# SODIFRANCE/NETAPSYS VOUS SOUHAITE LA BIENVENUE AU MEETUP GO

**1300**
COLLABORATEURS

**16**
IMPLANTATIONS


Spécialiste de la Transformation des SI

**+150**
Projets réalisés

Précurseur des architectures micros services

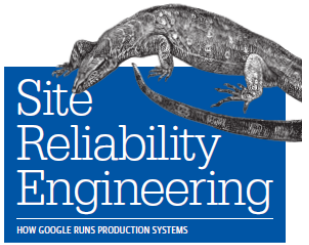
**+100**
Plusieurs milliards de lignes de code traitées

M€ d'investissements
**en R&D**
depuis 20 ans

Des solutions présentes dans
**14 pays**

Sodifrance
IT transformation to digital

# MOST METRICS ARE BETTER THOUGHT OF AS *DISTRIBUTIONS* RATHER THAN AVERAGES




*" Monitoring and alerting based only on the average latency would show no change in behavior "*

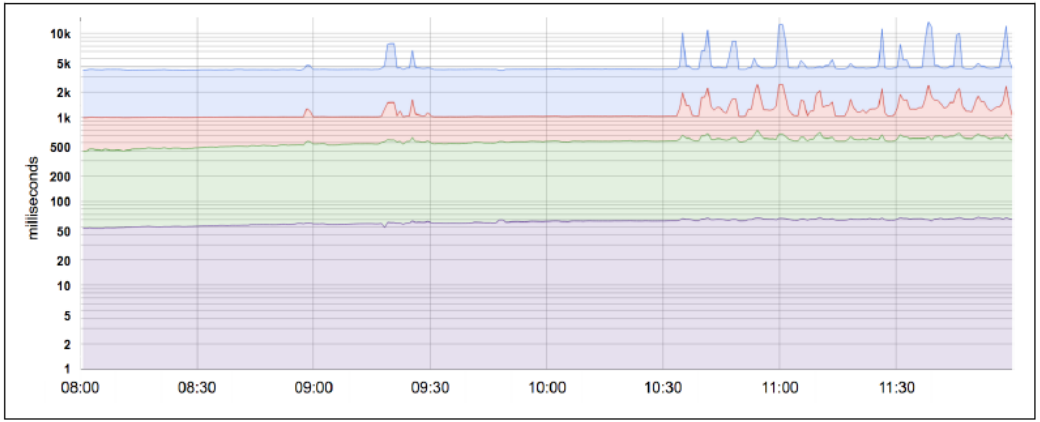



*Figure 4-1. 50th, 85th, 95th, and 99th percentile latencies for a system. Note that the Y-axis has a logarithmic scale.*

*" Using percentiles for indicators allows you to consider the shape of the distribution and its differing attributes: a high-order percentile, such as the* ***99th or 99.9th, shows you a plausible worst-case value****, while using the 50th percentile (also known as the median) emphasizes the typical case****. The higher the variance in response times, the more the typical user experience is affected by long-tail behavior****, an effect exacerbated at high load by* ***queuing effects****. User studies have shown that people typically prefer a slightly slower system to one with high variance in response time, so some* ***SRE teams focus only on high percentile values, on the grounds that if the 99.9th percentile behavior is good, then the typical experience is certainly going to be.****"*

# ROB PIKE'S 5 RULES OF PROGRAMMING

- **Rule 1.** You can't tell where a program is going to spend its time. Bottlenecks occur in surprising places, so don't try to second guess and put in a speed hack until you've proven that's where the bottleneck is.

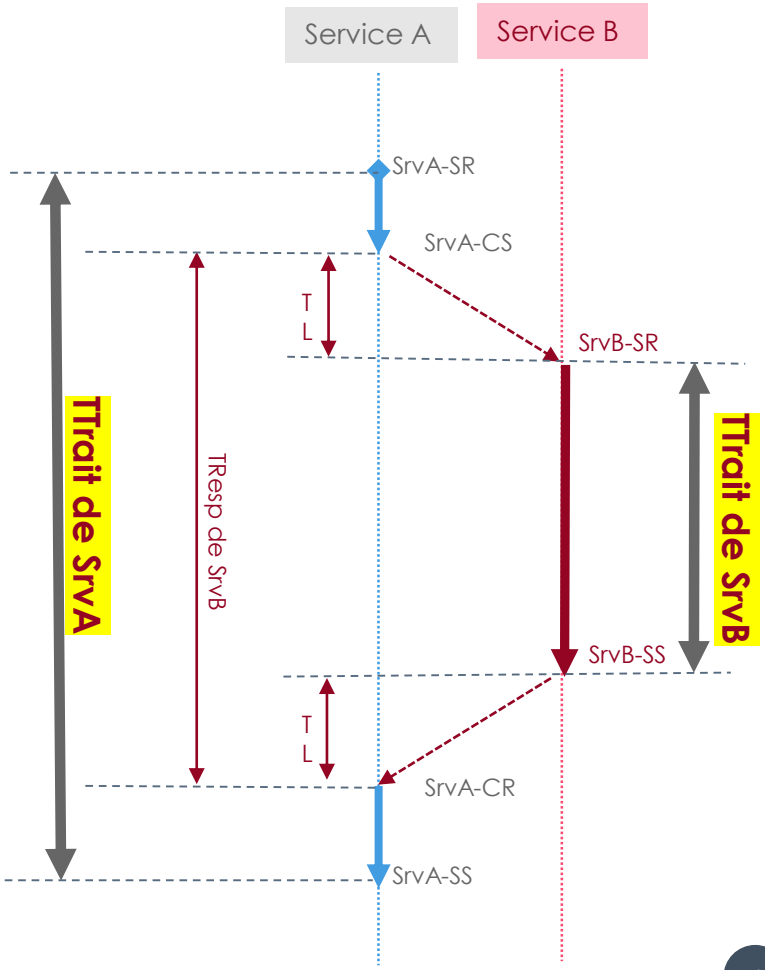
- **Rule 2.** Measure. Don't tune for speed until you've measured, and even then don't unless one part of the code overwhelms the rest.

- **Rule 3.** Fancy algorithms are slow when n is small, and n is usually small. Fancy algorithms have big constants. Until you know that n is frequently going to be big, don't get fancy. (Even if n does get big, use Rule 2 first.)
- **Rule 4.** Fancy algorithms are buggier than simple ones, and they're much harder to implement. Use simple algorithms as well as simple data structures.
- **Rule 5.** Data dominates. If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident. Data structures, not algorithms, are central to programming.

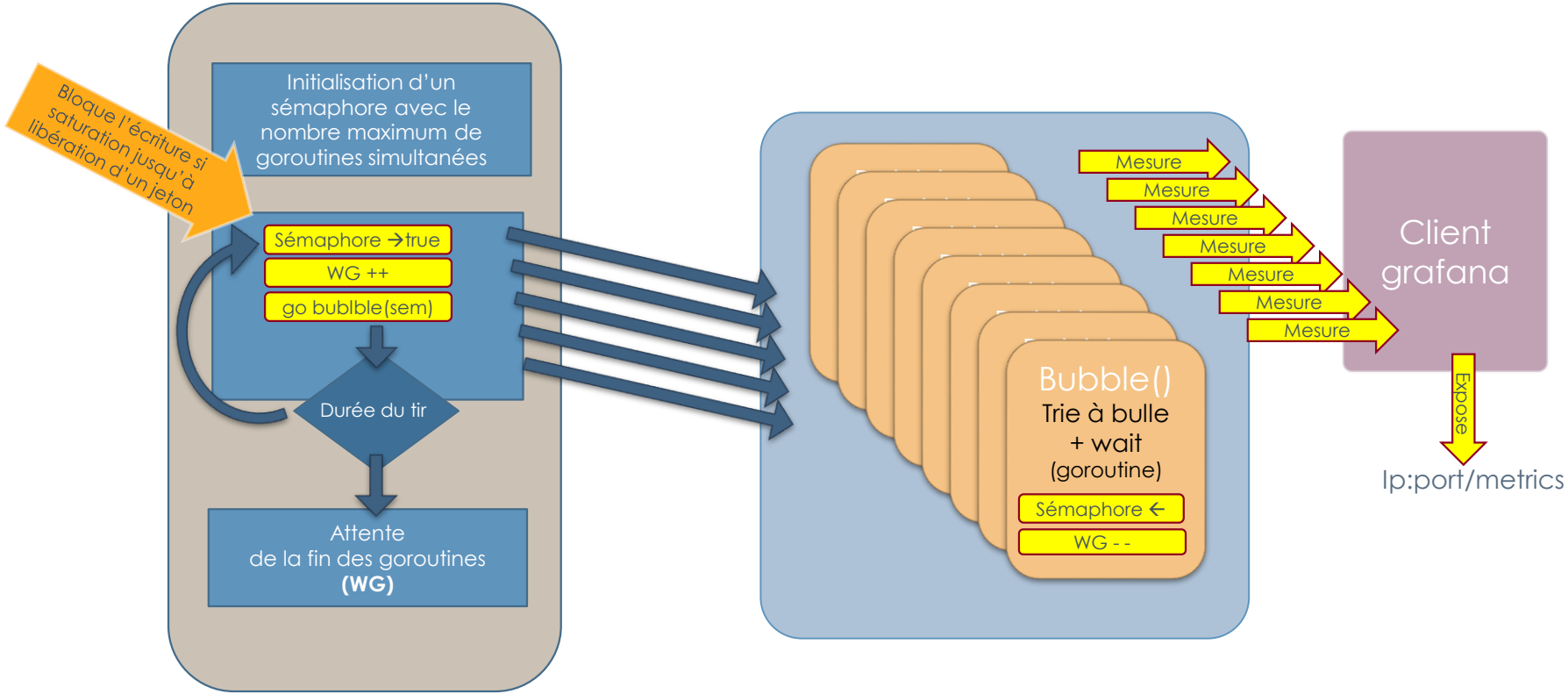
*Pike's rules 1 and 2 restate Tony Hoare's famous maxim "Premature optimization is the root of all evil." Ken Thompson rephrased Pike's rules 3 and 4 as "When in doubt, use brute force.". Rules 3 and 4 are instances of the design philosophy KISS. Rule 5 was previously stated by Fred Brooks in The Mythical Man-Month. Rule 5 is often shortened to "write stupid code that uses smart objects".*

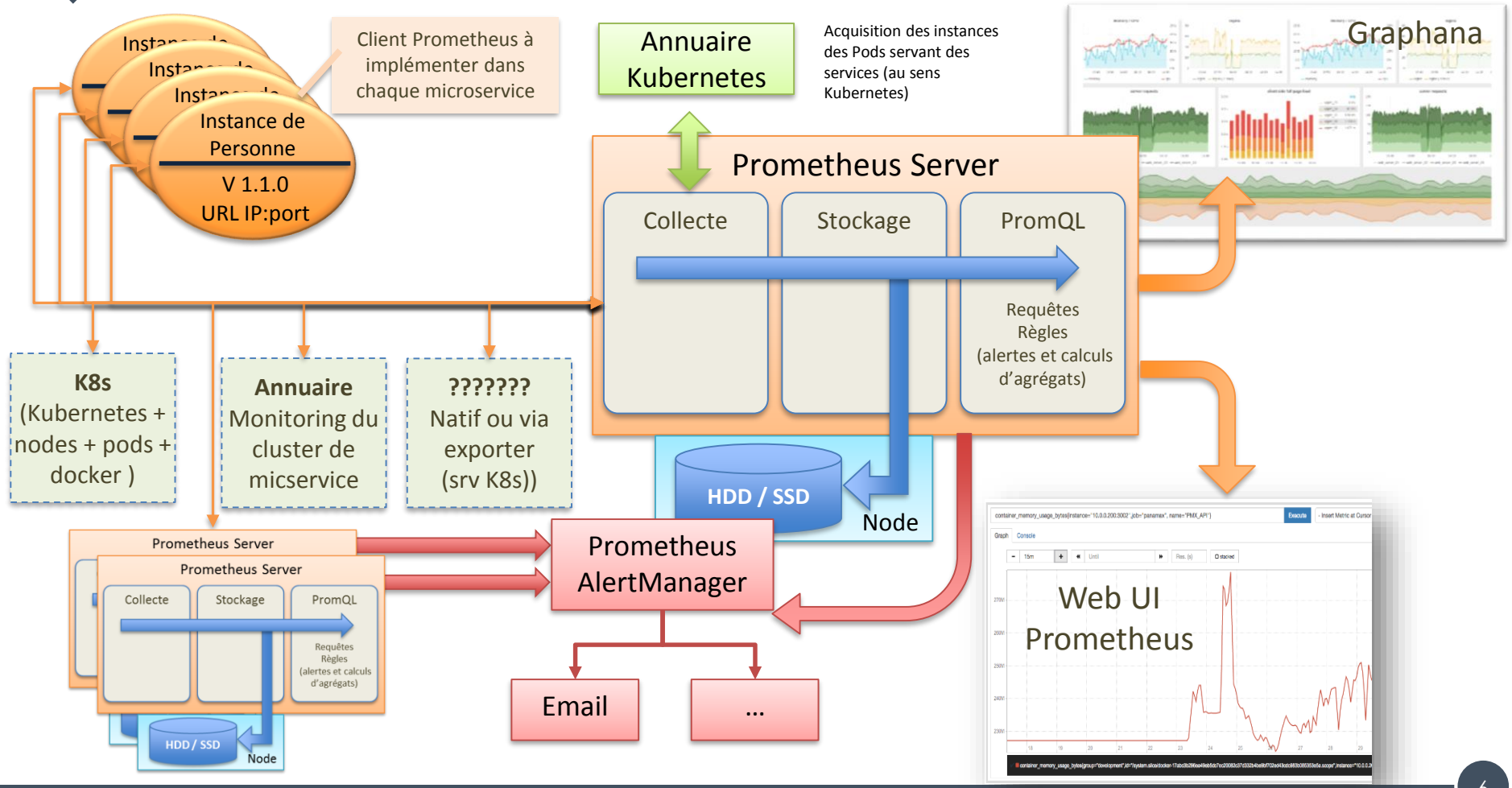
# CE QUE SIMULE BUBBLE

- Le temp de traitement serveur SR→SS
- Simule les CS→CR
- Isole strictement les problématiques de traitements pour évaluer:
  - La capacité du serveur a paralléliser l'exécution des requêtes.
  - Evaluer la capabilité du processus d'ordonnancement
- Ce qu'il ne fait pas encore …
  - D'appel RPC via les couches réseaux
  - En rest/json
  - Et Protobuf

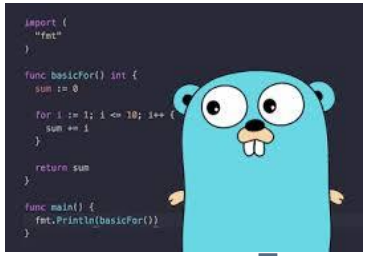


# SYNOPTIQUE DU PROGRAMME





# LE WORKFLOW DE LA DÉMO …



main.go

build/run/cp

docker

763MB
golang:latest

Bubble
(exe)

build/push

docker

Bubble
(Image)
5,47 MB
scatch

docker hub

docker

Pull

Prometheus

Grafana

Node-exporter

minikube

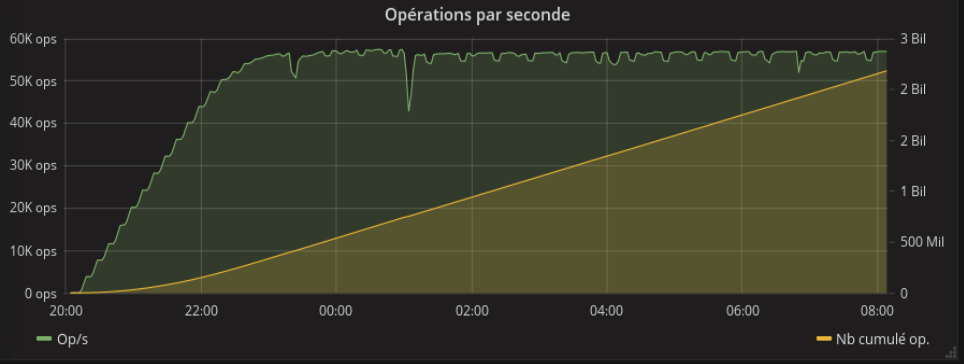
kubernetes

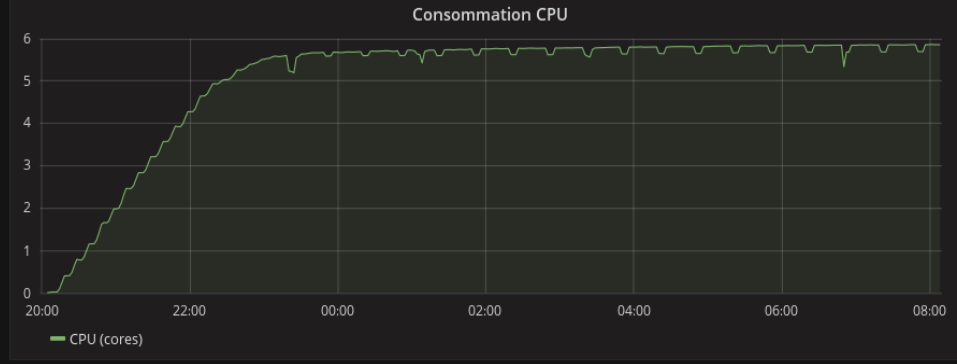
Core OS

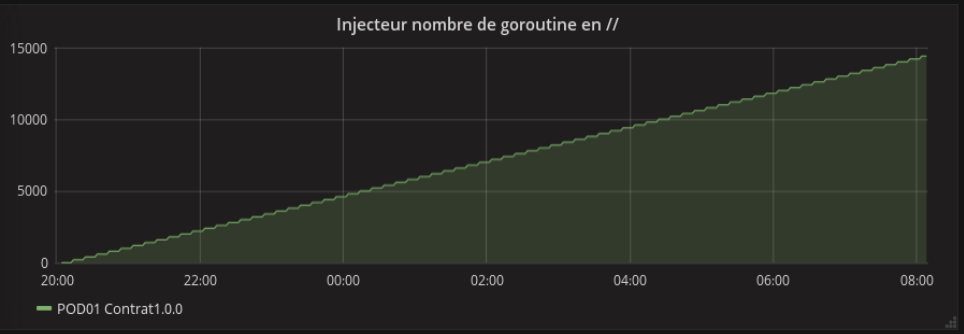
Kube state
metrcis

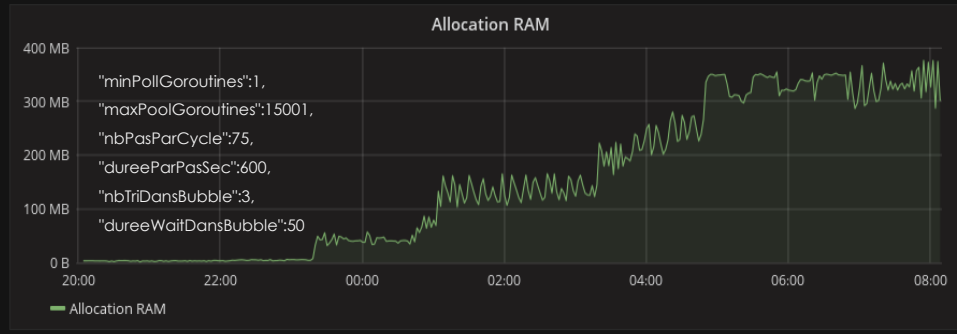
Core OS

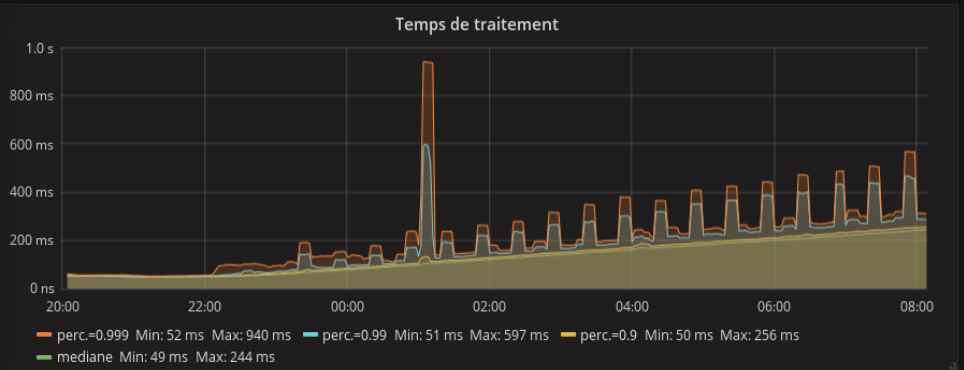
Prometheus
operator

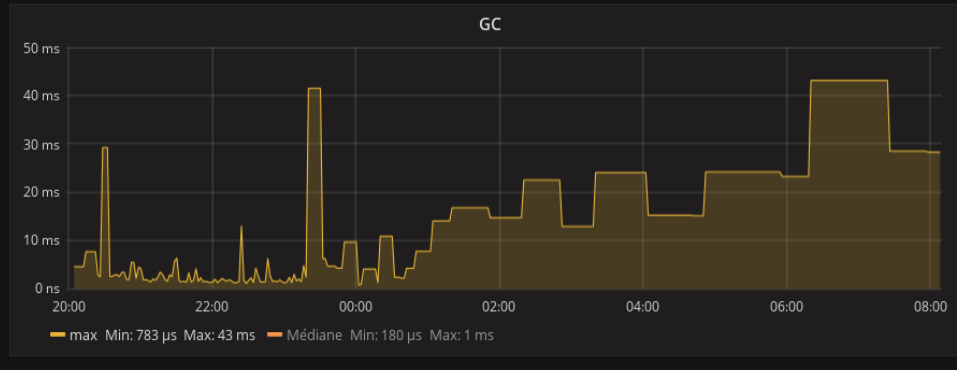

Opérations par seconde
60K ops
50K ops
40K ops
30K ops
20K ops
10K ops
0 ops
3 Bil
2 Bil
2 Bil
1 Bil
500 Mil
0
20:00
22:00
00:00
02:00
04:00
06:00
08:00
Op/s
Nb cumulé op.


Consommation CPU
6
5
4
3
2
1
0
20:00
22:00
00:00
02:00
04:00
06:00
08:00
CPU (cores)


Injecteur nombre de goroutine en //
15000
10000
5000
0
20:00
22:00
00:00
02:00
04:00
06:00
08:00
POD01 Contrat1.0.0


Allocation RAM
400 MB
300 MB
200 MB
100 MB
0 B
"minPollGoroutines":1,
"maxPoolGoroutines":15001,
"nbPasParCycle":75,
"dureeParPasSec":600,
"nbTriDansBubble":3,
"dureeWaitDansBubble":50
20:00
22:00
00:00
02:00
04:00
06:00
08:00
Allocation RAM


Temps de traitement
1.0 s
800 ms
600 ms
400 ms
200 ms
0 ns
20:00
22:00
00:00
02:00
04:00
06:00
08:00
perc.=0.999 Min: 52 ms Max: 940 ms
perc.=0.99 Min: 51 ms Max: 597 ms
perc.=0.9 Min: 50 ms Max: 256 ms
mediane Min: 49 ms Max: 244 ms


GC
50 ms
40 ms
30 ms
20 ms
10 ms
0 ns
20:00
22:00
00:00
02:00
04:00
06:00
08:00
max Min: 783 µs Max: 43 ms
Médiane Min: 180 µs Max: 1 ms

# QUELQUES LIENS …

- https://github.com/marcdivet/Bubble
- https://prometheus.io/docs/introduction/overview/
- https://grafana.com/
- https://coreos.com/blog/the-prometheus-operator.html
  - https://coreos.com/operators/prometheus/docs/latest/user-guides/getting-started.html
  - https://coreos.com/operators/prometheus/docs/latest/
- https://github.com/prometheus/node_exporter
- https://github.com/kubernetes/kube-state-metrics/
- https://kubernetes.io/docs/getting-started-guides/minikube/
- https://docs.docker.com/engine/installation/
- http://blog.cloud66.com/how-to-create-the-smallest-possible-docker-image-for-your-golang-application/
  - https://www.habitus.io

# C'EST FINI … MAINTENANT :

CONSEIL TECHNOLOGIQUE

TRANSFORMATION DIGITALE

DATA MANAGEMENT

MODERNISATION DES SI

SERVICE POUR LES APPLICATIONS

SERVICE D'INFRASTRUCTURES

Sodifrance

NETAPSYS

IT transformation to digital