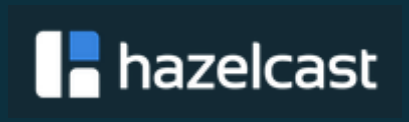
High-Performance Java EE with JCache and CDI

Steve Millidge - Founder Payara

Jaromir Hamala – Hazelcast



What are we going to talk about?

payara

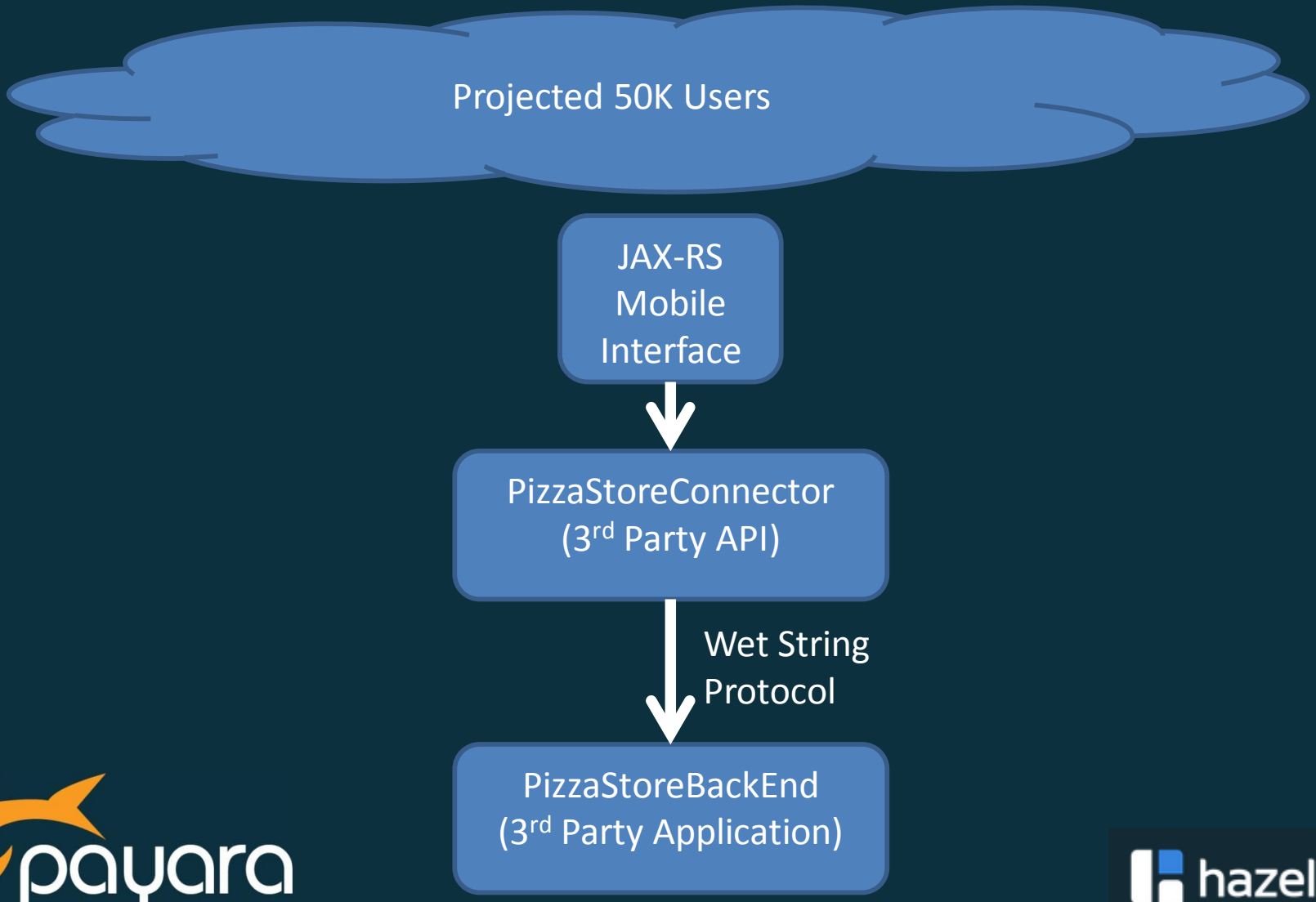
hazelcast

Imagine!

payara

hazelcast

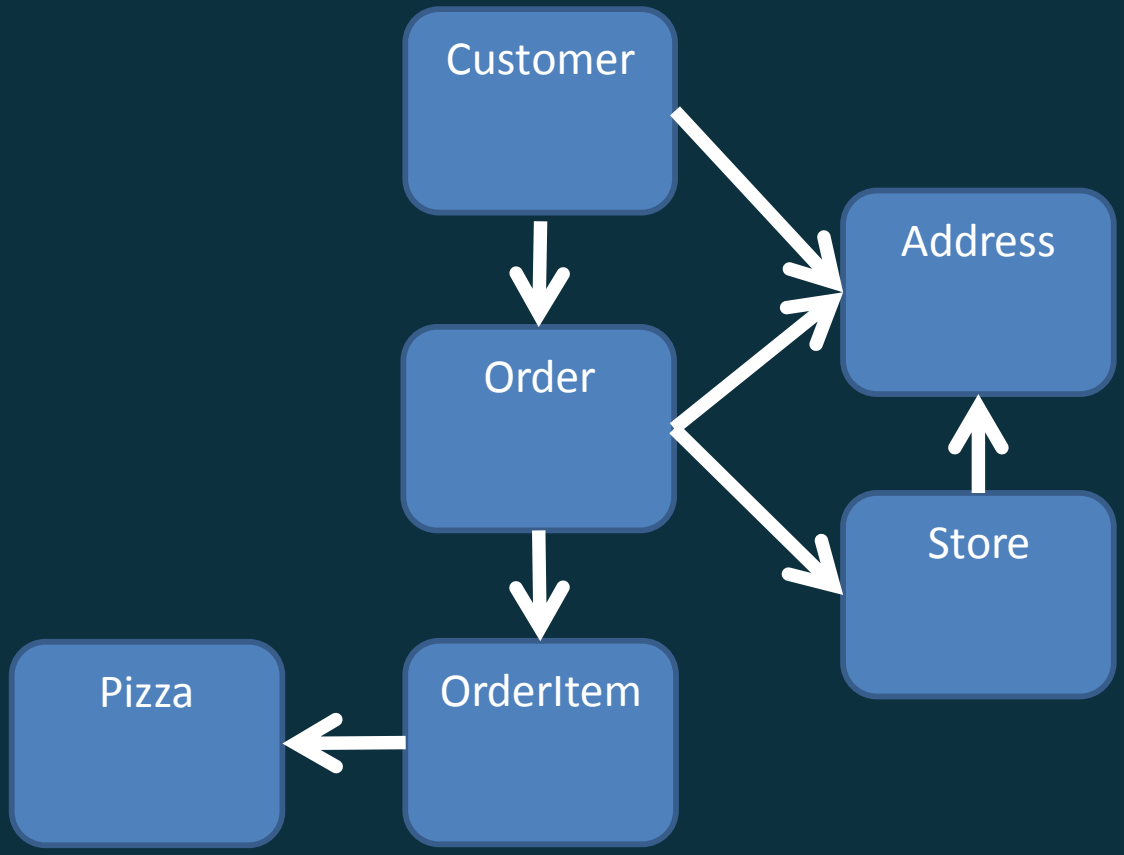
Pizza Architecture



payara

hazelcast

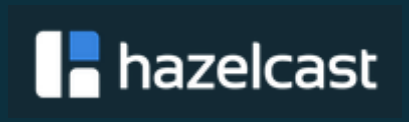
Pizza Store Model



payara

hazelcast

Pizza Connector Code



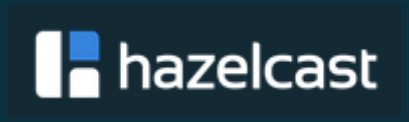
Pizza Connector Problems

- High Latency
- Multiple API round trips
  - Over Wet String
- Wrong Data Model
- Simply won't scale to user volumes
- Too costly to rewrite

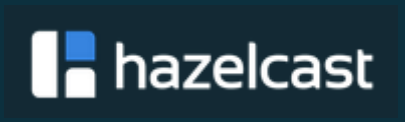
Recognize the Problem!

How do you speed it up?

JCACHE Java™ Temporary Caching API



Quick Cache Code



Beers All Round

payara

hazelcast

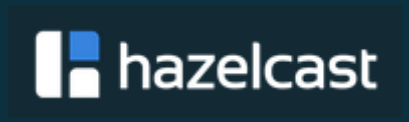
What Just Happened?

payara

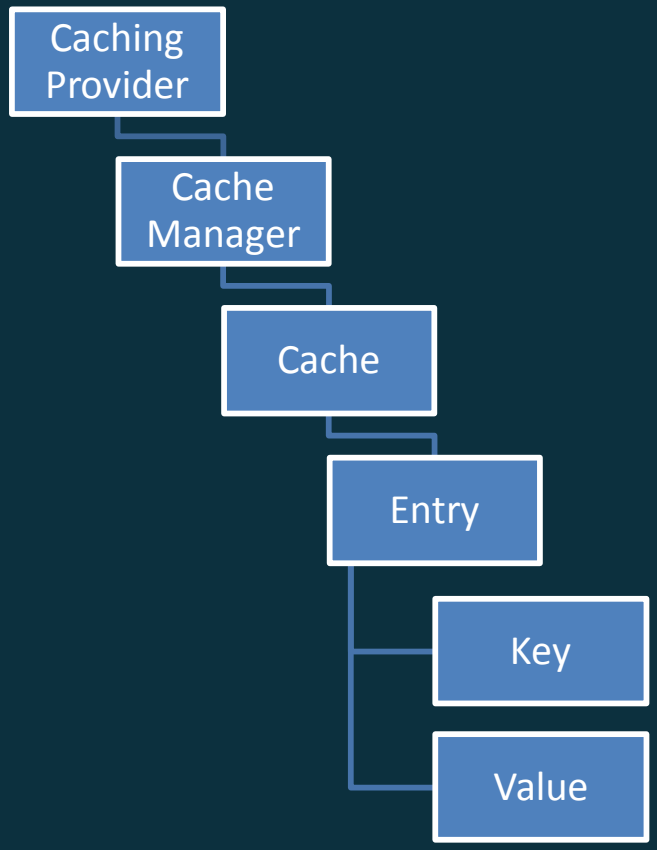
hazelcast

JCACHE

- Standard Java API for Caching
  - JSR107
- API and CDI binding
- Supported by Many Cache Providers
- Built in to Payara
  - Uses Hazelcast



Caching Concepts



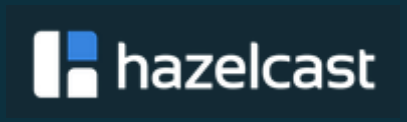
- **CachingProvider**
  - **Retrieves and closes Cache Managers**
- **CacheManager**
  - **Creates and destroys caches**
- **Cache**
  - **Contains Objects in Key Value Pairs**

payara

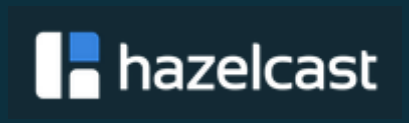
hazelcast

Magic Behind @CacheResult

- CDI Interceptor provided by Payara
- Obtains CacheManager from CachingProvider
- Creates Cache Name (Based on Method)
- Looks up Cache in CacheManager
- Takes Method Parameter Values and creates Cache Key
- Does Cache.get before invoking method
- If not in cache invokes method and does cache.put on the result.

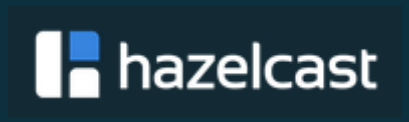


```
CachingProvider cp = Caching.getCachingProvider();

CacheManager cm = cp.getCacheManager();

MutableConfiguration<String, Stock> config
                = new MutableConfiguration<>();

config.setStoreByValue(true)
       .setTypes(String.class,String.class)
       .setManagementEnabled(true)
       .setStatisticsEnabled(true);

Cache<String, String> cache = cm.createCache("PizzaCache",
config);
String key = createKey();
Cache.put("Key",methodCall())
```



Further CDI

- @CacheResult
  - Caches the result of a method call
- @CachePut
  - Cache a specific method parameter
- @CacheRemove
  - Removes a cache entry based on parameters
- @CacheRemoveAll
  - Removes all entries in the cache



What is Hazelcast

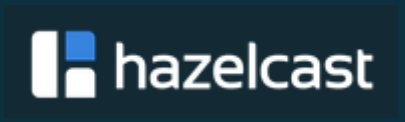
- Distributed implementation of j.u.Colections inside a single jar
- Distributed computing Swiss Army Knife
- **JCache Implementation**

And Much More!

What is Payara Micro

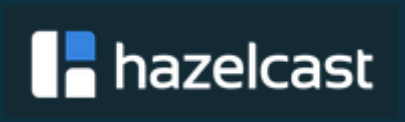
- Small Footprint GlassFish based Runtime
- Supports Java WAR deployments
- Embeds JCache Support
- Auto clusters using Hazelcast
- Fully Embeddable API

```
java -jar payara-micro.jar -deploy test.war
```



Payara Micro and Hazelcast

- Embeds Hazelcast for Session Persistence
- Enables Java EE developers to use JSR107 of raw Hazelcast api
- Provides CDI annotations for Hazelcast
- Supports Full Hazelcast API

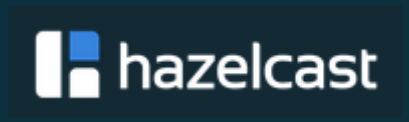


Payara CDI Extensions

```
@NamedCache(cacheName = "cache”)
@Inject Cache

@Inject CacheManager

@Inject HazelcastInstance
```

Clustering Demo

Beyond the Specification



Summary

- Using JCache on Payara is simple
- Hazelcast api is built in
- JCache and CDI are built in
- Build micro-services with distributed caching on Payara Embedded
- Hazelcast Provides Performance, Resilience and Scalability

More Info

- http://www.payara.co
  - Payara information and downloads
- https://github.com/payara/Payara
  - Payara Development
- http://www.hazelcast.com
    - GET Hazelcast
- https://github.com/smillidge/JCacheWebinar
    - Code from this talk
- https://github.com/payara/Payara-Examples/tree/master/JCache-Examples
    - Further JCache on Payara Examples