Scaling in a Distributed Spatial Cache Overlay

Alexander Gessler, Simon Hanna, Ashley Marie Smith
Universität Stuttgart, Institute for Parallel and Distributed Systems
{gessleah, hannasn, smithae} @studi.informatik.uni-stuttgart.de

Abstract: Location-based services (LBS) for mobile devices are a type of distributed system which utilize geographic behavior of its users. How to balance dynamic query workloads and skewed data remains a problem. Scale-in and scale-out are two proposals that temporarily remove or add resources, respectively. To characterize situations where scaling is more efficient, we implemented a distributed spatial cache overlay for 2D data with the goal of evaluating system performance with and without scaling-out. In this paper, we present an experimental setup to benchmark such a system, and measurements of relative scalability under different cache overlay sizes, query rates and workload distributions. Our results indicate that the system achieves almost linear relative scalability for both uniform and non-uniform query distributions.

1 Introduction

Location-based services (LBS) are data-intensive applications which use the geographic behavior of the user in order to process queries. These queries access spatial data, which correspond to physical geographic regions. A typical example is a route-planning application for smartphones. The user sends an address as a query, and in response to the query, the application delivers data, which could be the corresponding portion of the map. A fundamental problem of LBS is how to allocate the workload in the system, so as to guarantee low latency and efficiency in processing these queries. Much research has focused on developing load-balancing mechanisms for distributed systems with spatial data..

However in an LBS, both data and query workloads have spatial and temporal aspects, which dynamically change and thus require special considerations. Query loads can vary depending on time of day or geographic density of users. For example, fewer queries are sent late at night or in rural areas. Sometimes many queries access data from one location, like when large crowds gather for an event. Moreover, the distribution of data be skewed, i.e. non-uniformly clustered around certain spatial regions, like big cities.

Given these spatial and temporal characteristics, we are interested in how to minimize decreases in system performance under such loads or data distribution. A common technique is to partition data and then build a network overlay, which dedicates nodes to handle requests for certain partitions. [2] [6] Some approaches then focus on reducing data skew via migration or replication of data, but at higher costs for storage [1]. Other approaches estimate query loads or calculate weights to place more nodes around expected hotspots [3]. Yet it is difficult to predict or respond to changes, such as moving hotspots. A more effective approach dedicates nodes in the overlay to cache frequently accessed data; this network is referred to as a *distributed spatial cache overlay*.

However, none of the previously mentioned load-balancing approaches in distributed spatial overlays can increase throughput beyond what is already the system maximum. Thus extreme spikes in query workloads can exceed the system's maximum capabilities. Adding and then removing additional resources to the LBS can address these temporary spikes. The idea is that the overlay automatically decides when to add or remove nodes, based on self-measured load. Adding nodes is referred to as *scaling-in*, whereas removing nodes is referred to as *scaling-out.* In the scope of our project, we investigate whether scaling-out balances dynamic loads in a distributed spatial cache overlay. We consider whether the system scales as query workload increases or whether additional load-balancing, such as elastic load-balancing is necessary.

2 Foundations

Our system is based off on the work of Lübbe et al. [5], [4]. The data region is partitioned into a 2D grid. Then a distributed cache overlay is built on top and consists of nodes dedicated to caching data from certain grid partitions. We refer to a cache node's coordinate as its *cache focus*. The concept of cache focus is important during load-balancing and when nodes cache new data items. Suppose a node is serving a request but a local cache miss occurs. The cache node makes room in its cache list for the requested data by removing the entry with the greatest distance to its cache focus. Our grid of cache nodes forms a Delauney triangulation in a 2D metric space. Greedy routing is used to forward messages to the node that is closest to a certain coordinate. Upon receiving a message, a node checks if any of its neighbors is closer to the target location. If so, it forwards the message to its closest neighbor node. Otherwise, the node handles the message. We specifically chose the Delaunay triangulation, as Bose and Morin [8] have proven that greedy routing in a Delaunay triangulation is guaranteed to find a local optimum.

Load-balancing in a distributed spatial cache overlay under dynamic and non-uniform loads is the foundation of our investigation. One approach that has effectively responded to non-uniform loads is *elastic load-balancing.* Under elastic-load balancing, cache nodes are dynamically moved in the grid, so that cache foci are concentrated at the place which has come under a temporary spike in queries. Scaling-in/-out differ from the scaling mechanism described by Luebbe et al., since their primary focus is elastic load-balancing, i.e. moving cache foci, to balance load. In contrast, the only load-balancing mechanism in our system is that nodes can pass queries to their neighbors when their own workload exceeds a predefined threshold.

3 Architecture and Methodology

When designing our system architecture, as seen in 1, we had to deliver a grid of cache nodes that are organized spatially and deployed on a cluster. These nodes can receive and forward queries, but do not have to cache or process them. The system is launched and shut-down by an external GUI but is independent thereof. Flexibility is important: the system runs remotely on a cluster of unspecified size and hardware and can test the grid under different load-balancing and scaling mechanisms. Our development platform consisted of JRE 1.6 and Eclipse. In addition to the given specifications, we included an administrative node, asynchronous communication between nodes, and a basic logging system. The administrative node only has the task of initializing the overlay construction.

Our implementation of scaling-out relies on the asynchronous message-passing and handshake confirmation between nodes. The basic idea is that when a node's workload exceeds a given threshold, then the node communicates with its neighbors, in order to confirm whether an adjacent triangle could be subdivided by the addition of the new node.

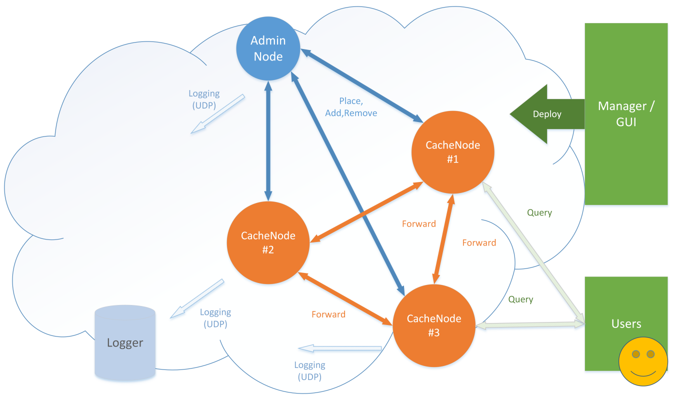

Abbildung 1: System Architecture of Cache as a Service

The goal of our experiment was to quantify latency under different kinds of query loads. Each query was tagged with a coordinate. We measured request latency of the cache overlay under four experimental groups: (1) under uniform queries and scaling-out, (2) under uniform query distribution and without scaling-out, (3) under non-uniform query distributions and scaling-out, (4) under non-uniform query distribution and without scaling-out. Uniform/non-uniform refer to how the target coordinate for each query is selected. In the uniform case, the target coordinate is picked from a uniform distribution. However in the non-uniform case, we can simulate the formation of hotspots: the target coordinate is obtained by sampling a Gaussian distribution with a standard deviation of 0.18 times the area of the data region that is centered around a point in the grid.

The distributed spatial cache overlay was launched on a cluster of up to 32 (virtual) machines. The benchmarking program that we wrote sent each cache node k queries per second. This query rate corresponds to about $1000/k$ milliseconds between two successive queries. For every experimental group, we ran the benchmark program with query rates of $k = 15$ and $k = 20$ and a given number of nodes that composed the grid ($n = 8, 12, 16$). Each run of the benchmark program lasted 10 seconds, so the total number of queries sent to the grid during each run was $n \cdot 10 \cdot k$.

4 Results and Discussion

For a given grid size n and query rate k, 2 show the median latencies in milliseconds of all queries sent, according to experimental group. The values for tests run under uniform distribution/without scaling-out are not shown. They were, as expected, worse than uniform/scaling-out but better than hotspot/scaling-out and hotspot/without scaling-out. The range of the axes differ: for the purpose of clearly showing median times when $k = 15$, the axis ranges only from 0 to 140 ms.

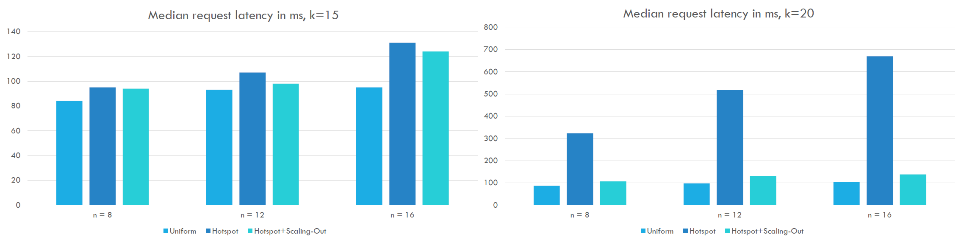

Abbildung 2: Median Request Latencies

We hypothesized that with scaling-out, our system could approximate linear scalability, relative to resources, in both the uniform and non-uniform cases. Our results support our hypothesis. Under the higher request rate $k = 20$, latency increases markedly for the non-uniform(no scaling-out) group. In contrast, the median times for the non-uniform(scaling-out) group when $n = 16$ remain within 30 percent of the median latencies when $n = 8$, although the total number of nodes has doubled and therefore the amount of queries being routed to the hotspot has also doubled.

System robustness was a factor that may have affected our results. We chose $k = 15$ after testing to see which query rates produced unacceptable amounts of node failures and overall system instability. Both the median and mean latencies were calculated. The median, however, is a more representative score of our sample, as extreme latencies in our dataset arose from factors like message timeouts. Improving grid fault tolerance would be good future work; for example, we already suggest replacing the administrative node by implementing a protocol for distributed Delaunay triangulation. [9]

5 Future Work

Based on our preliminary results, our primary focus will be to implement scaling-in and elastic load-balancing to handle changes in both global and relative load. Additionally, to reduce query latency arising from inefficient routing we propose 2D skip list routing and local routing shortlists. A 2D skip list could represent node connections, in that each cache node would not only maintain connections to their direct neighbors, but also to some of their second-degree and third-degree neighbors, and so on. With local routing shortlist, each cache node tracks recent queries which entered the overlay via that node, along with the cache foci of all the nodes the query visited. Nodes could then improve the first routing decision of subsequent queries that enter.

6 Conclusion

In our study, we designed, implemented and evaluated a distributed spatial cache overlay that is flexible enough to allow implementations of more sophisticated load-balancing schemes. This cache overlay serves as the basis for further comparisons of various load-balancing mechanisms to see whether linear scalability relative to resources can be achieved. We confirmed that non-uniform workloads cause the system's performance and scalability to degrade under higher query rates. We also showed that even with non-uniform workloads, the system with scaling-out is able to scale almost linearly relative to resources. Our preliminary results justify further evaluation of the system.

Literatur

[1] A. Mondal, K. Goda, and M. Kitsuregawa. "Effective Load-balancing via Migration and Replication in Spatial Grids."*Database and Expert Systems Applications: Lecture Notes in Computer Science, Vol. 2736*, Springer, 2003: pp.202-211.

[2] B. Godfrey, K. Lakshminarayanan, S. Surana, and R. Karp. "Load Balancing in Dynamic Structured P2P Systems.ïn *Proc. of 23. INFOCOM*, Hong Kong, China: IEEE Computer Society, 2004.

[3] T. Scholl, B. Bauer, et al. "Workload-Aware Data Partitioning in Community-Driven Data Grids.ïn *EDBT* 2009, St. Petersburg, Russia: ACM.

[4] C. Lübbe, and B. Mitschang, "Holistic load-balancing in a distributed spatial cache.ïn *Proc. of 24th Int. Conf. on Mobile Data Management*, Milan, Italy: IEEE Computer Society, 2013: pp. 267-270.

[5] C. Lübbe et.al. "DiSCO: A Distributed Semantic Cache Overlay for Location-Based Services."*Proc. of 15th Int. Conf. on Mobile Data Management*, Lulea, Sweden: EEE Computer Society, 2011: pp. 1-10.

[6] S. Wee and H. Liu. "Client-side load balancer using cloud."*Proc. of 25th Symp. on Applied Computing (SAC'10)*, Sierre, Switzerland: IEEE Computer Society, 2010, pp. 399-405.

[7] M. Randles, D. Lamb, and A. Taleb-Bendiab. "A Comparative study into distributed load balancing algorithms for cloud computing."*Proc. of 24th Int. Conf on Advanced Information Networking and Applications Workshop*, Perth, WA: IEEE Computer Society, 2010, pp.

551-556.

[8] P. Bose and P. Morin. “Online routing in triangulations.ïn *Proc. of 10 th Annual Int. Symp. on Algorithms and Computation (ISAAC’99)*, Berlin: Springer-Verlag, pp.113-122.

[9] D. Y. Lee, S. .S. Lam. “Efficient and accurate protocols for distributed Delaunay triangulation under Churn.”*Proc. of 16th Int. Conf. on Network Protocols*, Orlando, FL: IEEE Computer Society, 2008, pp. 124-136.