Using JCache to speed up your apps

Vassilis Bekiaris
Software Engineer, Hazelcast
@karbonized1

About me

- First computer:
  Amstrad 6128 green screen

- Favorite languages I never used in production: Ada, CLISP
- Freelancer (-2008)
- Software Architect, Team Leader, Jack of all Trades (2008-2016)
- Software Engineer, Hazelcast (2016-)

Outline

- Caching basics
- Introducing JSR-107 (JCache)
- Using JCache

Why cache?

- Performance
- Offload expensive or non-scalable parts of your architecture
- Buffer against load variability
- Usually fast and easy to apply

When to use caching

- When applications use the same data more than once
- When fetching or producing the data again is expensive

Caches in RAM and beyond

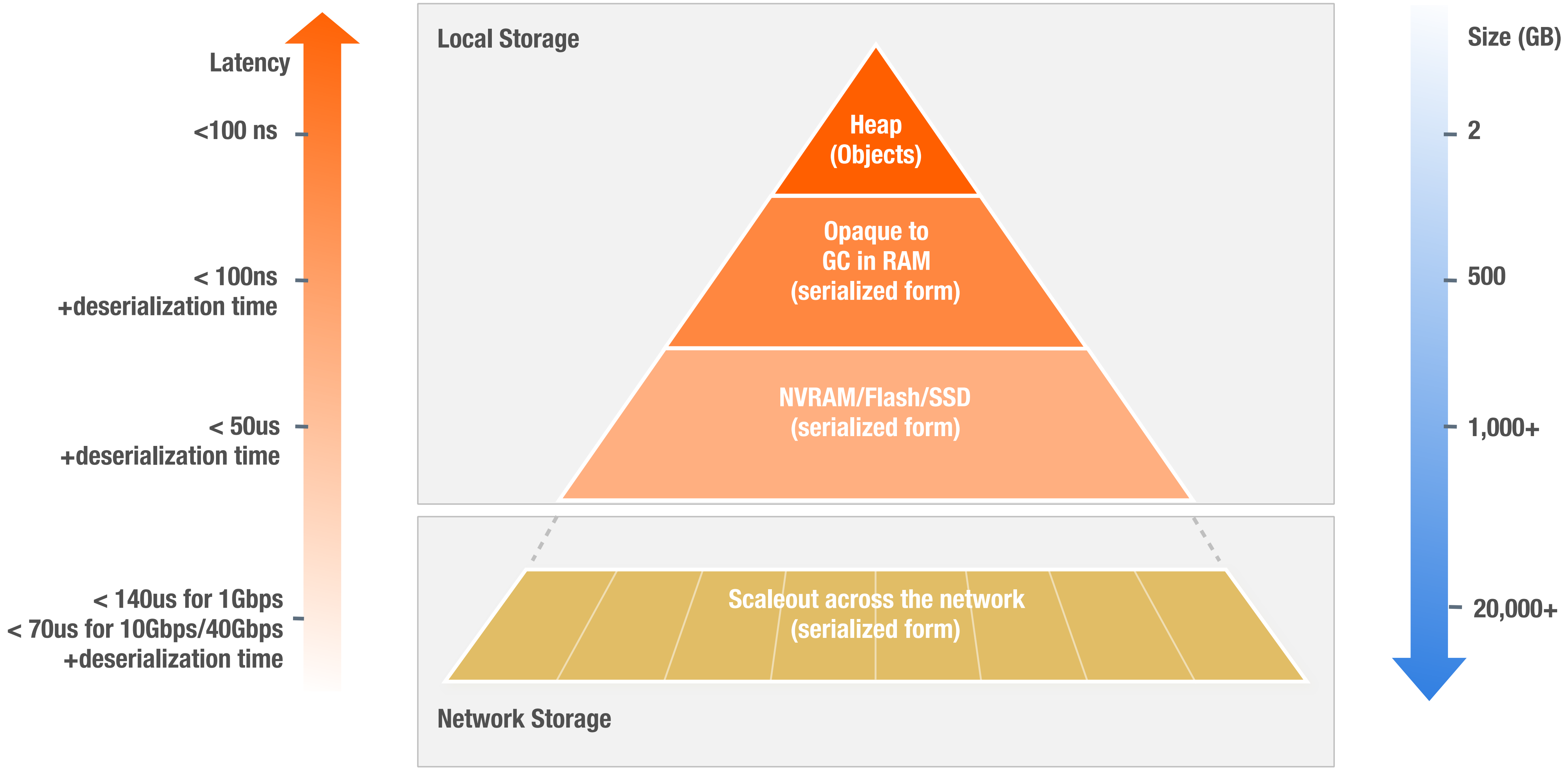
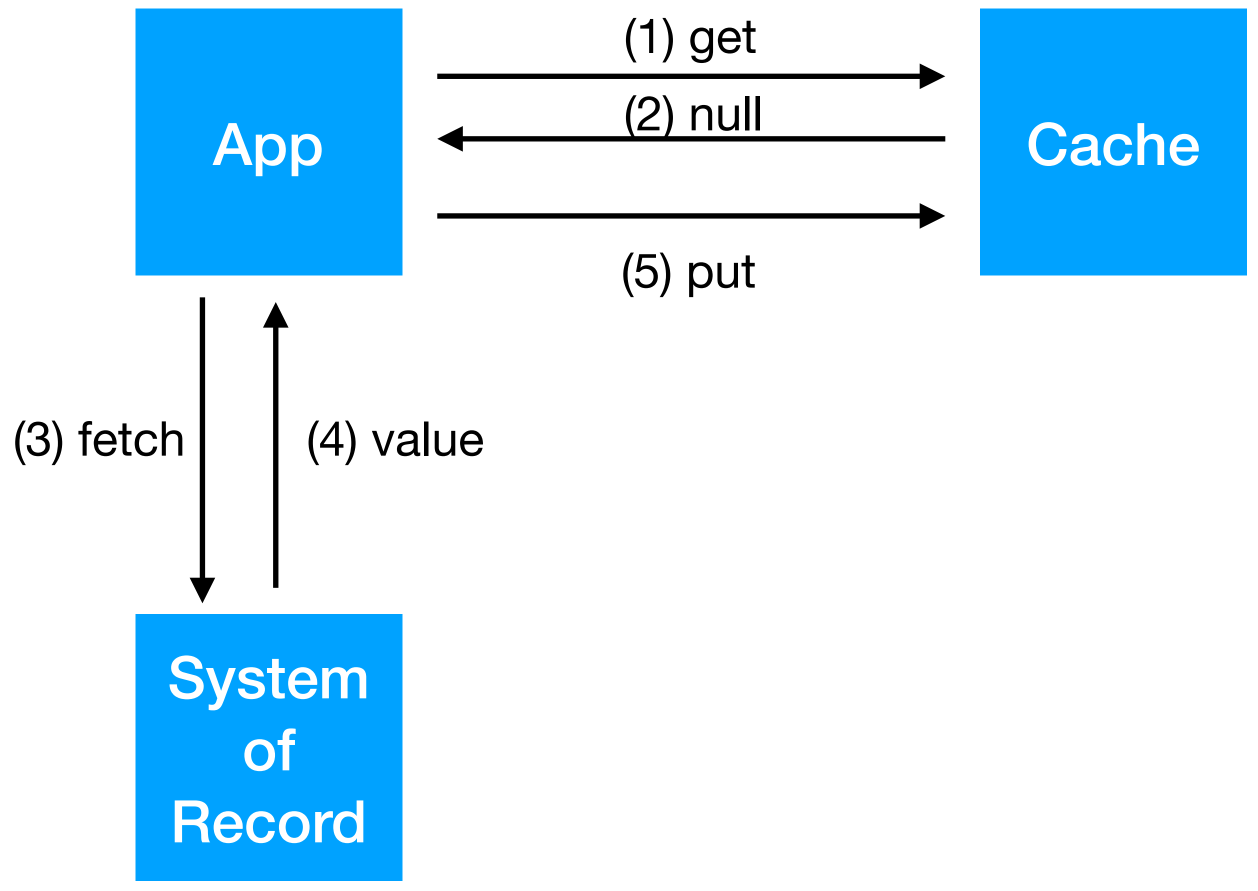


Diagram: Greg Luck

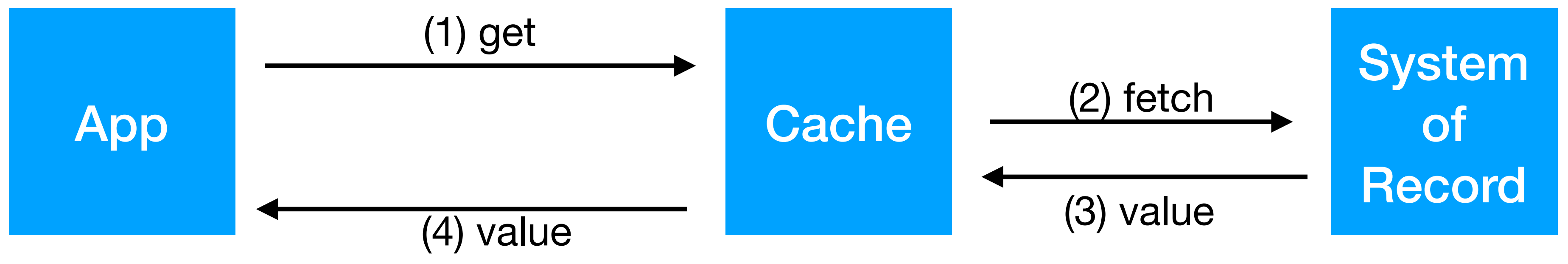
Implementation patterns

- Cache aside



Implementation patterns

- Cache through



Other implementation considerations

- In-process
- Distributed

Introducing JSR-107

- The standard way to cache for Java applications
- One of the longest running JSRs
  - Started in 2001
  - JSR-107 1.0 final released in March 2014
  - JSR-107 1.1 maintenance release (draft review done, expected mid-December 2017)
- Target: Java SE 6+

JCache implementations

- Apache Ignite
- Blazing Cache
- cache2k
- Caffeine
- Coherence (Oracle)
- Ehcache
- Hazelcast
- Infinispan
- …
  (https://jcp.org/aboutJava/communityprocess/implementations/jsr107/index.html)

Cache != Map

java.util.Map
Key-Value Based API
Supports Atomic Updates
Entries Don’t Expire
Entries Aren’t Evicted
Entries Stored On-Heap
Store-By-Reference

javax.cache.Cache
Key-Value Based API
Supports Atomic Updates
Entries May Expire
Entries May Be Evicted
Entries Stored Anywhere (ie: topologies)
Store-By-Value and Store-By-Reference
Supports Integration (ie: Loaders / Writers)
Supports Observation (ie: Listeners)
Entry Processors
Statistics

Cache now!

- Add javax.cache:cache-api:1.0.0 to your classpath
- Add an implementation
- Hello, world!

JCache API

- Caching
  - “Service loader”: locates implementation, supplies CachingProvider(s)
- CachingProvider
  - Creates & manages CacheManagers (per {URI, ClassLoader})
- CacheManager
  - Manages Caches lifecycle
- Cache
  - That’s what your app uses!

Entry processors

- Data transformations
- Computations

Listeners

- Caches are observable
    - `CacheEntryCreatedListener`
    - `CacheEntryUpdatedListener`
    - `CacheEntryRemovedListener`
    - `CacheEntryExpiredListener`

Cache Loader/Writer

- Integration with system of record
- Read-through, write-through

Annotations

- `@CacheResult`

  `@CachePut`

  `@CacheRemove`

  `@CacheRemoveAll`

- JSR107 RI provides support for CDI, Spring & Guice

Annotations

```
@CacheDefaults(cacheName = “users”)
class User {

  @CacheResult
  User getUser(long id);

  @CachePut
  void createUser(long id, @CacheValue User user);

  @CacheRemove
  User removeUser(long id);
  …
}
```

Management & Statistics

- Via JMX
- Can be enabled/disabled at runtime
  - `CacheManager.enableStatistics(cacheName, true)`
  - `CacheManager.enableManagement(cacheName, true)`

Integrations

- Spring
- CDI
- Payara server

JSR-107 links

- JCP Project:
  - http://jcp.org/en/jsr/detail?id=107
- Source Code:
  - https://github.com/jsr107
- Forum:
  - https://groups.google.com/forum/?fromgroups#!forum/jsr107