Dynamic Replica Management in the Service Grid

Byoung-Dai Lee and Jon B. Weissman
Department of Computer Science and Engineering,
University of Minnesota, Twin Cities
{blee, jon}@cs.umn.edu

Abstract

As the Internet is evolving away from providing simple connectivity towards providing more sophisticated services, it is difficult to provide efficient delivery of high-demand services to end-users due to the dynamic sharing of the network and connected servers. To address this problem, we propose the service grid architecture that incorporates dynamic replication and deletion of services

1. Introduction

The Service Grid [2] is an infrastructure for generic service delivery that has been designed to address several bottlenecks of the current Internet. Most notably, lack of reliability, transparency, and efficiency in service delivery. Our solution is to perform dynamic replication and deletion of services in response to user demand and system outages. Replication is the process by which one or more copies of a service are made. Although the idea of replication is not new, it presents several interesting challenges in the context of network services.

2. Architecture

Our architecture consists of three core components: Replication Manager (RM), Group Manager (GM) and Site Manager (SM). RM is the decision-maker for global replica selection, creation and deletion, and tracks the location and state of all the replicas.

The GM maintains a cache of local replicas allocated to it by the RM over time. This replica pool is available to the clients within the GM.

Every site in the service grid runs SM, whose primary job is to interact with GM to determine the network performance between replicas and client groups.

Each replica maintains a list of GMs that are currently using the replica and reports its load status to them periodically so that the GMs can have up-to-date information on the replica status. To reduce unnecessary network resource consumption, replicas can dynamically change window size for status report. Among all GMs that are sharing a replica, a primary GM is responsible for propagating the information to the RM so that the RM will also have an up-to-date global view of the system. With this protocol, the GM offloads much of the traffic that would otherwise reach the RM, promoting scalability. In addition to information collection and replica selection, the GM is also responsible for decision-making about when to acquire replicas and when to release replicas based on perceived performance and replica utilization.

Replica creation and deletion are initiated by the GMs in a distributed fashion. When the RM receives a replica acquisition request from the GMs, it decides whether to return an existing replica or to create a new replica based on the replica utilization by other groups. When a GM sends a replica release request and there is no other GM that is using the released replica, the RM put that replica in an idle replica pool.

3. Replica Management in GM

The GM runs three algorithms for replica management: replica selection, replica acquisition, and replica release. The challenge in designing algorithms for replica acquisition and replica release is that these algorithms should combine the goal of providing good performance to end-users with the goal of utilizing the system resources efficiently.

3.1 Replica Selection

Replica selection is the process by which the GM selects a replica among its local cache of replicas that is predicted to provide the best performance for the requesting client. Replica selection in the GM is based on response time prediction. With up-to-date state information about replicas in its local cache, the GM can predict the response time of the service accurately.

Response time (T_{resp}) is consists of four components: service time (T_s), waiting time (T_w), communication time (T_c) and overhead (T_o) and can be formulated as $T_{resp} = T_s + T_w + T_c + T_o$. The GM will select a replica that achieves a predicted minimum T_{resp} from its cache pool.

3.2 Replica Acquisition

Periodically, the GM computes the average response time (T_{resp}) for all local replicas over a recent time window. Once the average response time of each replica is computed, the GM next applies the replica acquisition algorithm to decide if it needs to acquire an additional replica from the RM.

The algorithm is based on a response time threshold ($T_{threshold}$) and two parameters, P and Q ($Q>P$), that control the degree of aggressiveness of replication. Each GM is free to select these parameters differently. With P, the GM can avoid acquiring unnecessary replicas due to temporary network congestion or transient increase of client demand. P requires that response time be monotonically increasing above the threshold for P consecutive time epochs. If the test on P fails, we apply the Q test which is less restrictive. Q requires that the response time be simply above the threshold for Q consecutive time epochs. Smaller values of P and Q lead to more aggressive replication. P allows an immediate response to rapidly growing demand, while Q permits some performance fluctuation and is more conservative.

3.3. Replica Release

If the GM caches more replicas than it needs to meet its current threshold, some replicas may be idle and system resources would in turn be wasted.

As a utilization metric, the number of requests that the replica has served within this group over the time window is used. This is the local utilization. The replica may be actively used by other groups. As in replica acquisition, the GM should not respond to a transient decrease in client demand. We apply the same principle within the release algorithm as in replica acquisition. The difference is that the algorithm should be applied against each local replica. Release does not mean that the replica is deleted. Deletion is a decision that is ultimately up to the RM, analogous to replica creation.

4. Replica Management in RM

The RM must perform the following replica management tasks: replica acquisition and replica release. Replica acquisition first examines the pool of available replicas not currently used by the group making the request. The RM determines whether an existing replica can provide predicted performance below the group's threshold while not compromising the performing of other groups sharing the replica. If these criteria cannot be met, the RM will create a new replica. Replica release simply indicates to the RM that the replica has been removed from the GMs cache. The RM notifies the replica of this change which allows the replica to eliminate any status updates to this GM saving network resources. In addition, the RM periodically checks the status of idle replicas (replicas released by all GMs). The RM is configured to maintain a minimum pool of idle replicas in the system. When this limit is exceeded, the RM will delete the replica that has been idle for the longest period of time.

5. Conclusion

We described a new architecture for the efficient delivery of high-demand network services, the Service Grid. To achieve scalable, reliable and adaptive performance, the Service Grid performs dynamic service replication and deletion in response to changing client demand. It implements an algorithmic framework for dynamic replica management that controls the degree of aggressiveness in creating and removing replicas.

A Service Grid prototype was built using the Legion system [1] and preliminary results for a compute-intensive service indicate that dynamic replica management can be done to meet end-user performance goals at a much lower cost than fully static replication.

6. References

[1] A.S. Grimshaw and W.A. Wulf, "The Legion Vision of a Worldwide Virtual Computer", *Communications of the ACM*, Vol. 40(1), 1997

[2] Jon B. Weissman and Byoung-Dai Lee, "The Service Grid Supporting Scalable Heterogeneous Services in Wide-Area Networks", *Proceedings of symposium on Applications and the Internet*, 2001