JCache

Using JCache

GREG LUCK, CO-SPEC LEAD JSR107 @GREGRLUCK
CEO | HAZELCAST
28 NOVEMBER 2015

Agenda

- Introduction to Caching
- Java Caching (JCache), JSR-107
- Code Demo

Introduction to Caching

Benefits of Caching

- Performance
- Offload expensive or non-scalable parts of your architecture
- Scale up – get the most out of one machine
- Scale out – add more capacity with more machines
- Excellent Buffer against load variability

And…

Usually very fast and easy to apply

When to Use Caching

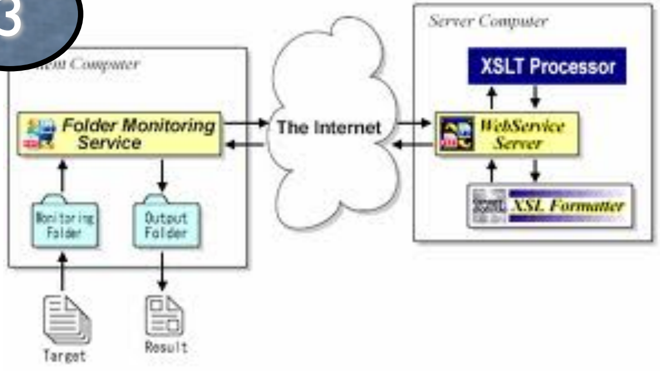
– When applications use the same data more than once

– When cost (time / resources) of making an initial copy is less than fetching or producing the data again or when faster to request from a Cache

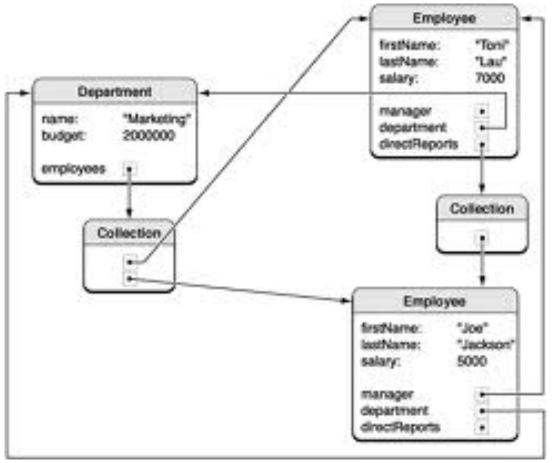
Common Problem Areas that Benefit

1 Anything Web Scale

3 Anything where the data is across the network

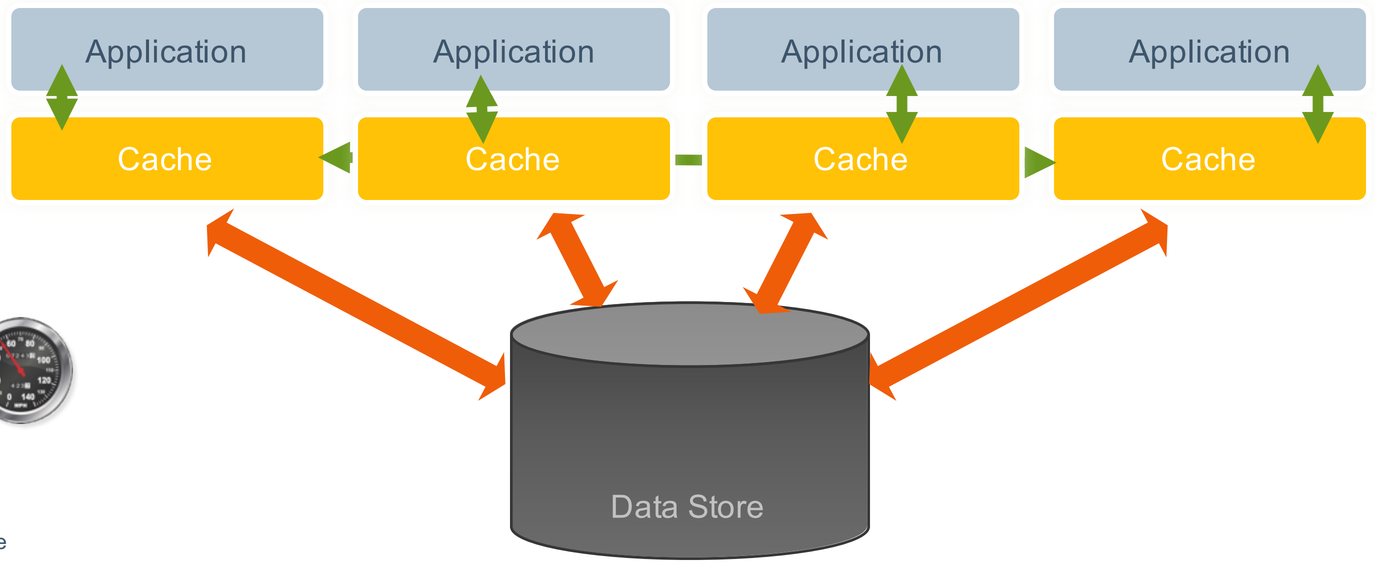


2 Compound Data Objects



4 Data Persistence

Database Caching

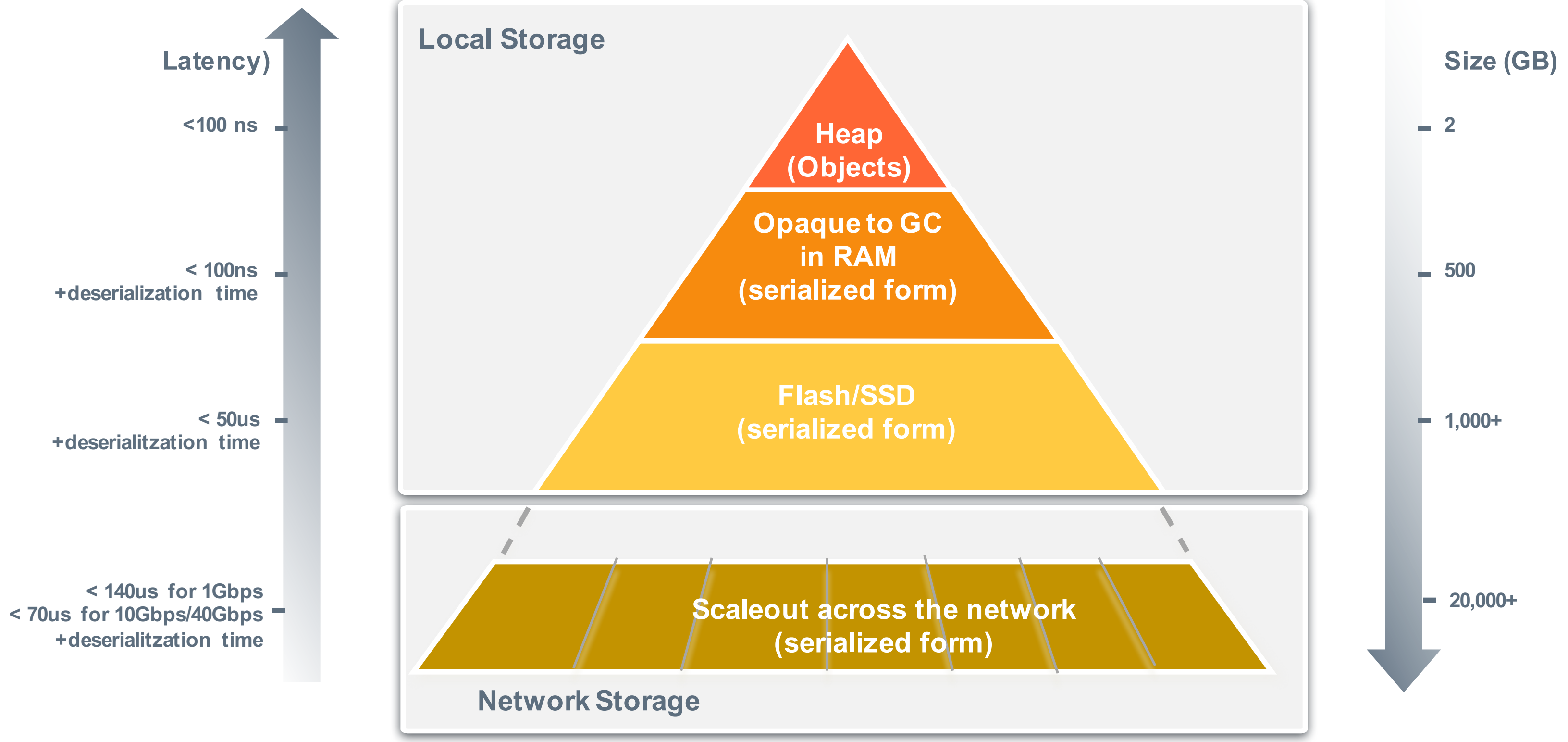


~200 us
Average Response Time

Speed	🟥	🟨	✓
Costs	🟥	🟨	✓
Scalability	🟥	🟨	✓

Moving data from the database into the cache increases processing speed and can reduce database licensing and maintenance costs.

Caches are built primarily in RAM in-process or distributed


Latency)
<100 ns
< 100ns +deserialization time
< 50us +deserialitzation time
< 140us for 1Gbps
< 70us for 10Gbps/40Gbps
+deserialitzation time
Local Storage
Heap (Objects)
Opaque to GC in RAM (serialized form)
Flash/SSD (serialized form)
Scaleout across the network (serialized form)
Network Storage
Size (GB)
2
500
1,000+
20,000+

Estimated Performance Improvements

amdahl's law

Predicted System Speedup

=

1 / ((1 – Proportion Sped Up) + Proportion Sped Up / Speed up))

Cache Efficiency

= cache hits / total hits

- High efficiency = high offload
- High efficiency = high performance
- How to increase:
  - Put reference data in the cache
  - Put long lived in the cache.
  - Consider frequency of mutability of data
  - Put highly used data in cache
  - Increase the size of the cache. Today you can create TB sized caches

Problems to Consider

- Standalone Caches and the N * problem
  - As each entry expires, the backing system gets N requests for data where n is the number of standalone caches. Solution: Use a distributed cache
- Consistency with the System of Record
  - How to keep the cache in sync with changes in a backing system. Solution: Match mutability of data with data safety configuration. Update the cache and backing store at the same time.
- Consistency with other cache nodes
  - How to keep all cache nodes in sync: Solution: Use a distributed cache and match consistency configuration with data mutability

Java Caching (JCache)

Java Caching (JCache)

- **What?**
  - Java Caching (JCache) standardized Caching for the Java Platform*
  - A common mechanism to create, access, update and remove information from Caches

- **How?**
  - JSR-107: Java Caching Specification (JCache)
  - Java Community Process (JCP) 2.9

Java Caching (JCache)

- **Why?**
  - Standardize! Standardize! Standardize!
    - Core Caching Concepts
    - Core Caching API
  - Provide application portability between Caching solutions
    - Big & Small, Open & Commercial
  - Caching is ubiquitous!

Java Caching (Jcache)

When?

Here Now!

Item	Date
JCache Final Spec Released	18 March 2014
Spring 4.1	September 2014
Hazelcast 3.3.1 TCK Compliant	September 2014
Hazelcast 3.4 (with High-Density Memory Store)	November 2014
Hazelcast 3.6 (with High-Density Caching)	July 2015

Implementations

- Implementations
  - *JCache Reference Implementation*
  - Hazelcast
  - Oracle Coherence
  - Terracotta Ehcache
  - Infinispan
  - GridGain
  - TayzGrid
- Keep Track
  - https://jcp.org/aboutJava/communityprocess/implementations/jsr107/index.html

Java Caching (JCache)

- **Which Platform?**

JCache Deliverable	Target Platform
Specification (SPEC)	Java 6+ (SE or EE)
Reference Implementation (RI)	Java 7+ (SE or EE)
Technology Compatibility Kit (TCK)	Java 7+ (SE or EE)
Demos and Samples	Java 7+ (SE or EE)

Java Caching (JCache)

Project Hosting

- 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

Java Caching (JCache)

How to get it.

Apache Maven: (via Maven Central Repository)

```
<dependency>
   <groupId>javax.cache</groupId>
   <artifactId>cache-api</artifactId>
   <version>1.0</version>
</dependency>
```

Caches and Caching

Caches and Caching

JSR107 Cache Definition: A high-performance, low-latency data-structure* in which an application places a temporary copy of information that is likely to be used more than once

Maps vs Cache APIs

java.util.Map (Java 6/7)
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 (Java 6)
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

JCache: Features

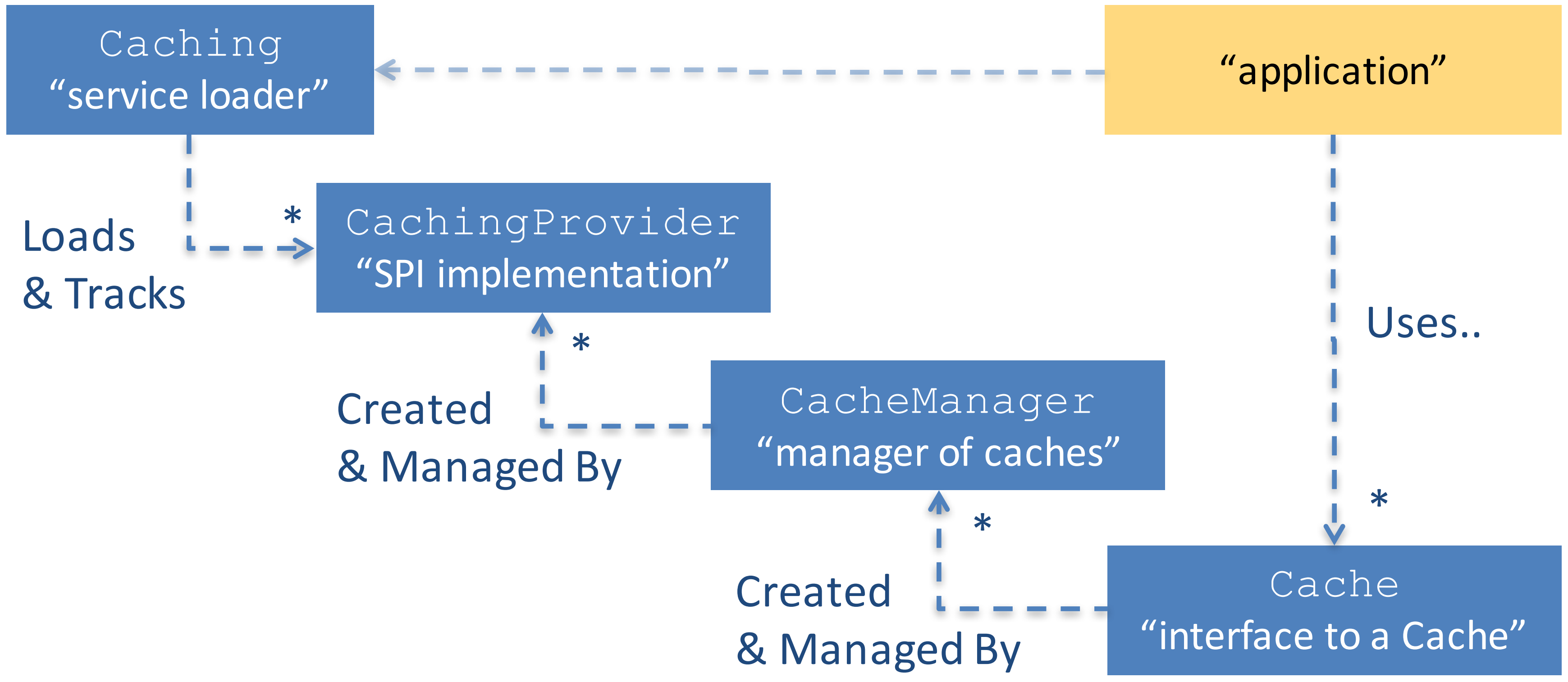
- `java.util.ConcurrentMap` like API
- Atomic Operations
- Lock-Free
- Read-Through / Write-Through Integration Support
- Cache Event Listeners
- Fully Generic API = type-safety
- Statistics
- Annotations (for frameworks and containers)
- Store-By-Value semantics (optional store-by-reference)

JCache: Features

- Topology Agnostic
  - Topologies not defined or restricted by the specification
- Efficiently supports:
  - “local” in-memory Caching and
  - “distributed” server-based Caching

JCache Key Classes/Interfaces

JCache: Runtime Structure



JCache: Cache Managers

`javax.cache.CacheManager`

- Establishes, configures, manages and owns named Caches
  - Caches may be pre-define or dynamically created at runtime
- Provides Cache infrastructure and resources
- Provides Cache “scoping” (say in a Cluster)
- Provides Cache ClassLoaders (important for store-by-value)
- Provides Cache lifecycle management

JCache: Hello World

(via a Cache Manager)

```
// acquire the default CacheManager
CacheManager manager = Caching.getCacheManager();

// acquire a previously configured cache (via CacheManager)
Cache<Integer, String> cache =
    manager.getCache("my-cache", Integer.class, String.class);

// put something in the cache
cache.put(123, "Hello World");

// get something from the cache
String message = cache.get(123);
```

Cache Interface & Methods

(in IDE)

JCache: Entry Processors

(custom atomic operations for everyone!)

```
// acquire a cache
    Cache<String, Integer> cache =
        manager.getCache("my-cache",
String.class, Integer.class);

    // increment a cached value by 42,
returning the old value
    int value = cache.invoke("key", new
IncrementProcessor<>(), 42);
```

JCache: Entry Processors

(custom atomic operations for everyone!)

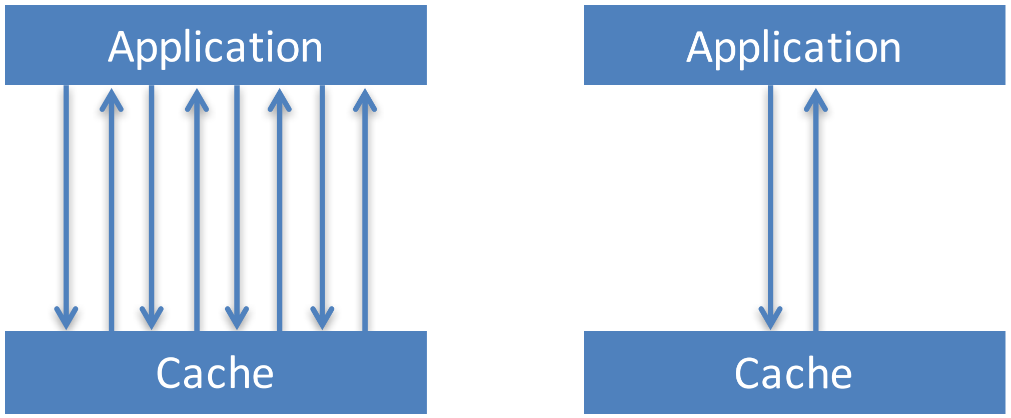
```
public class IncrementProcessor<K>
    implements EntryProcessor<K, Integer, Integer>, Serializable {

    @Override
      public Integer process(MutableEntry<K, Integer> entry, Object... arguments) {
        if (entry.exists()) {
            int amount = arguments.length == 0 ? 1 : (Integer)arguments[0];
            int current = entry.getValue();
            entry.setValue(count + amount);
            return current;
        } else {
            throw new IllegalStateException("no entry exists");
        }
}
```

JCache: Entry Processors

(custom atomic operations for everyone!)

- Eliminate Round-Trips! (in distributed systems)



- Enable development of a Lock-Free API! (simplifies applications)

*May need to be Serializable (in distributed systems)

JCache: Entry Processors

Java 8 Ready!

Which is better?

```
// using an entry processor?
int value = cache.invoke(“key”, new IncrementProcessor<>(), 42);

// using a lock based API?
cache.lock(“key”);
int current = cache.get(“key”);
cache.put(“key”, current + 42);
cache.unlock(“key”);
```

Annotations

- JSR107 introduces a standardized set of caching annotations, which do ***method level caching interception*** on annotated classes running in **dependency injection containers**.
- Caching annotations are becoming increasingly popular:
  - Ehcache Annotations for Spring
  - Spring 3’ s caching annotations.
- JSR107 Annotations will be added to:
  - Java EE 8 (planned?)
  - Spring 4.1 (released)

Annotation Operations

- The JSR107 annotations cover the most common cache operations:

- `@CacheResult`
- `@CachePut`
- `@CacheRemove`
- `@CacheRemoveAll`

Fully Annotated Class Example

```
@CacheDefaults(cacheName = "blogManager")
public class BlogManager {
    @CacheResult
    public Blog getBlogEntry(String title) {...}

    @CacheRemove
    public void removeBlogEntry(String title) {...}

    @CacheRemoveAll
    public void removeAllBlogs() {...}

    @CachePut
    public void createEntry(@CacheKey String title, @CacheValue Blog blog)
{...}

    @CacheResult
    public Blog getEntryCached(String randomArg, @CacheKey String title){...}
}
```

Specific Overrides

```
public class DomainDao {

  @CachePut(cacheName="domainCache")
  public void updateDomain(String domainId,
        @CacheKey int index,
     @CacheValue Domain domain) {
     ...
   }
 }
```

The Future?

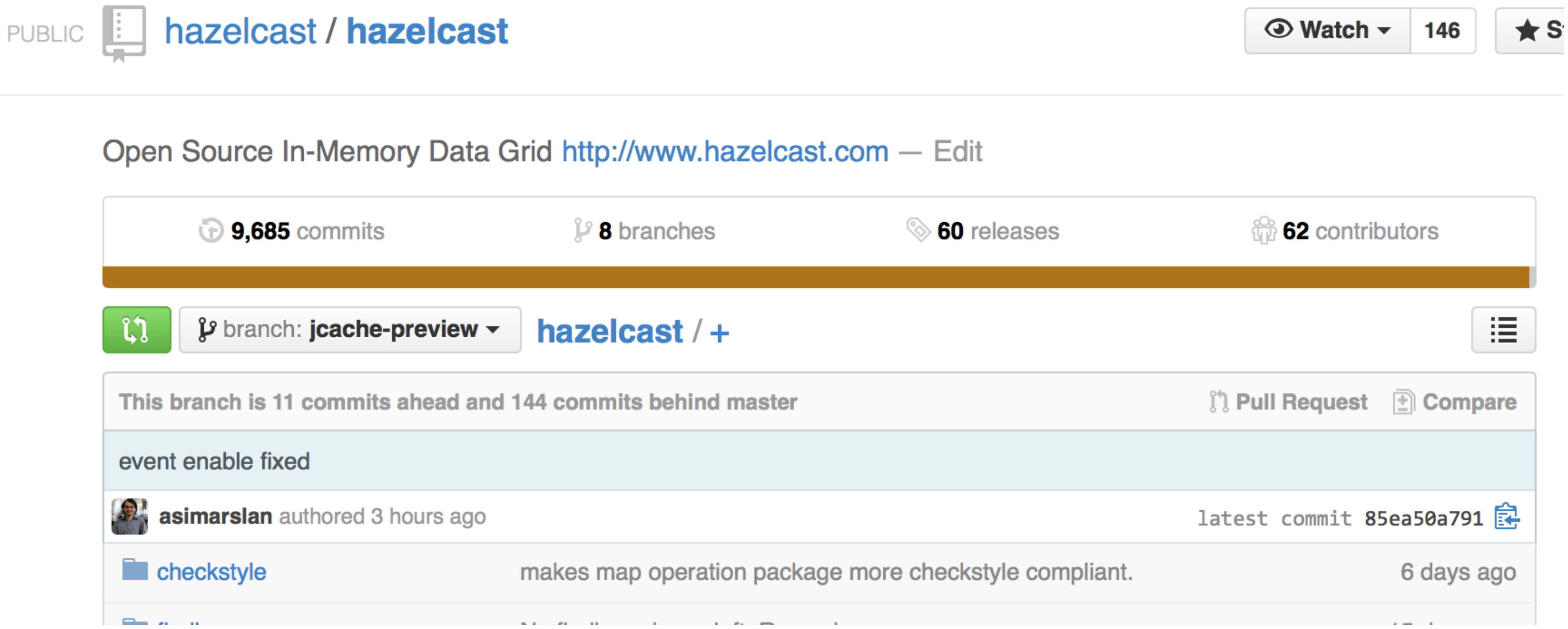
- **JCache 1.1** (2015)
  - Maintenance Release being worked on
- **JCache 2.0** (2016-)
  - Java 8 Language Features (Lambda & Streams)
  - Servlet 4.0 Integration / Session Caching?
  - Java EE 8 Alignment?
- **JCache 3.0** (2017?)
  - Java 10 Language Features?

Working with Hazelcast JCache

Hazelcast JCache Support

- Full implementation for:
  - Hazelcast.newHazelcastInstance()
  - HazelcastClient.newHazelcastClient()
- TCK Compliant
- JCache with Hi-Density Memory Store
- Docs: http://docs.hazelcast.org/docs/latest-dev/manual/html-single/hazelcast-documentation.html#jcache-overview

Check Out Hazelcast



PUBLIC hazelcast / **hazelcast**

Watch 146 ★ S

Open Source In-Memory Data Grid http://www.hazelcast.com — Edit

9,685 commits **8** branches **60** releases **62** contributors

branch: **jcache-preview** **hazelcast** / +

This branch is 11 commits ahead and 144 commits behind master Pull Request Compare

event enable fixed

asimarslan authored 3 hours ago latest commit 85ea50a791

checkstyle	makes map operation package more checkstyle compliant.	6 days ago

Or Download

- Download from hazelcast.org/download
- Maven:

```
<dependency>
    <groupId>com.hazelcast</groupId>
    <artifactId>hazelcast</artifactId>
    <version>3.5.4</version>
</dependency>
```

QUESTIONS?

- Greg Luck
  - (@gregrluck)
  - greg@hazelcast.com
- Terry Walters
  - (@tmwal7ers)
  - terry@hazelcast.com