What is a Quadtree & how is it used in location-based services?

Quadtrees

Quadtrees are a knowledge structure that encodes a two-dimensional space into adaptable cells. Similar to binary trees, quadtrees are a tree structure where every non-leaf node has four children. In the Quadtree data structure, every internal node has four children. They are usually a bi-dimensional analogue of octrees that are used for the division of a two-dimensional space by repetitively subdividing it into four quadrants.

Within the location, these nodes show the four quadrants:

- NorthWest
- NorthEast
- SouthWest
- SouthEast

During the technique of quadtrees, nodes are recursively divided into pieces, with successive subdivisions leading to smaller and smaller cells.

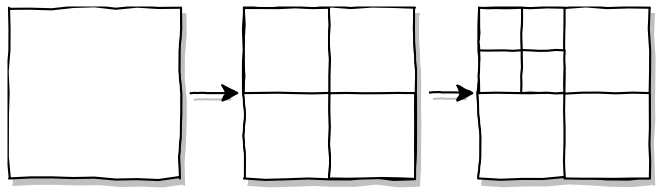
When are Quadtrees used for?

Quadtrees are used for scaling an internet service to handle thousands of requests every second. This requires an excellent caching strategy. When geolocation is the main parameter of those requests, conventional caching techniques and procedures fall through. Quadtrees are used to understand high cache hit ratios, while also keeping the responses relevant, even in intense areas.

Dynamic grids

While **10km** cells are overlarge to use in urban areas, there are many less dense areas outside of cities where two locations a couple of kilometres apart would have an equivalent list of **20** nearby Xone businesses. Ideally, we'd be able to use large, imprecise cells in sparse areas and smaller, more granular cells in dense areas.

Often a *quadtree is represented as a grid*, with each square representing a node within the tree. The subsequent image visualizes the method of subdividing nodes during a quadtree, starting with a uni-root node and finishing with a tree of depth:

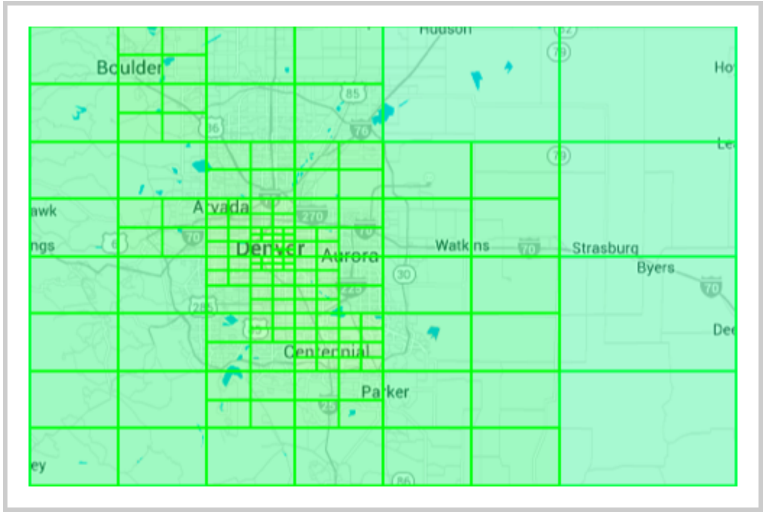


Quadtree as a Grid

We use quadtrees to keep a balance among the precision and validity of results. But let's say we wish to create a tree in which each leaf node is exact enough to search for a specific location within the node in the same group.

We can do this by developing a tree that begins with one node that is shown to the entire world and subsequently divides until every node satisfies this eligibility. The best case is when a tree that is precise has few nodes. This can attenuate the number of entries in the cache.

The following image shows the finished quadtree near Denver, CO to represent how quadtree depth changes with population intensity:



Colorado, USA

Auto-Update

Every night we recompute the quadtree to support our currently subscribed businesses. We use an easy heuristic to satisfy the criteria of consistent search results. If the number of search results entered on a node exceeds a predefined threshold, then we divide it further.

```java

//

public static void buildQuadtree(Noc
    if (searchBusinesses(node).size(
        node.subdivide();
        for (Node child : node.getCh
            buildQuadtree(child);
        }
    }
}

```

Quadtrees in action

When our server receives an invitation for businesses in a location, we first find the quadtree node for that location:

```java
public static void getNearbyBusinesses(f
```

Then, we use that node to create the cache key, and if necessary, to look for businesses:

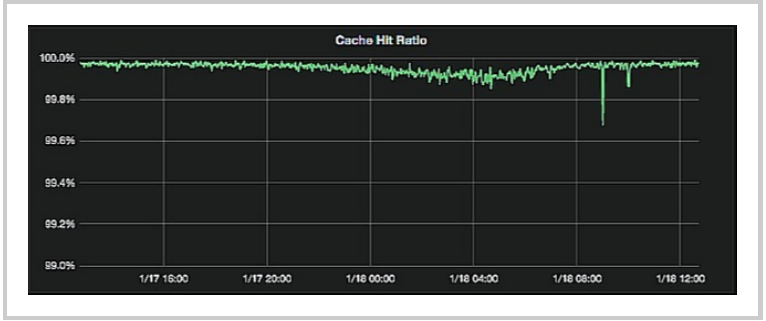
```

// Demo code to extract nearby Busir

public static void getNearbyBusiness
    QuadtreeNode node = LookupQuadTr
    String cacheKey = "quadtree_" +
    List<Result> results = Cache.get
    if (results == null) {
        results = searchBusinesses(r
        Cache.set(cacheKey, results,
    }
    render(results);
}

```

Extract Nearby Request

The search method that is applied in cache miss is identical to the search that is performed when creating the cache tree. The search is also centered on each node with a radius that is proportional to the dimensions of the node.

Conclusion

Quadtrees allow us to mix the advantages of caching the geo-locations with both high and low precisions. Against an 18 percentage cache hit ratio, we gained truncating locations up to 3 decimal places by using quadtrees. On peek time, it resulted in **cache hit ratio** of more than ***99.9%***.



Cache hit ratio graph

Relevant Answers

v to search in a row-wise and column-
e sorted matrix

What is Breadth First Search?

What is &nbsp in HTML?

Explore Courses

COURSE

Grokking the Coding Interview Patterns in C#

The ultimate guide to coding interviews in C#, developed by FAANG engineers. Master essential patterns, tackle tough questions, and prep faster for interviews at top tech companies.

85 h Intermediate

COURSE

Grokking the Coding Interview Patterns in JavaScript

The ultimate guide to coding interviews in JavaScript. Developed by FAANG engineers. Learn essential patterns, practice with real-world questions, and get interview-ready in just a few hours.

85 h Intermediate

COURSE

**Grokking the Coding Interv
Patterns in Go**

Grokking the Coding Interview
Patterns in Go helps you prep fa
with strategies developed by FA
engineers. Learn essential patte
to confidently tackle interview
questions from top companies.

85 h Interme