Distributed Caching Using the JCACHE API and ehcache, Including a Case Study on Wotif.com

Greg Luck

Maintainer, ehcache—http://ehcache.sf.net
Member, JSR 107 JCACHE Expert Group
Chief Architect, Wotif.com—http://wotif.com

TS-6175

Goal of This Talk

What You Will Gain

Learn how to:

- Use ehcache to make your apps fast
- Use General, Hibernate, and Web caches
- Configure caches as distributed caches for horizontal scaling
- Abstract the Cache Provider with Java™ Specification Request (JSR) 107 JCache
- Apply ehcache to real-world problems

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

Introduction

About ehcache

- Since 2003
- Probably the most widely used Java platform Cache
- Apache 2.0 License
- Integrated with Spring and Hibernate
- Java Development Kit (JDK™) 1.4 and higher (from ehcache 1.2.4)
- Web Caching, Persistence Caching, General Purpose and Cache Constructs
- Distributed Caching implementation in 2006
- JCACHE implementation released 2007

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

Make Your Application 10–1,000 Times Faster

How much faster will caching make an application?

- It depends on a multitude of factors being:
  - How many times a cached piece of data can and is reused by the application
  - The proportion of the response time that is alleviated by caching
  - In applications that are I/O bound, which is most business applications, most of the response time is getting data from a database; Therefore the speed up mostly depends on how much reuse a piece of data gets

Make Your Application 10–1,000 Times Faster

Cache Efficiency

- Factors which affect the efficiency of a cache are:
  - Required Liveness of Data
  - Proportion of total data cached
  - Read/Write ratio
  - Caching in a single VM vs. Caching Clusters

Make Your Application 10–1,000 Times Faster

How to calculate entire system speedup: Amdahl's Law

- Amdahl's law, after Gene Amdahl, is used to find the system speed up from a speed up in part of the system
  - 1/ ((1 - Proportion Sped Up) + Proportion Sped Up / Speed up)
- Things to Watch:
  - For a web application the "system" should include browser render time and network latency

Make Your Application 10–1,000 Times Faster

Speed up from a Web Page Cache

Uncached page time: 2 seconds

Cache retrieval time: 2ms

Proportion: 100%

The expected server-side system speedup is thus:

1 / ((1 - 1) + 1 / 1000)

= 1 / (0 + .001)

= 1,000 times system speedup

The to the browser “system” speedup is much less

Make Your Application 10–1,000 Times Faster

Speed up from a Database Level Cache

Uncached page time: 2 seconds

Database time: 1.5 seconds

Cache retrieval time: 2ms

Proportion: 75% (1.5/2)

The expected system speedup is thus:

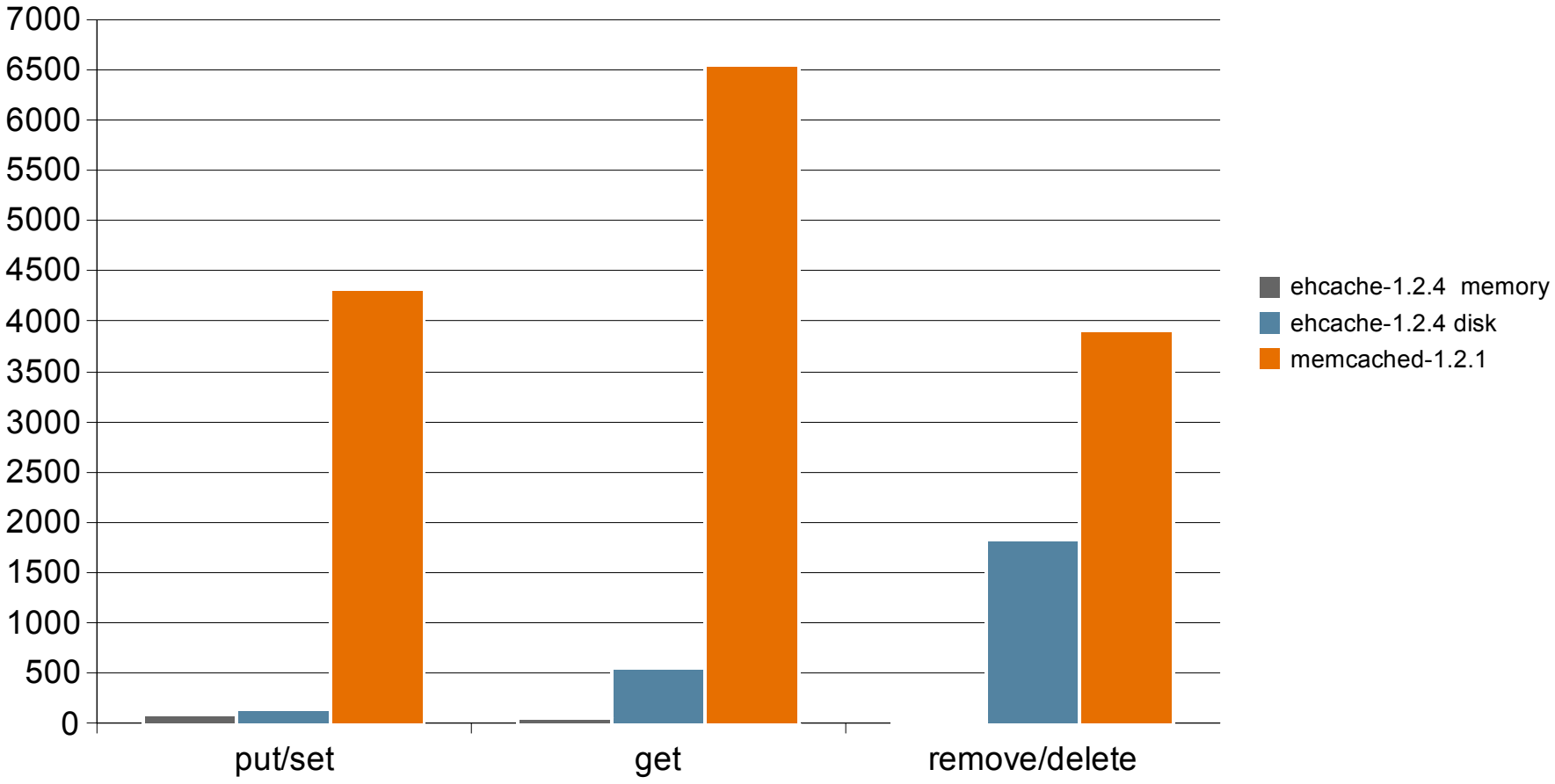
1 / ((1 - .75) + .75 / (1500/2)))

= 1 / (.25 + .75/750)

= 3.98 times system speedup

Make Your Application 10–1,000 Times Faster

Speed compared with Memcached (smaller is better)



Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

General Purpose Caching

Step by Step

1. Add ehcache Java Archive (JAR) to your classpath
2. Place configuration in ehcache.xml and add it to your classpath
3. Create a CacheManager
   ```
   CacheManager manager = CacheManager.create();
   ```
4. Reference a Cache
   ```
   ehcache sampleCache = manager.getCache();
   ```
5. Use it
   ```
   sampleCache.put(new Element("key", "value"));
   sampleCache.get("key");
   ```

General Purpose Caching

ehcache.xml Configuration

```
<ehcache>

    <diskStore path="java.io.tmpdir"/>

    ...

    <cache name="sampleCache1"
           maxElementsInMemory="10000"
           maxElementsOnDisk="1000"
           eternal="false"
           overflowToDisk="true"
           timeToIdleSeconds="300"
           timeToLiveSeconds="600"
           memoryStoreEvictionPolicy="LFU"
            />
</ehcache>
```

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

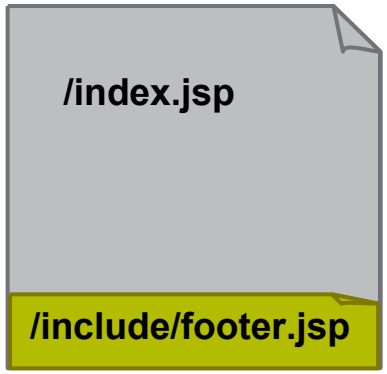
Web Caching

Step by Step

1. Use or Subclass SimpleCachingFilter or SimplePageFragmentCachingFilter
2. Configure filter in web.xml
3. Create a mapping in web.xml
4. Configure a cache entry matching the filter name

Web Caching

Example Scenario



Filter Configuration in web.xml

Full Page Cache

```
    <filter>
    <filter-name>SimplePageCachingFilter</filter-name>
    <filter-class>net.sf.ehcache.constructs.web.filter.
                SimplePageCachingFilter
    </filter-class>
</filter>

   <filter-mapping>
        <filter-name>SimplePageCachingFilter</filter-name>
        <url-pattern>/index.jsp</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>INCLUDE</dispatcher>
        <dispatcher>FORWARD</dispatcher>
    </filter-mapping>
```

Add Cache to ehcache Configuration

Full Page Cache

```
<ehcache>

    <diskStore path="java.io.tmpdir"/>

    ...

    <cache name="SimplePageCachingFilter"
           maxElementsInMemory="10000"
           maxElementsOnDisk="1000"
           eternal="false"
           overflowToDisk="true"
           timeToIdleSeconds="300"
           timeToLiveSeconds="600"
           memoryStoreEvictionPolicy="LFU"
            />
</ehcache>
```

Filter Configuration in web.xml

Page Fragment Cache

```
    <filter>
    <filter-name>SimplePageFragmentCache</filter-name>
    <filter-class>net.sf.ehcache.constructs.web.filter.
                        SimplePageFragmentCachingFilter
    </filter-class>
</filter>

    <!-- Page Fragment Cache -->
    <filter-mapping>
         <filter-name>SimplePageFragmentCachingFilter
    </filter-name>
         <url-pattern>/include/Footer.jsp</url-pattern>
   </filter-mapping>
```

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

JPA/Hibernate Caching

Overview

- Hibernate can either be used directly either or via the Hibernate Java Persistence API Provider in Hibernate 3.2
- ehcache is a CacheProvider for Hibernate 2.x and 3.x

Example

- Simple Country persistent object/Entity

JPA/Hibernate Caching

Step by Step

1. Configure Hibernate to use ehcache
2. Configure the Country object's cacheability; this can be done in a number of ways
3. Decide whether to use defaults, or to configure specific cache settings for the Country object cache in ehcache.xml

JPA/Hibernate Caching

Configure Hibernate to use ehcache

Add the following to hibernate.properties:

```
hibernate.cache.provider_class=net.sf.ehcache.hibernate.EhCacheProvider

<!-- optional configuration file parameter -->
net.sf.ehcache.configurationResourceName=/name_of_configuration_resource
```

JPA/Hibernate Caching

Native Hibernate Mapping File Configuration

```
<hibernate-mapping>

<class
    name="com.somecompany.someproject.domain.Country"
    table="ut_Countries"
    dynamic-update="false"
    dynamic-insert="false"
>
    <cache usage="read-write|nonstrict-read-write|read-only" />
</class>
...
</hibernate-mapping>


<cache
    usage="transactional|read-write|nonstrict-read-write|read-only"
    region="RegionName"
    include="all|non-lazy"
/>
```

JPA/Hibernate Caching

Entity Configuration With JPA and Hibernate Annotations

```
@Entity
...
@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public Class Country {

    private String name;

    ...

}
```

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

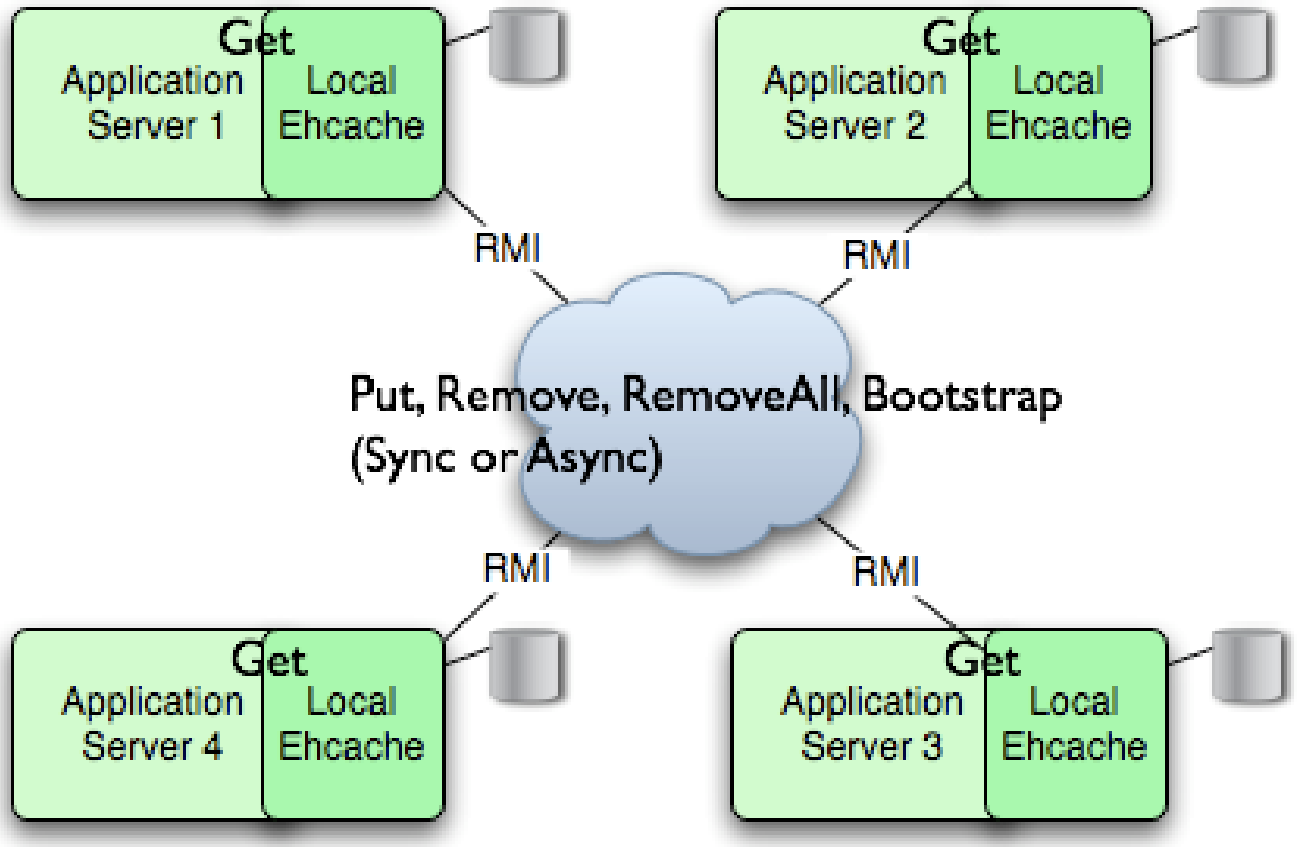
Distributed Caching for Clusters

Features

1. Pluggable distribution mechanism.
2. Comes with an RMI-based replicator. JGroups coming. Easy to add others.
3. Multiple “clusters”
4. Replication set per Cache. Options:
    1. Replicate adds
    2. Replicate removes
    3. Replicate update by copy or invalidate
    4. Async or Sync
5. Bootstrapping from existing Caches in a cluster

Distributed Caching for Clusters

Architecture



Distributed Caching for Clusters

RMI-Based Replication

1. Multicast Peer Discovery

```
<cacheManagerPeerProviderFactory  class=
"net.sf.ehcache.distribution.
RMICacheManagerPeerProviderFactory"
properties="peerDiscovery=automatic,
multicastGroupAddress=230.0.0.1,
multicastGroupPort=4446/>
```

2. RMI Listener

```
<cacheManagerPeerListenerFactory class=
"net.sf.ehcache.distribution.RMICacheManagerPeerLis
tenerFactory"properties="port=40001"/>
```

Distributed Caching for Clusters

Set Replication Per Cache

```
<cache ...>
   <cacheEventListenerFactory

        class="net.sf.ehcache.distribution.RMICacheReplicatorFactor
   y"
        properties="replicateAsynchronously=true,
        replicatePuts=true,
        replicateUpdates=true,
        replicateUpdatesViaCopy=true,
        replicateRemovals=true,
        asynchronousReplicationIntervalMillis=1000"/>
        ...
</cache>
```

Distributed Caching for Clusters

Set Bootstrapping Per Cache

```
<cache ...>
        <bootstrapCacheLoaderFactory class="
  net.sf.ehcache.distribution.RMIBootstrapCacheLoaderFactory"
   properties="bootstrapAsynchronously=true,
               maximumChunkSizeBytes=5000000"/>
</cache>
```

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

JSR 107—JCACHE: Java Temporary Caching API

Description and Status

- Formed 20/3/2001
- Abstracts user from the Cache Providers in much the same way as Java DataBase Connectivity (JDBC™) does for databases
- Many think it is moribund
- Too important not to be finalized
- Forthcoming ehcache-1.3 Implementation can be used with interfaces in net.sf.jsr107cache
- JCache expert group has reactivated and will be conducting meetings this week on the spec

JSR 107—JCACHE

Basic Usage

1. Add ehcache-1.3 and jsr107cache jar to your classpath
2. Place configuration in ehcache.xml and add it to your classpath
3. Specify a CacheFactory in META-INF/services/net.sf.jsr107cache.CacheFactory
   net.sf.ehcache.jcache.JCacheFactory
4. Create a CacheManager
   CacheManager manager = CacheManager.getInstance();

JSR 107—JCACHE

Basic Usage (Cont.)

5. Create a Cache

```
Map env = new HashMap();
env.put("name", "test4");
env.put("maxElementsInMemory", "1000");
env.put("overflowToDisk", "true");
env.put("eternal", "true");
env.put("timeToLiveSeconds", "0");
env.put("timeToIdleSeconds", "0");
cache = manager.getCacheFactory().createCache(env);
```

6. Reference a Cache

```
Cache sampleCache = manager.getCache();

or    manager.addCache(ehcache);
      Cache cache = new JCache(ehcache, null);
```

7. Use it

```
sampleCache.put(“key”, “value”));
sampleCache.get(“key”);
```

Agenda

Introduction

Make Your Application 10–1,000 Times Faster

General Purpose Caching

Web Caching

Java Persistence API (JPA)/Hibernate Caching

Distributed Caching for Clusters

Step-by-Step Coding Demo

Do It With Standards—JSR 107 JCACHE

Making It Real: A Wotif.com Case Study

Wotif.com Case Study

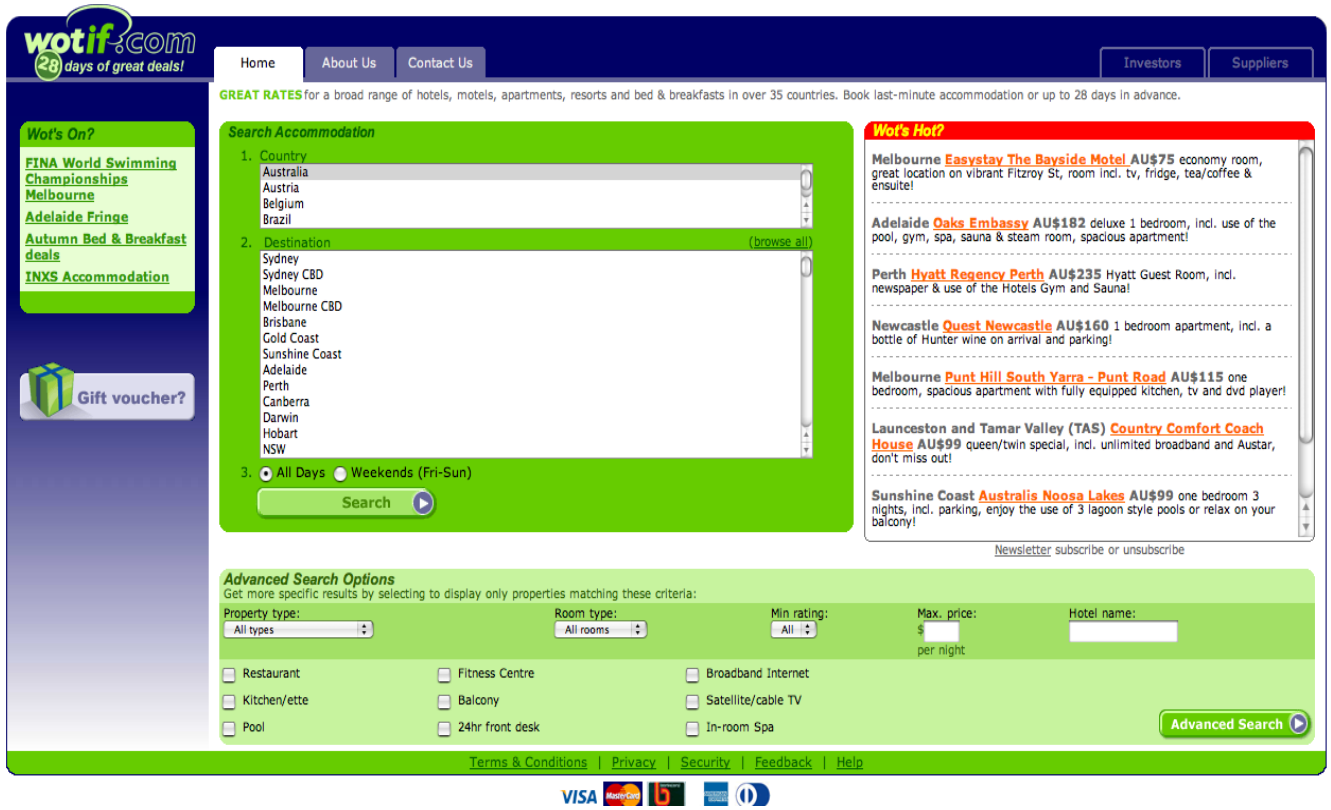
Key facts and figures

- Australia's No. 1 Hotel Booking Site
- Rapidly Growing elsewhere
- 1,000,000 users
- 10,000 concurrent users
- 3.6 million room nights sold per year
- 9,000 hotels
- Very Large Pages by design (some > 1.5MB)

Source: http://blogs.sun.com/stories/entry/wotif
http://www.networksasia.net/ena/article/articleDetail.jsp?id=393040
http://info.wotif.com/pdf/Wotif.com_Facts_Figures.pdf

Wotif.com Case Study

Screen shots—Home Page



Source: http://wotif.com

Wotif.com Case Study

Screen shots—Search Results

- + 15 more screenfuls for a total of 1.5MB
- Support many of these per second

Source: http://www.wotif.com/search/Simple?refine=simpleSearch&country=1&region=1&viewType=all

Wotif.com Case Study

Technology Platform

- Sun AMD64 Servers
- RHEL5
- 64-bit Java technology
- GlassFish™ Project
- Open Message Queue
- Oracle and MySQL
- Java Platform, Enterprise Edition (Java EE platform) 5 with Enterprise JavaBeans™ (EJB™) 3, JavaServer Pages™ (JSP™) 2
- Also lots of open source

Wotif.com Case Study

ehcache

- 150 caches
- Hibernate 2 and Hibernate 3 Persistent Object and Collection Caches
- Distributed Page Caches in a “Customer” cluster
- AJAX XML Response Caches
- Distributed Java Object Caches in a “Supplier” cluster
- Uses SelfPopulating and Blocking Cache
- Web Cache Full Page and Fragment Web Filters
- Disk Store and Persistent Disk Stores

Wotif.com Case Study

How we came to implement a Cache

- Started with Apache JCS. It had some serious threading bugs 3 years ago. Submitted a patch which was ignored by the maintainer.
- Mentioned on the Hibernate mailing list about JCS problems and linked the patch for people to apply.
- Gavin King suggested forking JCS. On the basis that Hibernate would use the forked version ehcache was born. It originally stood for Easy Hibernate Cache.
- ehcache's evolution mirrors Wotif.com's own need for caching features with the progression of MemoryStore -> DiskStore -> SelfPopulatingCache -> Web Response Caching -> Persistent Disk Store -> Distributed Caching.

DEMO

1. Web App on GlassFish Project without caching
2. Add caching to a single instance
3. Use distributed caching with n instances

Summary

- Most apps will benefit from caching
- You can calculate the speed up
- ehcache is easy to use directly
- You can also benefit from ehcache indirectly through frameworks that use it
- Distributed ehcache is a small configuration step
- Abstract yourself from Caching Providers using JCACHE

For More Information

- Project Website: http://ehcache.sf.net
- Downloadable Book: http://ehcache.sourceforge.net/EhcacheUserGuide.pdf
- My Email: gluck@gregluck.com

Q&A

Greg Luck
gluck@gregluck.com

Distributed Caching Using the JCACHE API and ehcache, Including a Case Study on Wotif.com

Greg Luck

Maintainer, ehcache—http://ehcache.sf.net
Member, JSR 107 JCACHE Expert Group
Chief Architect, Wotif.com—http://wotif.com

TS-6175