Dynamically Provisioning Distributed Systems to Meet Target Levels of Performance, Availability, and Data Quality

Amin Vahdat

Department of Computer Science
Duke University

1 Introduction

Increasingly, critical compute services are being delivered across large-scale networks. However, there are no mechanisms for provisioning distributed network resources (such as compute power, network bandwidth, storage, etc.) to meet target levels of performance, availability, or data quality. One approach to achieving application-specific targets is to over-provision distributed resources. Unfortunately, this imposes significant resource waste as all applications must plan for peak levels of demand. Further, there is no graceful way to degrade as network or application characteristics change, the system comes under attack, individual components of the distributed system fail, etc.

Thus, an important current challenge to distributed computing is automatically provisioning large-scale distributed network resources (hundreds of thousands of cooperating sites across the wide area, each composed of thousands of individual machines) to meet the requirements of a broad range of competing services. Important questions include determining per-application resource allocation, replica placement, and request routing. A common theme behind much of this work involves achieving target levels of application "utility" with the minimal global "cost".

As discussed earlier, utility may be measured in any number of application-specific ways, including performance, availability, and data quality. Performance typically lends itself to a straightforward application-specific definition, such as throughput. Availability is more difficult to quantify [9]. Traditional measures of availability for a network service typically focus on service uptime. However, in a system the size of the Internet, it is highly likely that a failure in the middle of the network will prevent a subset of clients from accessing an otherwise functional service. Along these lines, a service may be functioning correctly, but under such heavy load that it is unable to deliver satisfactory performance for a portion of the request stream. An appropriate definition of availability must account for all such considerations. Thus, we view appropriate availability metrics, benchmarks, and evaluation methodologies as an important research challenge in isolation. Data quality may be measured in a number of ways, including the percentage of the database represented by a given query [3, 5] or the consistency level of the returned data [8]. Overall system utility may take the form of more complicated convolutions of individual underlying metrics, for instance, maximizing performance while maintaining some minimal level of data quality. In general, delivering higher utility requires additional system resources, for instance, more CPUs or network bandwidth to

deliver higher throughput or additional service replicas to improve availability. Thus, an important research goal is to determine techniques for maximizing utility for minimal cost.

At a high level, we are pursuing key questions in the coordinated use of shared computing and storage resources across the Internet to systematically improve the performance and availability of Internet services. We envision a collection of server sites (e.g., data centers) distributed across the Internet and managed in a coordinated fashion as a shared physical infrastructure for decentralized Internet applications. Rather than forcing individual applications to reimplement significant functionality and to redundantly administer distributed service resources, an *overlay peer utility service* [1], Opus, dynamically allocates resources among competing applications. We see four key challenges to realizing this vision:

Infrastructures for decentralized, replicated services. A fundamental premise of the architecture is that application structure allows a dynamic mapping of functions and data onto physical resources, including dynamic replication. One key question involves whether it is possible for a shared infrastructure to simultaneously support the requirements of a broad range of network services, including replicated web services, application-layer multicast, and storage management.

Overlay construction and service quality. The number and placement of server sites affects multiple dimensions of service quality in a complex way. While allocation of additional server sites typically yields better performance and availability, in some cases additional resources can actually reduce overall performance and availability [9]. For a replicated service, the need to propagate updates between replicas imposes new network demands and may compromise availability for applications with strong consistency requirements. Further, the availability and performance of a candidate replica configuration depends in part on the overlay topology connecting the replicas. To configure overlays automatically, the system must efficiently generate valid candidate topologies that meet service quality goals while balancing network performance and cost. Finally, algorithms for building target overlay topologies must be scalable and adaptive to changing network characteristics.

SLAs for automated provisioning and configuration. One key issue is expressing application targets for service quality and using these targets to balance competing demands on the system. We are investigating the utility of expressing such goals in economic terms as continuous *utility functions* that specify values ("revenue") associated with various levels of service volume and quality for each customer. These utility functions form the basis of Service Level Agreements incorporating multiple dimensions of service quality, including availability and consistency quality as well as performance. Our recent work [2, 4] shows that utility functions can capture dynamic tradeoffs of service quality and cost; a key challenge of our future work is to extend this approach to balance competing dimensions of service quality in an application-directed way.

Scalable resource mapping. The core of the Opus architecture is a set of resource allocation algorithms to map services and their workloads onto the physical server resource pool. Based on a prioritization of hosted applications, individual utility nodes must make local decisions, based on potentially outdated information, to approximate the global good. In the wide area, highly variable network conditions make replica

placement as important a decision as the overall percentage of resources allocated to a particular application. In fact, fewer well-placed replicas are likely to outperform more poorly-placed replicas, especially when considering consistency requirements and per-application scalability issues.

As described in more detail below, Opus uses utility functions to evaluate candidate resource allotments by balancing their costs and benefits, and allocates resources to maximize global value. The resulting optimization problems are practical and efficient given reasonable constraints on the form of the utility functions [6]. Moreover, the utility functions provide a common measure of value that can serve as a foundation of a market-based approach to dynamic resource allocation for large-scale Internet services.

2 An Overlay Peer Utility Service

We are pursuing our agenda of dynamically placing functionality at appropriate points in the network in the context of Opus [1], an overlay peer utility service. As discussed earlier, individual applications specify their performance and availability requirements to Opus. Based on this information, Opus maps applications to individual nodes across the wide area. The initial mapping of applications to available resources is only a starting point. Based on observed access patterns to individual applications, Opus dynamically reallocates global resources to match application requirements. For example, if many accesses are observed for an application in a given network region, Opus may reallocate additional resources close to that location. One key challenge to achieving this model is determining the relative utility of a given candidate configuration. That is, for each available unit of resource, we must be able to *predict* how much any given application would benefit from that resource. Existing work in resource allocation in clusters [2] and replica placement for availability [10] indicate that this can be done efficiently in a variety of cases.

Many individual components of Opus require information on dynamically changing system characteristics. Opus employs a global *service overlay* to interconnect all available service nodes and to maintain soft state about the current mapping of utility nodes to hosted applications (group membership). The service overlay is key to many individual system components, such as routing requests from individual clients to appropriate replicas, and performing resource allocation among competing applications. Individual services running on Opus employ *per-application overlays* to disseminate their own service data and metadata among individual replica sites.

Clearly, a primary concern is ensuring the scalability and reliability of the service overlay. In an overlay with n nodes, maintaining global knowledge requires $O(n^2)$ network probing overhead and $O(n^2)$ global storage requirements. Such overhead quickly becomes intractable beyond a few dozen nodes. Peer-to-peer systems can reduce this overhead to approximately $O(\lg n)$ but are unable to provide any information about global system state, even if approximate. Opus addresses scalability issues through the aggressive use of hierarchy, aggregation, and approximation in creating and maintaining scalable overlay structures. Opus then determines the proper level of hierarchy and aggregation (along with the corresponding degradation of resolution of global system state) necessary to achieve the target network overhead.

A primary use of the service overlay is to aid in the construction of the *per-application overlays*. Once resource allocation and replica placement decisions are made, Opus must create an interconnection graph for propagating data (events, stock quotes, replica updates) among application nodes. It is important to match an overlay topology to the characteristics of individual applications, for instance, whether an application is bandwidth, latency, or loss sensitive. In general, adding more edges to an application overlay results in improved "performance", with a corresponding increase in consumed network resources. Our economic model for resource allocation helps determine the proper balance between performance and cost on a per-application basis.

Security is an important consideration for any general-purpose utility. Opus allocates resources to applications at the granularity of individual nodes, eliminating a subset of the security and protection issues associated with simultaneously hosting multiple applications in a utility model. We believe that a cost model for consumed node and network resources will motivate application developers to deploy efficient software for a given demand level. Initially, we consider applications that are self-contained within a single node. Longer term, however, a utility service must support multi-tiered applications (e.g., a web server that must access a database server or a storage tier). In such a context, one interesting issue is determining the proper ratio of allocation among application tiers, e.g., the number of storage nodes needed to support a particular application server.

3 Future Challenges

There are many open question associated with realizing the above vision. To date however, we have made progress on a number of fronts as briefly summarized below.

- *Continuous consistency:* We have defined a continuous consistency model that captures the semantic space between optimistic and strong consistency [8]. Applications can dynamically trade relaxed consistency for increased performance and availability, defining another dimension along which Opus can allocate resources while balancing performance, availability, and consistency. In this space, we have also shown how to optimally place (both in terms of number and location) replicas to achieve maximum availability for a given workload and faultload (network failure characteristics) [10].
- *Cluster-based resource allocation policies:* In Muse [2], we have shown how a centralized load balancing switch can measure per-application throughput levels and service level agreements to determine how to allocate resources to maximize throughput (more generally "revenue"). One key challenge to extending this work to the wide area is to make similar resource allocation decisions in a decentralized manner based on incomplete and potentially stale information and to incorporate other metrics beyond throughput, such as availability or consistency, into our calculations.
- *Adaptive Cost, Delay Constrained Overlays:* As discussed above, application overlays form a key components of the service architecture. We are exploring the space between small-scale overlays that use global knowledge (making them unscalable)

to construct and maintain appropriate topologies and large-scale peer-to-peer systems that randomly distribute functionality across the network to achieve scalability (while sacrificing control over exact performance and reliability of the resulting topology). We have designed and built a scalable overlay construction technique that eliminates any centralized control or global knowledge [7]. Our performance evaluation indicates that we are able to export a flexible knob that allows application developers to quickly converge to structures that deliver the performance of shortest path trees (with associated high cost) at one extreme and the low resource utilization of a minimum cost spanning tree at the other extreme (with associated low performance). Applications are able to specify where on this spectrum they wish to reside based on current network and application characteristics. The structure is also adaptive to changing network conditions, quickly adapting to achieve target levels of performance or cost with low per-node state and network overhead.

The key challenge is to pull together the individual pieces described above and to weave in additional best practices to build and deploy a distributed service that fairly allocates global resources among competing applications while dynamically adapting to changing client access patterns and network characteristics.

References

1. Rebecca Braynard, Dejan Kostić, Adolfo Rodriguez, Jeffrey Chase, and Amin Vahdat. Opus: an Overlay Peer Utility Service. In *Proceedings of the 5th International Conference on Open Architectures and Network Programming (OPENARCH)*, June 2002.
2. Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat, and Ronald P. Doyle. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP)*, October 2001.
3. Armando Fox and Eric Brewer. Harvest, Yield, and Scalable Tolerant Systems. In *Proceedings of HotOS-VII*, March 1999.
4. Yun Fu and Amin Vahdat. Service Level Agreement Based Distributed Resource Allocation for Streaming Hosting Systems. In *Proceedings of the Seventh International Workshop on Web Caching and Content Distribution (WCW)*, August 2002.
5. Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online Aggregation. In *Proc. ACM-SIGMOD International Conference on Management of Data*, 1997.
6. Toshihide Ibaraki and Naoki Katoh, editors. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, Cambridge, MA, 1988.
7. Dejan Kostić, Adolfo Rodriguez, and Amin Vahdat. The Best of Both Worlds: Adaptivity in Two-Metric Overlays. Technical report, Duke University, May 2002. `http://www.cs.duke.edu/~vahdat/ps/acdc-full.pdf`.
8. Haifeng Yu and Amin Vahdat. Design and Evaluation of a Continuous Consistency Model for Replicated Services. In *Proceedings of Operating Systems Design and Implementation (OSDI)*, October 2000.
9. Haifeng Yu and Amin Vahdat. The Costs and Limits of Availability for Replicated Services. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.
10. Haifeng Yu and Amin Vahdat. Minimal Replication Cost for Availability. In *Proceedings of the ACM Principles of Distributed Computing*, July 2002.