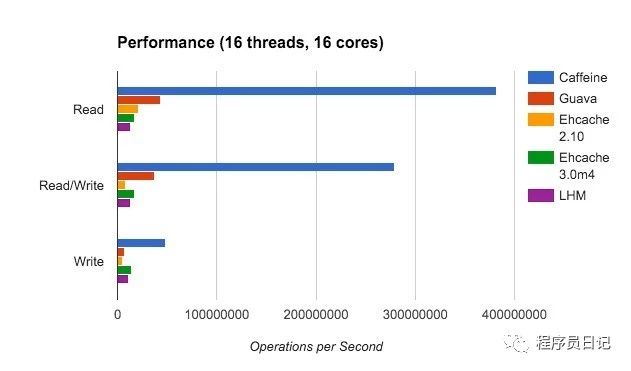
案例：<https://gitee.com/zx19890628/spring-boot-example/tree/master/lab_032_cache_caffeine>

# Caffeine

Spring Boot 2.0 引入了缓存组件 Caffeine 舍弃了大家熟知的 Google Guava 想必 Spring 大佬必有他的理由，各大缓存组件性能 PR 图如下



## 依赖

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-cache</artifactId>

</dependency>

<dependency>

<groupId>com.github.ben-manes.caffeine</groupId>

<artifactId>caffeine</artifactId>

</dependency>

## 配置类

@Data

@Slf4j

@EnableCaching

@ConfigurationProperties(prefix = "spring.cache") // 加载cache的配置

public class CaffeineConfiguration {

/\*\*

\* 加载 name\_spaces 的数据，保存多种类型缓存的信息

\*/

private Map<String, CacheSpec> nameSpaces;

@Bean

public CacheManager cacheManager(Ticker ticker) {

SimpleCacheManager manager = new SimpleCacheManager();

if (nameSpaces != null) {

List<CaffeineCache> caches = nameSpaces.entrySet().stream()

.map(entry -> this.buildCache(entry.getKey(), entry.getValue(), ticker))

.collect(Collectors.toList());

manager.setCaches(caches);

}

return manager;

}

private CaffeineCache buildCache(String name, CacheSpec cacheSpec, Ticker ticker) {

log.info("【缓存管理】加载缓存={} expireTime={} s, maxSize={}", name, cacheSpec.getExpireTime(), cacheSpec.getMaxSize());

// 用法与guava的localCache一样，但是性能要高很多

final Caffeine<Object, Object> caffeineBuilder = Caffeine.newBuilder()

.expireAfterWrite(cacheSpec.getExpireTime(), TimeUnit.SECONDS)

.maximumSize(cacheSpec.getMaxSize())

.removalListener(new RemovalListener<Object, Object>() {

@Override

public void onRemoval(@Nullable Object key,

@Nullable Object value,

@NonNull RemovalCause cause) {

log.info("【缓存管理】缓存过期， 过期原因={}，删除 key={},删除 value={}",cause, key.toString(), value.toString());

}

})

.ticker(ticker);

return new CaffeineCache(name, caffeineBuilder.build());

}

@Bean

public Ticker ticker() {

return Ticker.systemTicker();

}

}

@Data

public class CacheSpec {

/\*\* 过期时间,单位second \*/

private Integer expireTime = 100;

/\*\* 缓存最大保存数量 \*/

private Integer maxSize = 200;

}

## 访问接口

@RestController

public class DemoController {

@Autowired

private CacheManager cacheManager;

@RequestMapping("/get")

@Cacheable(value="one-cache", sync=true)

public String get(String name) {

System.out.println(name);

return "12222";

}

@RequestMapping("/go")

@Cacheable(value="two-cache", sync=true)

public String go(String name) {

System.out.println(name);

return "12222";

}

@RequestMapping("/all")

public String all() {

Collection<String> cacheNames = this.cacheManager.getCacheNames();

for (String cacheName : cacheNames) {

System.out.println(cacheName);

// 获得这个name下的全部的缓存

CaffeineCache cache = (CaffeineCache)this.cacheManager.getCache(cacheName);

@NonNull

ConcurrentMap<@NonNull Object, @NonNull Object> asMap = cache.getNativeCache().asMap();

System.out.println(asMap);

}

return "12222";

}

}