The Geo-aware State Deployment Problem for Mobile Distributed Applications

Diogo Lima
LaSIGE, Faculdade de Ciências,
Universidade de Lisboa
Lisbon, Portugal
dlima@lasige.di.fc.ul.pt

ABSTRACT

The geographical barrier between mobile devices and mobile application servers (typically hosted in cloud datacenters) imposes an unavoidable latency and jitter that negatively impacts the performance of modern mobile systems. Fog Computing architectures can mitigate this impact, but such architectures directly depend on a middleware service able to correctly partition and deploy the state of an application at optimal locations. Geo-aware state deployment is challenging as it must consider the mobility of the devices and the dependencies arising when multiple devices concurrently manipulate the same application state.

In this work we start by proposing our system model based on the Fog Computing paradigm. Then, our research focuses on geographically aware state deployment strategies able to benefit from the proposed system model to avoid latency and increase quality-of-experience, by having most of traffic handled by servers located in close proximity to users. Finally, we plan to investigate the real impact and role of the Oracle component of our system model, responsible for managing the state deployment decisions and coordinating the multiple servers of the system.

CCS CONCEPTS

• **Computing methodologies**; • **Distributed computing methodologies**; • **Distributed algorithms**; • **Self-organization**;

KEYWORDS

Mobile Distributed Applications, Fog Computing, Geographically-aware State Deployment

1 INTRODUCTION

The evolution of wireless networking technologies is leading to an increasing number of large scale distributed mobile applications over wireless networks in domains such as smart cities, augmented reality games (e.g. Ingress, Pokemon Go) and social networks (e.g. Foursquare). Distributed mobile applications are characterized by concurrently connecting a large number of users that retrieve, publish and manipulate significant amounts of application state. The current trend tends to concentrate consistency and concurrency control in a supporting infrastructure hosted in Cloud datacenters. However, this model creates a *geographical barrier* between the end users and the application's state managers. The latency and jitter that unavoidably results from this distance negatively impacts the application performance.

The Fog Computing approach [2] proposes to overcome this geographical barrier by deploying *surrogate servers* at the edge of the network, in particular on access networks. Although this approach has the potential to mitigate jitter and latency, its performance strongly depends of the ability of a middleware service to partition application state and deploy each of its components at its most convenient location. A problem referred as *geo-aware state deployment*.

A good geo-aware state deployment must be aligned with the distributed access pattern of each state component. This is challenging as it must consider the application semantic, the mobility of the devices and the dependencies established between multiple devices concurrently manipulating application state.

2 PROBLEM STATEMENT

We assume a wide scale distributed application that is supported by servers deployed at distinct locations at the edge of the network, hereafter named *surrogates*. Surrogates serve as access points to clients connecting to the application, storing state and providing computing power, and are placed at a neighbourhood level in stores, restaurants or public services. Expectations are that, similar to the benefits of providing free wifi access to their costumers, the increasing offer of storing and computing power will allow stores to attract more costumers and for longer periods of stay. Surrogates also connect to other surrogates and to cloud datacenter(s) by a high speed wired network. End users devices include desktop computers, mobile devices and sensors using wired or wireless networks.

Figure 1 graphically depicts the implementation of our model. The geo-aware state deployment is managed by a service that conceptually acts as an oracle (1), becoming eventually aware of the source of all accesses to state items. The oracle decides at run-time the most suitable location of each state item and coordinates its transfer between surrogates. The oracle collects transaction logs, delivered in the background by the surrogates (2,3).

We assume that application state is composed of three sorts of data: *(i)* data that is unique to each user or device (*personal state*), *(ii)* collaborative data relevant to a specific geographical location (*geo-aware state*), and *(iii)* general application logic data (*global state*). Personal and geo-aware state items are location dependent, i.e., they are associated with one or more surrogates. In the particular case of personal state, these surrogates change over time, as users move.

Expectations are that, to avoid latency and increase quality-of-experience, most of the traffic produced by clients is handled at surrogates. To address these expectations, the system must implement a geo-aware state deployment strategy, allowing surrogates to store each state item at a location most likely to reduce latency and long-distance accesses. However, application operations (hereafter

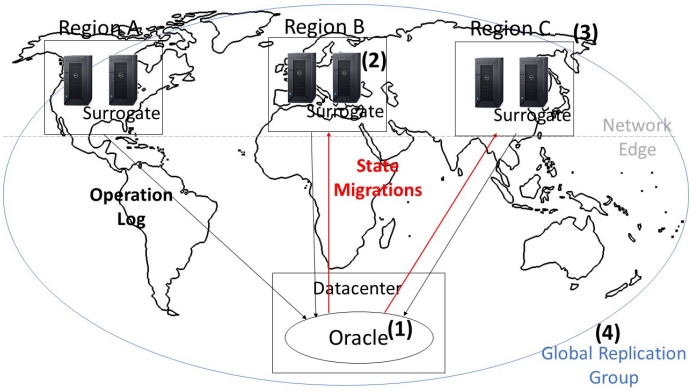

Figure 1: System model.

referred as *transactions*) may affect more than one state item, possibly hosted at more than one surrogate. Depending on the number of sites involved, transactions can be either *local* or *global*. Local transactions are those that access data stored at a single site, while global transactions involve multiple sites.

The complexity and latency of coordinating global transactions motivate the oracle to keep commonly accessed state components at the same site. Each migration decision must consider the cost of the migration itself, the benefits of increasing the number of local transactions and the cost of future global transactions. We consider three metrics to evaluate the performance of a geo-aware state deployment strategy: (i) the number of remote accesses represent the operations over state items stored in other surrogates, (ii) the number of state item migrations from one surrogate to another at the end of each state deployment evaluation, and (iii) the number of enclosed transactions in a single surrogate. A good strategy should maximize the last metric (privileging transactions involving a single surrogate), while minimizing the first and second. However, remote accesses are considered the highest latency inducing operation hence its minimization must be priority.

3 GRAPH-BASED GEO-AWARE STRATEGIES

Preliminary results [8] have shown that geo-aware policies can positively impact system performance in comparison with off-the-shelf approaches that either always or never migrate objects from one region to another.

So far, we have been researching geographically aware state deployment strategies based on graphs. Originally proposed to partition database items across storage nodes [3], graph partitioning is an interesting approach to distribute state components across different locations so that most of the transactions only need to access one. Geo-aware state deployment extends the graph partitioning problem with the additional requirement that the accessed location should be preferably in proximity of the transaction initiator.

One natural application of graph partitioning to geo-aware state deployment consists in mapping both application state items and surrogate locations to vertices, and add two sort of weighted edges [9]: (i) transaction edges connect pairs of state items used in the same transaction ; (ii) location edges link a location and a state item if this state item is used in a least one transaction initiated at this location.

The fundamental idea consists in periodically executing a graph partitioning algorithm on a graph capturing the relationship between data items and surrogates. State items are then proactively relocated to surrogate servers according to the outcome of the partitioning algorithm. In its basic form, the graph used for the partitioning contains all state items and all surrogates as vertices. The graph then records weighted edges between pairs of items used in the same transaction (counting the number of such transactions), and between a surrogate and the items contained in a transaction initiated at this surrogate (again counting such occurrences).

Unfortunately, in this approach, the graph partitioning approach is oblivious to the number of vertexes that change partition on each iteration. Spurious migrations penalize the application performance by increasing network traffic, load at the surrogates and consequently, transaction latency [9].

In an effort to reduce the number of state item migrations while still avoiding awareness of the application semantics by the graph partition approach, we are investigating the contribution and modeling of historical data to enrich this basic graph, and thus influence the partitioning algorithm. Historical data can be used *i*) to introduce some inertia on the graph partitioning algorithm, encouraging it to maintain the previous deployment; and *ii*) to select the subset of state items used as input to the graph partitioning algorithm. The goal is to understand if this information, which can be trivially extracted from transaction logs, can contribute to reduce the number of remote accesses and possibly the amount of state item migrations while increasing the number of local transactions, i.e. those that use exclusively state items hosted at the surrogate that is closer to the transaction initiator.

4 THE ROLE OF THE ORACLE IN GLOBAL TRANSACTIONS

A second research question to be answered focuses on determining the real impact and role of the Oracle in our system model, especially in face of global transactions. Expectations are that better state deployment strategies able to store *personal state* at the most convenient surrogate servers will benefit latency by maximizing the amount of local transactions obtained. However, it is also clear that a perfect state deployment may be unachievable and the system will inevitably have to deal with global transactions having either *geo-aware* or global state. This means that a trade-off needs to be established between involving and coordinating more surrogate servers in proximity to validate transactions, or instead relying on the Oracle (expected to be located in a distant datacenter in background) to solely validate and commit global transactions, while paying the negative latency overhead imposed by the physical distance to the surrogate servers.

5 RELATED WORK

The geo-aware state deployment problem shares with Content Delivery Networks (e.g. Akamai[1]) the goal of deploying data close to the clients. However, CDN deploys read-only replicas of the data and its biggest challenges are to cope with the limited bandwidth

[1] https://www.akamai.com/

between the replicas and the storage space. In contrast, geo-aware state deployment aims to define the ideal location of a single copy of the data, knowing that clients can edit and create new content. Therefore, the development of efficient geo-aware state deployment strategies leveraged on previous research results on different fields, namely database partitioning and on geo-replicated cloud storage.

5.1 Data Partitioning

Data partitioning was originally proposed to address scalability and performance requirements of database management systems. Horizontal and vertical partitioning were some of the earliest strategies to emerge [5]. In horizontal partitioning, the rows of each table are evenly distributed by the partitions. Vertical partitioning follows the same approach but distributes columns by partitions. The rationale behind these approaches is to make the amount of data more manageable while aggregating related data (sets of columns or rows) in the same node.

Graph partitioning algorithms are a particular approach as they consider the implicit association created between state items by transactions. The objective of the graph-based approach reported in [3] is to partition database items across storage nodes so that most transactions only need to access one node thus centralizing locking and consistency. The approach is very similar to the one used in this paper. It represents data items as vertexes and edges to associated those sharing at least one transaction. Vertex weight is used to represent the absolute access frequency of an item, while edge weight represents the number of transactions that accessed both state items. Mapping of state items in nodes is performed by a graph partitioning algorithm such as k-way min/cut [7].The graph modeling approach and the partitioning algorithm show that the focus of [3] is to improve load-balancing. This is in contrast with our model as we assume that partition imbalances are expected to reduce latency, as long as they represent the effective utilization patterns of the data.

Other mechanisms have been proposed for example in [4, 10, 11]. These share our goal of minimizing the number of distributed transactions. However, they assume that all partitions are geographically collocated, thus ignoring the negative impact of distance on latency.

5.2 Geo-replicated Cloud Storage

In contrast with the geo-oblivious approaches pursued by the databases community, cloud storages bring the notion of physical distance to data partitioning. Solutions at different scales have been proposed. At a server-level scale, geo-replicated cloud storages aim to reduce latency and improve load balancing. At a datacenter scale, the goal is to provide fault-tolerance for datacenter level outages.

5.2.1 Server-level Scale. AdaptCache [1] proposes a cooperative and integrated cache framework for web enterprise systems where application servers cooperatively share their caches. Similar to our approach, AdaptCache also has an "oracle" which dynamically evaluates and manages state items placement. However, it goes a step further and distributes requests across servers to achieve load balancing and simplify consistency management. Request management is facilitated by the collocation of servers, which is in contrast with our system model.

SPANStore [13] is a geo-replicated key-value store that unifies in a single framework storage at multiple datacenters. The goal is to reduce the operation cost, by taking advantage of pricing discrepancies. SPANStore ignores the location of the clients or the correlation of the data items.

The introduction of geographical constraints in graph partitioning algorithms by using special "surrogate" vertexes was originally proposed by the authors in [9]. Although promising, the advantages of such approach could only been observed in the limited number of scenarios where the large number of state items migrations would not contribute for an increase in latency and traffic at the surrogates.

5.2.2 Datacenter Scale. A possible approach to handle datacenter-level outages is using distributed transactional SQL databases. In CockroachDB[2]. Replica location is decided automatically, based on user configured criteria such as the types of failures to tolerate and distinct locations. Unfortunately, CockroachDB cannot encompass any concerns related with client access latency neither the dynamism associated to mobile applications.

Regardless of the data partitioning strategy used, multi partition transactions are by themselves a complex subject. These tend to negatively impact system performance and several solutions try to mitigate their effect in the system. The goal of DS-SMR [6] is to design a dynamic and scalable replicated state machine to exploit workload locality. Like in our approach, data partitioning is managed by an oracle. State reconfiguration follows a simple approach that transfers all required data items to a single partition prior to initiating the transaction. However, it neglects the optimal deployment of the data, introducing non-negligible latency with state item migrations during transaction execution.

Workload analysis and location manipulation can also be found in the geo-replicated storage system SDUR [12]. Considering that local transactions take less latency to be validated and committed than distributed transactions, this paper claims that latency may be hampered in the presence of mixed workloads where a local transaction delivered after a global transaction will experience a longer delay. The paper focuses on strategies to improve system performance by reordering transaction to give priority to transactions using a single partition and deferring global transactions. On the contrary, we aim at reducing the overall number of global transactions.

ACKNOWLEDGMENT

Work described in this paper was partially supported by Fundação para a Ciência e Tecnologia, Portugal, under the Individual Doctoral Grant SFRH/BD/120631/2016.

REFERENCES

[1] Omar Asad and Bettina Kemme. 2016. AdaptCache: Adaptive Data Partitioning and Migration for Distributed Object Caches. In *Procs. of the 17th Int'l Middleware Conf. (Middleware '16)*. ACM, Article 7, 13 pages. https://doi.org/10.1145/2988336.2988343

[2] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. *Fog Computing: A Platform for Internet of Things and Analytics*. Springer International Publishing, 169–186.

[2] https://www.cockroachlabs.com/

[3] Carlo Curino, Evan Jones, Yang Zhang, and Sam Madden. 2010. Schism: A Workload-driven Approach to Database Replication and Partitioning. *Proc. VLDB Endow.* 3, 1-2 (Sept. 2010), 48–57. https://doi.org/10.14778/1920841.1920853

[4] Sameh Elnikety, Steven Dropsho, and Willy Zwaenepoel. 2007. Tashkent+: Memory-aware Load Balancing and Update Filtering in Replicated Databases. In *Procs. of the 2nd ACM SIGOPS/EuroSys European Conf. on Computer Systems 2007 (EuroSys '07)*. ACM, 399–412. https://doi.org/10.1145/1272996.1273037

[5] L. Gruenwald and M. H. Eich. 1990. Choosing the best storage technique for a main memory database system. In *Procs. of the 5th Jerusalem Conf. on Next Decade in Information Technology*. 1–10. https://doi.org/10.1109/JCIT.1990.128263

[6] L. L. Hoang, C. E. Bezerra, and F. Pedone. 2016. Dynamic Scalable State Machine Replication. In *46th IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN)*.

[7] George Karypis and Vipin Kumar. 1998. Multilevelk-way Partitioning Scheme for Irregular Graphs. *J. Parallel Distrib. Comput.* 48, 1 (Jan. 1998), 96–129. https://doi.org/10.1006/jpdc.1997.1404

[8] Diogo Lima, Hugo Miranda, and François Taïani. 2016. Partial Replication Policies for Dynamic Distributed Transactional Memory in Edge Clouds. In *Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets (MECC '16)*. ACM, New York, NY, USA, Article 2, 6 pages. https://doi.org/10.1145/3017116.3022872

[9] Diogo Lima, Hugo Miranda, and François Taïani. 2017. Can Graphs Solve the Geo-aware State Deployment Problem?. In *INFORUM 2017 - Atas do 9 Simpósio de Informática*, João Paulo Barraca, Helena Rodrigues, António Teixeira, and José Maria Fernandes (Eds.). UA Editora, 221–232.

[10] Vivek S. Pai, Mohit Aron, Gaurov Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, and Erich Nahum. 1998. Locality-aware Request Distribution in Cluster-based Network Servers. In *Procs. of the 8th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS VIII)*. ACM, 205–216. https://doi.org/10.1145/291069.291048

[11] Andrew Pavlo, Carlo Curino, and Stanley Zdonik. 2012. Skew-aware Automatic Database Partitioning in Shared-nothing, Parallel OLTP Systems. In *Procs. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD '12)*. ACM, 61–72. https://doi.org/10.1145/2213836.2213844

[12] Daniele Sciascia and Fernando Pedone. 2014. Geo-Replicated Storage with Scalable Deferred Update Replication. In *Procs. of the 2014 IEEE 33rd Int'l Symposium on Reliable Distributed Systems Workshops (SRDSW '14)*. IEEE Computer Society, 26–29. https://doi.org/10.1109/SRDSW.2014.21

[13] Zhe Wu, Michael Butkiewicz, Dorian Perkins, Ethan Katz-Bassett, and Harsha V. Madhyastha. 2013. SPANStore: Cost-effective Geo-replicated Storage Spanning Multiple Cloud Services. In *Procs. of the 24th ACM Symposium on Operating Systems Principles (SOSP '13)*. ACM, 292–308. https://doi.org/10.1145/2517349.2522730