S2 Geo Library

badoo

badoo tech

badoo_tech

Hello, my name is

Marko Kevac

Software Engineer in Platform team at Badoo.

I love low level stuff, learning new things, watching Coub weekly, yogurt and of course Go language.

marko@kevac.org
blog.kevac.org

Mobile phones are ubiquitous

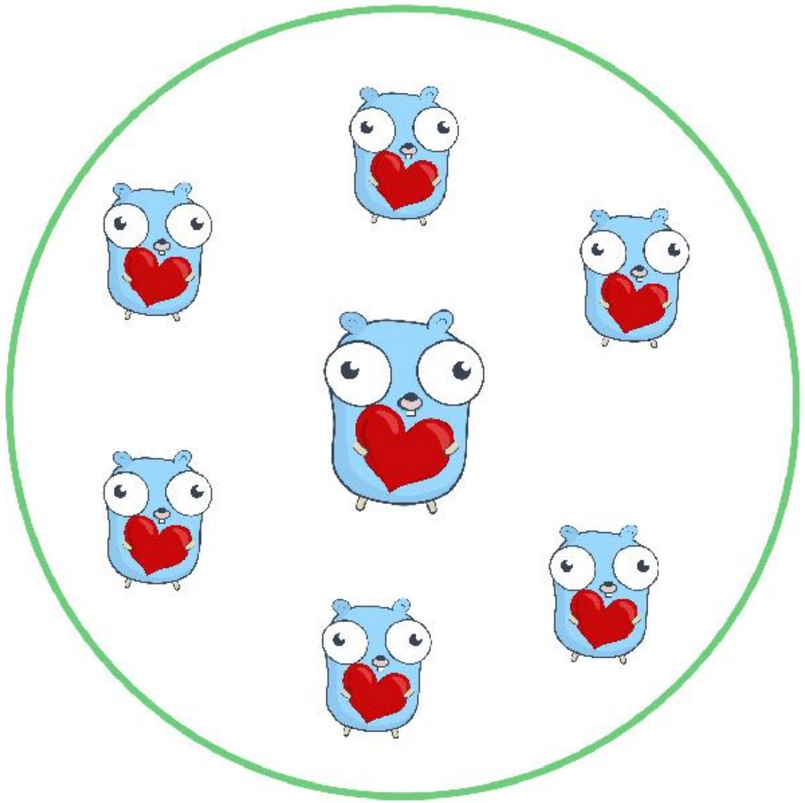
- 5B people have phones
- 60% of them have internet access

badoo_tech

badoo
379 450 646
People on Badoo worldwide
Realtime matches
March 5, 16:46

Meetmaker/ Geousers

People expect to see people that are near them, not 1000 km away.



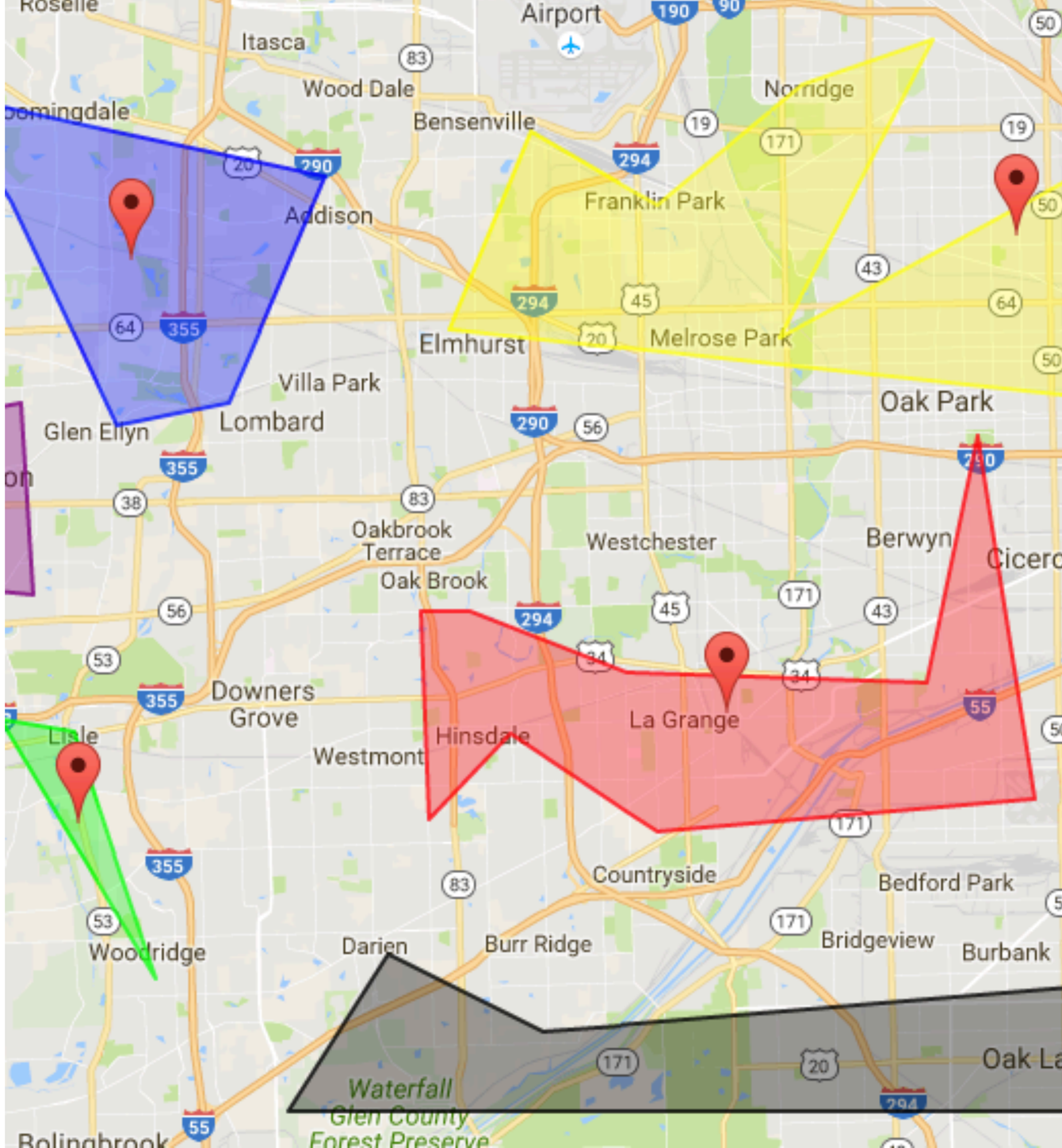
Geo in Badoo

Geoborder

Geofencing with Z-Index and nearest polygon search.

Massive database compiled from open sources and with help from our moderators.

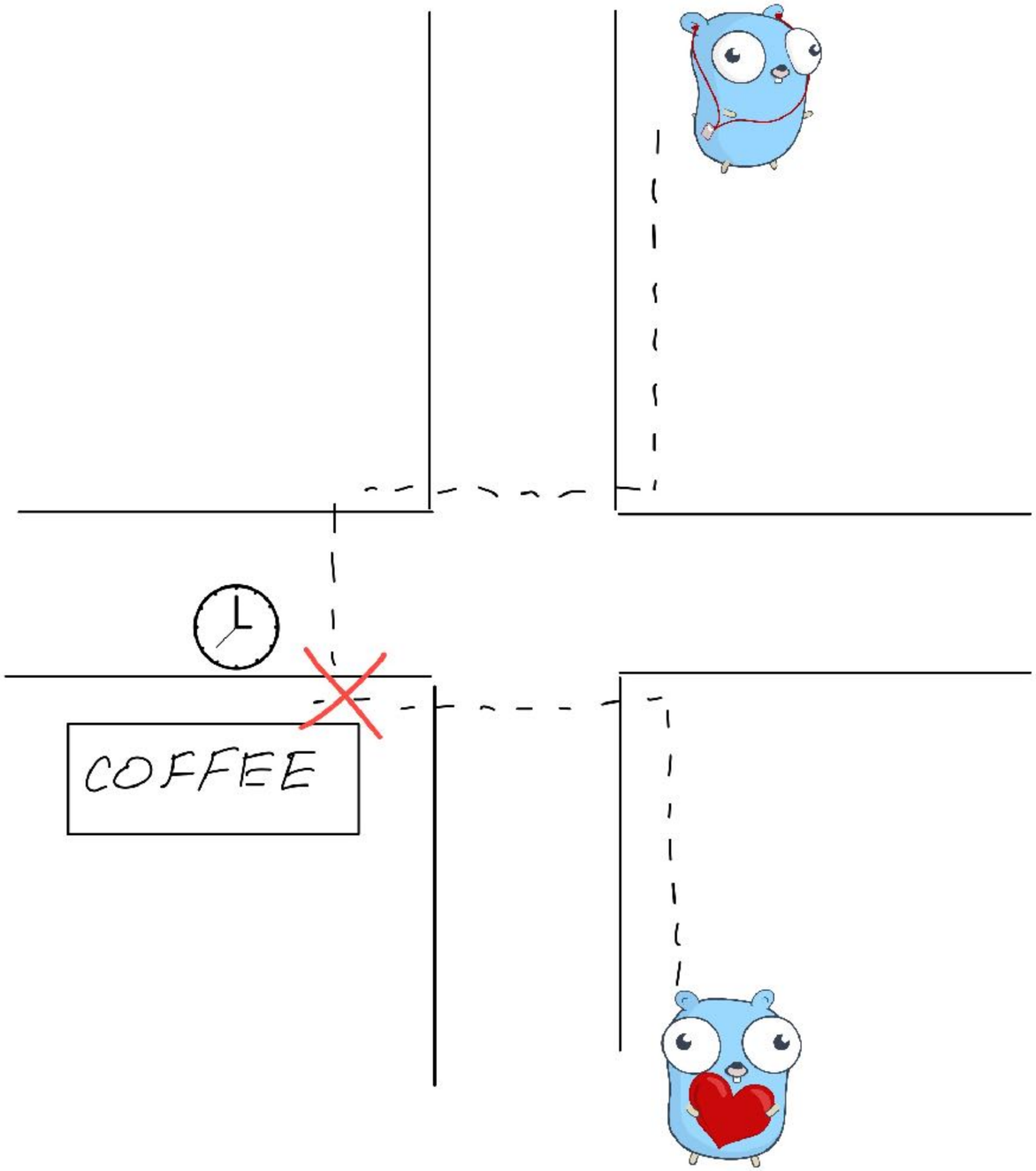
Countries, States, Cities, Neighbourhoods, Hospitals, Airports, Universities, etc.



Bumpd

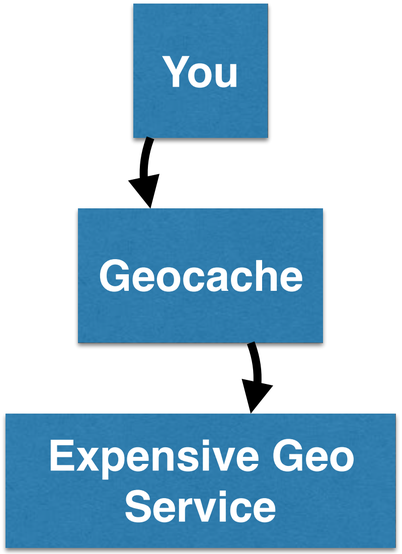
It is fun to learn about people that you unknowingly meet each day.

And the fact that you go to same places can be a great conversation opener.



Geocache

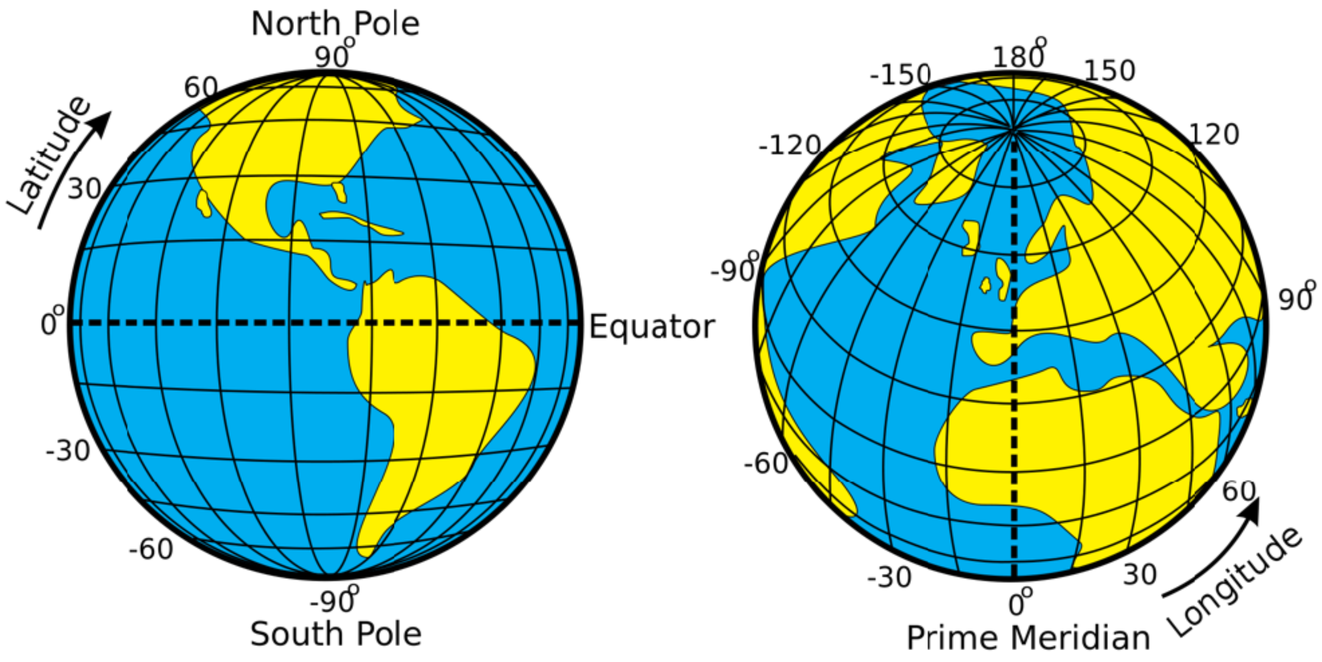
Sometimes it is too slow to get external geo data and you want to cache it.



What is so special?

Two dimensional

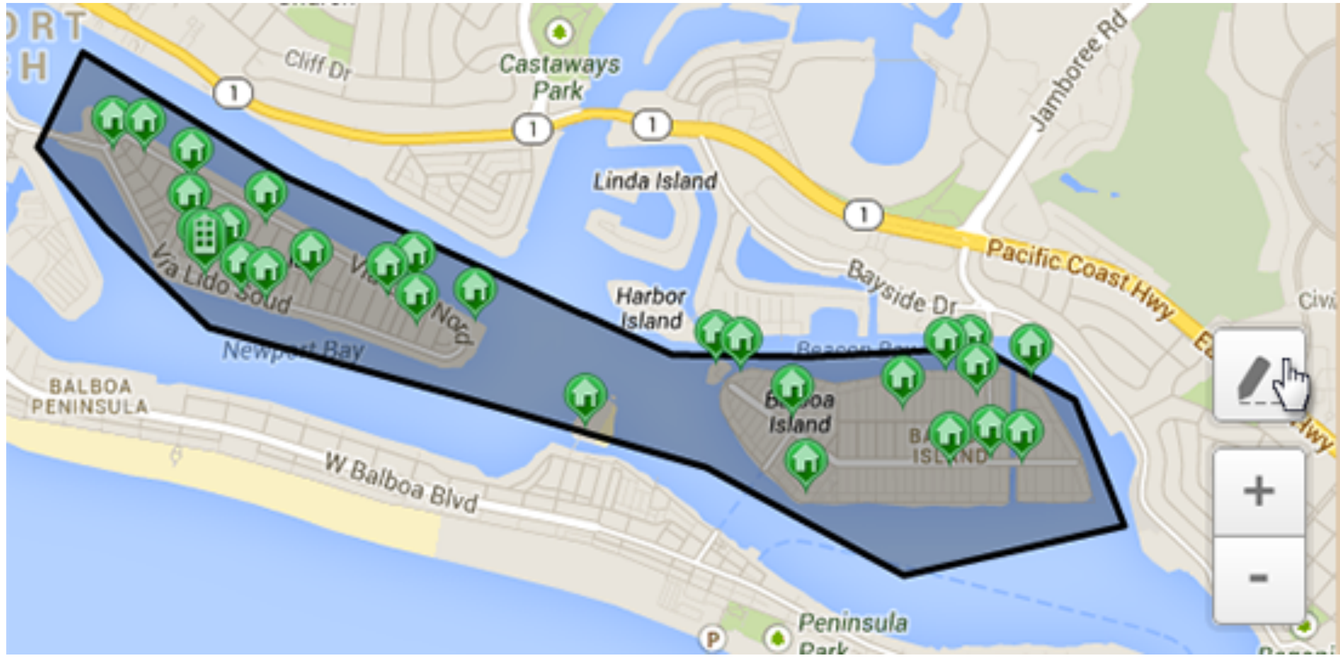
- Coordinates consist of latitude and longitude.
- Search usually involve both of them.
- Usual DB indexes don’t support search by both at the same time.



What is so special?

Search vocabulary is bigger

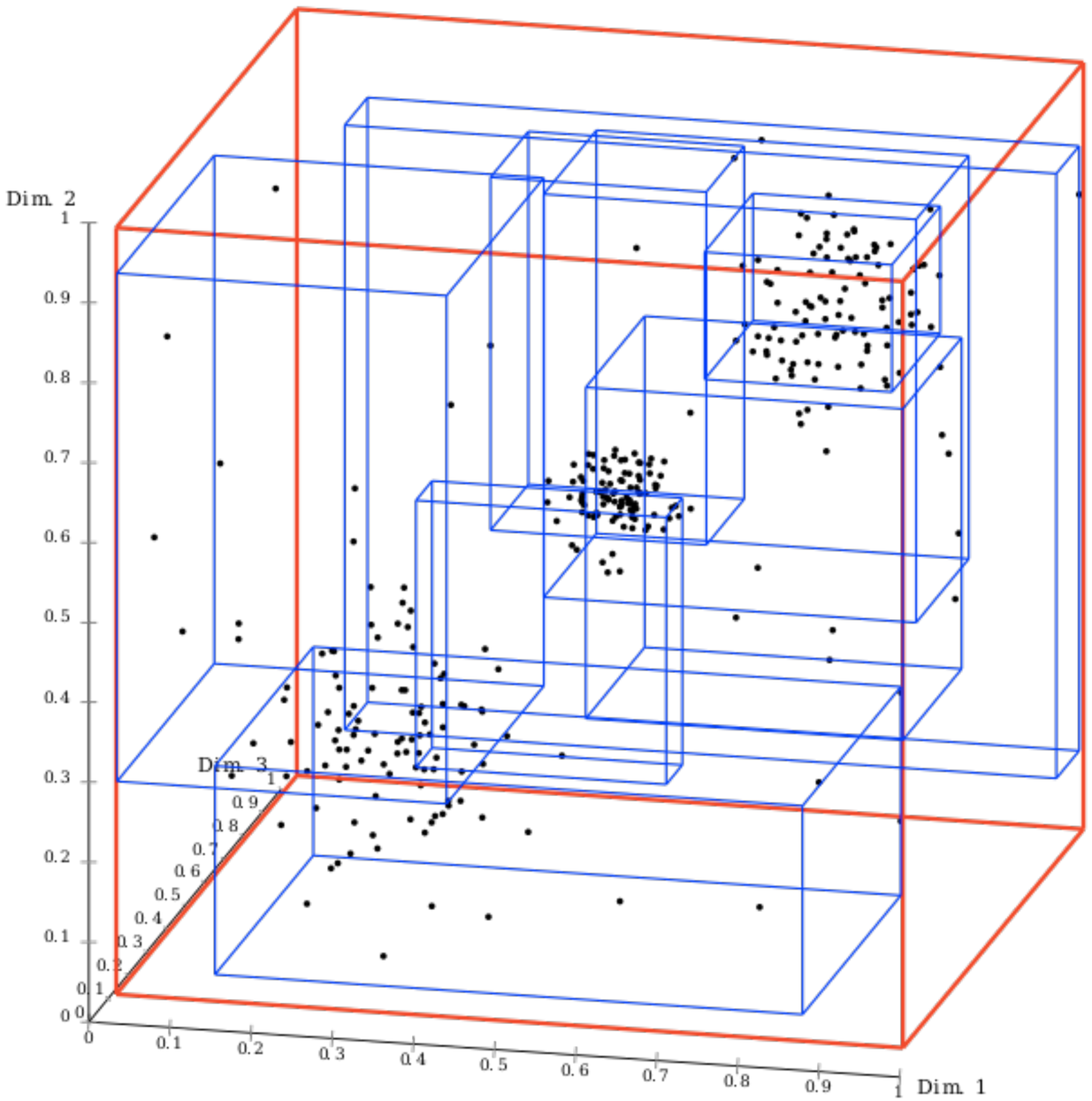
Polygons, polylines, regions, distance to region, union, intersection...



What is so special?

Special indexes

- PostGIS is famous for extensive geospatial support.



What is so special?

Write your own

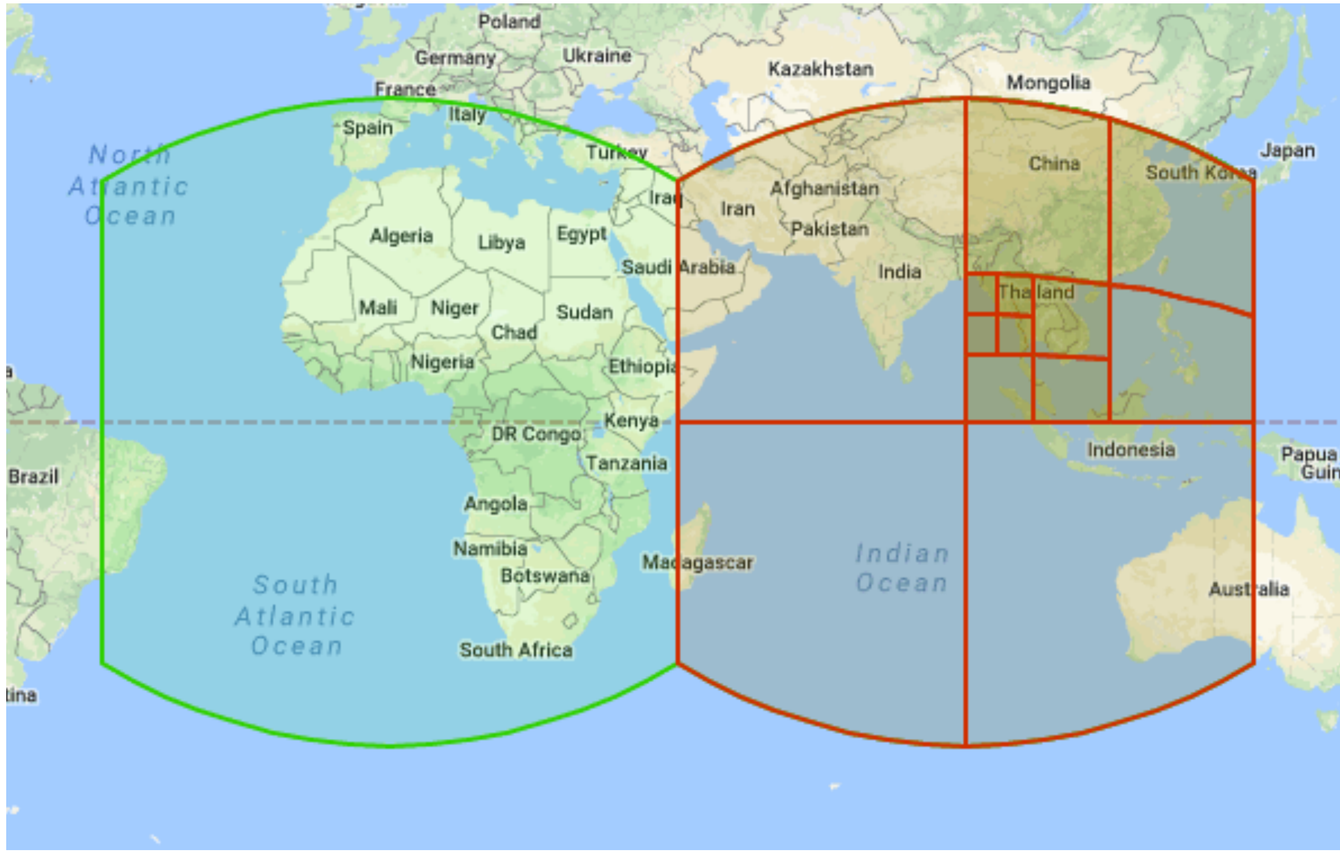
- Keep business logic in one place.
- Be in charge.
- Don’t be constrained by DB limits.

PROCEED WITH CAUTION!

52 Geometry

S2 Geometry

- Geometry functions for r1, r2, r3, s1 and s2 (sphere)
- Spatial indexing.

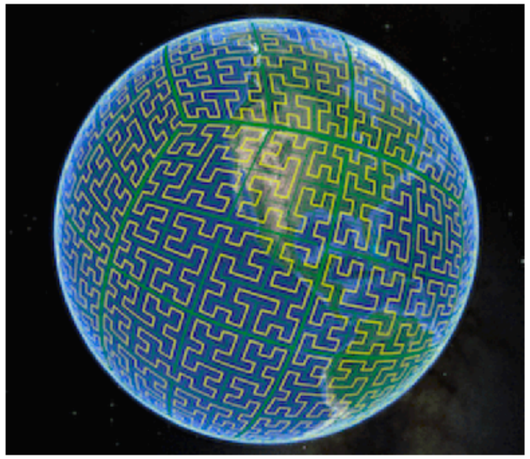


S2 Geometry

- Reanimated in December 2017 with official support promise.
- Written in C++
- Various ports, including Go

Announcing the S2 Library: Geometry on the Sphere

Tuesday, December 5, 2017



Google has always embraced new approaches to organizing all the world's information, and this includes all the world's geography. Today we are announcing the open source release of Google's S2 library, the core geometric library on which Google's global geographic database is built.

A unique feature of the S2 library is that unlike traditional geographic information systems, which represent data as flat two-dimensional projections (similar to an atlas), the S2 library represents all data on a **three-dimensional sphere** (similar to a globe). This makes it possible to build a worldwide geographic database with no seams or singularities, using a single coordinate system, and with low distortion everywhere compared to the true shape of the Earth. While the Earth is not quite spherical, it is much closer to being a sphere than it is to being flat!

badoo_tech

Google

yelp

UBER

mongoDB

badoo_tech

S2 Geometry

Internals introduction

Mercator projection

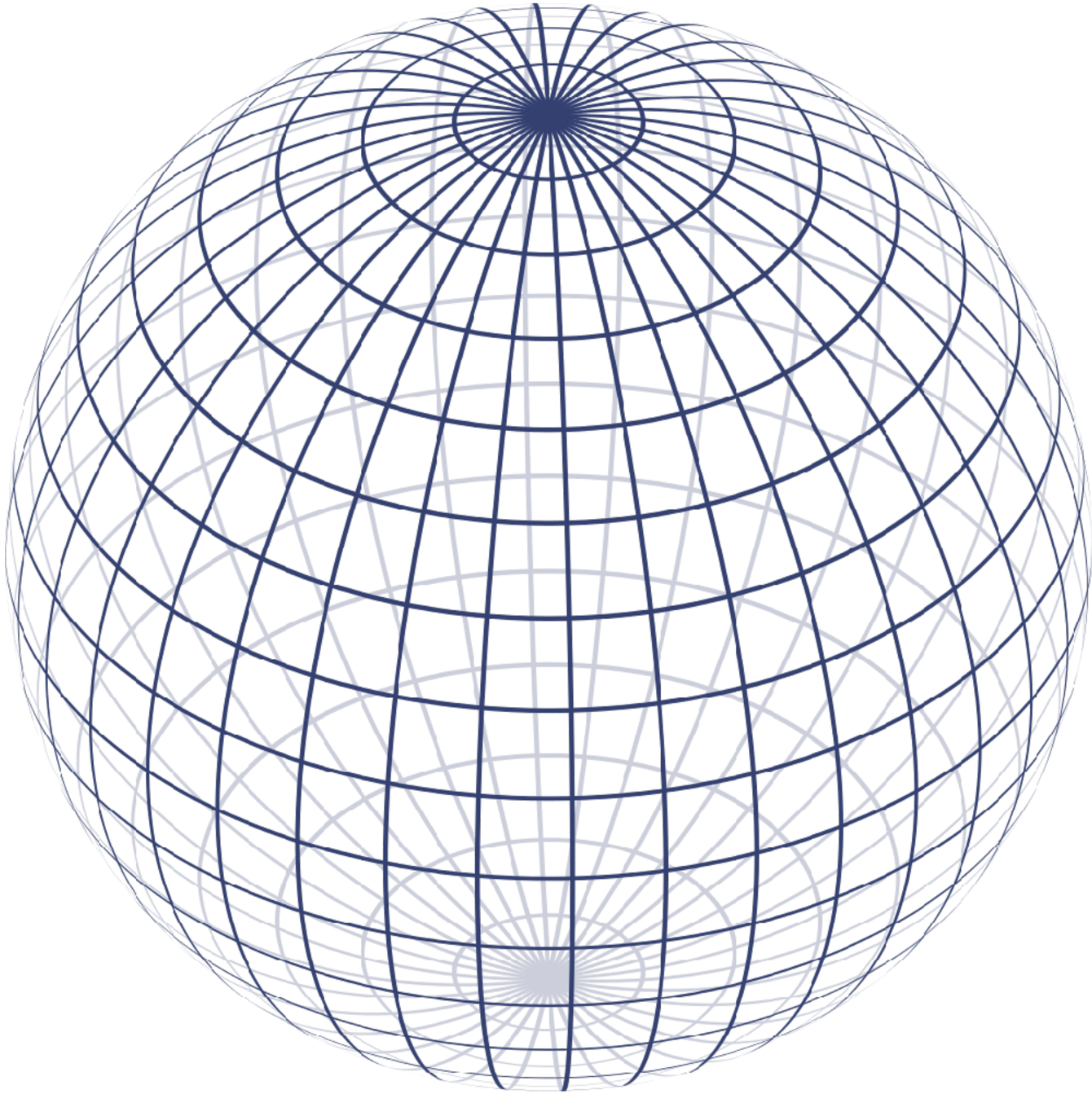
- Earth is not a plane, so we get massive distortions.

Mercator projection

Africa is 2x bigger than Russia, which is not straightforward looking at normal map.

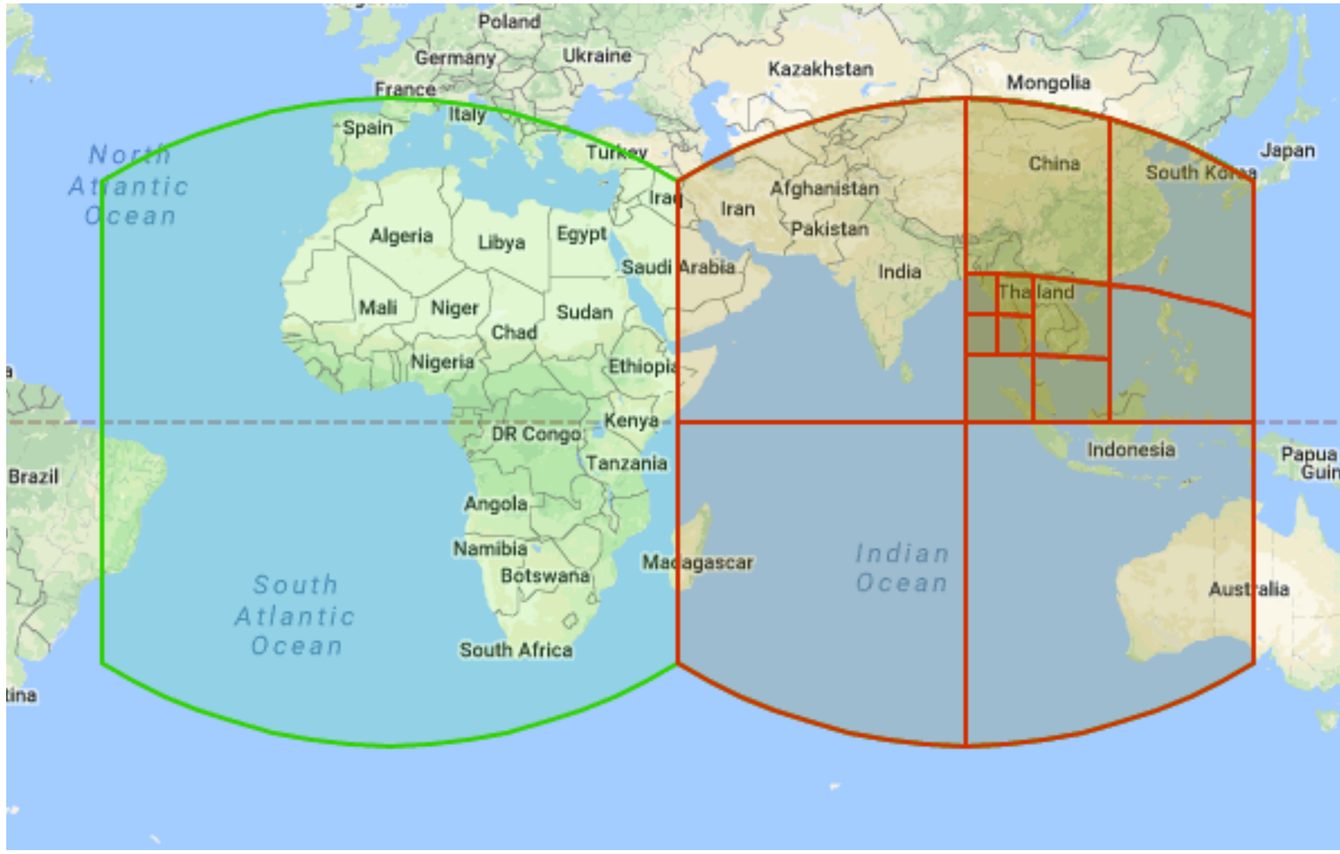
Unit sphere

- Distortion < 1%
- Simpler computations if compared to ellipsoid or geoid, but good enough precision for most tasks.



Cell Hierarchy

- 31 level of division (0-30)
- 30th level cell has size less than 1 cm.



Cell Hierarchy

- 31 level of division (0-30)
- 30th level cell has size less than 1 cm.

Level	Min Area	Max Area
0	85,011,012 km^2	85,011,012 km^2
1	21,252,753 km^2	21,252,753 km^2
12	3.31 km^2	6.38 km^2
30	0.48 cm^2	0.93 cm^2

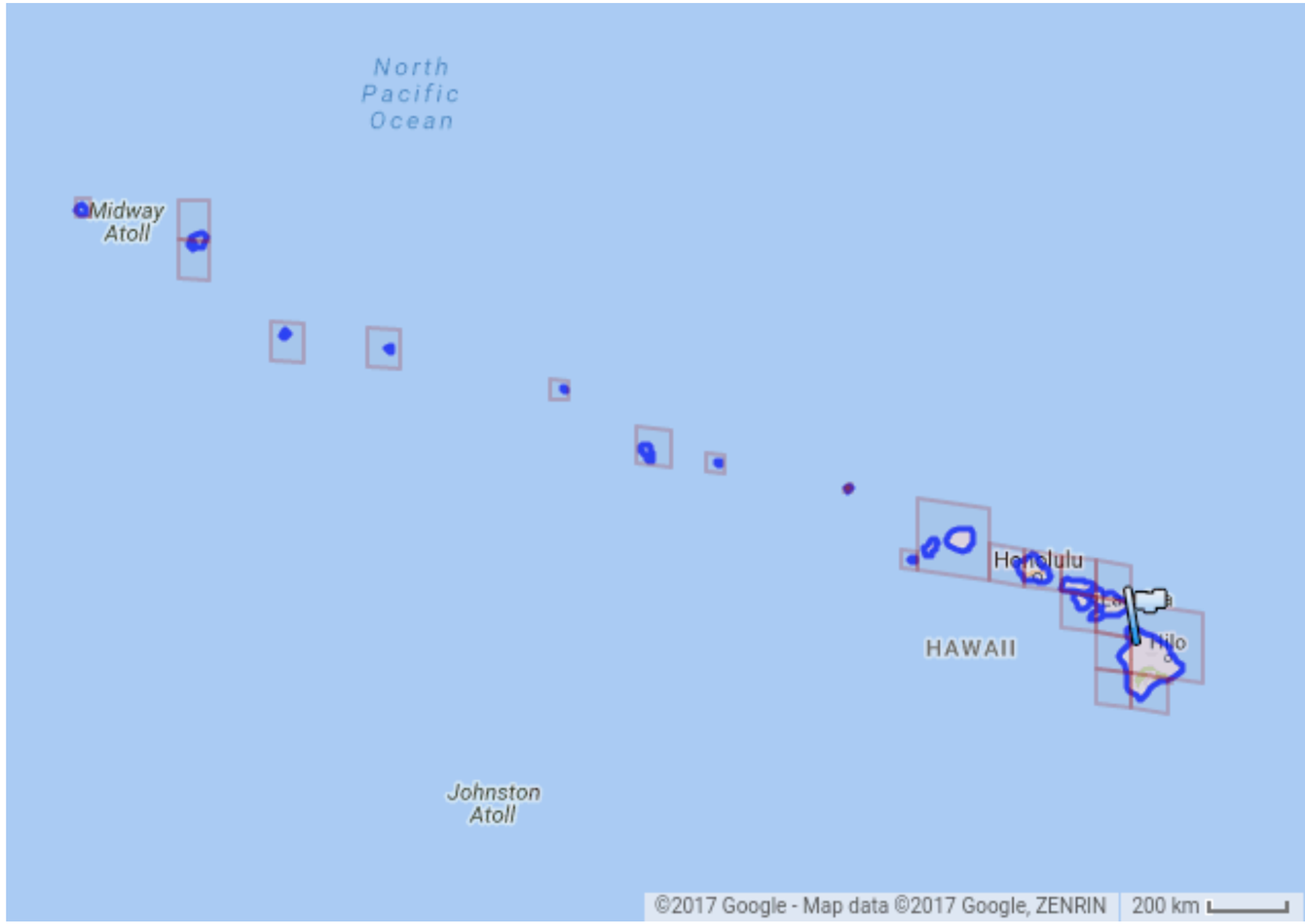
smallest cell

Cell Union

- Union of cells with various levels can be used to represent more complex structures.

Cell Representation

- 64bit number called CellID

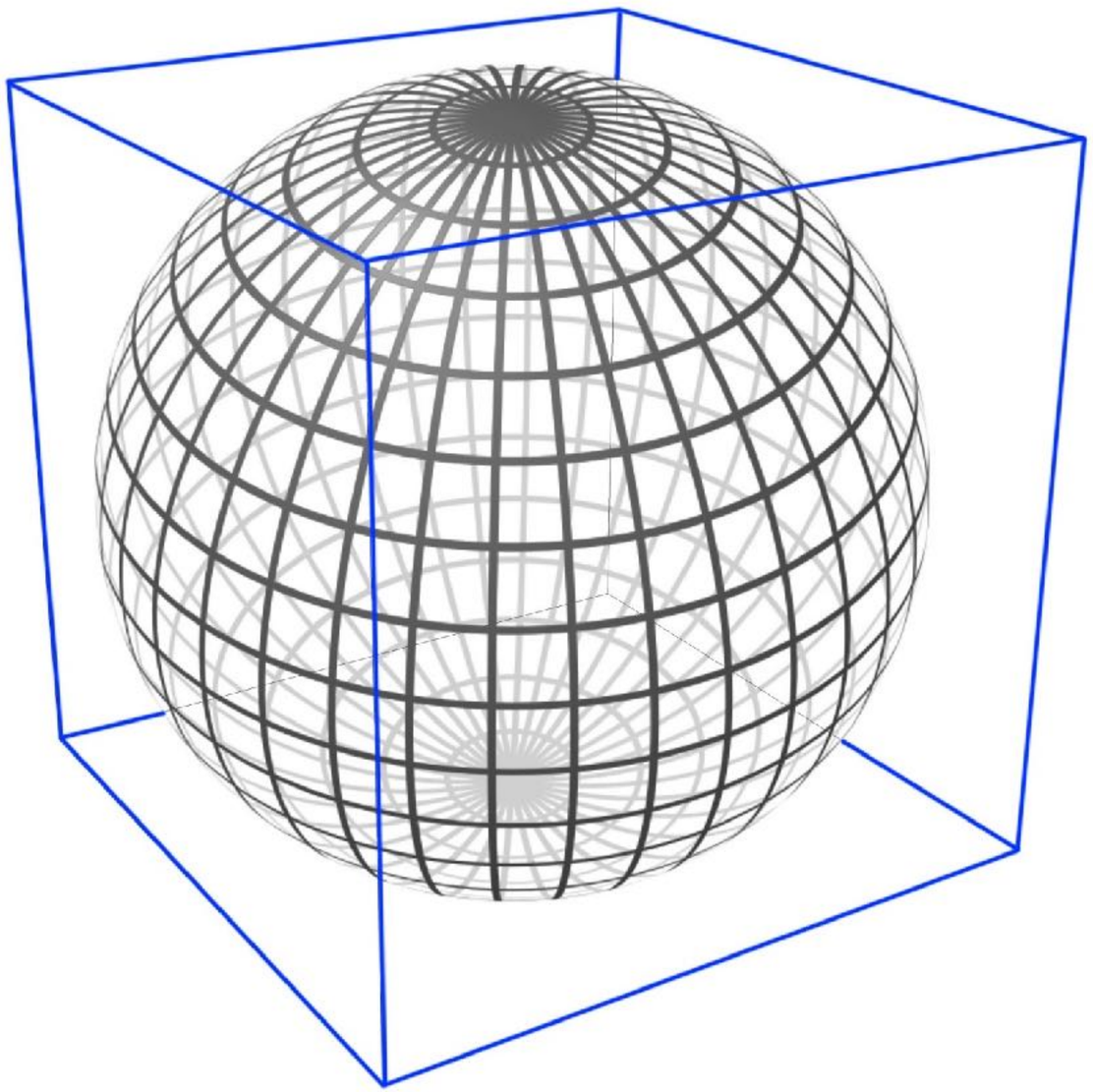


52 Geometry

From 2d to 1d

Level 0

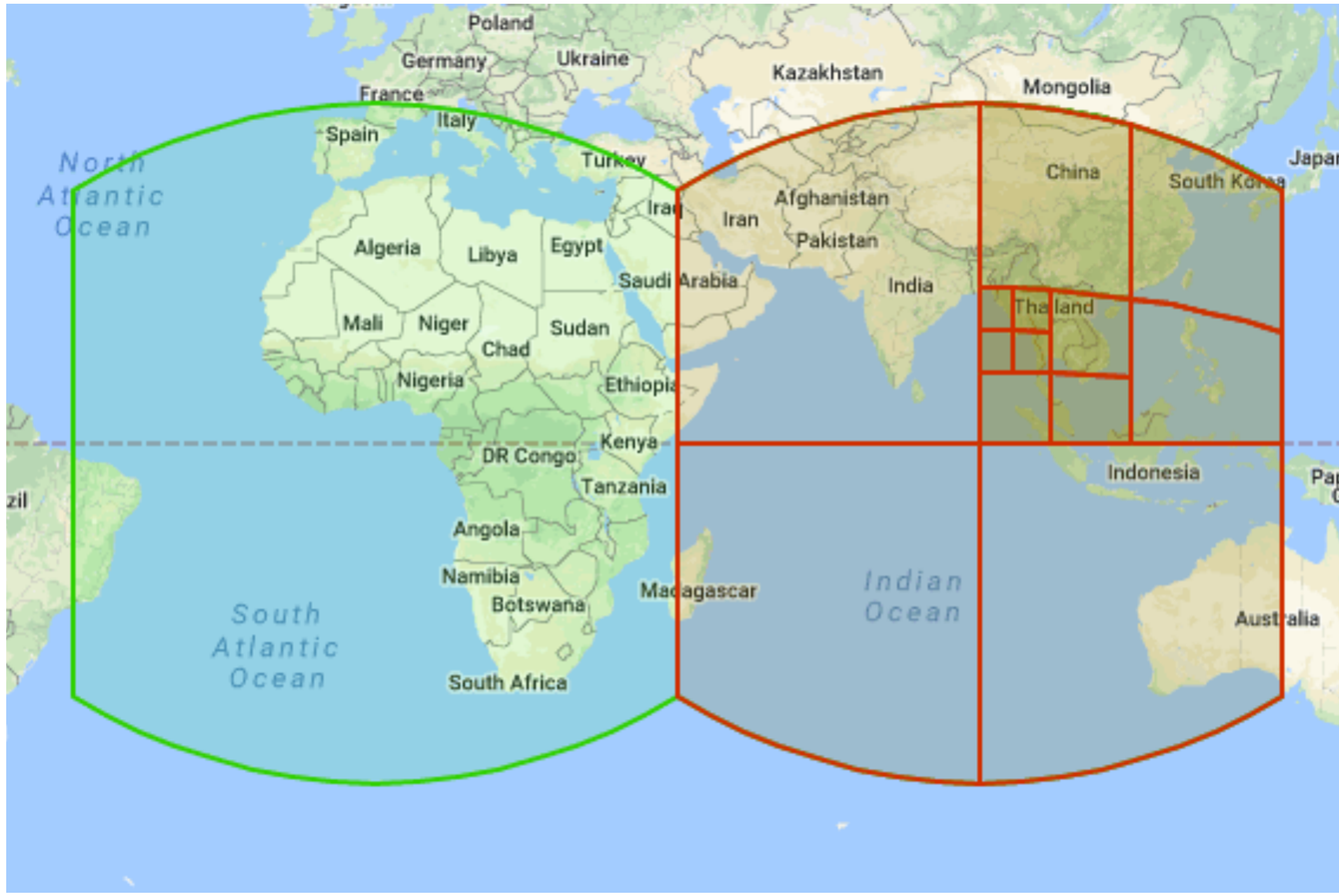
Earth projected to 6 faces of a cube with non linear transformation to correct deformation.



S2 Geometry From 2d to 1d

Levels 1-30

- Divide by 4
- Quadtree



Hilbert curve

One of space filling curves invented by Hilbert long time ago.

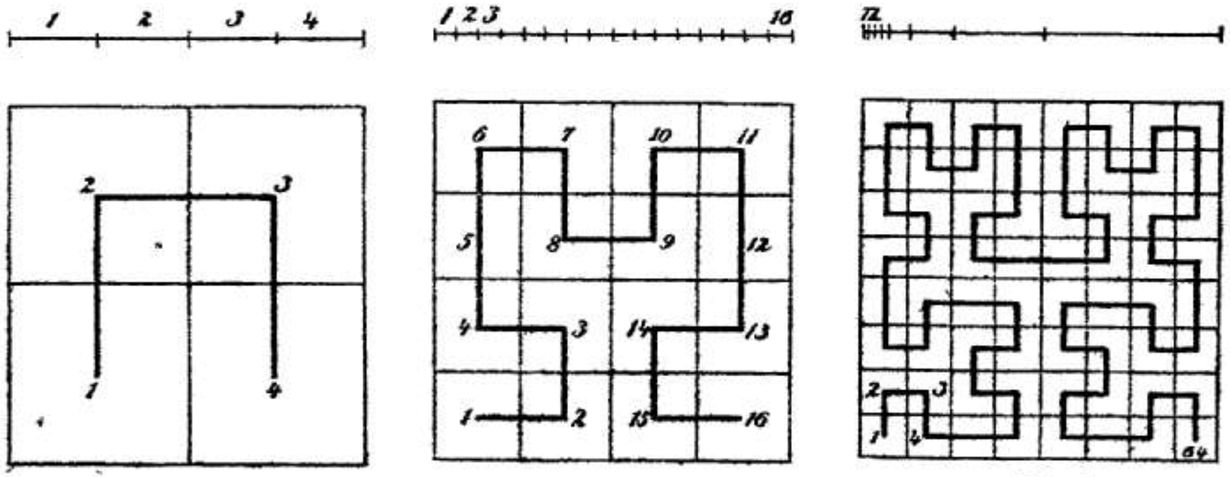
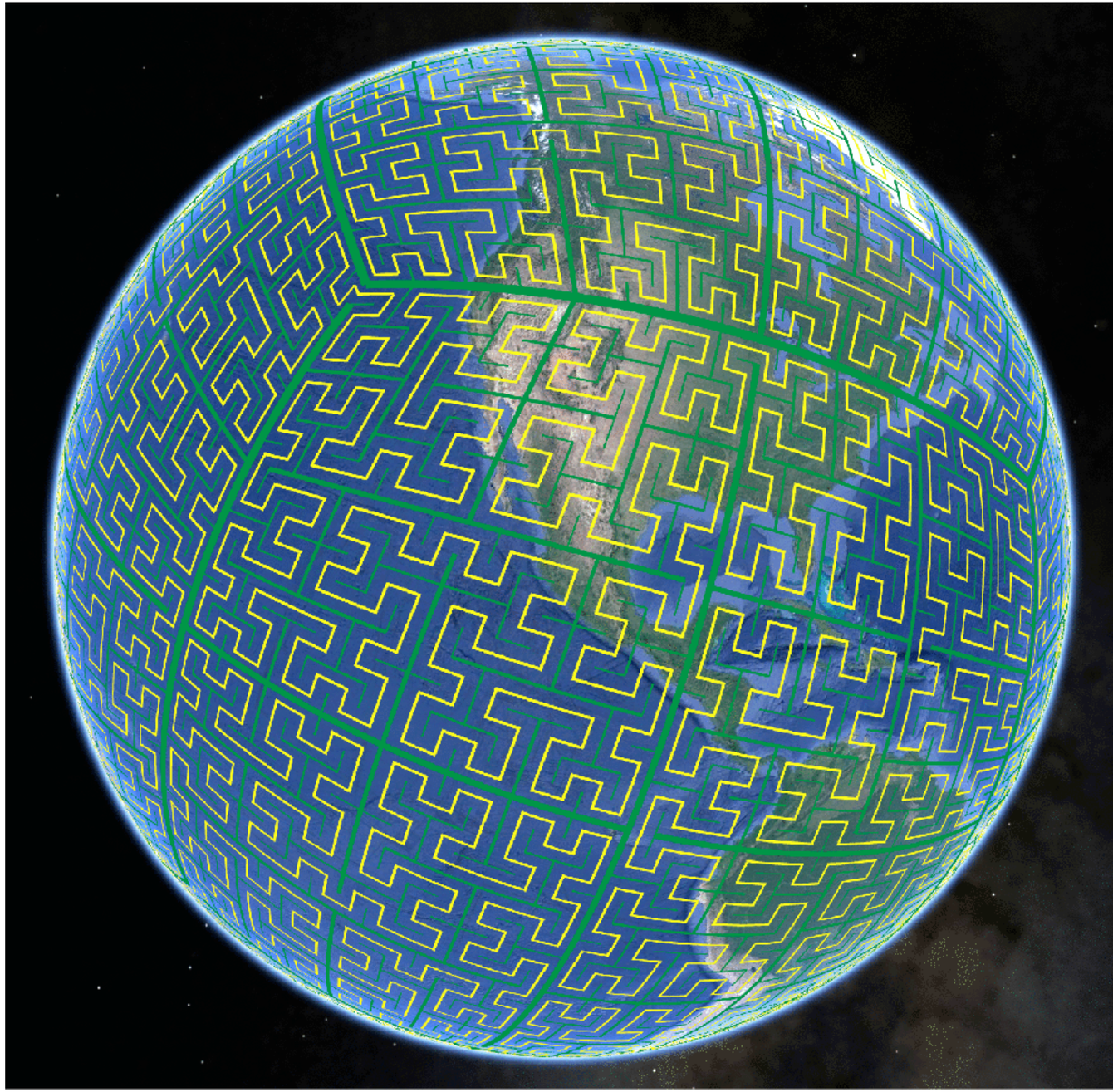


Fig. 1. Fig. 2. Fig. 3.

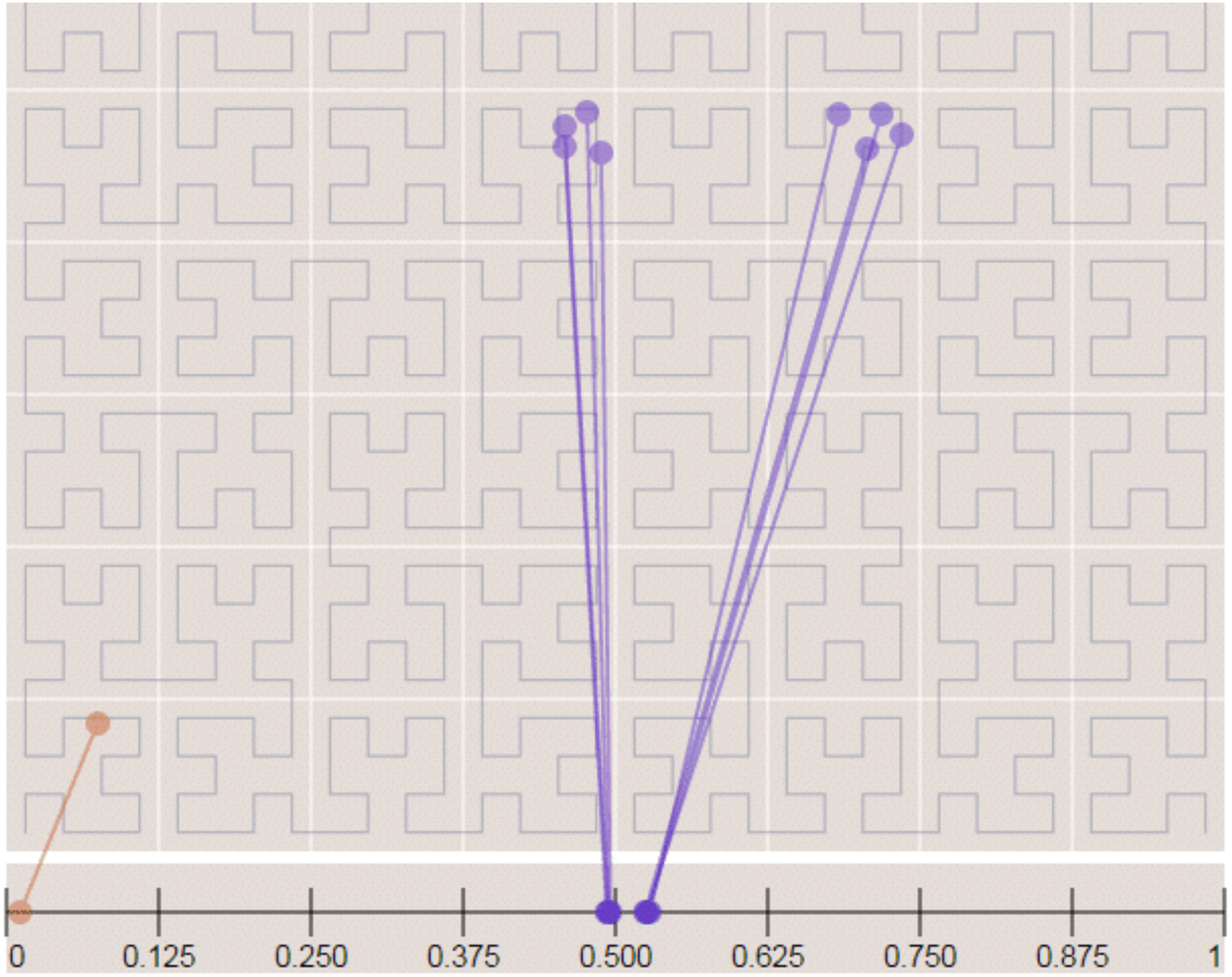
Hilbert curve

One of space filling curves invented by Hilbert long time ago.



Spatial locality

Points that are near each other on the Hilbert curve are near each other in space.

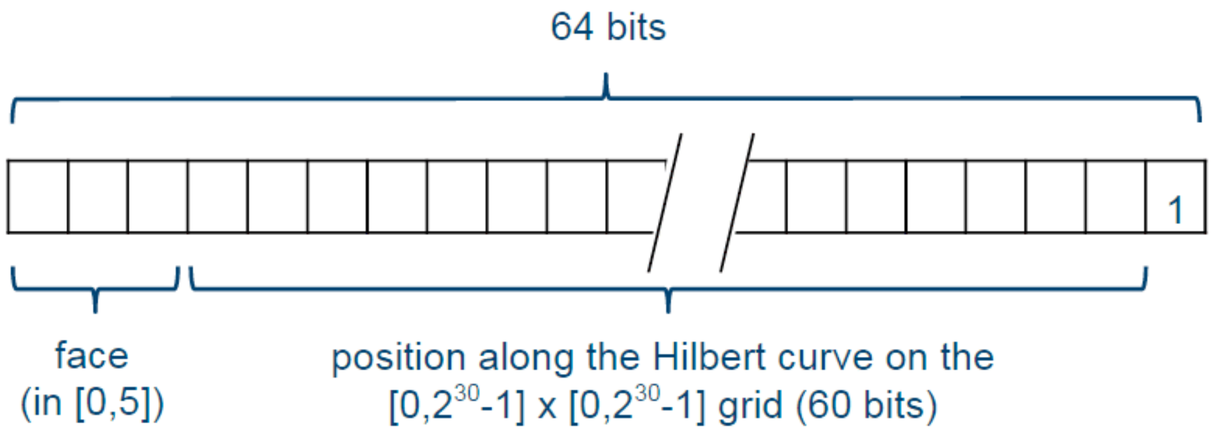


S2 Geometry From 2d to 1d

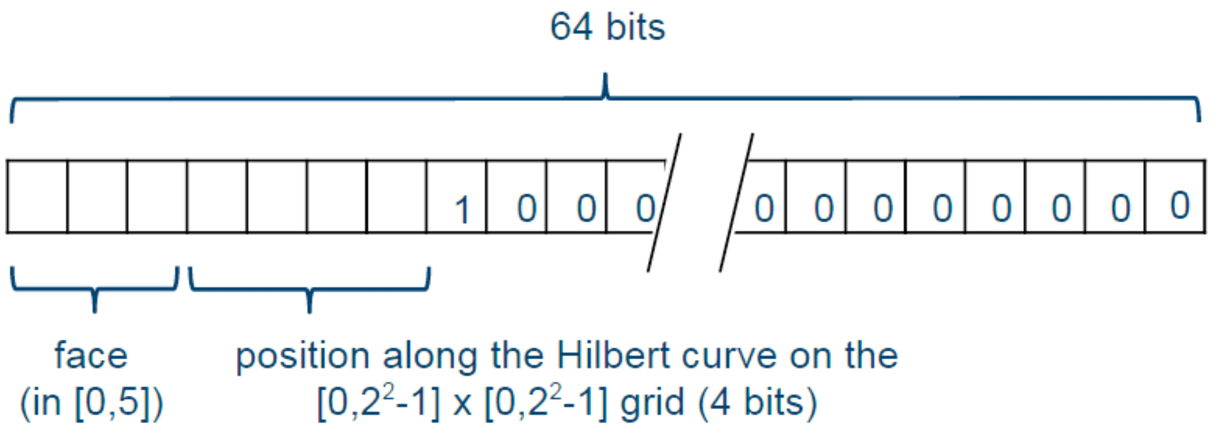
64bit CellID

- 3 bits for cube face
- 2 bits per level
- 1 last bit to represent CellID level
- 3 + (2*30) + 1 = 64 bits

S2 Cell ID of a **leaf** cell (level 30):

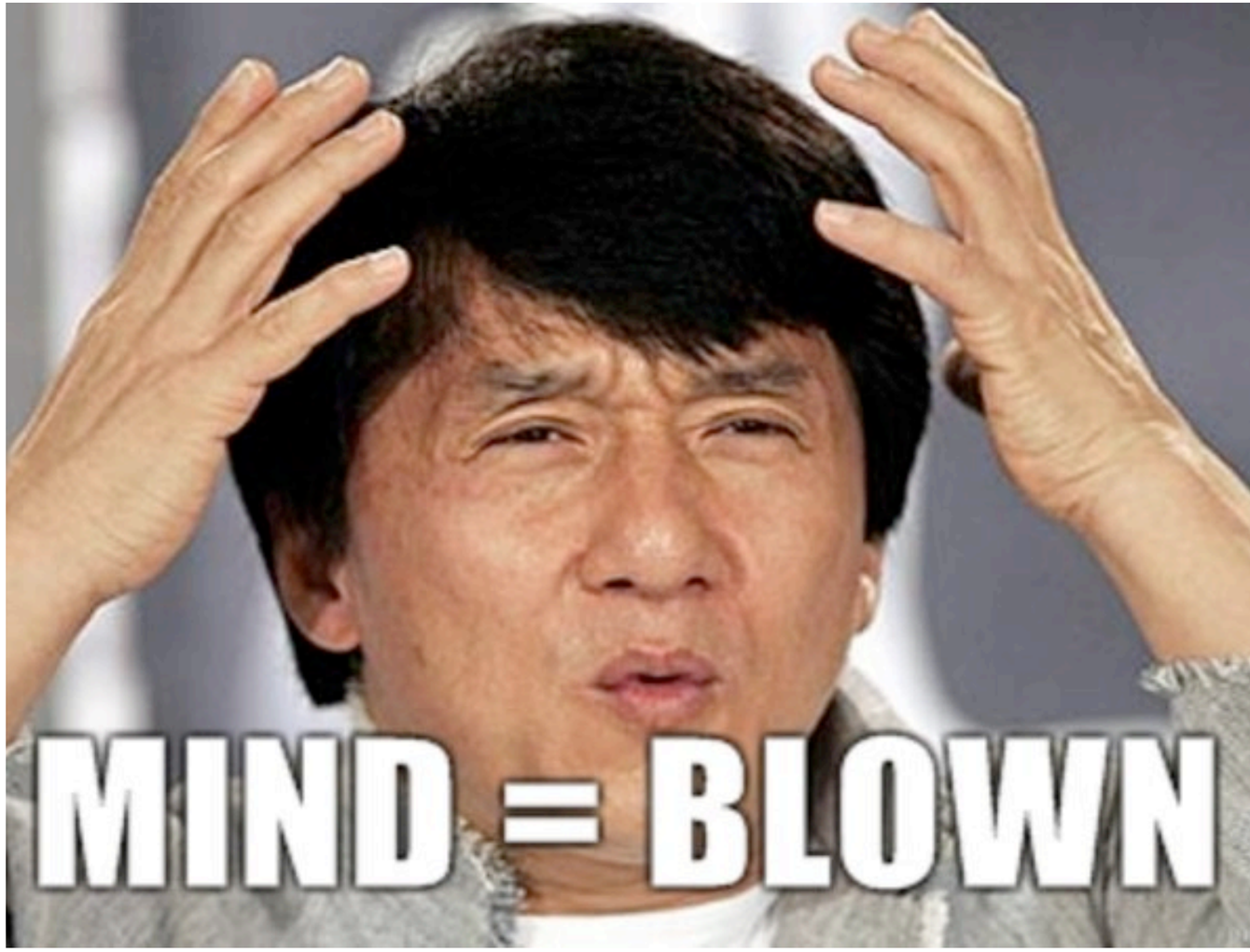


S2 Cell ID of a **level-2** cell:



And then what?

- Every point on Earth can be represented by cell with various accuracy.
- Every object on earth can be represented by union of cells of different levels.
- Every cell can be represented by 64bit variable with spatial locality. This enables us to use standard range search tree like B-tree.
- Hierarchical cells give us the framework for sharding our service.



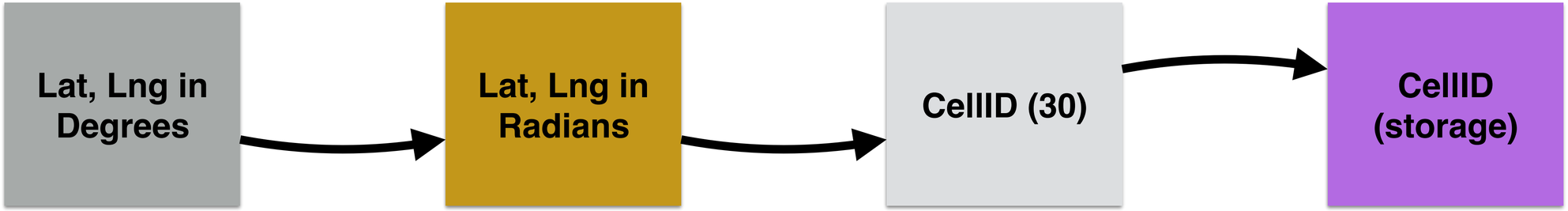
Search around me in some radius

```go
type Index struct {}

func NewIndex(storageLevel int) *Index {}

func (i *Index) AddUser(userID uint32, lon, lat float64) error {}

func (i *Index) Search(lon, lat float64, radius uint32) ([]uint32, error) {}

func TestSearch(t *testing.T) {}
```

badoo_tech

```go
type Index struct {
 storageLevel int
 bt           *btree.BTree
}

type userList struct {
 cellID s2.CellID
 list   []uint32
}

func (ul userList) Less(than btree.Item) bool {
 return uint64(ul.cellID) < uint64(than.(userList).cellID)
}
```

```go
func (i *Index) AddUser(userID uint32, lon, lat float64) error {

  latlng := s2.LatLngFromDegrees(lat, lon)
  cellID := s2.CellIDFromLatLng(latlng)
  cellIDOnStorageLevel := cellID.Parent(i.storageLevel)

  // ...

}
```

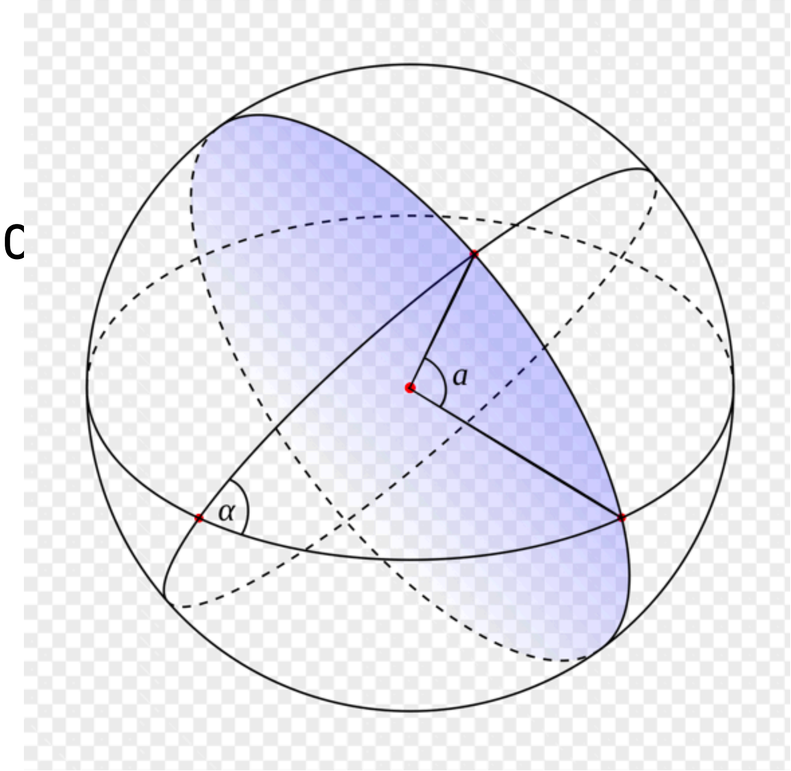


```go
func (i *Index) AddUser(userID uint32, lon, lat float64) error {

  // ...

  ul := userList{cellID: cellIDOnStorageLevel, list: nil}

  item := i.bt.Get(ul)
  if item ≠ nil {
    ul = item.(userList)
  }

  ul.list = append(ul.list, userID)

  i.bt.ReplaceOrInsert(ul)

  return nil
}
```

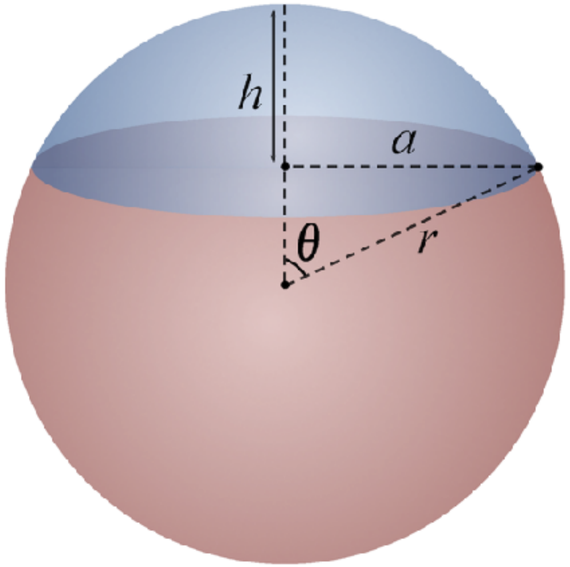
```go
func (i *Index) Search(lon, lat float64, radius uint32) ([]uint32, error) {

  latlng := s2.LatLngFromDegrees(lat, lon)
  centerPoint := s2.PointFromLatLng(latlng)

  centerAngle := float64(radius) / EarthRadiusM
  cap := s2.CapFromCenterAngle(centerPoint, s1.Angle(centerAngle))

  // ...

  return result, nil
}
```

```go
func (i *Index) Search(lon, lat float64, radius uint32) ([]uint32, error) {

  latlng := s2.LatLngFromDegrees(lat, lon)
  centerPoint := s2.PointFromLatLng(latlng)

  centerAngle := float64(radius) / EarthRadiusM
  cap := s2.CapFromCenterAngle(centerPoint, s1.Angle(c

  // ...

  return result, nil
}
```

1



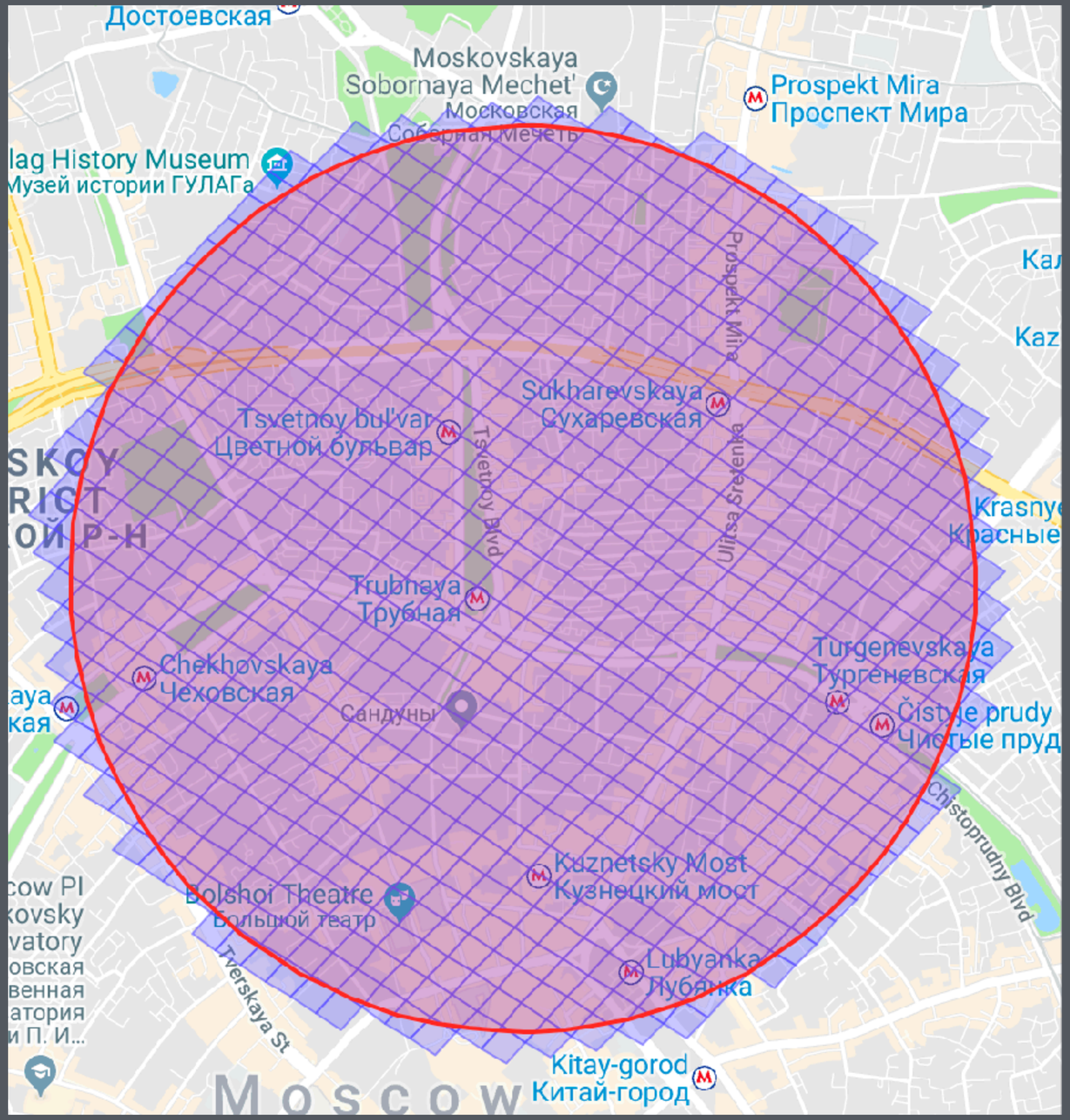
```go
func (i *Index) Search(lon, lat float64, radius uint32) ([]uint32, error) {

  latlng := s2.LatLngFromDegrees(lat, lon)
  centerPoint := s2.PointFromLatLng(latlng)

  centerAngle := float64(radius) / EarthRa
  cap := s2.CapFromCenterAngle(centerPoint                    ngle))

  // ...

  return result, nil
}
```



2

```go
func (i *Index) Search(lon, lat float64, radius uint32) ([]uint32, error) {

  // ...

  rc := s2.RegionCoverer{MaxLevel: i.storageLevel, MinLevel: i.storageLevel}
  cu := rc.Covering(cap)

  // ...

  return result, nil
}
```

badoo_tech


Достоевская
Moskovskaya Sobornaya Mechet'
Московская Соборная Мечеть
Prospekt Mira
Проспект Мира
lag History Museum
Музей истории ГУЛАГа
Kaz
Sukharevskaya
Сухаревская
Tsvetnoy bul'var
Цветной бульвар
Tsvetnoy Blvd
Prospekt Mira
Ulitsa Sretenka
Krasnye
Красные
Trubnaya
Трубная
Turgenevskaya
Тургеневская
Chekhovskaya
Чеховская
Сандуны
Čistye prudy
Чистые пруды
Chistoprudny Blvd
Kuznetsky Most
Кузнецкий мост
Bolshoi Theatre
Большой театр
Lubyanka
Лубянка
Tverskaya St
Kitay-gorod
Китай-город
Moscow

```go
func (i *Index) Search(lon, lat float64, radius uint32) ([]uint32, error) {

  // ...

  var result []uint32

  for _, cellID := range cu {
    item := i.bt.Get(userList{cellID: cellID})
    if item ≠ nil {
      result = append(result, item.(userList).list...)
    }
  }

  return result, nil
}
```

```go
func prepare() *Index {
 i := NewIndex(13 /* ~ 1km */)
 i.AddUser(1, 14.1313, 14.1313)
 i.AddUser(2, 14.1314, 14.1314)
 i.AddUser(3, 14.1311, 14.1311)
 i.AddUser(10, 14.2313, 14.2313)
 i.AddUser(11, 14.0313, 14.0313)
 return i
}
```

badoo_tech

```go
func TestSearch(t *testing.T) {
 indx := prepare()

 found, _ := indx.Search(14.1313, 14.1313, 2000)
 if len(found) ≠ 3 {
  t.Fatal("error while searching with radius 2000")
 }

 found, _ = indx.Search(14.1313, 14.1313, 20000)
 if len(found) ≠ 5 {
  t.Fatal("error while searching with radius 20000")
 }
}
```

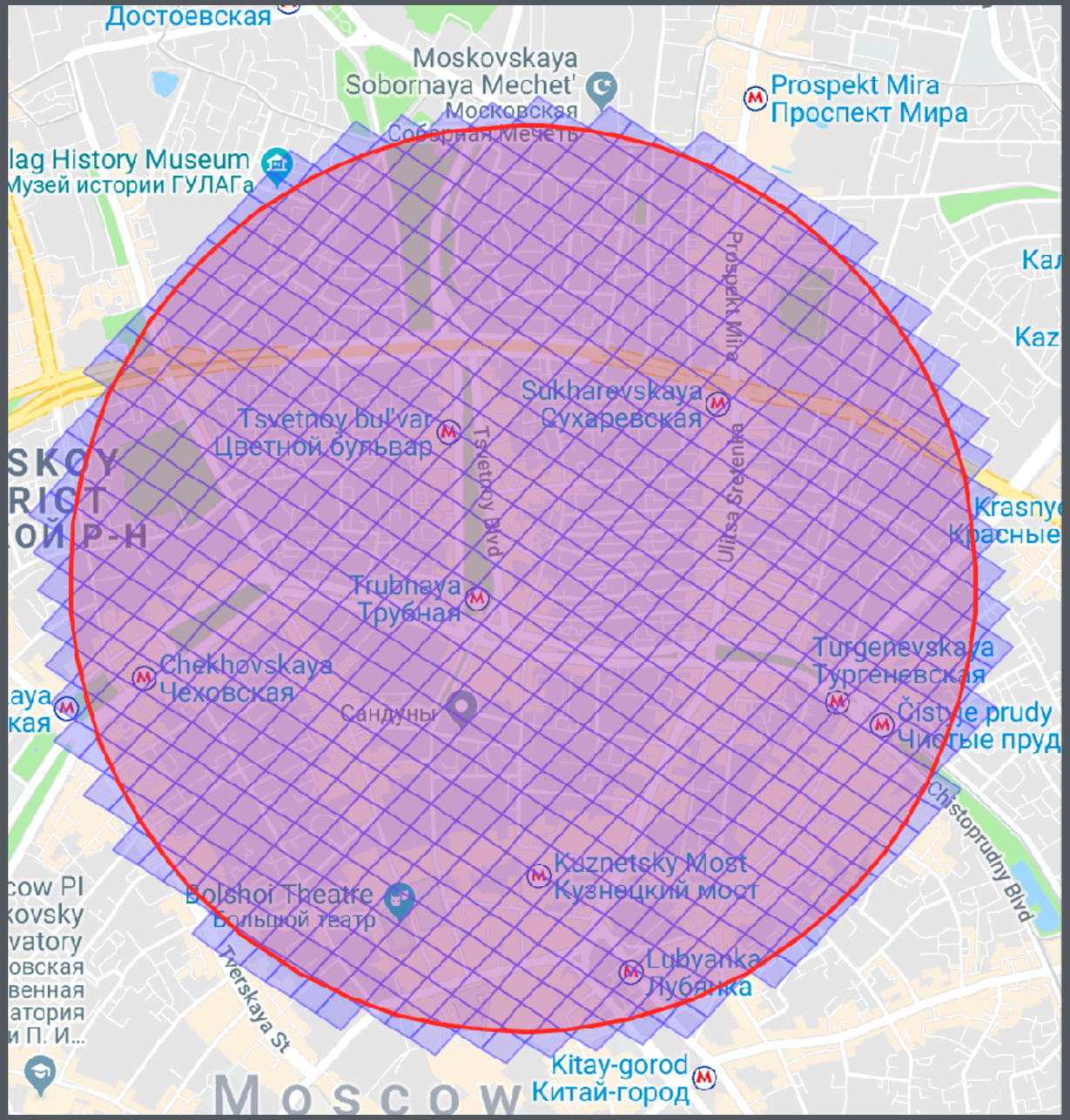
1

2

3

```
$ go test -run 'TestSearch$'
PASS
ok  github.com/mkevac/gophercon-russia-2018/geosearch   0.008s
```

badoo_tech

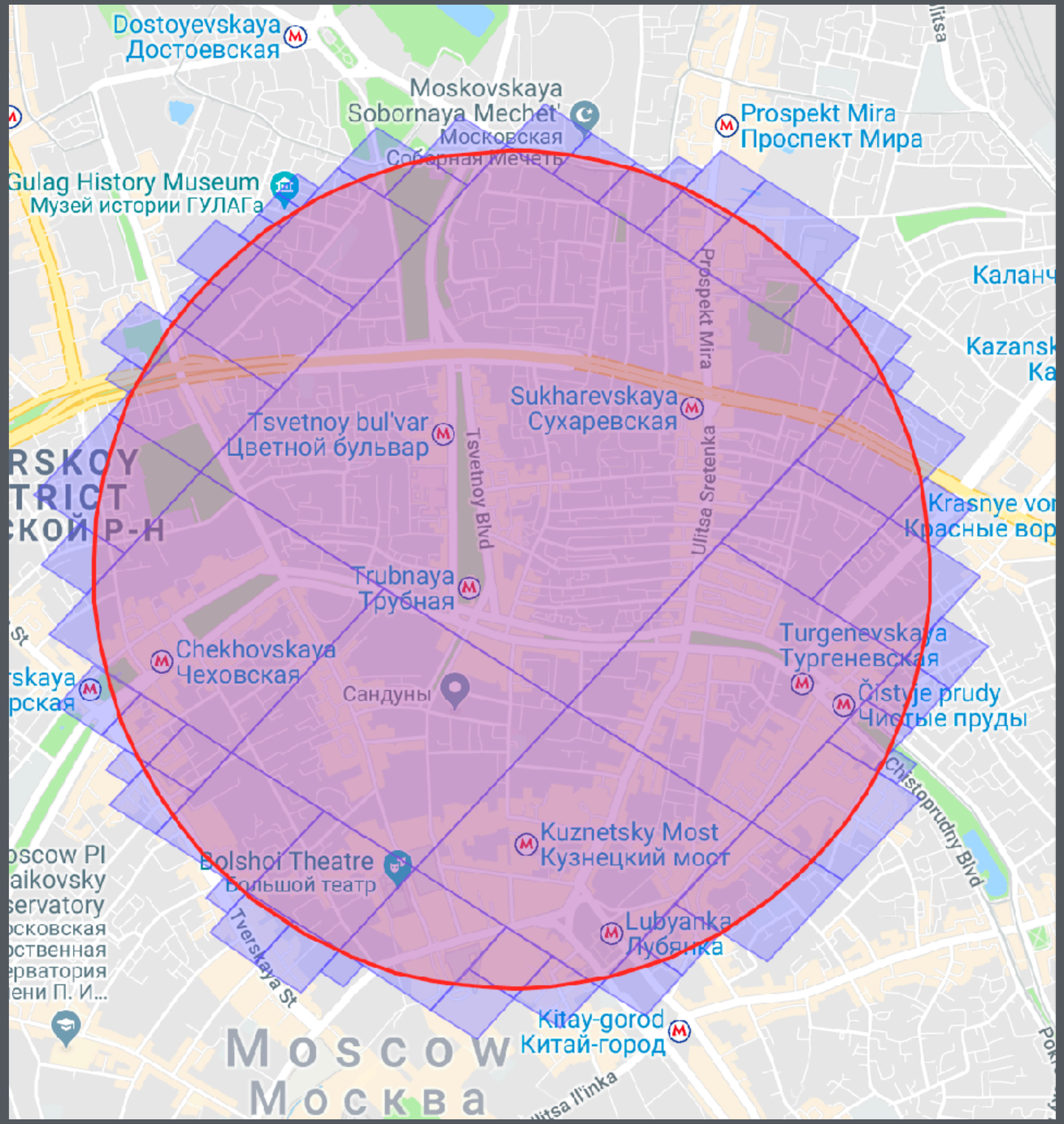

Достоевская
Moskovskaya Sobornaya Mechet'
Московская Соборная Мечеть
Prospekt Mira
Проспект Мира
lag History Museum
Музей истории ГУЛАГа
Kaz
Sukharevskaya
Сухаревская
Tsvetnoy bul'var
Цветной бульвар
Tsvetnoy Blvd
Prospekt Mira
Ulitsa Sretenka
Krasnye
Красные
Trubnaya
Трубная
Turgenevskaya
Тургеневская
Chekhovskaya
Чеховская
Сандуны
Čistye prudy
Чистые пруды
Chistoprudny Blvd
Kuznetsky Most
Кузнецкий мост
Bolshoi Theatre
Большой театр
Lubyanka
Лубянка
Tverskaya St
Kitay-gorod
Китай-город
Moscow

```
rc := s2.RegionCoverer{MaxLevel: i.storageLevel, MinLevel: i.storageLevel}
cu := rc.Covering(cap)
```

1

```
rc := s2.RegionCoverer{MaxLevel: i.storageLevel}
cu := rc.Covering(cap)
```

2


Dostoyevskaya
Достоевская
Moskovskaya
Sobornaya Mechet'
Московская
Соборная мечеть
Prospekt Mira
Проспект Мира
Gulag History Museum
Музей истории ГУЛАГа
Prospekt Mira
Sukharevskaya
Сухаревская
Tsvetnoy bul'var
Цветной бульвар
Tsvetnoy Blvd
Ulitsa Sretenka
Trubnaya
Трубная
Chekhovskaya
Чеховская
Сандуны
Turgenevskaya
Тургеневская
Čistyje prudy
Чистые пруды
Chistoprudny Blvd
Kuznetsky Most
Кузнецкий мост
Bolshoi Theatre
Большой театр
Lubyanka
Лубянка
Tverskaya St
Kitay-gorod
Китай-город
Moscow
Москва

badoo_tech

badoo_tech

```go
func (i *Index) SearchFaster(lon, lat float64, radius uint32) ([]uint32, error) {
  // ...

  for _, cellID := range cu {

    if cellID.Level() < i.storageLevel

      begin := cellID.ChildBeginAtLevel(i.storageLevel)
      end := cellID.ChildEndAtLevel(i.storageLevel)

      i.bt.AscendRange(userList{cellID: begin}, userList{cellID: end.Next()},func(item btree.Item) bool {
        result = append(result, item.(userList).list...)
        return true
      })

    } else {
      // ...
    }
  }

  return result, nil
}
```

1

2

badoo_tech

```go
var res []uint32

func BenchmarkSearch(b *testing.B) {
  indx := prepare()

  for i := 0; i < b.N; i++ {
    res, _ = indx.Search(14.1313, 14.1313, 50000)
  }
}

func BenchmarkSearchFaster(b *testing.B) {
  indx := prepare()

  for i := 0; i < b.N; i++ {
    res, _ = indx.SearchFaster(14.1313, 14.1313, 50000)
  }
}
```

```
$ go test -bench=.
goos: darwin
goarch: amd64
pkg: github.com/mkevac/gophercon-russia-2018/geosearch
BenchmarkSearch-8                 500    3097564 ns/op
BenchmarkSearchFaster-8        200000       7198 ns/op
PASS
ok   github.com/mkevac/gophercon-russia-2018/geosearch  3.393s
```

3 orders of magnitude

Geofencing

```go
type Index struct {
 storageLevel int
 bt           *btree.BTree
 polygons     map[uint32]*s2.Polygon
}

func NewIndex(storageLevel int) *Index {
 return &Index{
  storageLevel: storageLevel,
  bt:           btree.New(35),
  polygons:     make(map[uint32]*s2.Polygon),
 }
}
```

```go
type IndexItem struct {
 cellID         s2.CellID
 polygonsInCell []uint32
}

func (ii IndexItem) Less(than btree.Item) bool {
 return uint64(ii.cellID) < uint64(than.(IndexItem).cellID)
}
```

```go
func (i *Index) AddPolygon(polygonID uint32, vertices []s2.LatLng) error {}
func (i *Index) Search(lon, lat float64) ([]uint32, error) {}
func (i *Index) SearchNearest(lon, lat float64) ([]uint32, error) {}
```

```go
func (i *Index) AddPolygon(polygonID uint32, vertices []s2.LatLng) error {

  points := func() (res []s2.Point) {
    for _, vertex := range vertices {
      point := s2.PointFromLatLng(vertex)
      res = append(res, point)
    }
    return
  }()

  loop := s2.LoopFromPoints(points)
  loop.Normalize()
  polygon := s2.PolygonFromLoops([]*s2.Loop{loop})

  // ...

  return nil
}
```

1

2

badoo_tech

```go
func (i *Index) AddPolygon(polygonID uint32, vertices []s2.LatLng) error {

  // ...

  coverer := s2.RegionCoverer{MinLevel: i.storageLevel, MaxLevel: i.storageLevel}
  cells := coverer.Covering(polygon)

  for _, cell := range cells {
    // ...

    ii.polygonsInCell = append(ii.polygonsInCell, polygonID)
    i.bt.ReplaceOrInsert(ii)
  }

  i.polygons[polygonID] = polygon

  return nil
}
```

1

2

3

```go
func (i *Index) Search(lon, lat float64) ([]uint32, error) {
  latlng := s2.LatLngFromDegrees(lat, lon)
  cellID := s2.CellIDFromLatLng(latlng)
  cellIDOnStorageLevel := cellID.Parent(i.storageLevel)

  item := i.bt.Get(IndexItem{cellID: cellIDOnStorageLevel})
  if item ≠ nil {
    return item.(IndexItem).polygonsInCell, nil
  }

  return []uint32{}, nil
}
```

1

2

badoo_tech

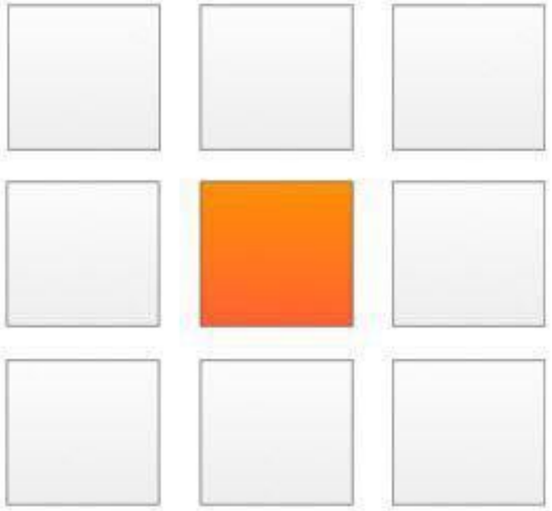
```go
func (i *Index) SearchNearest(lon, lat float64) ([]uint32, error) {

  // ...

  alreadyVisited := []s2.CellID{cellID}
  var found []IndexItem
  searched := []s2.CellID{cellID}

  for {
    found, searched = i.searchNextLevel(searched, &alreadyVisited)
    if len(searched) == 0 {
      break
    }
    if len(found) > 0 {
      return i.filter(lon, lat, found), nil
    }
  }

  return []uint32{}, nil
}
```

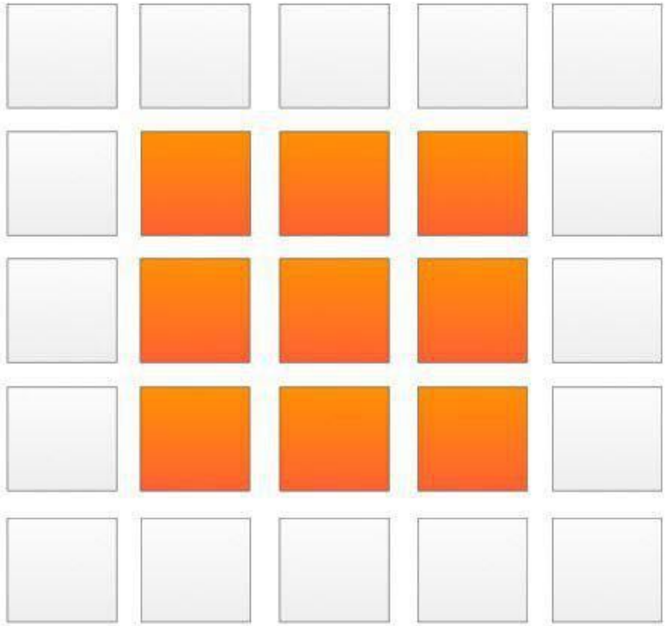
1

2

3

badoo_tech





badoo_tech

```go
func (i *Index) searchNextLevel(radiusEdges []s2.CellID,
  alreadyVisited *[]s2.CellID) (found []IndexItem, searched []s2.CellID) {

  for _, radiusEdge := range radiusEdges {
    neighbors := radiusEdge.AllNeighbors(i.storageLevel)
    for _, neighbor := range neighbors {

      if in(*alreadyVisited, neighbor) {
        continue
      }
      *alreadyVisited = append(*alreadyVisited, neighbor)
      searched = append(searched, neighbor)

      item := i.bt.Get(IndexItem{cellID: neighbor})
      if item ≠ nil {
        found = append(found, item.(IndexItem))
      }
    }
  }
  return
}
```

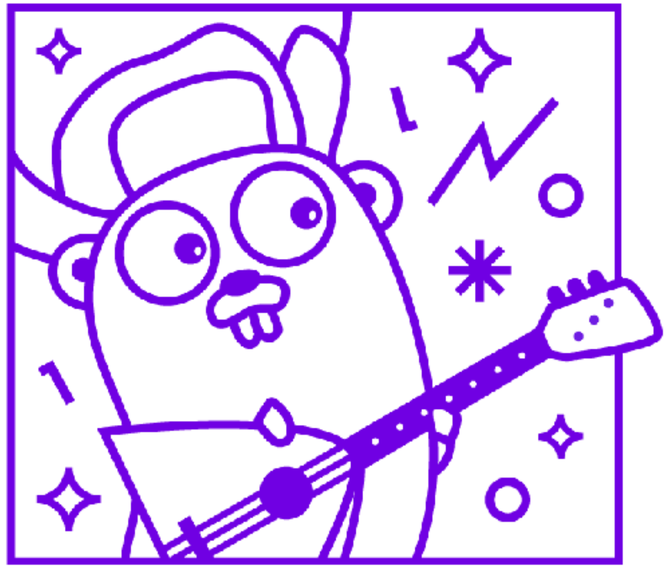
1

2

https://github.com/mkevac/gophercon-russia-2018

Final words

badoo_tech



1

If you've decided to write your own geospatial service, consider using S2.

2

S2 gives you most of what you would need to write that service. Including very convenient framework for service scaling.

3

S2 is very simple and very powerful. We at Badoo love it!

Thank you!

github.com/mkevac/gophercon-russia-2018
techblog.badoo.com
habrahabr.ru/company/badoo

github.com/mkevac
blog.kevac.org
marko@kevac.org

badoo

badoo tech