Devoxx France 2017

Caching et Performance

Ludovic Orban @bitronix
Aurélien Broszniowski @AurBroszniowski

Avant de commencer

https://github.com/aurbroszniowski/DevoxxFr2017

```
mvn install
```

et écoutez nous !

Qui sommes nous ?

Ludovic Orban

- Lead engineer chez Terracotta
- Auteur de Bitronix Transaction Manager
- Use ZFS or die!

Aurélien Broszniowski

- Lead engineer chez Terracotta
- Auteur de Rainfall Perf testing

Agenda

- Introduction
- JCache (JSR-107)
- Cache aside
- Cache through
- Tests de performance et caching
- Statistiques JSR-107
- Sizing du cache
- Warmup et persistance
- Plus de perf : Serialization
- Configurations et trade-offs

Sondage

- Qui ne connait rien au Caching ?

- Qui utilise déjà du Caching en production ?

- Qui a déjà eu des problèmes de cache en production ?

- Qui connait JSR-107 ?

Qu’est-ce qu’un cache ?

Une structure de données contenant une ***copie temporaire*** de vos données.

Ensemble de compromis, par exemple entre une utilisation plus élevée de la mémoire et la latence d’accès aux données.

Qu’est-ce qu’un cache n’est pas ?

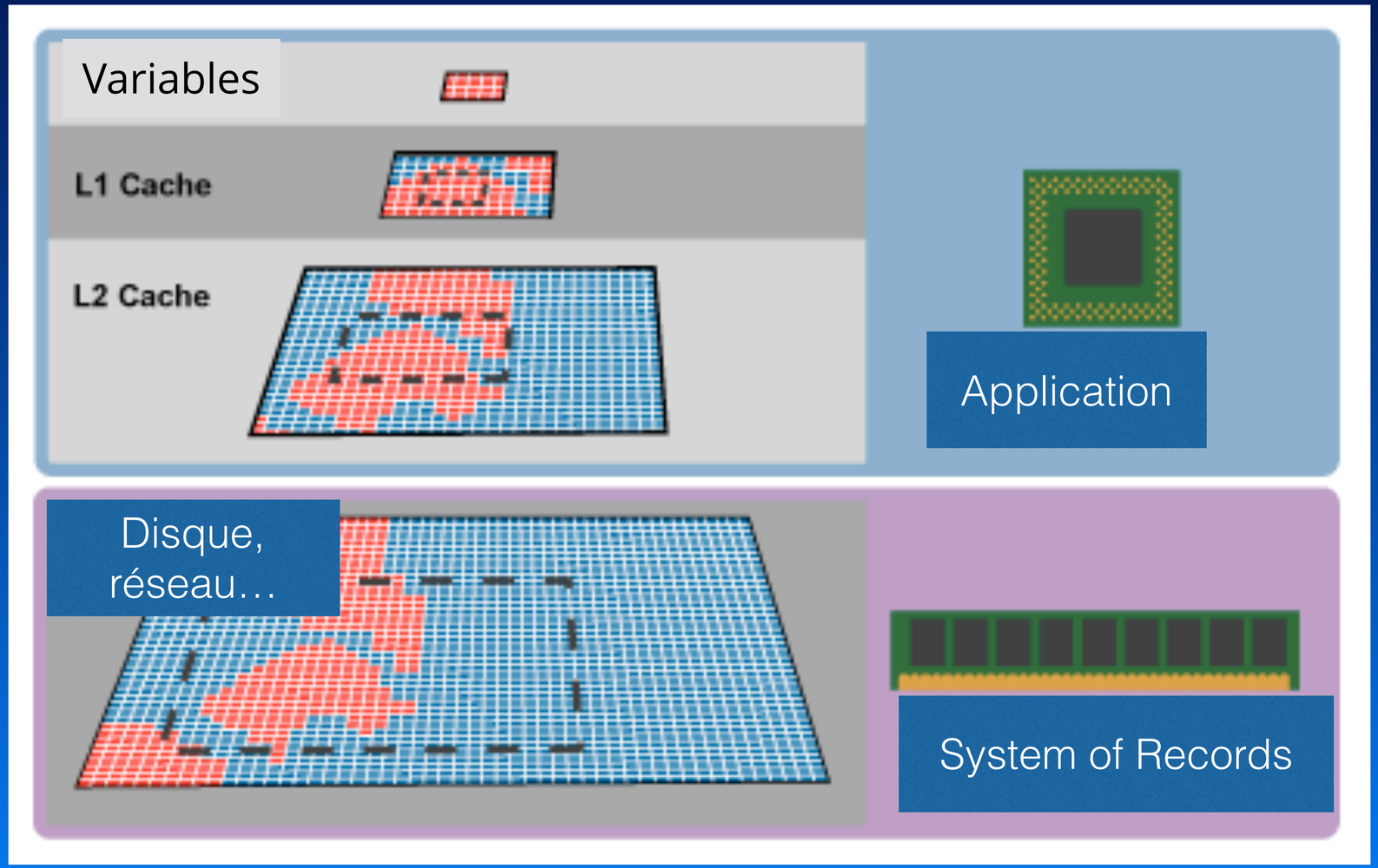
Une zone de stockage permanente de données.

Une copie complète de vos données.

La source unique d’informations.

Modèle du cache

Hardware



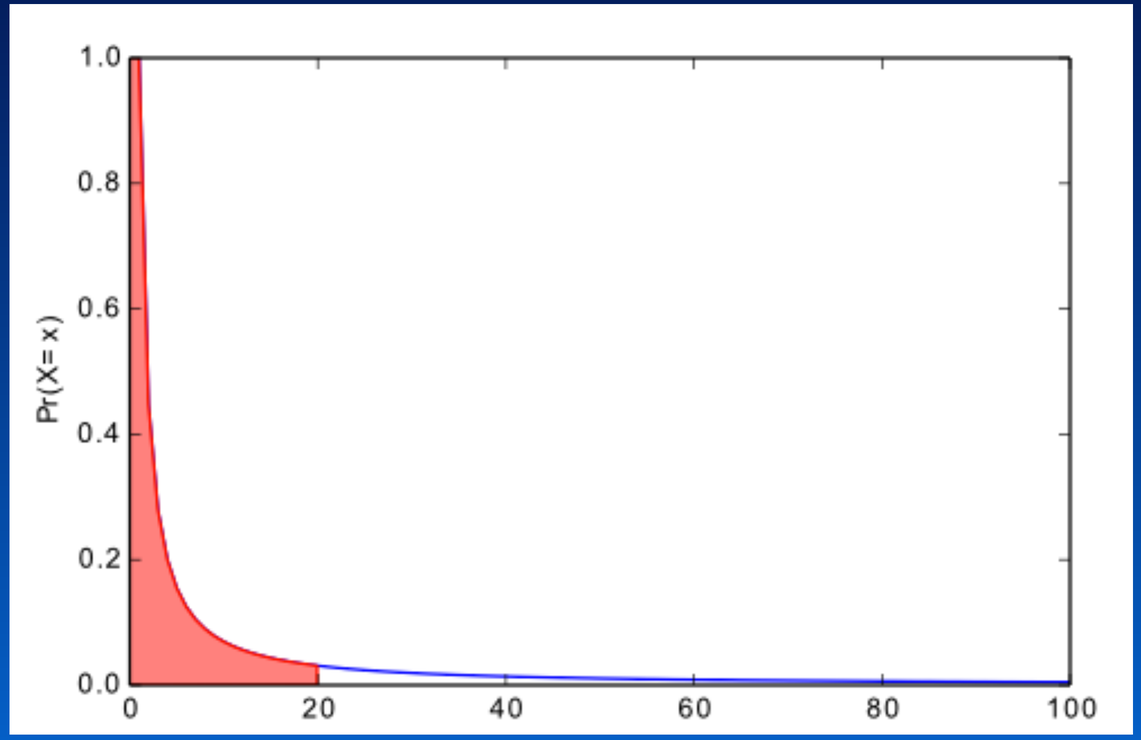
Données cibles

Les cibles de données principales lors de l'utilisation du caching sont:

Les données qui sont souvent réutilisées.

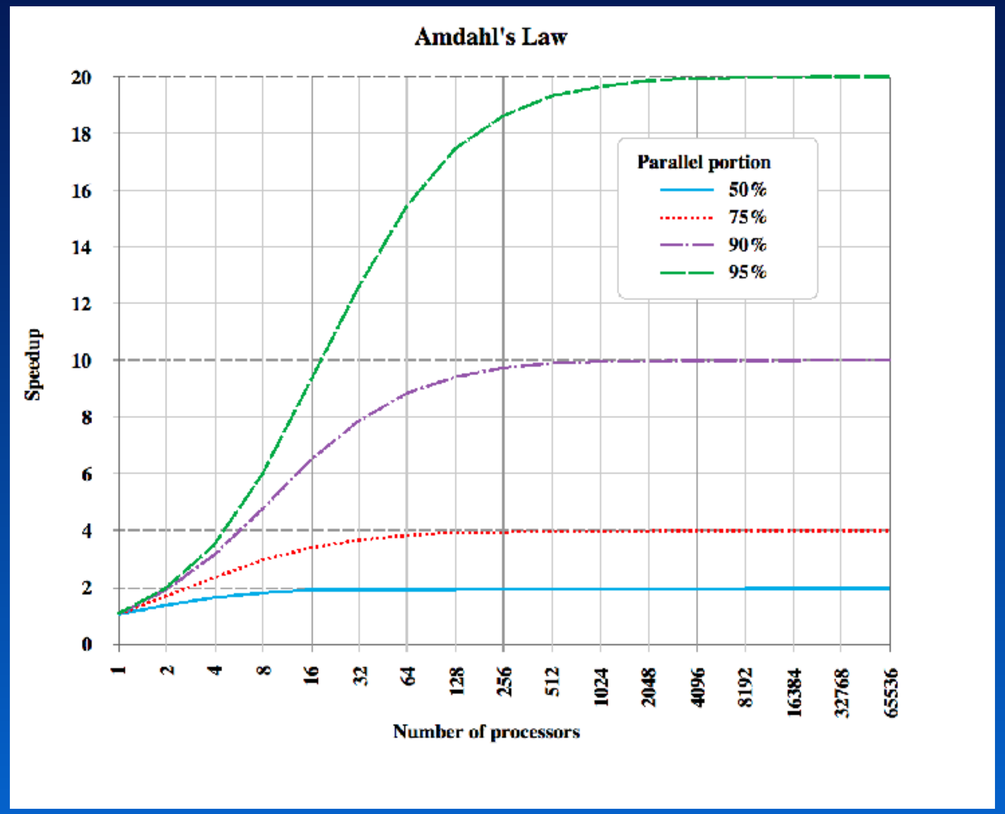
Les données qui sont cher à calculer ou à récupérer.

Pourquoi le caching fonctionne ?



La distribution de Pareto : 80/20

La loi d'Amdahl



La parallélisation des tâches améliore la performance jusqu'à une limite

Usages du caching

Applications consommatrice de temps CPU:

- Solution classique : Amélioration algorithmique, parallélisation
- Alternative : Cacher les traitements intensifs (ex: génération de pages web)

Applications limitées par les IO (réseau, disque, base de données)

- Solution classique : Upgrader le hardware
- Alternative : Le cache stocke les données localement

Exercice 0
Pas de cache

Description de l'appli

- Web app spring multi-couches
  - Repository (Accès à la base de données)
  - Service (Logique métier)
  - API Rest
- Tests de Performance

Exercice 1

JCache

JSR-107

- Standard guidé par le Java Community Process
- Spécifie l'API et la sémantique pour le caching temporaire en mémoire d'objets Java, incluant l'accès partagé aux données, la validation et la cohérence des données entre les JVMs.

javax.cache API

Conteneur d'instances de `CacheManager`, identifié par une `URI` et un `ClassLoader`

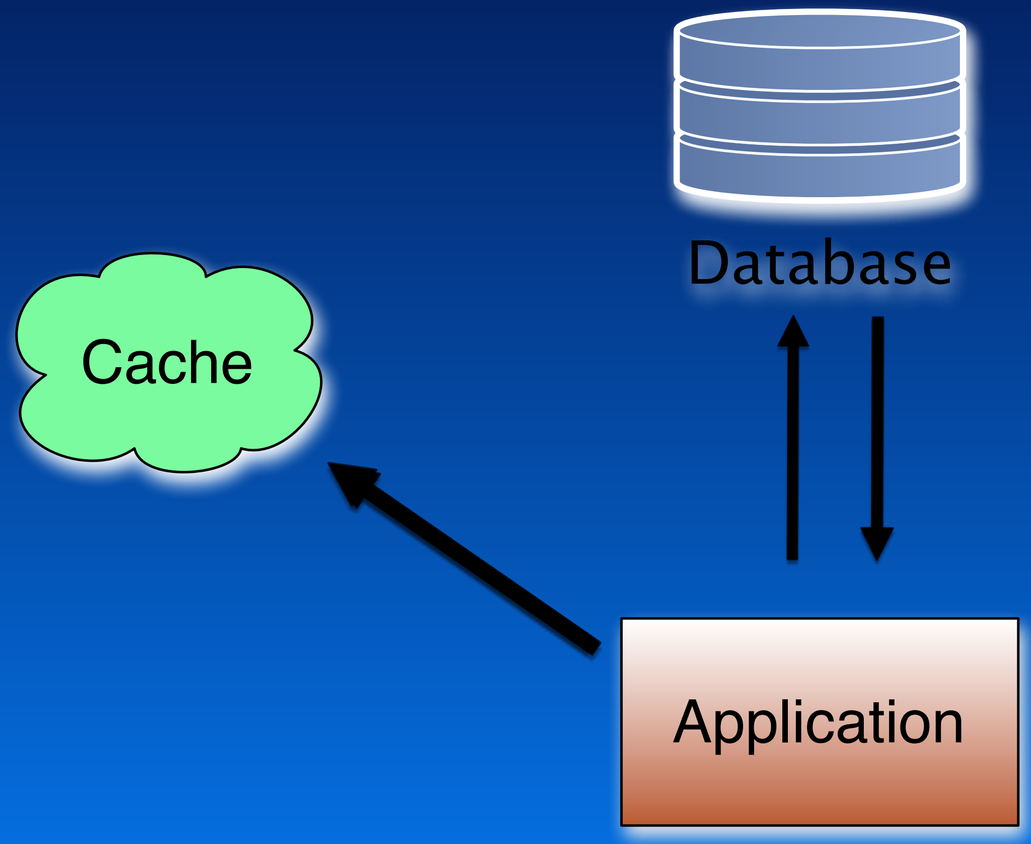
```
CachingProvider provider = Caching.getCachingProvider();

CacheManager cacheManager = provider.getCacheManager();

Cache<Long, String> myCache = cacheManager.getCache("myCache", Long.class, String.class);
```

Conteneur d'instances de `Cache`, qui gère leur cycle de vie

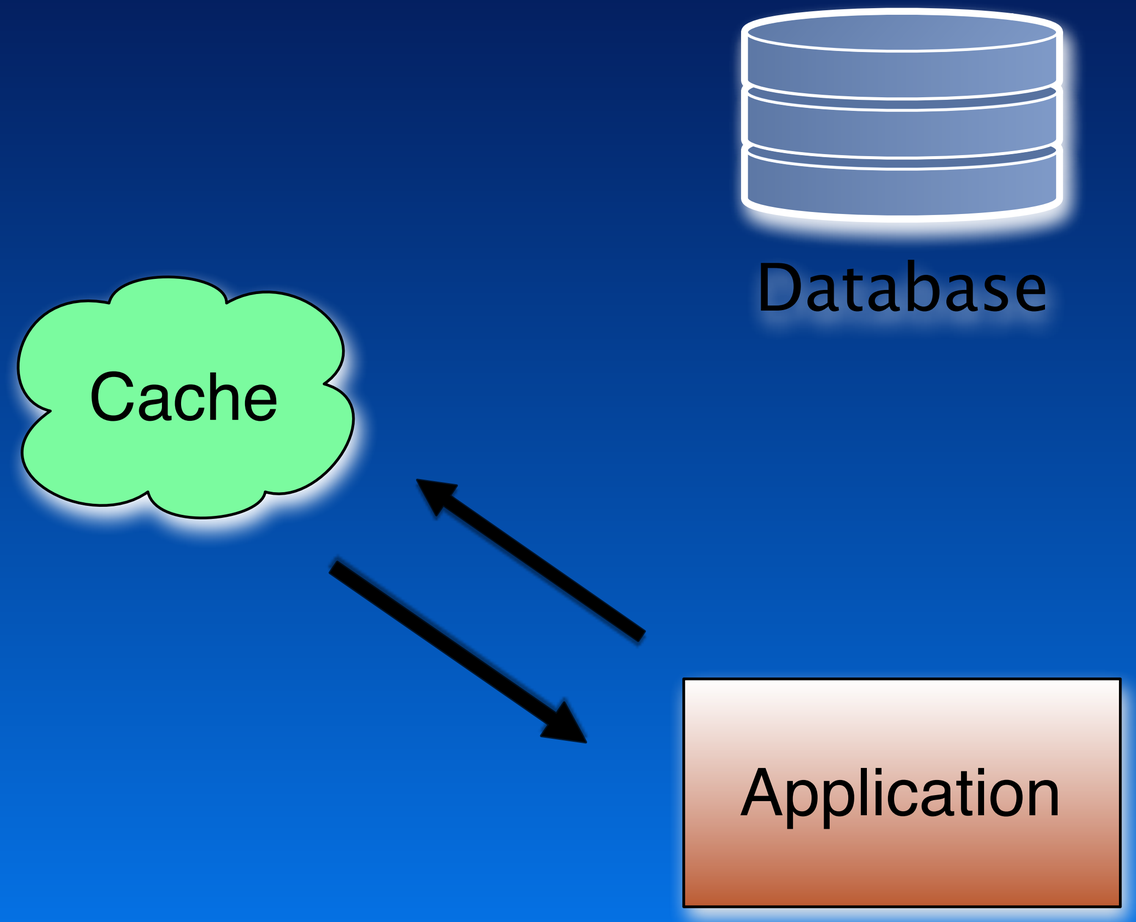
Homologue de la `ConcurrentMap<K, V>`, point d'entrée principal.

Cache aside

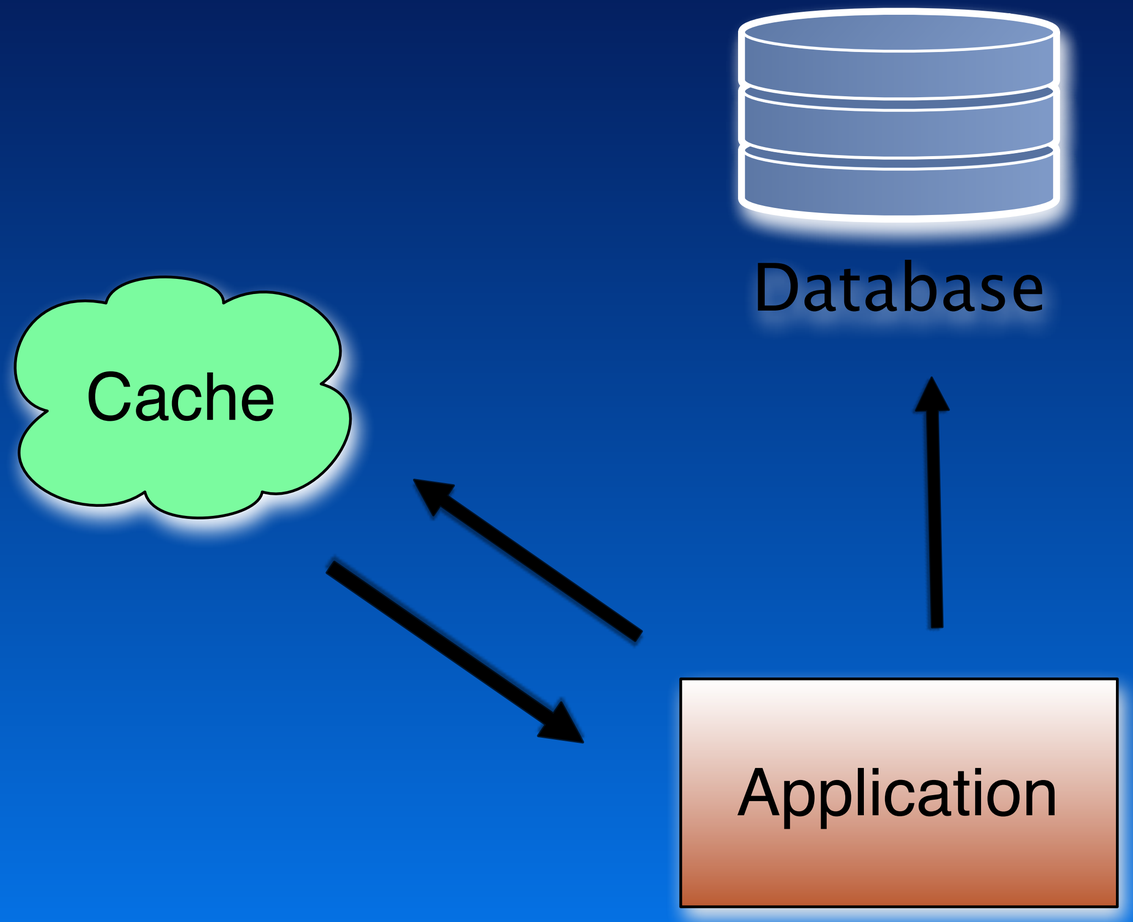
Cache aside - miss



Cache aside - hit



Cache aside - write

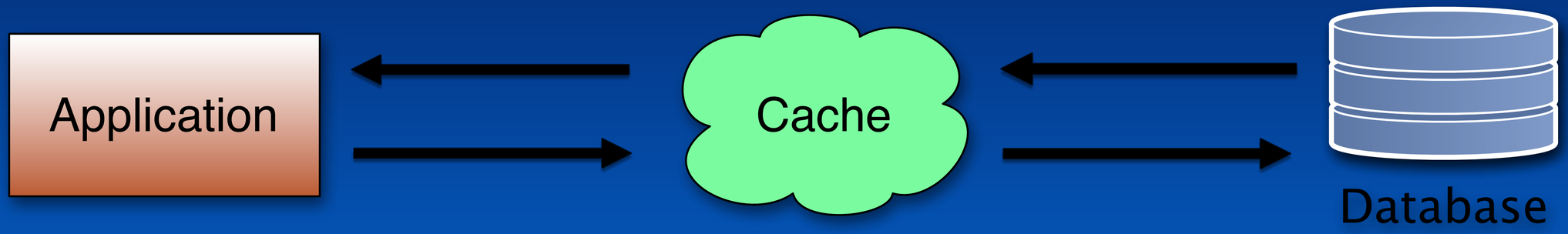


Exercice 1
Cache Aside
A vous de coder!

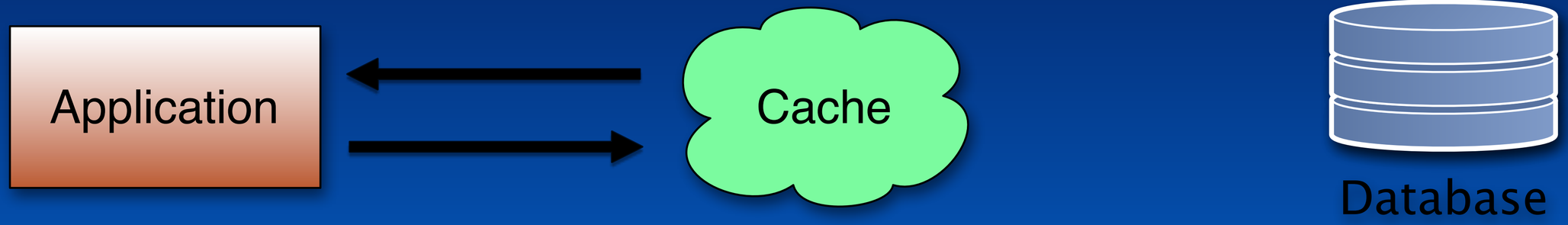
Exercice 2

Cache through

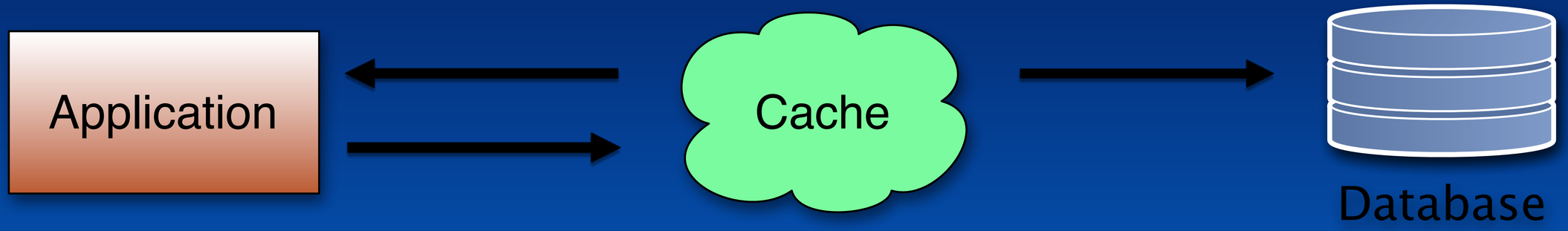
Cache through - miss



Cache through - hit



Cache through - write



Exercice 2
Cache Through
A vous de coder!

Exercice 3

Tests de charge

Niveau de benchmarking

- Macro-benchmark (JMeter, Gatling) : Test end to end
- Micro-benchmark (JMH) : Test d’optimisation de code
- Meso-benchmark (Rainfall) : Test centré sur un composant

Tests de performance

Définir un scénario d’opérations

Exécuter le Scénario

Collecter les statistiques

Rapporter les résultats

Framework Rainfall

```
Runner.setUp(
    Scenario.scenario("load test")
        .exec(new Operation() {
          @Override
          public void exec(…) throws TestException {

            long start = getTimeInNs();
            // This is what we measure
            service.someLogic(id);
            //
            long end = getTimeInNs();
            statisticsHolder.record(“READ”, (end - start), READ);
          }
       }))
    .warmup(during(20, TimeDivision.seconds))
    .executed(during(1, TimeDivision.minutes))
    .config(report(Results.class).log(text(), html()))
    .start();
```

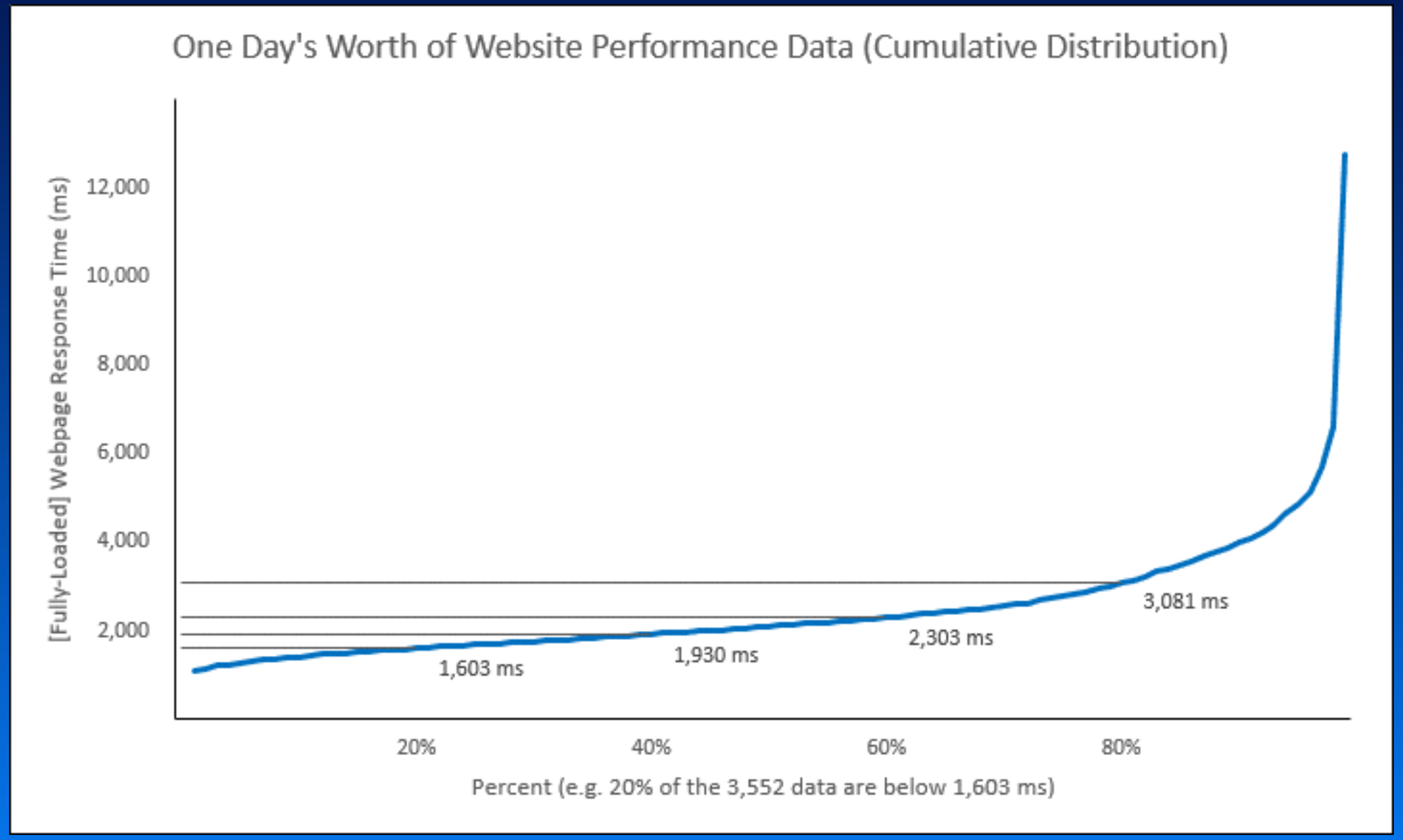
Exercice 3
Tests de charge
A vous de jouer!

Performance et investigation

Performance et SLA

- La première tâche dans l'amélioration de performances : Définir le SLA.
- Amazon
- SLA et percentile distribution

Percentile distribution



Investigation et optimisation

Investigation :

- Prendre connaissance du contexte de l’application
- Rechercher les causes possibles de problèmes

Optimisation :

- Définir un but
- Travailler incrémentalement

Exercice 4

Les statistiques JSR-107

Statistiques JSR-107

Désactivées par défaut

Accès via un MBean

Compteurs standards

- Gets
- Miss count
- Hit percentage
- Eviction count

Exercice 4
Statistiques JSR-107
A vous de jouer!

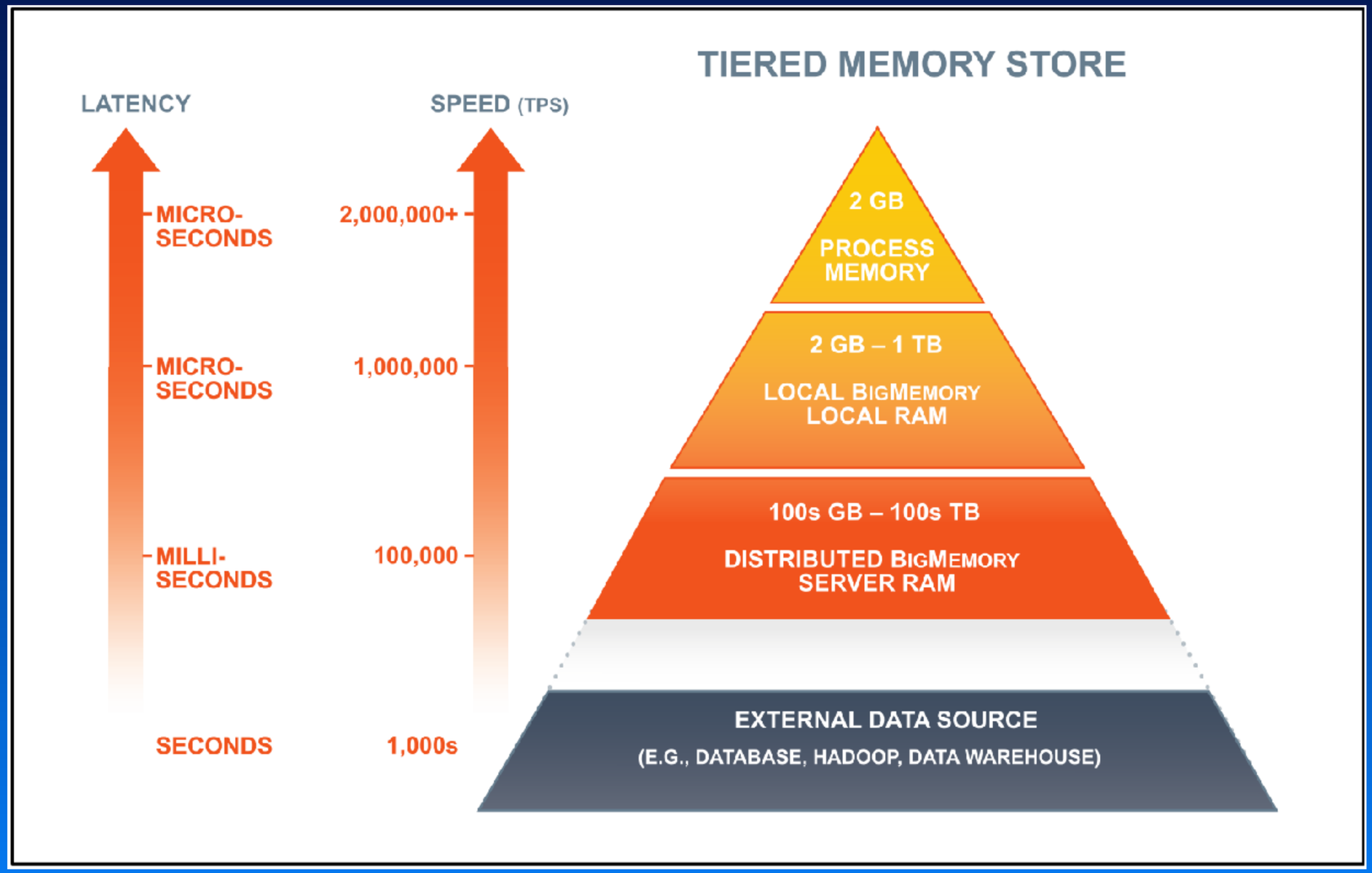
Statistiques JSR-107

- Gets
- Miss count
- Hit percentage
- Eviction count

Exercice 5

Cache size

Cache size


TIERED MEMORY STORE
LATENCY
SPEED (TPS)
MICRO-SECONDS
MICRO-SECONDS
MILLI-SECONDS
SECONDS
2,000,000+
1,000,000
100,000
1,000s
2 GB
PROCESS MEMORY
2 GB – 1 TB
LOCAL BIGMEMORY
LOCAL RAM
100s GB – 100s TB
DISTRIBUTED BIGMEMORY
SERVER RAM
EXTERNAL DATA SOURCE
(E.G., DATABASE, HADOOP, DATA WAREHOUSE)

JSR-107 et sizing

Rien dans le standard pour configurer la taille d'un cache

Ehcache config:

```
<cache alias="someCache5">
 …
 <resources>
    <heap unit="entries">1</heap>
    <offheap unit="MB">10</offheap>
    <disk unit="MB">200</disk>
  </resources>
</cache>
```

Statistiques Ehcache

Pas de statistiques pour les tiers dans le standard

Statistiques Ehcache (comparables à JSR-107):

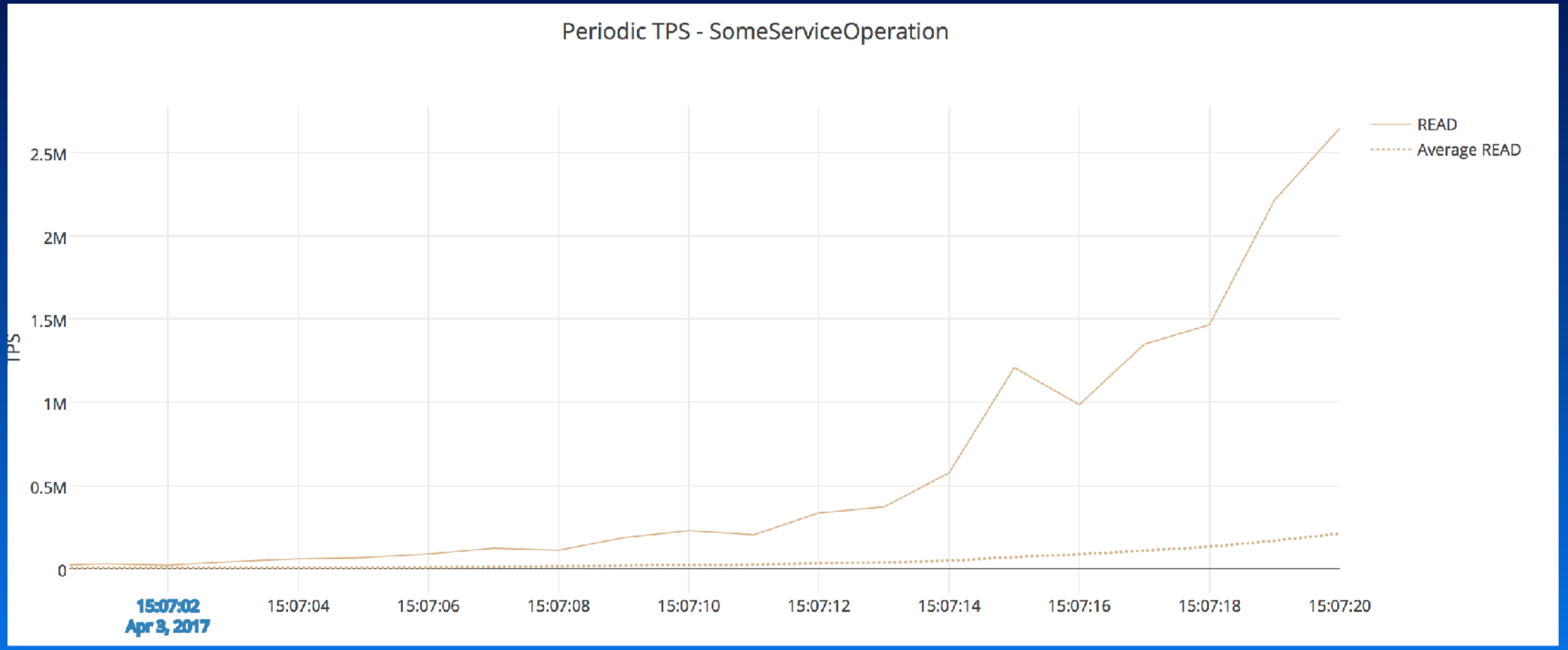
- Cache
- Par tier (indépendantes)

Exercice 5
Cache size
A vous de jouer!

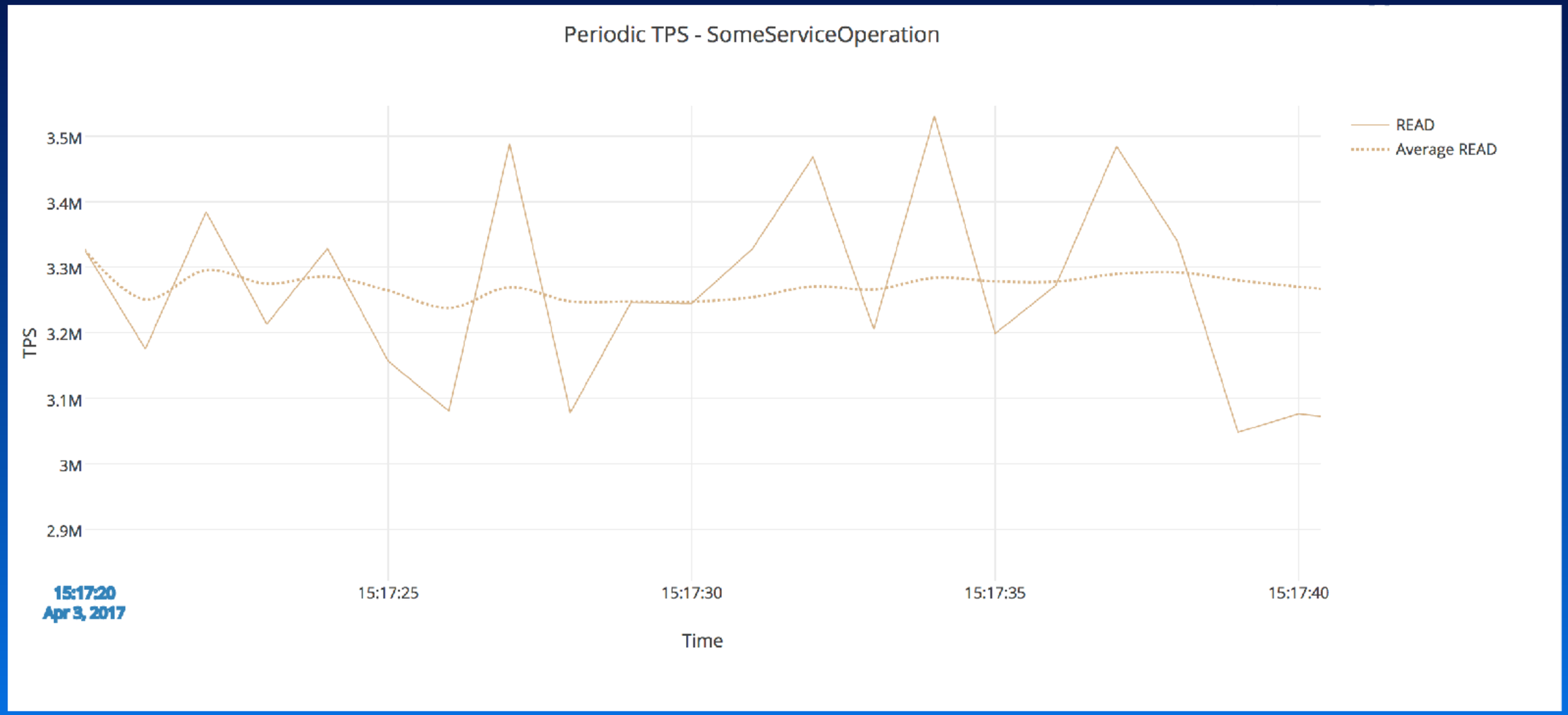
Exercice 6

Warmup et Persistance

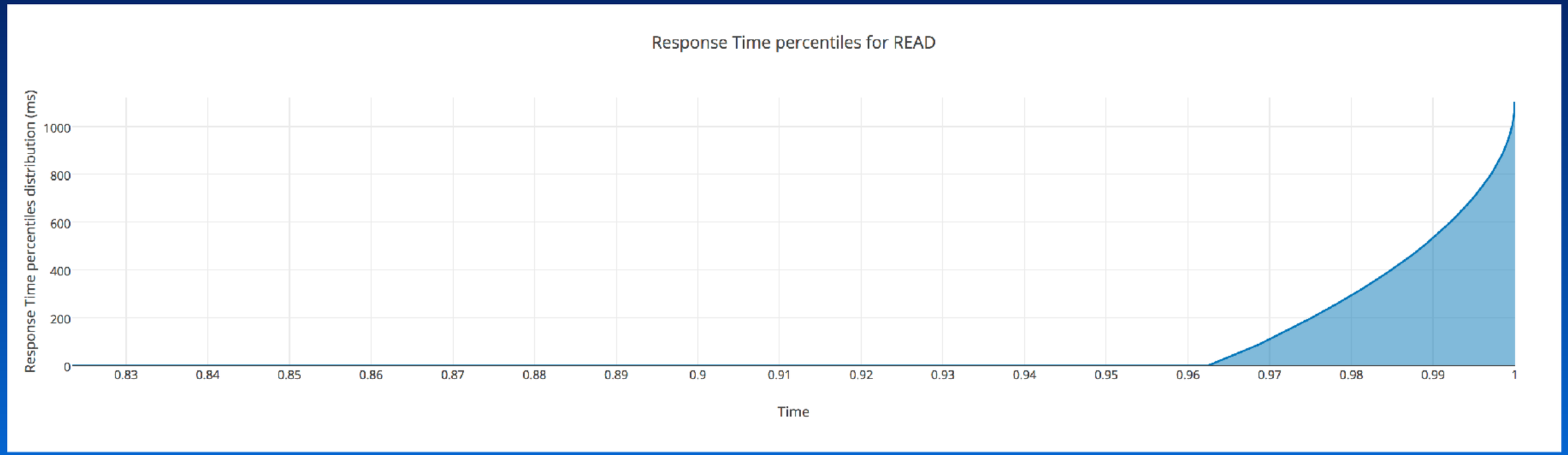
Sans warmup


Periodic TPS - SomeServiceOperation
READ
Average READ
TPS
2.5M
2M
1.5M
1M
0.5M
0
15:07:02
Apr 3, 2017
15:07:04
15:07:06
15:07:08
15:07:10
15:07:12
15:07:14
15:07:16
15:07:18
15:07:20

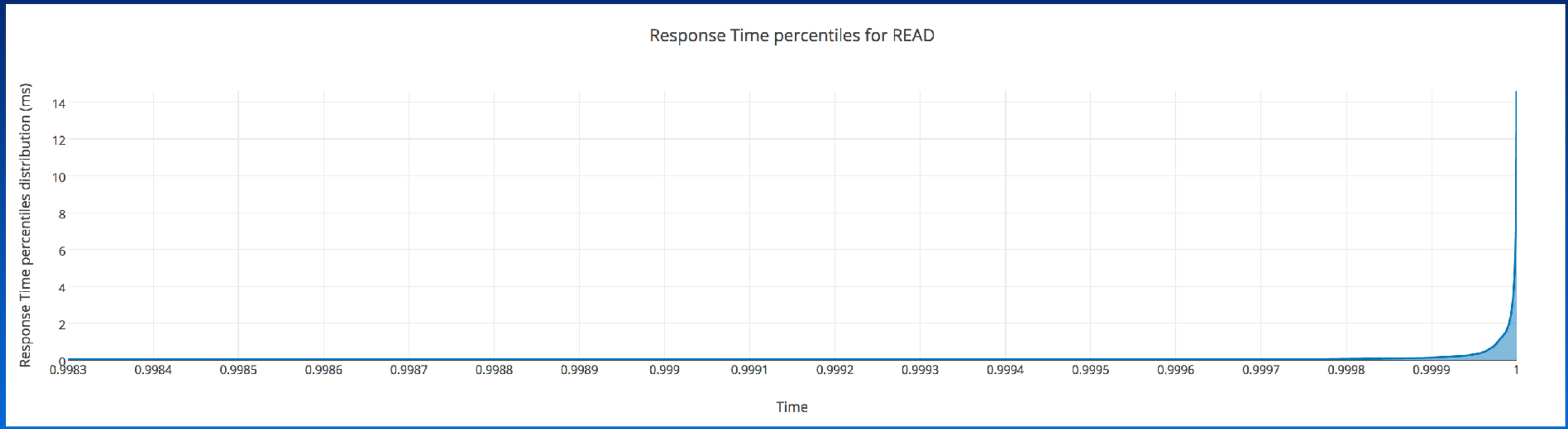
Avec warmup


Periodic TPS - SomeServiceOperation
READ
Average READ
TPS
3.5M
3.4M
3.3M
3.2M
3.1M
3M
2.9M
15:17:20
Apr 3, 2017
15:17:25
15:17:30
15:17:35
15:17:40
Time

Percentile distribution : Pas bien



Percentile distribution: bien ! (mais avec modération)



Persistance

Qu’est ce que la persistance

```
<disk unit="MB" persistent="true">200</disk>
```

Avantages

- Le cache est rempli après redémarrage
- Le warmup est plus court

Persistant != résistant au crash

Exercice 6
Warmup et persistance
A vous de jouer!

Exercice 7

Serializers

La serialization

Pourquoi la serialization?

Par défaut : Serialization Java

Serialization : Un mécanisme complexe - la solution universelle existe mais est forcément moins rapide qu'une version spécialement dédiée

Voir le talk Devoxx : Terracotta Off-Heap pour les nuls: https//www.youtube.com/watch?v=ezTrDlrIz5o

Exercice 7
Serializers
A vous de coder!

Exercice 8

Configuration et Trade-offs

ARC - byte sizing

- ARC byte sizing (plutôt que par nombre d’entrées)

```
<heap unit=“MB">10</heap>
```

- Plus pratique
- Moins performant pour le remplissage

Expiration

- Expiration : TTL

```
<expiry>
  <ttl unit="seconds">10</ttl>
</expiry>
```

- Fraîcheur des données
- Impact sur les performances: le cache “refroidit” naturellement
- Expiration != eviction !!!

Exercice 8
Configuration et trade-offs
A vous de jouer!

Conclusion