Spring cloud et Zipkin à la rescousse du tracing distribué

Florian Garcia

Sommaire

1. Problématique
2. Le tracing distribué
3. Les outils
4. Démo
5. Conclusion

Problématique

Ca pourrait commencer comme ça ...

Temps de réponse du site !

Robert <bebert.cto@bigcompany.com> 7 févr. 2017 14:28

à : Florian GARCIA <fgarcia@ippon.fr>

Bonjour,
Nous constatons une forte dégradation des temps de réponse du site depuis hier soir.
De nombreux clients se sont plaints !!!
Merci de fixer ce problème ASAP

Robert le CTO

En vérifiant

On investigue !

Type d’application :

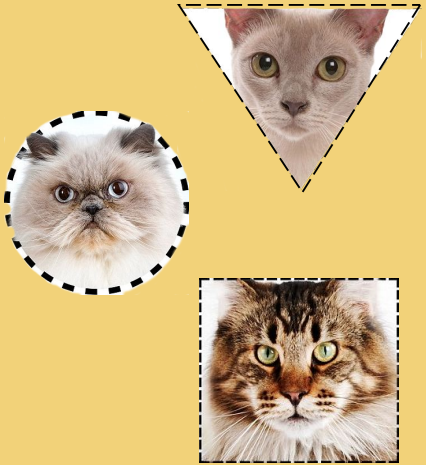
- Web
- Batch
- …

Types d’architecture :

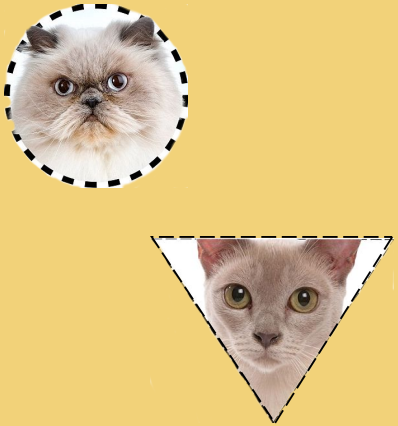
- Monolithe
- Micro-services
- Event driven
- …

µ-services ?

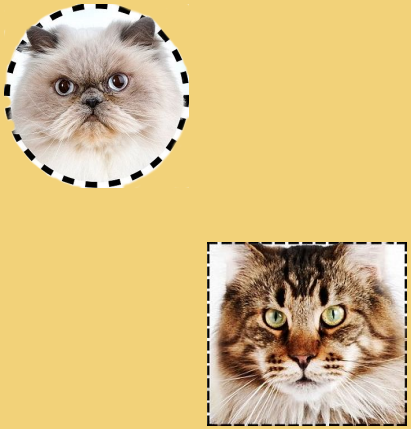
Machine 1



Machine 2

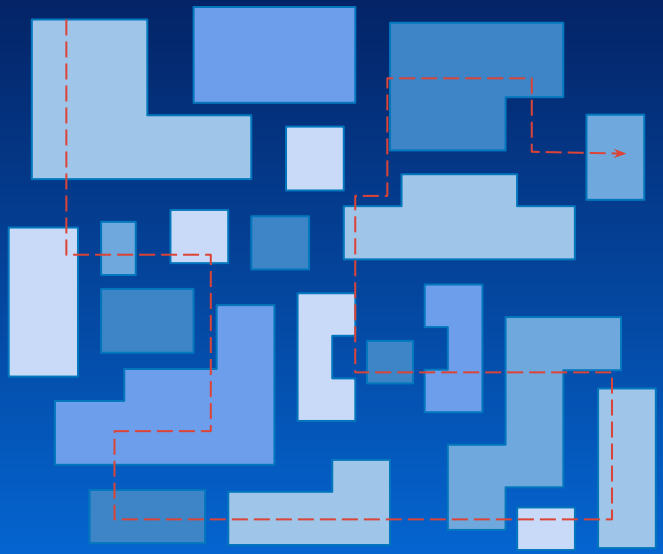


Machine 3



Du réseau ?

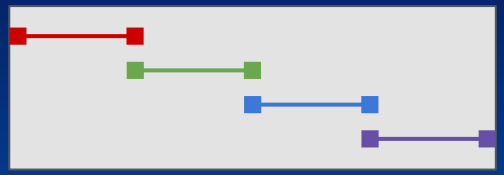
- De nombreux appels réseaux
  - Introduction de latence
  - Introduction de pannes
- Difficile à suivre et tracer



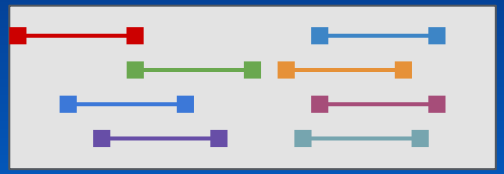
Source: *Ben Sigelman*

De la concurrence ?

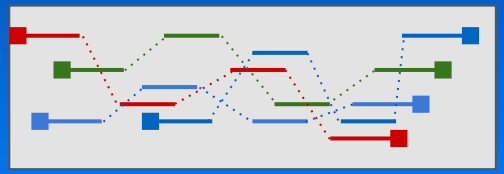
Simple



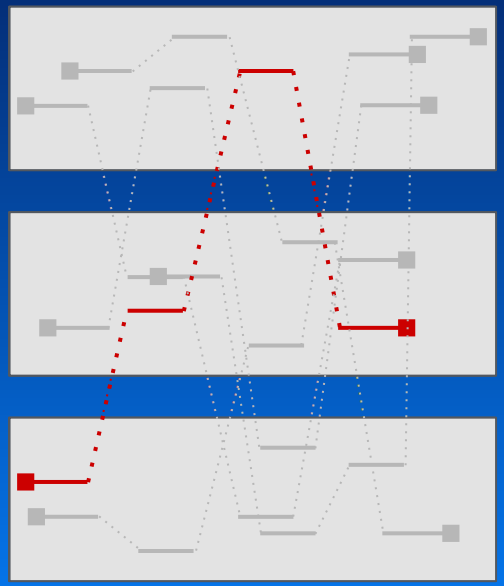
Basic Concurrency



Async Concurrency



Distributed Concurrency



Source : *Ben Sigelman*

Le tracing distribué

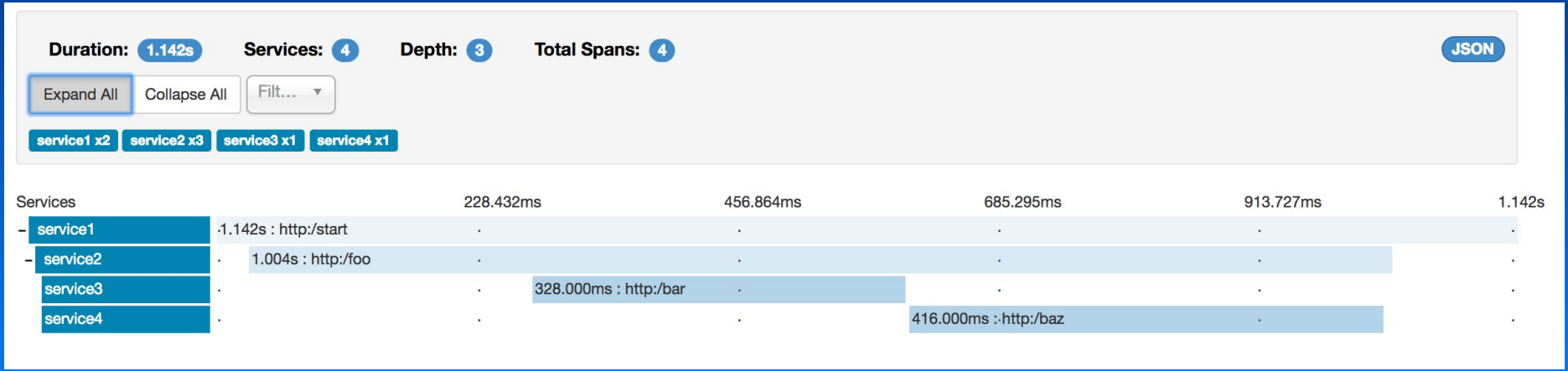
Historique

- Google utilise “Dapper” depuis 12 ans
- Le Dapper paper est en ligne depuis 6 ans
- Twitter Zipkin open source depuis 4 ans
- Création d’OpenTracing il y a 2 ans
- Novembre 2016 Opentracing rejoints la CNCF
- Décembre 2016 Amazon annonce AWS X-ray

Trace et spans

Span : une opération indivisible qui a eu lieu, composée d'événements horodatés

Trace : graphe d'une requête complète, composé de plusieurs spans



Tags

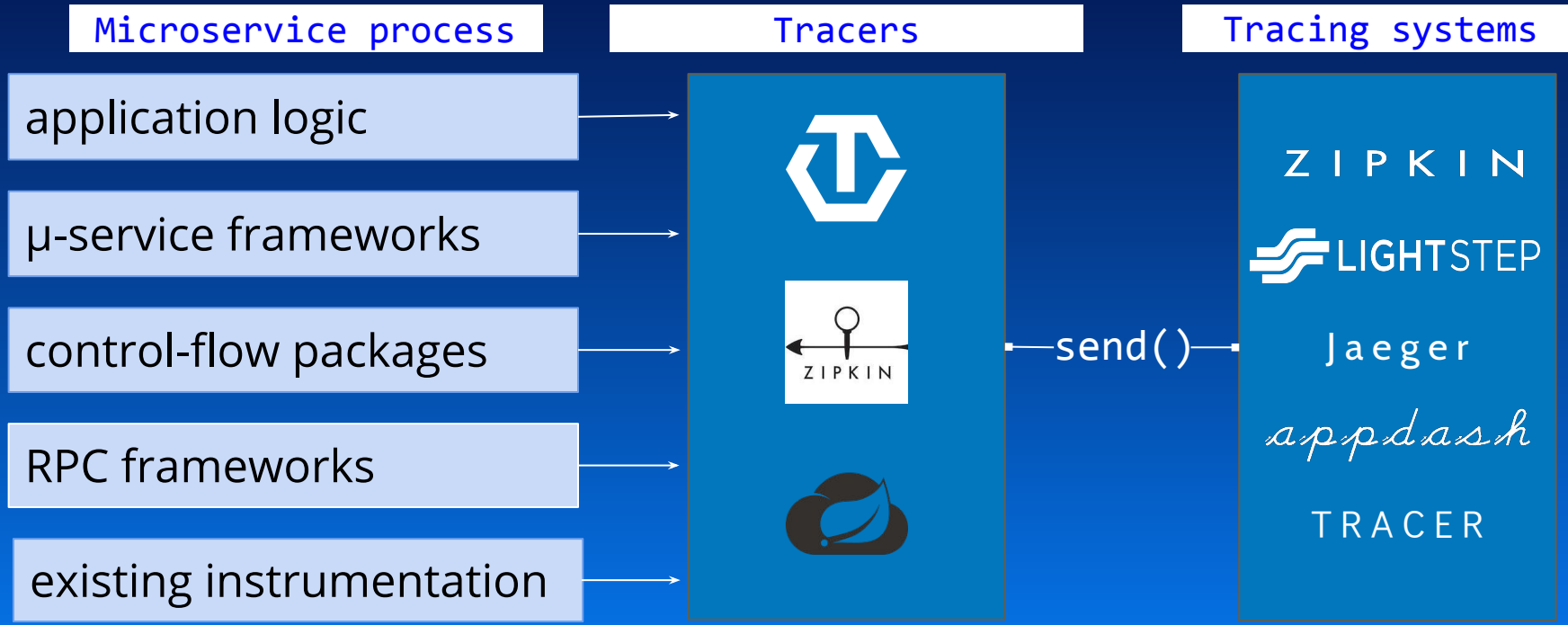
service2.http:/readtimeout: 3.557s

AKA: service1,service2

Date Time	Relative Time	Annotation	Address
19/12/2016, 14:19:23	307.000ms	Client Send	127.0.0.1:8081 (service1)
19/12/2016, 14:19:23	310.000ms	Server Receive	127.0.0.1:8082 (service2)
19/12/2016, 14:19:26	3.836s	Server Send	127.0.0.1:8082 (service2)
19/12/2016, 14:19:27	3.864s	Client Receive	127.0.0.1:8081 (service1)

Key	Value
error	Request processing failed; nested exception is org.springframework.web.client.ResourceAccessException: I/O error on GET request for "http://localhost:8082/blowup": Read timed out; nested exception is java.net.SocketTimeoutException: Read timed out
http.host	localhost
http.method	GET
http.path	/readtimeout
http.status_code	500
http.url	http://localhost:8082/readtimeout
mvc.controller.class	BasicErrorController
mvc.controller.method	error

Implémentation



Source: *Ben Sigelman*

Overhead ?

- Sampling
- Quelques headers dans la requête
  - Id du span parent
  - Id de la trace
  - Id du span
  - Flag concernant l'exportation
- Pour le tracing system
  - Tout est envoyé
  - Par chaque service
  - Asynchrone

Les outils

Pour tout le monde !

- Java
- Go
- Ruby
- Python

- Javascript
- C#
- Scala
- ...

Java : Spring cloud Sleuth

- Tracer
- Version 1.0.0 en mai 2016
- Corrélation de log et sampling
- Instrumentation spring boot
  - Servlet filters
  - Async endpoints
  - Rest template
  - Feign clients
  - zuul filters
  - ...

Zipkin server

- Tracing system
- Stocke les spans
  - Cassandra
  - Elasticsearch
  - Mysql
- UI pour analyser et rechercher les traces / spans / tags
- Graphe de dépendances entres les applications

Notre cas d'utilisation

L’application

2

1

Démo !

http://github.com/imflog/tracing-playground

Conclusion

Ce qu'il faut retenir

- Outil d'analyse de latence
- Peut ajouter de la corrélation de logs
- Propagation de contexte
- En temps quasi réel en production
- Sans overhead
- Très simple et peu coûteux !
- Bonus : carte des intéractions entres services
- Des communautés très actives

Sources

Google dapper paper

Martin Fowler blog

microservices.io

opentracing.io

Spring cloud sleuth

zipkin.io

Adrian Cole

Ben Sigelman

Marcin Grzejszczak

PARIS
BORDEAUX
NANTES
LYON
MARRAKECH
WASHINGTON DC
NEW-YORK
RICHMOND
MELBOURNE

contact@ippon.fr
www.ippon.fr - www.ippon-hosting.com - www.ippon-digital.fr
@ippontech
-
01 46 12 48 48