Hibernate Search
In your face(ting)

Emmanuel Bernard
JBoss by Red Hat

What you will learn

- What Hibernate Search can do for your project
- How to use it
- Zoom on Faceting
- Explore advanced features
  - Mapping, indexing, clustering, query

Emmanuel Bernard

- JBoss: data, hibernate, ceylon
- Hibernate [ORM|Search|Validator|OGM]
- Podcasts:
  - Les Cast Codeurs - http://lescastcodeurs.com
  - JBoss Community Asylum - http://asylum.jboss.org

- More at http://emmanuelbernard.com

Hibernate Search
What's that?

- Object Index Mapper
  - Full-text search engine library based on Lucene
    - API at the Object level

- Integrates with Hibernate ORM, OGM and Infinispan
- Cluster friendly

Why?

- Ease of use
- Lower barrier of entry for search
  - You focus on query
  - We bring the infrastructure and optimized patterns
- Scalable query computation
- Close to the application logic

Use case

- Google-like search engine for your entities
- Return the “best” matching entities first
- Recover from typos / faulty orthography
- Find from words with the same meaning
- Find by synonyms
- Find exact phrases
- Find similar entities
- Regular query

Faceting

What's faceting

Demo

Indexing

Conditional indexing

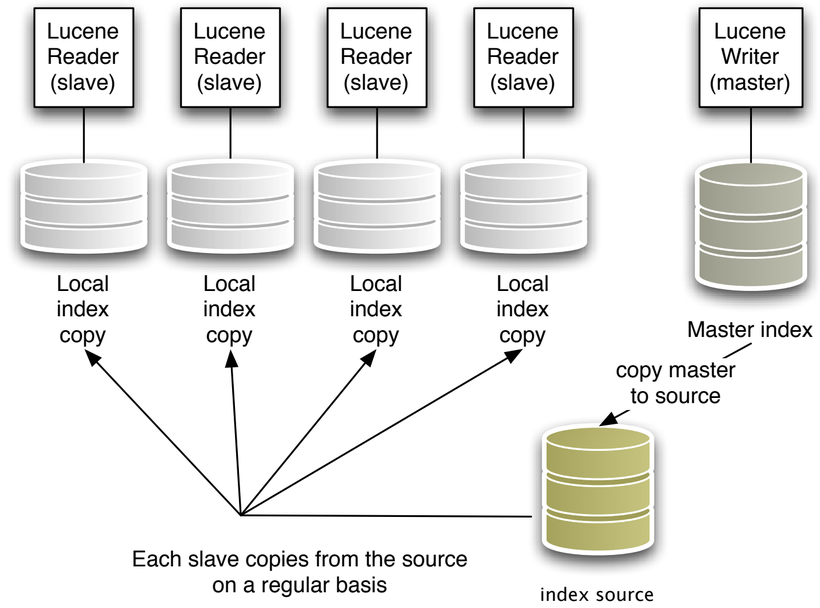
- Index only if entity is in a given state
  - Only approved data
  - Exclude old or obsolete data

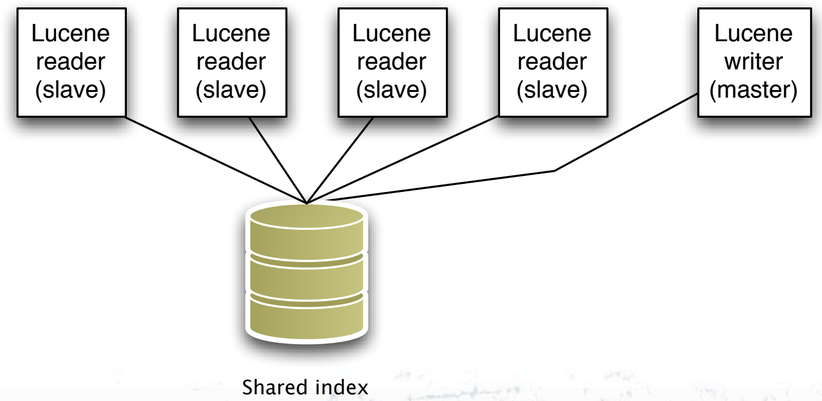
```
@Entity
@Indexed(interceptor=IndexWhenPublishedInterceptor.class)
public class Blog {
    public BlogStatus getStatus() { return status; }
    public void setStatus(BlogStatus status) {  this.status = status; }
    private BlogStatus status;
    ...
}
```

```java
public class IndexWhenPublishedInterceptor
    implements EntityIndexingInterceptor<Blog> {
    @Override
    public IndexingOverride onAdd(Blog entity) {
        if (entity.getStatus() == BlogStatus.PUBLISHED) {
            return IndexingOverride.APPLY_DEFAULT;
        }
        return IndexingOverride.SKIP;
    }

    @Override
    public IndexingOverride onUpdate(Blog entity) {
        if (entity.getStatus() == BlogStatus.PUBLISHED) {
            return IndexingOverride.UPDATE;
        }
        return IndexingOverride.REMOVE;
    }

    @Override
    public IndexingOverride onDelete(Blog entity) {
        return IndexingOverride.APPLY_DEFAULT;
    }

    @Override
    public IndexingOverride onCollectionUpdate(Blog entity) {
        return onUpdate(entity);
    }
}
```

Clustering

- Cluster the indexing event
  - None
  - JMS
  - JGroups
- Share the index
  - Incremental file copy
  - NFS
  - Infinispan





Near Real Time indexing

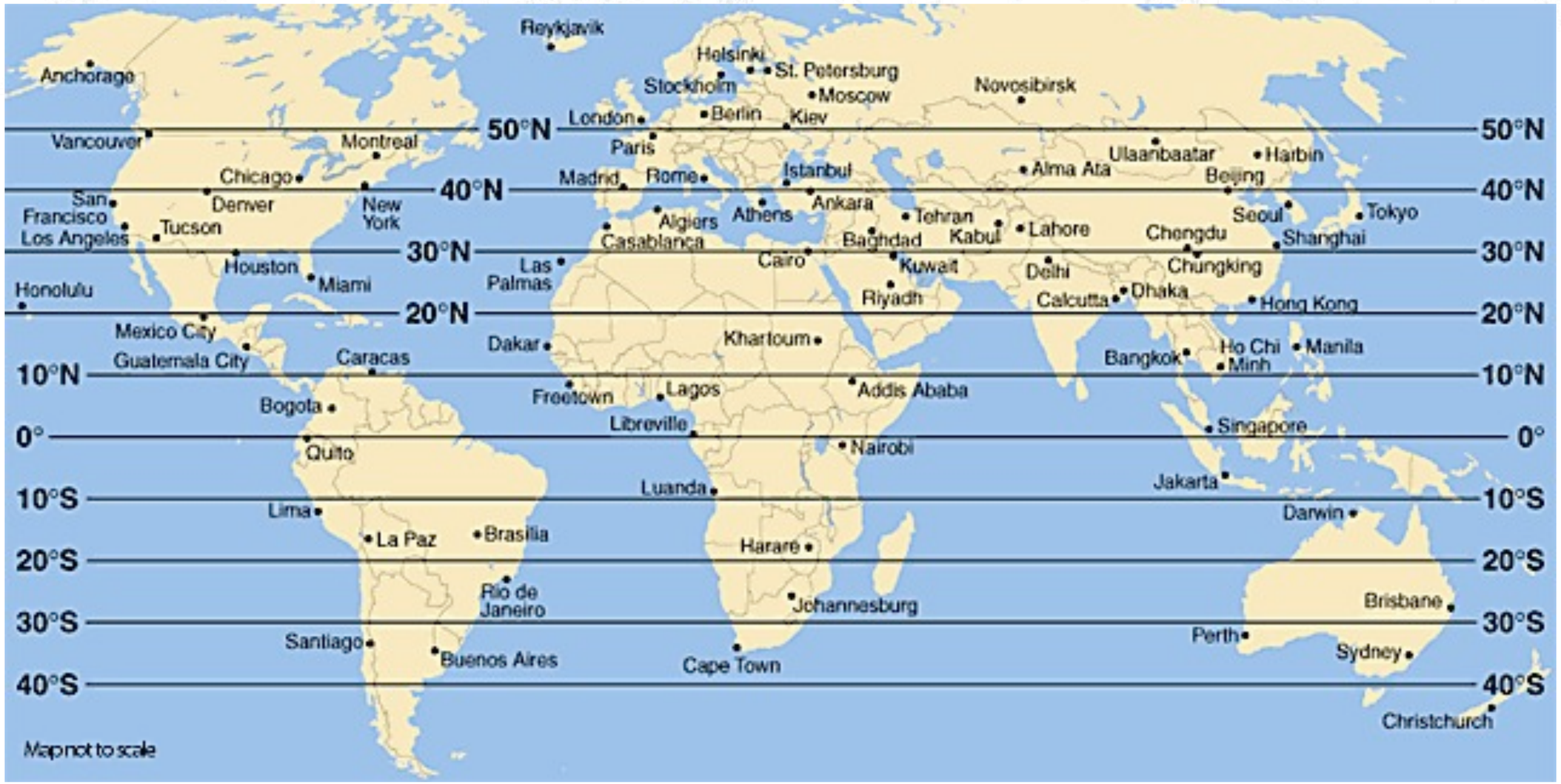
- Without NRT
  - Visible on commit changes
  - + Propagation time
- NRT
  - Read uncommitted changes in memory
  - Non clusterable
  - Screaming fast

Spatial query

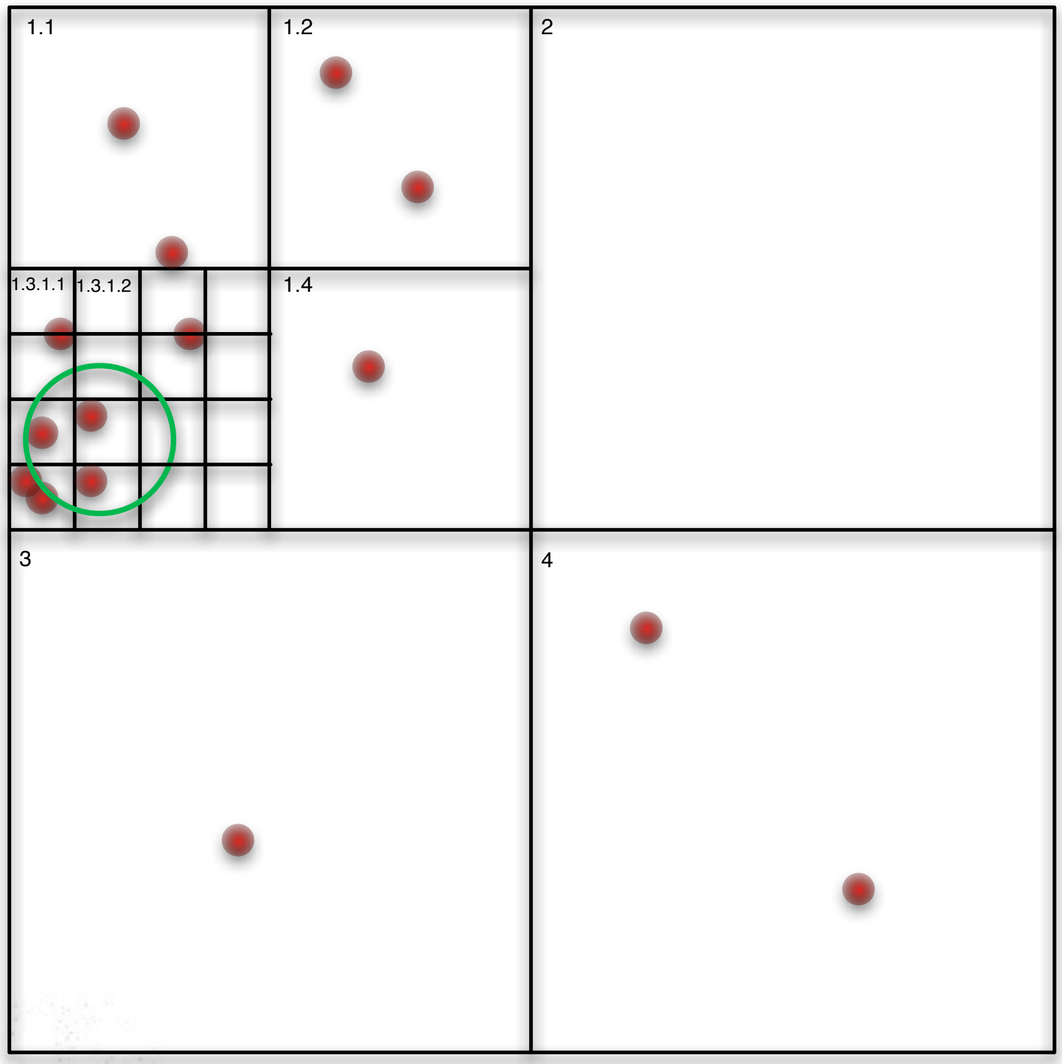
How I did nothing and delegated everything to the community

Indexing

- Boolean query on longitude and latitude
  - Good for small corpus (100k)
  - Smaller index
- Quad Tree
  - Good for big corpus
  - Works nice with heterogenous distribution


Reykjavik
Anchorage
Helsinki
St. Petersburg
Stockholm
Moscow
Novosibirsk
London
Berlin
Kiev
50°N
Vancouver
Montreal
Paris
Ulaanbaatar
Harbin
Chicago
Madrid
Rome
Istanbul
Alma Ata
Beijing
40°N
San Francisco
Denver
New York
Algiers
Athens
Ankara
Tehran
Seoul
Tokyo
Los Angeles
Tucson
Casablanca
Baghdad
Kabul
Lahore
Chengdu
Shanghai
30°N
Houston
Las Palmas
Cairo
Kuwait
Delhi
Chungking
Miami
Honolulu
Riyadh
Calcutta
Dhaka
Hong Kong
20°N
Mexico City
Dakar
Khartoum
Guatemala City
Caracas
Bangkok
Ho Chi Minh
Manila
10°N
Freetown
Lagos
Addis Ababa
Bogota
Libreville
Singapore
0°
Quito
Nairobi
Jakarta
10°S
Luanda
Lima
Darwin
La Paz
Brasilia
Harare
20°S
Rio de Janeiro
Johannesburg
Brisbane
30°S
Santiago
Perth
Buenos Aires
Cape Town
Sydney
40°S
Christchurch
Map not to scale

Quad tree



```
@Entity
@Indexed
public class POI {
    @Spatial(gridMode=true)
    @Embedded
    public Coordinates getLocation() {
        return new Coordinates() {
            @Override
            public double getLatitude() {
                return latitude;
            }

            @Override
            public double getLongitude() {
                return longitude;
            }
        };
    }
    ...
}
```

```
Query query = builder
        .spatial()
            .onCoordinates( "location" )
            .within( 51, Unit.KM )
                .ofCoordinates( coordinates )
            .createQuery();

List results = fullTextSession.createFullTextQuery( query, POI.class ).list();
```

Many more improvements

- Caching of index fields
- Lock factory
- Exception Handling
- Static and dynamic boosting
- Cross-version wire format
- Per index backend
- Sharding
- Filters
- Declarative analyzers
- Mass Indexer

Future

- More features
- Über scalable
- Master autodiscovery
- Part of a NoSQL search engine initiative

More info

- http://search.hibernate.org
- @hibernate

- Hibernate Search in Action
  - Old but damn good

- Part of JBoss Web Framework 2.0

Discuss!