Project Result-Page-Cache Syncher

Assume you are working at the image search division of Google, and lets say Google image search runs on two datacenters in US, (say one in East Coast and one in West Coast) and each datacenter serves to tens of millions of image queries every minute. Since computing the results to image queries are expensive, at each Google datacenter, the result pages (the pages that return as answer to image search queries and contain lots of images) are cached.

You are responsible from making sure that the cached result pages to these queries are available in both datacenters. The idea is, if these queries arrive again, Google will not have to recompute the results pages and instead will display the cached result pages.

One obvious way of doing this is at the end of each minute sharing the result pages to the queries that arrived in that minute with the other datacenter. However, now you have a new boss who thinks this approach is not efficient, as image result pages are huge (hundreds of kilobytes) and most of the queries arriving to both datacenter are identical . She asks you to come up with a way of reducing the number of result pages exchanged by possibly identifying some of the identical result pages efficiently, and not exchanging those.

You immediately suggest to first exchange the queries that arrived, and then using the identical queries identify identical result pages, and then exchange only the result pages that are not identical. Your boss likes this way of thinking, however she thinks even exchanging tens of millions of queries will be too much to communicate and plus this solution brings another communication iteration, which she believes might increase latency especially in heavily loaded times. Instead she asks you to come up with a solution that involves exchanging a fixed tiny amount of data and then synching non-identical result pages using that information. She accepts that you can still possibly exchange some identical pages but should definitely not miss to exchange any non-identical pages.

To test possible solutions, she asks you to create an application instances of which can talk to each other over the internet given each others IP addresses. These applications should have a UI that allows them to admit as input text files containing millions of image queries along with their popularities (the number of times they are queried). After admitting those files, your programs should build a data structure from these queries. When clicked on “sync”, they should sync the set of queries using as little communication as possible. The synched query file should be written to another text file and your app should display the amount of communication made for synching.

Sample query sets are provided.

Once you have a working program for two datacenters, she also asks you to extend your application such that it can be used for simulating the result cache synchronization of three datacenters.

Also, after synchronizing query sets, she asks you to build another data structure from query strings that can be used for query completion. The scenario where your data structure will be used for query completion is as follows: as the user starts typing a query, the system should start suggesting most popular four queries starting with the typed characters so far. e.g. as the user types “bos” the system suggests “boston weather”, “boston university”, “boston globe”, “boston airport” as these four are the most popular queries that start with the characters “bos”, but as the user types “boson” the suggestions are now “boson sampling”, “boson x”, “boson netsim”, “boson and fermion” as now these four are the most popular queries that start with the characters “boson”.

She asks you to demonstrate your data structure with a search box in your app that suggests query completions as a user types in queries.

Project Milestones - Develop a program instances of which can talk to each other (send data to each other) over the internet given each others IP addresses

- Completed project that supports all required operations.
- Project presentations

Comparative Features

Your algorithms and data structures will be compared against others in the class and ranked according to:

- Reductions obtained in amount of communication
- Speed of key operations