Feature Cache Framework for GeoTools
A Google Summer of Code 2007 Project

Christophe Rousson, Université Laval
project mentored by Ian Turton
Outline

• Project motivations
• GeoTools Data Access Model
• Cache Framework Design
• Benchmarks
• Future work
Outline

• Project motivations
• GeoTools Data Access Model
• Cache Framework Design
• Benchmarks
• Future work
Project Motivations
Project Motivations

Client side
Project Motivations

Client side

Don’t download same data again and again
Project Motivations

- **Client side**
- **Save bandwidth**
- **Don’t download same data again and again**
Project Motivations

Client side

Don’t download same data again and again

Save bandwidth

Ability to disconnect from network
Outline

- Project motivations
- GeoTools Data Access Model
- Cache Framework Design
- Benchmarks
- Future work
GeoTools Data Module
Feature

- a common interface to encapsulate geographic vector data
- based on the OGC Reference Model
- SimpleFeature sub-interface (OGC SFS)
DataStore vs. FeatureSource
DataStore vs. FeatureSource

Low-level API

DataStore
DataStore vs. FeatureSource

Low-level API

DataStore

High-level API
DataStore vs. FeatureSource

- Low-level API: DataStore
- High-level API: FeatureSource (read-only)
DataStore vs. FeatureSource

Low-level API

DataStore

Transaction

High-level API

FeatureSource (read-only)

FeatureStore (read-write, transactional)
DataStore vs. FeatureSource

Low-level API

- DataStore
- Transaction
- LockingManager

High-level API

- FeatureSource (read-only)
- FeatureStore (read-write, transactional)
- FeatureLocking
WFS DataStore Example

HashMap params = new HashMap();
WFSDataStoreFactory factory = new WFSDataStoreFactory();

try {
    params.put(WFSDataStoreFactory.URL.key,
            new URL("http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap?
version=1.0.0&request=getcapabilities&service=wfs"));
    DataStore ds = (WFSDataStore) factory.createNewDataStore(params);
    FeatureSource source = ds.getFeatureSource("road");
    FeatureCollection fc = source.getFeatures(filter);
}

} catch (MalformedURLException e) {
    // ...
}

} catch (IOException e) {
    // ...
}
Supported Data Stores

- Shapefile
- Postgis, Oracle, DB2
- WFS, ArcSDE
- many other unsupported data plugins
Outline

• Project motivations
• GeoTools Data Access Model
• Cache Framework Design
• Benchmarks
• Future work
Cache basics
Cache basics
Cache basics
Cache basics

CLIENT

SOURCE

get
Cache basics

CLIENT → CACHE → STORAGE → SOURCE

get
Cache basics
Cache basics

CLIENT → CACHE
  get

CACHE → STORAGE
  peek
  data

SOURCE
Cache basics

CLIENT → CACHE
get

CACHE → STORAGE
peek
data

SOURCE

SOURCE → STORAGE
get
data

SOURCE → CLIENT
Cache basics

CLIENT \(\rightarrow\) CACHE \(\rightarrow\) STORAGE \(\rightarrow\) SOURCE

get \(\rightarrow\) peek \(\rightarrow\) data \(\leftarrow\) get \(\leftarrow\) data \(\rightarrow\) store
Cache basics

CLIENT → CACHE:
- get

CACHE → STORAGE:
- peek
- data
- get
- data
- store
Requirements

- optimized for how features are usually accessed
Filter
Rewrite

Filter = BBox AND Filter'
Rewrite

\[ \text{Filter} = \text{BBox \AND Filter'} \]

\( \text{Cache nodes} \)
Rewrite

Filter = BBox AND Filter'

Cache nodes

Blocks of Data
Storage
Storage

associate with
Divide space into regions

associate with

Storage
Divide space into regions

Storage

associate with

Group and store features by regions
Divide space into regions

Node ID

Storage

Group and store features by regions
Divide space into regions

Group and store features by regions
Divide space into regions

Known & unknown regions
Valid & invalid nodes

Group and store features by regions
Divide space into regions

Known & unknown regions
Valid & invalid nodes

Group and store features by regions

Storage

Node ID
How to divide space into regions?
How to divide space into regions?

regular space division
grid
How to divide space into regions?

regular space division
grid

regular space division
quadtree
How to divide space into regions?

Regular space division:
- Grid
- Quadtree

Irregular space division:
- R-tree
How to divide space into regions?

- Regular space division: grid
- Regular space division: quadtree
- Irregular space division: R-tree

The most efficient method is the irregular space division using an R-tree.
How to divide space into regions?

- Regular space division: grid
- Regular space division: quadtree
- Irregular space division: R-tree

The grid is most simple, and the R-tree is most efficient.
Feature Cache Framework
Feature Cache Framework

FeatureCache -> Storage
Feature Cache Framework

FeatureCache

Tracker

Storage
Feature Cache Framework

FeatureCache

Tracker

EvictionPolicy

Storage
Feature Cache Framework

- FeatureCache
- Tracker
- EvictionPolicy
- Storage
Feature Cache Framework

- FeatureCache Class Diagram (Simple) 2007/09/23

... <<interface>>Storage
... <<interface>>Tracker

- FeatureSource
- FeatureCache
- FeatureListener
- SpatialIndex
- Storage
- Tracker
- EvictableTree
- EvictionPolicy
- LRUEvictionPolicy

AbstractFeatureCache
Feature Cache Framework
Feature Cache Framework

FeatureCache Class Diagram (Simple) 2007/09/23

- FeatureSource
- FeatureCache
- FeatureListener
- Tracker
- SpatialIndex
- Storage
- AbstractFeatureCache
- EvictionPolicy
- LRUEvictionPolicy
- EvictableTree
Feature Cache Framework
Using spatial indexation

- spatial index framework proposed by Marios Hadjieleftheriou
- http://research.att.com/~marioh/spatialindex
Example code

```java
int feature_type_index = 0;
int number_of_nodes = 500;
int cache_feature_capacity = 1000;
DataStore ds = ... ; // get any instance of a datastore
FeatureCache cache = new GridFeatureCache(
    ds.getFeatureSource(
        ds.getTypeNames([feature_type_index]),
        number_of_nodes,
        cache_feature_capacity,
        BufferedDiskStorage.createInstance()));
// cache may be used anywhere you would normally use FeatureSource
```
FeatureCache Class Diagram (Simple) 2007/09/23

- <<interface>> FeatureSource
- <<interface>> FeatureCache
- <<interface>> FeatureListener
- <<interface>> SpatialIndex
- <<interface>> Tracker
- <<interface>> EvictableTree
- <<interface>> EvictionPolicy
- LRUEvictionPolicy
- AbstractFeatureCache
- Storage
Extension points

Inner mechanisms
Extension points

Type of index

Inner mechanisms
Extension points

Type of index

1. Inner mechanisms

2. Storage

3. Backend

FeatureSource <<interface>> FeatureCache

FeatureCache <<interface>> FeatureListener

FeatureSource <<interface>> FeatureSource

FeatureListener <<interface>> FeatureListener

Storage <<interface>> Storage

Tracker <<interface>> Tracker

SpatialIndex <<interface>> SpatialIndex

EvictableTree <<interface>> EvictableTree

EvictionPolicy <<interface>> EvictionPolicy

LRUEvictionPolicy <<interface>> LRUEvictionPolicy

AbstractFeatureCache

FeatureSource

FeatureCache

FeatureListener

Tracker

SpatialIndex

Storage

EvictableTree

EvictionPolicy

LRUEvictionPolicy

Inner mechanisms

Storage/Backend

FeatureCache Class Diagram (Simple) 2007/09/23
Extension points

Type of index

1. Inner mechanisms
   - AbstractFeatureCache
   - FeatureSource
   - FeatureCache
   - FeatureListener

2. Storage/Backend
   - SpatialIndex
   - Storage
   - EvictableTree
   - Tracker

3. Storage Backend

4. Eviction policy
   - EvictionPolicy
   - LRUEvictionPolicy
Project Current State
Project Current State

✓ Grid Spatial Index
Project Current State

✓ Grid Spatial Index
✓ Memory and Disk Storage
Project Current State

✓ Grid Spatial Index
✓ Memory and Disk Storage
✓ Thread safe
Project Current State

✓ Grid Spatial Index
✓ Memory and Disk Storage
✓ Thread safe
✓ LRU eviction policy
Project Current State

✓ Grid Spatial Index
✓ Memory and Disk Storage
✓ Thread safe
✓ LRU eviction policy
✓ Eviction based on a maximum number of features
Project Current State

✓ Grid Spatial Index
✓ Memory and Disk Storage
✓ Thread safe
✓ LRU eviction policy
✓ Eviction based on a maximum number of features
✓ Reusable framework
DEMO
Outline

- Project motivations
- GeoTools Data Access Model
- Cache Framework Design
- Benchmarks
- Future work
Test Case

- 5000 random features in the unit square, mean size of 0.05
- 300 queries
- Query window size of 0.05, 0.02, 0.01
- Queries in the neighborhood of previous one
Relative execution time difference compared to MemoryDataStore (%)
Relative execution time difference compared to MemoryDataStore (%)
Outline

• Project motivations
• GeoTools Data Access Model
• Cache Framework Design
• Benchmarks
• Future work
Current limitations
Current limitations

• support only for SimpleFeature
Current limitations

- support only for SimpleFeature
- per feature type cache
Current limitations

- support only for SimpleFeature
- per feature type cache
- read-only implementation
TODO LIST

• write-thru
TODO LIST

- write-thru
- disconnectable
TODO LIST

- write-thru
- disconnectable
- warm start from storage
• write-thru
• disconnectable
• warm start from storage
• support for other types of index: quadtree, r-tree
Wish list
Wish list

• cache listens to update feeds
Wish list

- cache listens to update feeds
- cache downloads pre-built index from third-party service (r-tree case)
Questions ?

For more information, please contact:
christophe.rousso@gmail.com
Relative execution time difference compared to MemoryDataStore (%)
Relative execution time difference compared to MemoryDataStore (%)

-700%  -500%  -300%  -100%  100%  250 nodes  500 nodes

50 nodes  100 nodes  250 nodes  500 nodes

Worst case  Optimal case  Mean

-684%  -230%  -340%  -255%  95%  84%  96%  91%  97%  80%  96%  77%