Massive Technical Interviews Tips

Programming interview questions, tips and knowledge: system design, big data.

Newer Post Slider(Newer 20) Home Random Post Slider(Random 20) Slider(Older 20) Older Post

Distributed BFS algorithms

http://www.1point3acres.com/bbs/thread-148865-1-1.html

4. Pirate. Design Wikipedia crawler.

followup 1: No global status.

followup 2: deal with machine failure

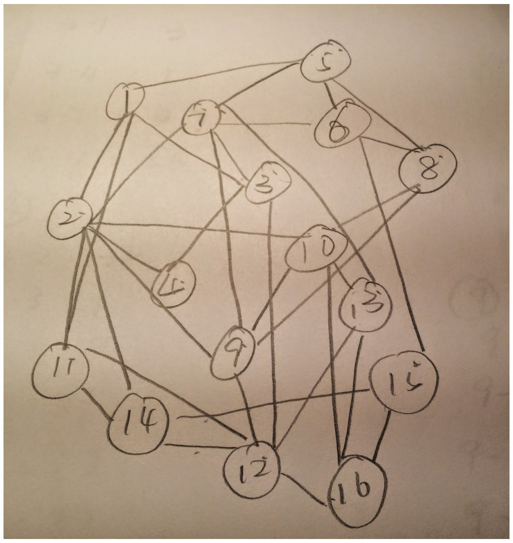
followup 3: make the wiki site unaware of this crawler.

1. distributed bfs
2. consistent hashing
3. assign task with a random delay

http://shirleyisnotageek.blogspot.com/2015/03/breadth-first-search-using-distributed.html

> How would you design the data structures for a very large social network (Facebook, LinkedIn, etc)? Describe how you would design an algorithm to show the connection, or path, between two people (e.g., Me -> Bob -> Susan -> Jason -> You).

This problem is from the Cracking code interview. However, I saw a video on YouTube about how to use distributed system (MapReduce) to do Breadth first search. So I guess that would be a good answer.



How to store the graph: The nodes a list of adjacent nodes (if all weights are 1).

At each iteration, we will start from the original node and grow the frontier by one level. The distance to the start node (DistanceTo(startNode) = 0). For all nodes n directly reachable from startNode, DistanceTo(n) = 1.

Using the above graph, if startNode = 1, then DistanceTo(2) = 1, DistanceTo(11) = 1, DistanceTo(5) = 1, ...etc....
For all nodes reachable from other set of nodes S, DistanceTo(n) = 1 + min(DistanceTo(m), m in S).
So if 4, and 7 is reachable by 2, DistanceTo(4) = 2, DistanceTo(7) = 2.

Not the entire adjacency matrix(sparse matrix, adjacent nodes) to the mapper. Each mapper receives a single row, describing who can be reached from some nodes that we've already known about.
Key: node n that is processing
Value: DistanceTo(n), a list of adjacency nodes (nodes n points to).

So for 1:
Key:1
Value: 0, (2, 3, 5, 11)

Then from those nodes it can reach, we emit those nodes as keys, DistanceTo = D + 1 (shuffle & sort phrase).
So output from mapper:
Key 2, Value 1
Key 3, Value 1
Key 5, Value 1
Key 11, Value 1

The reducer then receive all these values and select the minimum as the new distance. So if 3 can be reached by:
1 -> 3 (1)

Popular Posts

Design a chess game using OO principles | Runhe Tian Coding Practice

[CC150v5] 8.4 Design a Parking Lot - Shuatiblog.com

Avro vs Protocol Buffers vs Thrift

thought-works: Object Oriented design for Elevator in a multi-storied apartment

How to design a tiny URL or URL shortener? - GeeksforGeeks

Design Hit Counter - how to count number of requests in last second, minute and hour - Stack Overflow

Solr vs. Elasticsearch

Design a chat server | Hello World

Difference between System.exit() and System.halt() method?

Design an in-memory file system

Labels

- Algorithm (13)
- Architecture Principles (11)
- Big Data (24)
- Book Notes (57)
- Brain Teaser (12)
- Code Quality (10)
- Company - LinkedIn (11)
- Concurrency (10)
- Cracking Code Interview (22)
- Database (24)
- Design Patterns (29)
- Distributed (19)
- Google (11)
- How to (13)
- Interview (53)
- Interview - Review (21)
- Interview Q&A (20)
- Interview-Java (32)
- Interview-System Design (65)
- Java (93)
- Java - Code (21)
- Linux (35)
- Miscs (25)
- MultiThread (22)

Blog Archive

▼ 2016 (312)
► Septemb
► August (
► July (12)
► June (8)
► May (28)
► April (21
▼ March (5
Solr Cros
AWS Hig
Software
Java Mis
Design T
How to b
SLA - Se
Jersey 1.
jenkins tu
How to w
Lessons Devel
HBASE
Scalabilit
YARN - Y
Jersey 2.
How to b
Mockito T
How to S Instea
dotfiles
JSON
C C++ M
Reason f
Linkedin
Software Intervi
Redis Ar
How to D
System D
Event Bu
Cross Da
Distribute
Java Use
Google's Datac
How to th
Domain-I Comp
Domain-I
Reposito
Asynchro
How to W
How to D
How to b
Database
How to C
Java Enu
Java Dec

- Operating System (13)
- Product Architecture (27)
- Security (11)
- System Design (156)
- System Design - Practice (19)
- Testing (10)
- to-do (40)
- Tools (10)

Pages

- Home
- System Design: Lesson Learned
- Video: System Desgin
- resources

1 -> 11 -> 12 -> 3 (3)
The reducer will select 1.

Then we will pick those output keys and move to the next iteration: a non- MapReduce component then feeds the output of this step back into the MapReduce task for another iteration
Mapper emits the node itself and the points-to list as well. So 1 will be sent back to mapper again, so the shortest distanceTo will not be changed.

Eventually all DistanceTo will converge to their shortest distance, so the algorithm will stop if no shorter distance is found.

Add the edge weight to the adjacency nodes, DistanceTo(n) = DistanceTo(m) + weight(m, n).

http://www.cs.ucsb.edu/~gilbert/cs140/old/cs140Win2011/bfsproject/bfs.html

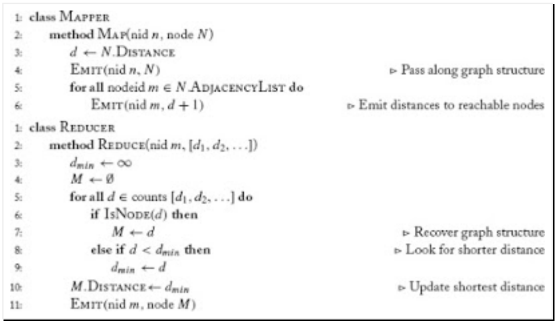
Parallel Breadth-First Search

The idea of doing BFS in parallel is that, in principal, you can process all the vertices on a single level at the same time. That is, once you've found all the level-1 vertices, you can do a parallel loop that explores from each of them to find level-2 vertices. Thus, the parallel code will have an important sequential loop over levels, starting at 0.
In the parallel code, it's possible that when you're processing level i, two vertices v and w will both find the same level-i+1 vertex x as a neighbor. This will cause a data race when they both try to set level[x]=i+1, and also when they each try to set parent[x] to themselves. But if you're careful this is a "benign data race" -- it doesn't actually cause any problem, because there's no disagreement about what level[x] should be, and it doesn't matter in the end whether parent[x] turns out to be v or w.

https://lintool.github.io/Cloud9/docs/content/bfs.html

The key to Dijkstra's algorithm is the priority queue that maintains a globally sorted list of nodes by current distance. This is not possible in MapReduce, as the programming model does not provide a mechanism for exchanging global data. Instead, we adopt a brute force approach known as parallel breadth-first search. First, as a simplification let us assume that all edges have unit distance.

Distance to each node is directly stored alongside the adjacency list of that node, and initialized to ∞ for all nodes except for the source node. In the pseudo-code, we use *n* to denote the node id (an integer) and *N* to denote the node's corresponding data structure (adjacency list and current distance). The algorithm works by mapping over all nodes and emitting a key-value pair for each neighbor on the node's adjacency list. The key contains the node id of the neighbor, and the value is the current distance to the node plus one. This says: if we can reach node *n* with a distance *d*, then we must be able to reach all the nodes that are connected to *n* with distance *d* + 1. After shuffle and sort, reducers will receive keys corresponding to the destination node ids and distances corresponding to all paths leading to that node. The reducer will select the shortest of these distances and then update the distance in the node data structure.



```
class Mapper
    method Map(nid $n$, node $N$)
        $d \leftarrow N$.Distance
        Emit(nid $n$, $N$)                                  ▹ Pass along graph structure
        for all nodeid $m \in N$.AdjacencyList do
            Emit(nid $m$, $d + 1$)                          ▹ Emit distances to reachable nodes
class Reducer
    method Reduce(nid $m$, $[d_1, d_2, \ldots]$)
        $d_{min} \leftarrow \infty$
        $M \leftarrow \emptyset$
        for all $d \in$ counts $[d_1, d_2, \ldots]$ do
            if IsNode($d$) then
                $M \leftarrow d$                            ▹ Recover graph structure
            else if $d < d_{min}$ then                      ▹ Look for shorter distance
                $d_{min} \leftarrow d$
        $M$.Distance $\leftarrow d_{min}$                   ▹ Update shortest distance
        Emit(nid $m$, node $M$)
```

It is apparent that parallel breadth-first search is an iterative algorithm, where each iteration corresponds to a MapReduce job. The first time we run the algorithm, we "discover" all nodes that are connected to the source. The second iteration, we discover all nodes connected to those, and so on. Each iteration of the algorithm expands the "search frontier" by one hop, and, eventually, all nodes will be discovered with their shortest distances (assuming a fully-connected graph). Before we discuss termination of the algorithm, there is one more detail required to make the parallel breadth-first search algorithm work. We need to "pass along" the graph structure from one iteration to the next. This is accomplished by emitting the node data structure itself, with the node id as a key.

In the reducer, we must distinguish the node data structure from distance values and update the minimum distance in the node data structure before emitting it as the final value. The final output is now ready to serve as input to the next iteration.

How to b
Trouble S
Preventa
Wireshar
JDK 8 La
Java Cor
How to b
Code Re
Cloud De Archite
Crack the
随风而逝 问答] |
一道关于

- ► February
- ► January
- ► 2015 (724)
- ► 2014 (115)
- ► 2013 (2)
- ► 2011 (8)

Popular Posts

Design a ch Runhe Tian
[CC150v5] 8 Shuatiblog.
thought-wor Elevator in a
Avro vs Prot
How to desi - GeeksforG
Design Hit C requests in Stack Overfl
Solr vs. Elas
The Uber sc
Design an i
Uber Desig

Typically, execution of an iterative MapReduce algorithm requires a non-MapReduce “driver” program, which submits a MapReduce job to iterate the algorithm, checks to see if a termination condition has been met, and if not, repeats. Hadoop provides a lightweight API for constructs called “counters”.

http://puffsun.iteye.com/blog/1905524

```java
public class ParallelDijkstra extends Configured implements Tool {

    public static String OUT = "output";
    public static String IN = "inputlarger";

    public static class DijkstraMapper extends Mapper<LongWritable, Text, LongWritable, Text> {

        public void map(LongWritable key, Text value, Context context)
                throws IOException, InterruptedException {

            //From slide 20 of Graph Algorithms with MapReduce (by Jimmy Lin, Univ @ Maryland)
            //Key is node n
            //Value is D, Points-To
            //For every point (or key), look at everything it points to.
            //Emit or write to the points to variable with the current distance + 1
            Text word = new Text();
            String line = value.toString();//looks like 1 0 2:3:
            String[] sp = line.split(" ");//splits on space
            int distanceAdded = Integer.parseInt(sp[1]) + 1;
            String[] pointsTo = sp[2].split(":");
            for (String distance : pointsTo) {
                word.set("VALUE " + distanceAdded);//tells me to look at distance value
                context.write(new LongWritable(Integer.parseInt(distance)), word);
                word.clear();
            }
            //pass in current node's distance (if it is the lowest distance)
            word.set("VALUE " + sp[1]);
            context.write(new LongWritable(Integer.parseInt(sp[0])), word);
            word.clear();

            word.set("NODES " + sp[2]);//tells me to append on the final tally
            context.write(new LongWritable(Integer.parseInt(sp[0])), word);
            word.clear();

        }
    }

    public static class DijkstraReducer extends Reducer<LongWritable, Text, LongWritable, Text> {
        public void reduce(LongWritable key, Iterable<Text> values, Context context)
                throws IOException, InterruptedException {

            //From slide 20 of Graph Algorithms with MapReduce (by Jimmy Lin, Univ @ Maryland)
            //The key is the current point
            //The values are all the possible distances to this point
            //we simply emit the point and the minimum distance value

            String nodes = "UNMODED";
            Text word = new Text();
            int lowest = 10009;//start at infinity

            for (Text val : values) {//looks like NODES/VALUES 1 0 2:3:, we need to use the first as a key
                String[] sp = val.toString().split(" ");//splits on space
                //look at first value
                if (sp[0].equalsIgnoreCase("NODES")) {
                    nodes = null;
                    nodes = sp[1];
                } else if (sp[0].equalsIgnoreCase("VALUE")) {
                    int distance = Integer.parseInt(sp[1]);
                    lowest = Math.min(distance, lowest);
                }
            }
            word.set(lowest + " " + nodes);
            context.write(key, word);
            word.clear();
        }
    }
```

http://www.johnandcailin.com/blog/cailin/breadth-first-graph-search-using-iterative-map-reduce-algorithm

every Map iteration "makes a mess" and every Reduce iteration "cleans up the mess". let's say we by representing a node in the following text format

```
ID    EDGES|DISTANCE_FROM_SOURCE|COLOR|
```

where EDGES is a comma delimited list of the ids of the nodes that are connected to this node. in the beginning, we do not know the distance and will use Integer.MAX_VALUE for marking "unknown". the color tells us whether or not we've seen the node before, so this starts off as white.

suppose we start with the following input graph, in which we've stated that node #1 is the source (starting point) for the search, and as such have marked this one special node with distance 0 and color GRAY.

```
1       2,5|0|GRAY|
2       1,3,4,5|Integer.MAX_VALUE|WHITE|
3       2,4|Integer.MAX_VALUE|WHITE|
4       2,3,5|Integer.MAX_VALUE|WHITE|
5       1,2,4|Integer.MAX_VALUE|WHITE|
```

the mappers are responsible for "exploding" all gray nodes - e.g. for exploding all nodes that live at our current depth in the tree. for each gray node, the mappers emit a new gray node, with distance = distance + 1. they also then emit the input gray node, but colored black. (once a node has been exploded, we're done with it.) mappers also emit all non-gray nodes, with no change. so, the output of the first map iteration would be

```
1       2,5|0|BLACK|
2       NULL|1|GRAY|
5       NULL|1|GRAY|
2       1,3,4,5|Integer.MAX_VALUE|WHITE|
3       2,4|Integer.MAX_VALUE|WHITE|
4       2,3,5|Integer.MAX_VALUE|WHITE|
5       1,2,4|Integer.MAX_VALUE|WHITE|
```

note that when the mappers "explode" the gray nodes and create a new node for each edge, they do not know what to write for the edges of this new node - so they leave it blank.

the reducers, of course, receive all data for a given key - in this case it means that they receive the data for all "copies" of each node. for example, the reducer that receives the data for key = 2 gets the following list of values :

```
2       NULL|1|GRAY|
2       1,3,4,5|Integer.MAX_VALUE|WHITE|
```

the reducers job is to take all this data and construct a new node using

- the non-null list of edges
- the minimum distance
- the darkest color

using this logic the output from our first iteration will be :

```
1       2,5,|0|BLACK
2       1,3,4,5,|1|GRAY
3       2,4,|Integer.MAX_VALUE|WHITE
4       2,3,5,|Integer.MAX_VALUE|WHITE
5       1,2,4,|1|GRAY
```

the second iteration uses this as the input and outputs :

```
1       2,5,|0|BLACK
2       1,3,4,5,|1|BLACK
3       2,4,|2|GRAY
4       2,3,5,|2|GRAY
5       1,2,4,|1|BLACK
```

and the third iteration outputs:

```
1       2,5,|0|BLACK
2       1,3,4,5,|1|BLACK
```

```
3       2,4,|2|BLACK
4       2,3,5,|2|BLACK
5       1,2,4,|1|BLACK
```

subsequent iterations will continue to print out the same output.

how do you know when you're done? you are done when there are no output nodes that are colored gray. note - if not all nodes in your input are actually connected to your source, you may have final output nodes that are still white.

http://www.johnandcailin.com/files/blog/Node_0.java

```
public Text getLine() {
  StringBuffer s = new StringBuffer();

  for (int v : edges) {
    s.append(v).append(",");
  }
  s.append("|");

  if (this.distance < Integer.MAX_VALUE) {
    s.append(this.distance).append("|");
  } else {
    s.append("Integer.MAX_VALUE").append("|");
  }

  s.append(color.toString());

  return new Text(s.toString());
}
```

```
public static class MapClass extends MapReduceBase implements
    Mapper<LongWritable, Text, IntWritable, Text> {


  public void map(LongWritable key, Text value, OutputCollector<IntWritable, Text> output,
      Reporter reporter) throws IOException {


    Node node = new Node(value.toString());


    // For each GRAY node, emit each of the edges as a new node (also GRAY)
    if (node.getColor() == Node.Color.GRAY) {
      for (int v : node.getEdges()) {
        Node vnode = new Node(v);
        vnode.setDistance(node.getDistance() + 1);
        vnode.setColor(Node.Color.GRAY);
        output.collect(new IntWritable(vnode.getId()), vnode.getLine());
      }
      // We're done with this node now, color it BLACK
      node.setColor(Node.Color.BLACK);
    }


    // No matter what, we emit the input node
    // If the node came into this method GRAY, it will be output as BLACK
    output.collect(new IntWritable(node.getId()), node.getLine())


  }
}
  public void reduce(IntWritable key, Iterator<Text> values,
      OutputCollector<IntWritable, Text> output, Reporter reporter) throws IOException {


    List<Integer> edges = null;
    int distance = Integer.MAX_VALUE;
    Node.Color color = Node.Color.WHITE;

    while (values.hasNext()) {
      Text value = values.next();

      Node u = new Node(key.get() + "\t" + value.toString());

      // One (and only one) copy of the node will be the fully expanded
      // version, which includes the edges
      if (u.getEdges().size() > 0) {
        edges = u.getEdges();
      }

      // Save the minimum distance
      if (u.getDistance() < distance) {
        distance = u.getDistance();
      }

      // Save the darkest color
      if (u.getColor().ordinal() > color.ordinal()) {
        color = u.getColor();
      }

    }

    Node n = new Node(key.get());
    n.setDistance(distance);
    n.setEdges(edges);
    n.setColor(color);
    output.collect(key, new Text(n.getLine()));

  }
}
```

https://catalystcode.github.io/case-studies/hadoop/hdinsight/big-data/batch-processing/2015/07/21/Parallel-breadth-first-aggregation-algorithm.html

https://github.com/DNyaika/BFS-with-MapReduce/blob/master/src/main/java/it/unitn/bd/bfs/BfsSpark.java

You might also like

- Scalability Rules: 50 Principles for Scaling Web Sites
- Learning Scala
- Linkedin Data Architecture
-
- Implement your own tail (Read last n lines of a huge file) - GeeksforGeeks
- Linux iptables
- Http Misc
- Design data structures for an online book reader system ~ KodeKnight
- 面试题一 交通灯管理系统 | Flyne
- Design Twitter

Posted by Jeffery yuan at 10:51 PM

Labels: Distributed, Map Reduce, to-do

No comments:

Post a Comment

Enter your comment...

Comment as: tejas49 (Googl ▼ Sign out

Publish Preview □ Notify me

Links to this post

Create a Link

Newer Post Slider(Newer 20) Home Random Post Slider(Random 20) Slider(Older 20) Older Post

Subscribe to: Post Comments (Atom)

Labels

System Design (156) Java (93) Interview-System Design (65) Book Notes (57) Interview (53) to-do (40) Linux (35) Interview-Java (32) Design Patterns (29) Product Architecture (27) Miscs (25) Big Data (24) Database (24) Cracking Code Interview (22) MultiThread (22) Interview - Review (21) Java - Code (21) Interview Q&A (20) Distributed (19) System Design - Practice (19) Algorithm (13) How to (13) Operating System (13) Brain Teaser (12) Architecture Principles (11) Company - LinkedIn (11) Google (11) Security (11) Code Quality (10) Concurrency (10) Testing (10) Tools (10) Redis (9) Architecture Model (8) Cache (8) Company - Uber (8) Java67 (8) Linux - Shell (8) OO Design principles (8) SOLID (8) Scalability (8) Spark (8) Interview - MultiThread (7) Java Basics (7) Resource (7) Web Dev (7) Amazon (6) Better Programmer (6) C++ (6) File System (6) Git (6) Highscalability (6) How to Ace Interview (6) Network (6) NoSQL (6) Search (6) Solr (6) Spring (6) Design (5) How to Better (5) How to Interview (5) Interview-Operating System (5) JDK Source Code (5) JavaScript (5) Math (5) Must Known (5) Soft Skills (5) Big Fata (4) C (4) Code Review (4) Company - Twiiter (4) Company Product Architecture (4) Design Principles (4) Facebook (4) Hardware (4) Hash (4) Interview Corner (4) JDK8 (4) JVM (4) Leetcode (4) Optimization (4) Product + Framework (4) Shopping System (4) Source Code (4) Trouble Shooting (4) node.js (4) Back-of-Envelope (3) Cassandra (3) Coding (3) Company - Pinterest (3) Consistent Hash (3) GOF (3) GeeksforGeeks (3) Generics (3) Google Interview (3) Growth (3) Guava (3) Interview-Big Data (3) Interview-Linux (3) Interview-Network (3) Java EE Patterns (3) Javarevisited (3) Kafka (3) Map Reduce (3) OOD Design (3) Performance (3) Puzzles (3) Python (3) Resource-System Desgin (3) Restful (3) UML (3) Web Service (3) geeksquiz (3) AI (2) API Design (2) AngularJS (2) Bugs (2) CareerCup (2) Data Structure Design (2) Data structures (2) Database-Shard (2) Debugging (2) Elasticsearch (2) Garbage Collection (2) GeoHash (2) Hadoop (2) Html (2) Interview-Database (2) Interview-Miscs (2) Interview-Web (2) JDK (2) Logging (2) Machine Learning (2) Math - Probabilities (2) POI (2) Papers (2) Programming (2) Project Practice (2) Random (2) Scala (2) Software Desgin (2) System Design - Feed (2) Thread Synchronization (2) Video (2) reddit (2) Ads (1) Advanced data structures (1) Android (1) Approximate Algorithms (1) Base X (1) Bash (1) Be Architect (1) Behavior Question (1) C# (1) CSS (1) Chrome (1) Client-Side (1) Cloud (1) Coding Interview (1) CodingHorror (1) Company - Facebook (1) Company - Netflix (1) Company - Yelp (1) Counter (1) Crawler (1) Cross Data Center (1) DSL (1) Dead Lock (1) Difficult Puzzles (1) Docker (1) Eclipse (1) Facebook Interview (1) Function Design (1) Functional (1) Game Design (1) Go (1) GoLang (1) ID Generation (1) IO (1) Important (1) Internals (1) Interview - Dropbox (1) Interview - Project Experience (1) Interview - Soft Skills (1) Interview Tips (1) Interview-Algorithm Desgin (1) Interview-Brain Teaser (1) Interview-How (1) Interview-Mics (1) Interview-Process (1) Jeff Dean (1) Joda (1) LinkedIn (1) Mac (1) Micro-Services (1) Mini System (1) MySQL (1) Nigix (1) NonBlock (1) Productivity (1) Program Output (1) Programcreek (1) RPC (1) Raft (1) RateLimiter (1) Reactive (1) Reading (1) Reading Code (1) Resource-Java (1) Resource-System Design (1) Resume (1) SQL (1) Sampling (1) Shuffle (1) Slide Window (1) Spotify (1) Stability (1) Storm (1) Summary (1) System Design - TODO (1) Tic Tac Toe (1) Time Management (1) Web Tools (1) ZooKeeper (1) algolist (1) corejavainterviewquestions (1) martin fowler (1) mitbbs (1)

Popular Posts

Design a chess game using OO principles | Runhe Tian Coding Practice

http://k2code.blogspot.com/2014/03/design-chess-game-using-oo-principles.html http://swcodes.blogspot.in/2012/09/chess-game-design.html ...

[CC150v5] 8.4 Design a Parking Lot - Shuatiblog.com

[CC150v5] 8.4 Design a Parking Lot - Shuatiblog.com A parking lot has multiple Levels. A Level has multiple Parking Spot. A Spot can pa...

thought-works: Object Oriented design for Elevator in a multi-storied apartment

thought-works: Object Oriented design for Elevator in a multi-storied apartment A typical lift has buttons(Elevator buttons) inside the ca...

Avro vs Protocol Buffers vs Thrift

https://martin.kleppmann.com/2012/12/05/schema-evolution-in-avro-protocol-buffers-thrift.html Thrift, Protobuf and Avro all support schem...

How to design a tiny URL or URL shortener? - GeeksforGeeks

How to design a tiny URL or URL shortener? - GeeksforGeeks How to design a system that takes big URLs like "http://www.geeksforgeeks....

Design Hit Counter - how to count number of requests in last second, minute and hour - Stack Overflow

algorithm - how to count number of requests in last second, minute and hour - Stack Overflow There is a hypothetical web server which supp...

Solr vs. Elasticsearch

-- I already uses Solr for several years, here just try to know more about Elasticsearch and the difference and Elasticsearch's advanta...

The Uber software architecture

http://highscalability.com/blog/2015/9/14/how-uber-scales-their-real-time-market-platform.html Video http://www.infoq.com/br/presentation...

Design an in-memory file system

http://www.careercup.com/question?id=13618661 1> Use Structure with n children, on parent pointer , one data field and one name field. ...

Uber Design Excel

http://codeinterviews.com/Uber-Design-Excel/ How would you design an Excel sheet’s Data structure. You need to perform operations like ...