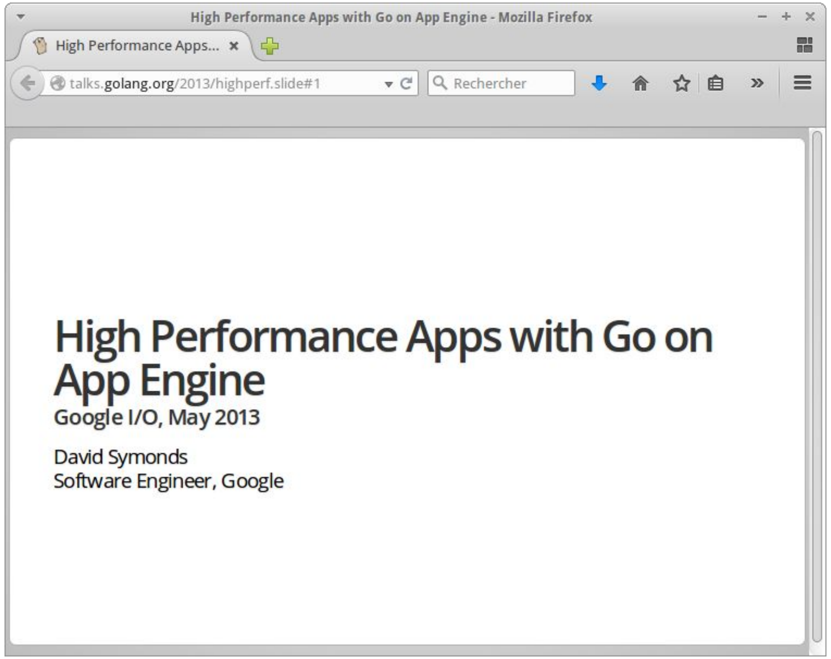
- ✓ go
- ✓ java
- ✓ py



| | |
|---|---|
| logservice.Read | 275ms |
| memcache.Get | 1ms |
| memcache.Get | 1ms |
| datastore_v3.RunQuery | 26ms |
| memcache.Set | 2ms |
| memcache.Get | 1ms |
| memcache.Get | 2ms |
| memcache.Set | 2ms |
| memcache.Get | 2ms |
| datastore_v3.Get | 17ms |
| memcache.Set | 2ms |
| memcache.Get | 1ms |
| app_identity_service.GetAccessToken | 11ms |
| memcache.Set | 30ms |
| urlfetch.Fetch | 46ms |
| memcache.Get | 2ms |
| memcache.Set | 2ms |
| memcache.Get | 1ms |
| datastore_v3.Get | 10ms |
| memcache.Set | 2ms |
| urlfetch.Fetch | 54ms |
| memcache.Set | 2ms |
| taskqueue.BulkAdd | 4ms |
| taskqueue.BulkAdd | 3ms |
| **RPC Total** | 499ms |
| **Grand Total** | 626ms |

## RPC calls waterfall

http://talks.golang.org/2013/highperf.slide


High Performance Apps with Go on App Engine - Mozilla Firefox
High Performance Apps...
talks.golang.org/2013/highperf.slide#1
Rechercher

High Performance Apps with Go on App Engine

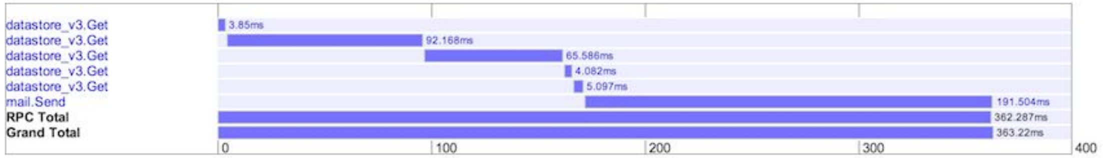
Google I/O, May 2013

David Symonds

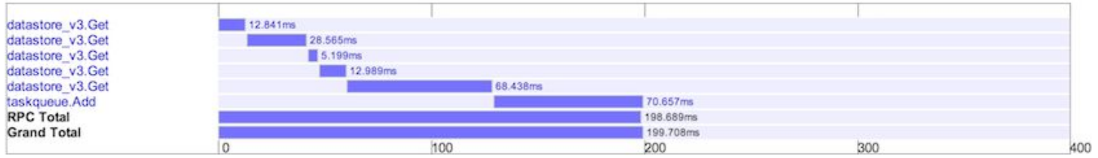
Software Engineer, Google

# RPC calls waterfall

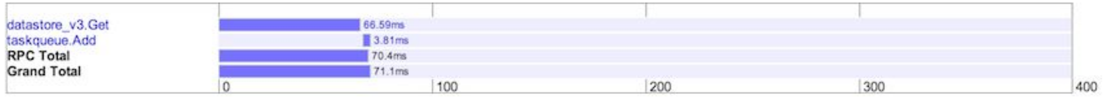
Baseline:



Defer work:



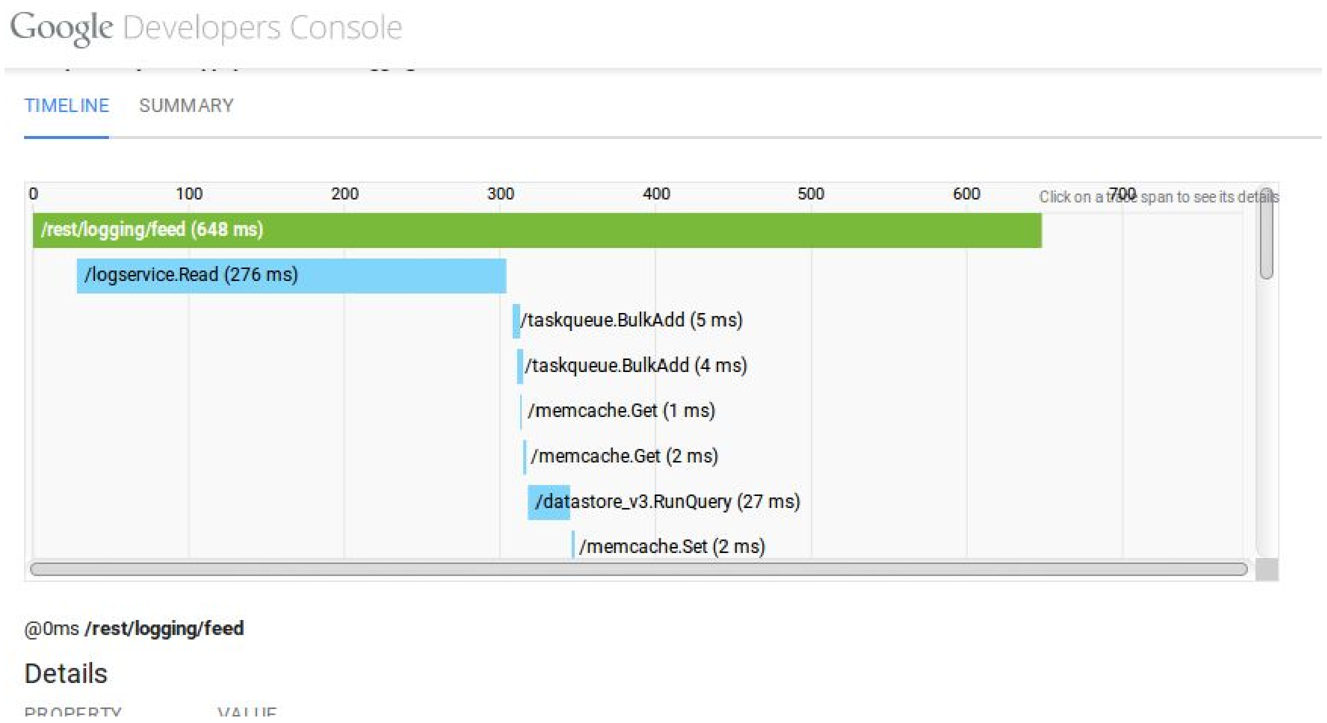
Batching:



## RPC calls waterfall

### Traces

- ✓ go
- ✓ java
- ✓ php
- ✓ py



Google Developers Console

TIMELINE SUMMARY

/rest/logging/feed (648 ms)
/logservice.Read (276 ms)
/taskqueue.BulkAdd (5 ms)
/taskqueue.BulkAdd (4 ms)
/memcache.Get (1 ms)
/memcache.Get (2 ms)
/datastore_v3.RunQuery (27 ms)
/memcache.Set (2 ms)

@0ms **/rest/logging/feed**

Details

PROPERTY VALUE

# Libraries *part I*

| | Go | Java | Php | Py |
|---|---|---|---|---|
| URL Fetch | ✔ | ✔ | ✔ | ✔ |
| Datastore | ✔ | ✔ | ✘ | ✔ |
| Memcache | ✔ | ✔ | ✔ | ✔ |
| Cloud Storage | ✔ | ✔ | ✔ | ✔ |
| Cloud SQL | ✔ | ✔ | ✔ | ✔ |
| BigQuery | ✘ | ✘ | ✘ | ✘ |

| | Go | Java | Php | Py |
|---|---|---|---|---|
| User auth | ✔ | ✔ | ✔ | ✔ |
| OAuth | ✔ | ✔ | ✘ | ✔ |
| Task Queues (Push) | ✔ | ✔ | ✔ | ✔ |
| Task Queues (Pull) | ✔ | ✔ | ✘ | ✔ |
| Task Queues (REST) | ✔ | ✔ | ✘ | ✘ |
| MapReduce | ✘ | ✔ | ✘ | ✔ |

# Libraries *part II*

| | Go | Java | Php | Py |
|---|---|---|---|---|
| Text Search | ✔ | ✔ | ✘ | ✔ |
| Channel * | ✔ | ✔ | ✘ | ✔ |
| Endpoints | ? | ✔ | ? | ✔ |
| Images | ✔ | ✔ | ✘ | ✔ |
| Logs | ✔ | ✔ | ✔ | ✔ |
| Mail | ✔ | ✔ | ✔ | ✔ |

| | Go | Java | Php | Py |
|---|---|---|---|---|
| Multitenancy / NS | ✔ | ✔ | ✘ | ✔ |
| SMS / Voice | ✘ | ✔ | ✔ | ✔ |
| Sockets | ✔ | ✔ | ✔ | ✔ |
| XMPP | ✔ | ✔ | ✘ | ✔ |

| | | | | |
|---|---|---|---|---|
| Total | 18 | 21 | 10 | 20 |

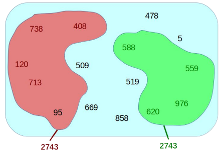
## Sample apps

App 1 Fortune teller

*(hello world)*

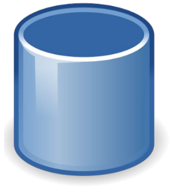
App 2 Equal Sums

*CPU and RAM intensive*



App 3 Business App

*Access database*



## Load test

10mn

18 000 requests

Increasing rate from 1 req/s up to 60 req/s

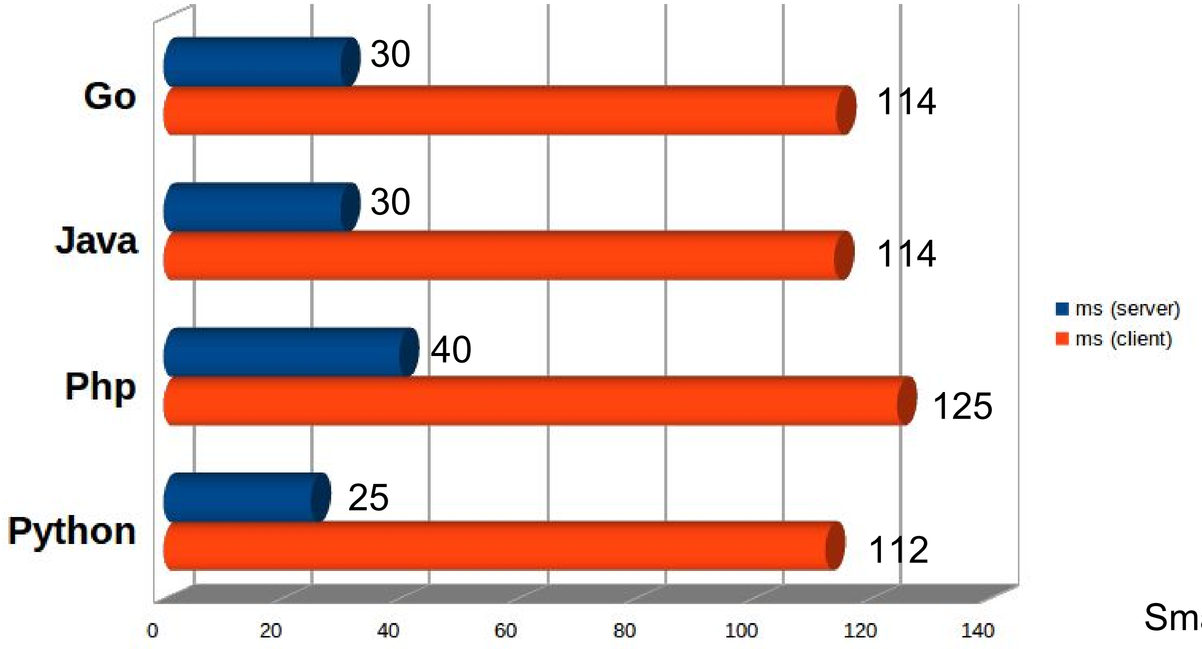
From 6 injectors in 3 continents

## App 1 : Fortune teller

Tonight you will get ***N*** beers.

# App 1 : Fortune teller

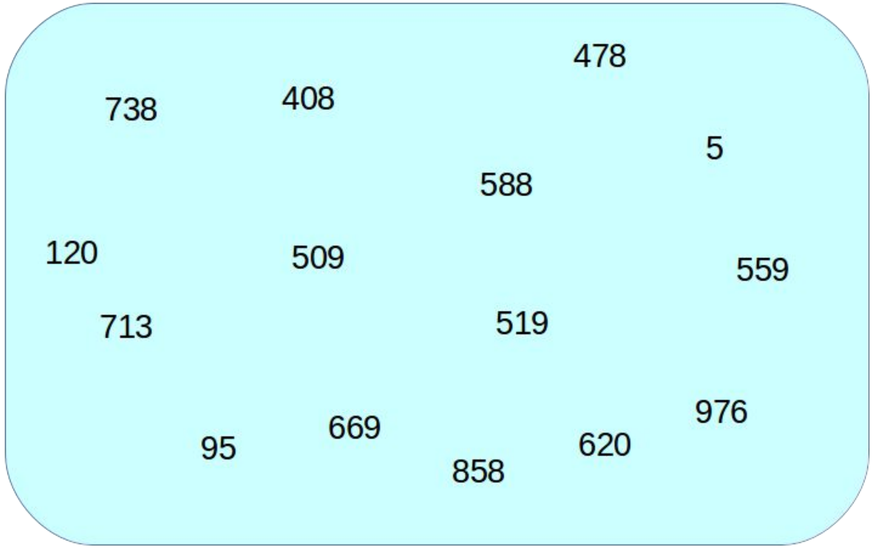
## Latency



Smaller is better

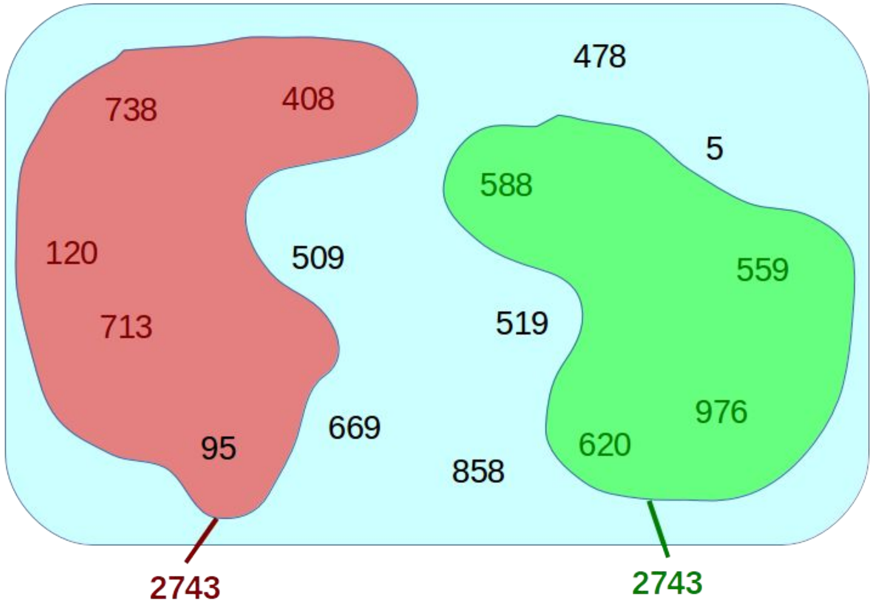
## App 2 : Equal Sums

Problem from *Google Code Jam* 2012



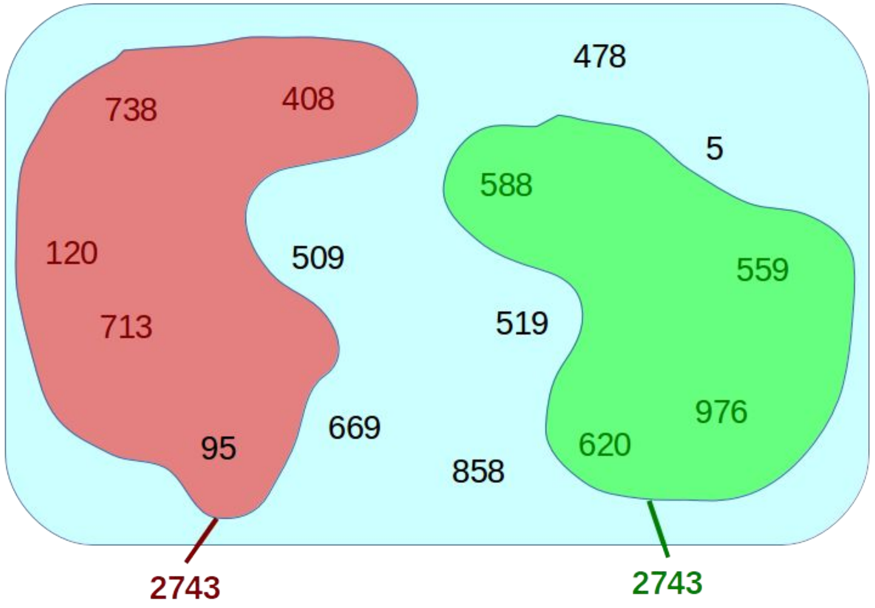
# App 2 : Equal Sums

Problem from *Google Code Jam* 2012



# App 2 : Equal Sums

Problem from *Google Code Jam* 2012



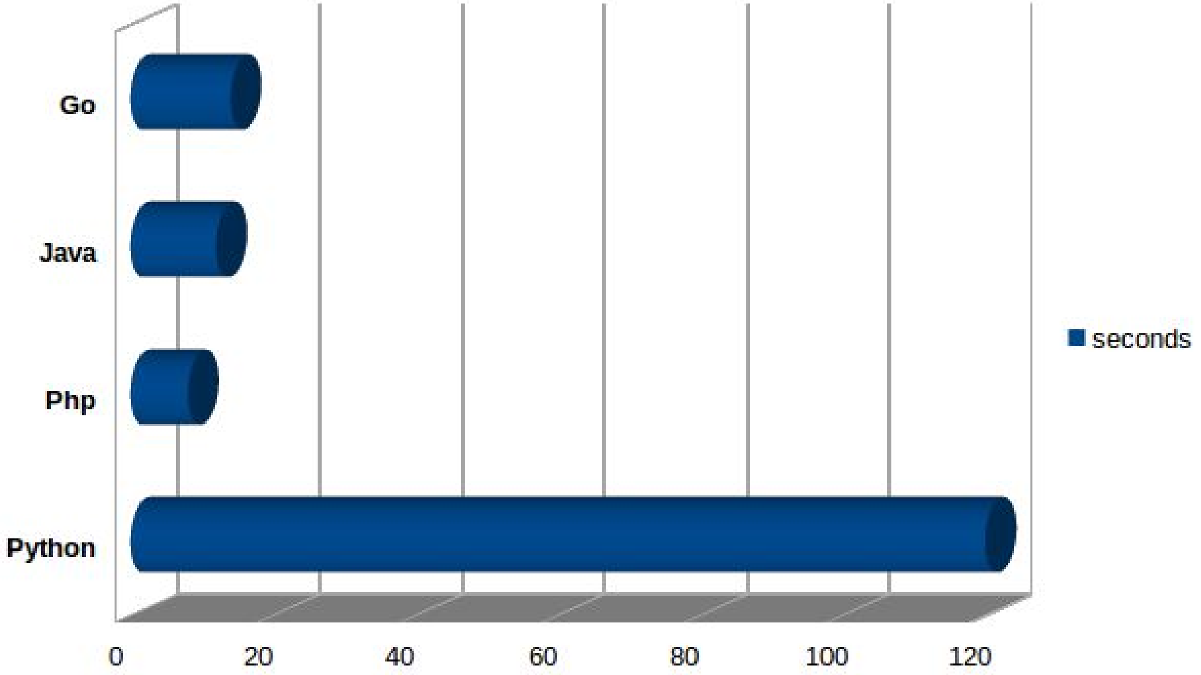
Nice algorithm uses :

- random
- hashmap

# App 2 : Equal Sums

## Offline

on my workstation

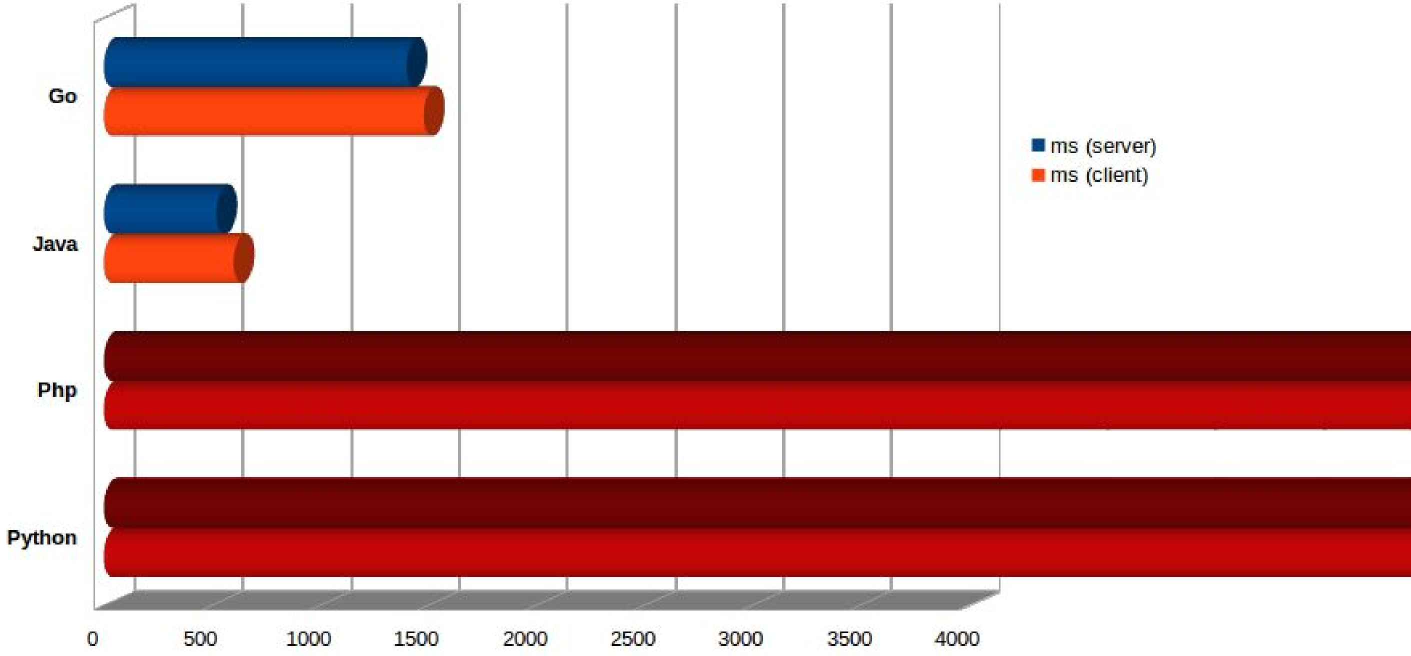


Smaller is better

# App 2 : Equal Sums

## Latency

on GAE

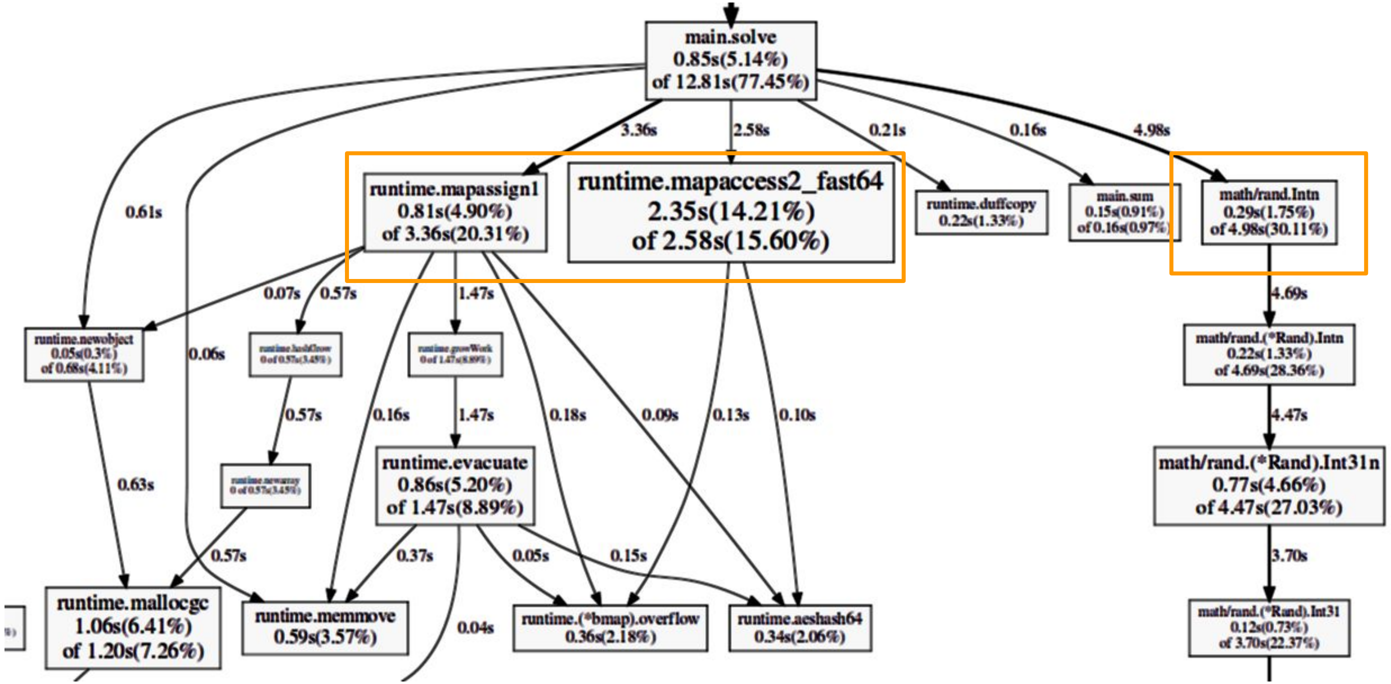


Smaller is better

# Advertisement

**pprof**

is your friend

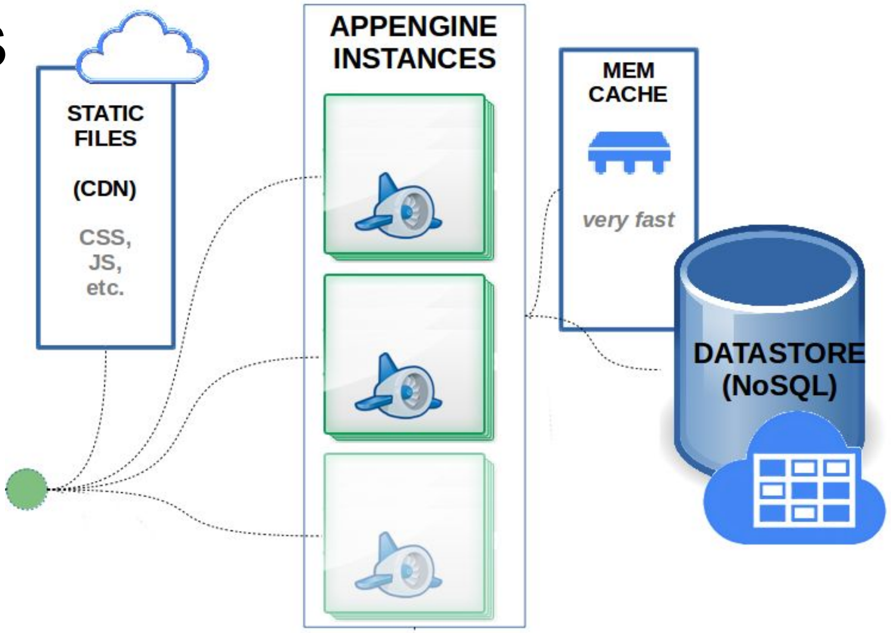


# App 3 : Business App

Lots of data reads

Lots of Memcache hits

Some data writes

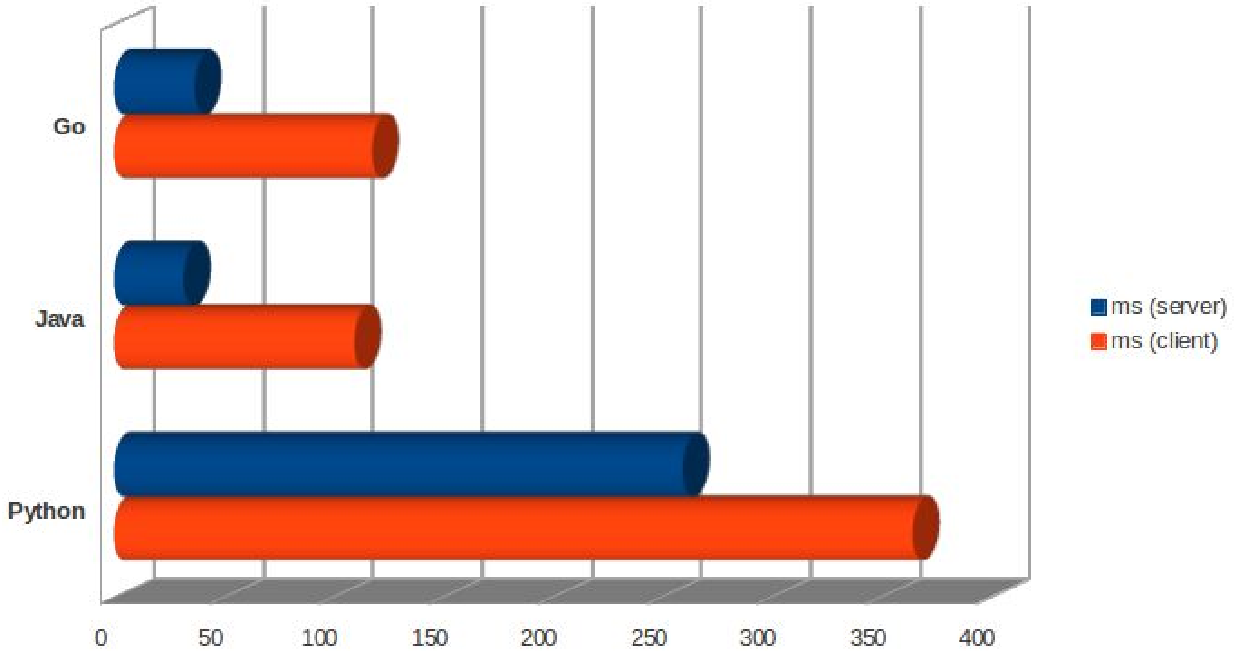


# App 3 : Business App

## Latency

on GAE

Smaller is better



# Conclusion

## Why choose Go on GAE ?

- Because you can
- It’s fun
- Fast cold start
- Most frugal
  - compiled
  - memory footprint
  - runtime overhead
- Native concurrency
- goapp

## Why **not** choose Go on GAE ?

- Your team doesn’t know Go well and is already efficient in Java || Php || Python
- You absolutely need a library which doesn’t exist in Go
  - But **modules** can help you
- You expect phenomenal speedup
  - Response times are dominated by many other factors than runtime language

## Thank you !

## Questions ?

Sample apps : bench-fortune.appspot.com

Sources : github > Ripounet > gae-language-bench

deleplace2015@gmail.com

## Batching

e.g.

```
datastore.Get(c, key, item)
        ->  datastore.GetMulti(c, keys, items)

datastore.Put(c, key, item)
        ->  datastore.PutMulti(c , keys, items)
```

## appengine/delay Appendix

1) *Register*

```
var historyDelayer = delay.Func("save-history-item",
    func(c appengine.Context, gopherKey *datastore.Key) error {
        var historyItem GopherHistory
        err := datastore.Get(c, gopherKey, &historyItem.Gopher)
        if err != nil {
                return err
        }
        _, err = datastore.Put(c, newHistoryKey(c), &historyItem)
        return err
    })
```

## appengine/delay Appendix

*2) Call async*

```
func saveGopher(c appengine.Context, gopher *Gopher) (*datastore.Key, error) {
    key, err := datastore.Put(c, datastore.NewIncompleteKey(c, "Idiom", nil), idiom)
    if err != nil {
        return key, err
    }

    // Save an history item : asynchronously
    historyDelayer.Call(c, key)

    return key, err
}
```

## Appendix

## Perf fine-tuning of Equal Sums

- custom poor man’s random
- custom hash-like array

## Load testing tool

gatling.io

+ Compute Engine VMs

Google Cloud Platform

## Powered by Go + GAE

secret.ly

goread.io

gorillatoolkit.org

programming-idioms.org