Locality Aware Mechanisms for Large-scale Networks

Ben Y. Zhao, Anthony D. Joseph, and John Kubiatowicz
Computer Science Division
University of California, Berkeley *
{ravenben, adj, kubi}@cs.berkeley.edu

Abstract

Recent advances in decentralized object location and routing (DOLR) systems provide an attractive platform for wide-area network applications. These systems, also referred to as distributed hashtables, provide routing and location algorithms which scale logarithmically with the size of the network. In this paper, we contend that many properties of these systems are not yet well understood in the context of deployed large-scale networks. Specifically, we identify the property of locality-awareness and show how locality-aware design mechanisms play a positive role in providing greater scalability in wide-area networks.

1 Introduction

As applications are scaled to wide-area networks, distributed middleware layers are increasingly used to abstract away complex issues faced by these applications. In particular, recent work on decentralized object location and routing (DOLR) systems ([6, 7, 9, 11]) combine location and routing in an overlay network layer, promising several important desirable properties such as decentralized operation, scalability, fault-tolerance, and load-balancing.

We contend in this paper, however, that many properties of these systems are not yet fully understood in the context of deployed large scale networks. We take a closer look at the issue of scalability, and examine design decisions which in combination may detract from scalable operation when deployed in the wide-area.

DOLRs offer scalability as the main advantage over existing services, such as Domain Name Service, web search engines, and especially Peer-to-Peer (P2P) file sharing technologies. These systems make tradeoffs between decentralized operation, efficient bandwidth usage, and deterministic operation. In comparing DOLRs to existing P2P file-sharing systems, (*e.g.*, as Gnutella, Morpheus, Napster, and Freenet), scalability is usually measured with respect to three key resources: CPU processing time, local storage, and network bandwidth. DOLRs use novel search algorithms based on coordinate or high-dimensionality routing to yield routing table storage and a number of hops between nodes that are both logarithmic to the size of the total network, while guaranteeing consistent operation.

When examining scalability, it is important to consider not just the routing algorithm, but the entire infrastructure. For wide-area deployment, DOLR algorithms require support mechanisms that could impact the scalability of the overall system. Mechanisms such as proximity routing, replication, soft state, etc., enhance the performance of these systems at the cost of limited overhead. Unfortunately, the effects of these mechanisms as the system scales are not well understood.

This paper examines properties of several recent research projects, including Content-Addressable Networks (CAN) [6], Pastry [7], Chord [9], and Tapestry [11], that use DOLRs to build scalable wide-area network infrastructures and applications. These four efforts all provide wide-area object location by routing messages to objects named with an associated key or unique identifier.

There are two fundamental mechanisms that can be used to differentiate between these projects and their scalability characteristics. The first is the core algorithm used by each system to build and manage local routing tables. The second is the caching or replication scheme used by each system on objects or location pointers to provide a highly available location service.

To study the performance of these systems in the absence of actual large-scale deployments, we propose an analytical approach based on examining their "locality awareness." In the following sections, we examine the locality awareness of several commonly used system techniques, and show how their inclusion in existing DOLRs have impacted overall system scalability.

2 Locality Awareness

We have identified a key metric, *locality awareness*, that helps define the impact of design techniques on scalability. We define locality awareness as the ability to exploit local resources over remote ones whenever possible. Here, "local" and "remote" are defined in the context of

*We gratefully acknowledge support for this project by NSF funds ANI-9985129, ANI-9985250 and DARPA fund N66001-99-2-8913.

heterogeneous network links with varying latency, bandwidth, and distance between the source (requesting) and destination nodes. Intuitively, locality awareness is also the property that enables a system to limit the impact of local operations on wide-area performance, both during regular operation and under fault conditions.

In this section, we examine commonly used overlay network mechanisms, discuss whether they are locality aware, and show how their usage can impact the overall wide-area scalability. Specifically, we examine mechanisms used to optimize key operations in distributed systems, including proximity routing, distribution of responsibility, data replication, and soft-state.

2.1 Proximity in Overlay Hop Selection

Routing overhead is a key performance metric for distributed object location and routing systems and other overlay infrastructures. When an overlay network increases in number of nodes, there are two likely consequences: the number of wide-area physical network hops traversed by any single logical overlay hop increases, and the number of multiple overlay hops between two nodes also increases. The former means that for a given overlay path traversing N overlay hops, end to end latency and bandwidth usage both increase. The latter increase in number of expected overlay hops per path further increases both latency and bandwidth usage. Along with an increase in the number of nodes, the network will scale in reach, implying there will likely be a corresponding increase in the average number of requests per time on the overlay network. Considered as a whole, these consequences can easily overwhelm a network's resources and significantly degrade performance.

However, an important counterbalance to this problem is the increase in the interconnections between nodes. The various DOLR projects all use a selection mechanism to choose between paths, with the simplest choice being random choice [6, 9]. The primary alternative to random selection of an overlay path is to consider proximity in selecting neighbor links. The challenge with this approach is constructing a selection list of nearby next hop neighbors using only locally available information about the physical network. This *nearest neighbor selection* problem has recently been examined in [3, 4], and two of the DOLR projects use some form of proximity metrics [7, 11]. The tradeoff is that building routing tables to support proximity routing is a difficult task in a distributed system, resulting in more complex node insertion algorithms.

To better understand the impact of proximity routing[1], consider a node that routes a message to a nearby resource using Tapestry or a non-proximity routing infrastructure (NPRI). Using a NPRI, a message routed to a neighboring node may travel an arbitrarily long distance across multiple physical networks before it arrives at the neighbor. With Tapestry, however, a message to a local resource never leaves the local network, because each Tapestry node keeps nearby nodes in its routing table. If node X chooses to route a message to destination Z via some node Y, then node Y must satisfy the routing table requirement that there is no node closer to X that satisfies the prefix matching routing requirement of Tapestry. The destination node Z always matches itself for all prefix lengths. Therefore, unless there is a wide-area node that is closer to node X than Z, node X will not route outside the local network because Z would be a closer neighbor than any wide-area node.

We also note that proximity routing can be achieved incrementally. The Chord [9] system uses a run-time heuristic that gradually modifies its finger table to point to nodes closer in network distance. In a stable wide-area system, this can provide significant performance benefits as nodes slowly optimize their finger tables over time.

There are two additional approaches that further leverage node differentiation for better routing and location performance. One approach can be found in Brocade [10], which utilizes a secondary overlay network to direct wide-area traffic. Brocade applies locality in routing at the level of Autonomous Systems (AS) to avoid routing messages through unnecessary AS domains. Instead, Brocade routes messages as directly as possible from one AS domain to another. This approach has been shown to significantly reduce the number of wide-area (*i.e.* high latency) hops traversed per overlay route, and substantially reduce bandwidth usage in the wide area, two benefits both of which improve network scalability.

A method to leverage network link heterogeneity for improved performance in the local area is to add bias or "weight" factors to be used when selecting overlay hops. The bias factors enable more robust or resource plentiful nodes to be favored over nodes with average resources.

We observe that in using these proximity approaches, there is a tradeoff between routing performance and the robustness gained from random selection of hop links. Previous studies have shown that a randomly organized network can provide unexpected robustness in the face of targeted attacks while more structured networks are more resilient to random failures. [1]. We believe this tradeoff warrants further study, particularly into the fault-tolerant impact of reducing random hops across the network. Overall, proximity metrics and differentiation between local and distant network links are excellent examples of how locality-aware mechanisms can improve system scalability.

[1] Note that we assume that the latency to wide-area nodes is greater than to nodes in the local network.

2.2 Distribution of Responsibility

It is well understood that in a large scale system, any single point of responsibility will become a centralized point of congestion and failure. Most distributed systems avoid centralization by using intelligent replication and data partitioning techniques. We observe that replication and partitioning for load-balancing alone are insufficient to maintain scalability in a wide-area system. To minimize network distance traversed by requests, we must associate client nodes with their closest server or object replicas.

Two examples of locality-aware replication and partitioning can be found in the Scribe [8] and Bayeux [12] application-level multicast systems. In both systems, the challenge is to distribute responsibility for handling join and leave requests for a multicast group. In Bayeux, membership messages are handled by the root node of the multicast tree. Using a single root node alone is not scalable, so Bayeux supports root replication by treating root nodes as Tapestry objects. Bayeux leverages Tapestry's object location mechanism, which supports location of object replicas, by having listeners join by using Tapestry location. Since Tapestry routing locates the nearest copy of an object, members automatically self-organize around the closest root node replica. By tying the amount of replication to the number of members and using proximity-based routing, this approach minimizes network distance and reduces overall bandwidth in a scalable manner.

Scribe uses an alternative approach by allowing intermediate nodes in the dissemination tree to handle membership messages. Assuming clients sign up with a nearby branch of the multicast tree, Scribe membership messages need only travel the distance to a nearby parent for processing, reducing bandwidth and latency while contributing to greater system scalability.

Once again, we see how scalability can be aided by the appropriate choice of routing mechanism. In this instance, by dividing the amount of work performed by overlay nodes, latency and bandwidth can be conserved in a scalable manner.

2.3 Data Replication

Given the scale of current wide-area network applications, service and object location are becoming key infrastructure services. The use of caching and replication has proven to be an effective technique for improving locality, and has resulted in substantial performance improvements and increased scalability.

We examine the design of Tapestry, Pastry, and CAN to show different usages of this technique. These three systems provide similar functionality, but use replication differently in terms of the number of replicas, their placement, and when they are placed. In Tapestry, clients searching for an object may find any one of $\log N$ cached object pointers, with most copies located near or close to the object. This approach provides the benefit that clients near to the object quickly locate it without searching networks that are farther away.

Pastry replicates objects themselves, and distributes a small number of replicas randomly in the network. Since Pastry replicas are not placed in a locality optimizing manner, there is no correlation between the location of the client and the original object. Thus, any relationships between objects and nearby clients are lost when the replicas are distributed.

CAN takes a different approach by replicating copies of location information towards sources of high query traffic. Copies of location pointers are propagated towards the source of high queries. While this type of on-demand caching can be highly effective, it is a run-time heuristic. Therefore, there can be significant initial delay before the mechanism reaches its maximum effectiveness.

In comparing these systems, we observe that the effectiveness of replication depends on the number of replicas and placement algorithm. However, the cost of maintaining consistency must also be considered as a factor in system scalability. For instance, Tapestry refreshes its cached location pointers using a soft-state mechanism that incurs a bandwidth overhead, while CAN avoids the consistency issue by assuming that objects are immutable.

2.4 Soft-state

Soft-state is a well-known design principle that provides a simple information propagation mechanism with graceful data recovery after failures. The technique was first developed for Internet Group Management Protocol [2] and further refined for the Session Announcement Protocol [5].

Rather than relying on explicit (hard state) algorithms to recover data after faults, soft state advocates the use of periodic data broadcasts by an information source to interested listeners. Listeners learn the information they need by listening to the periodic broadcasts. Unlike hard state, which requires a separate fault recovery protocol, soft state uses the same broadcast mechanism to recover after a failure. A listener automatically recovers the data by simply listening for a periodic broadcast. By tuning the broadcast period, the time to recover from a failure can be balanced against the bandwidth overhead of repeated transmissions.

While soft-state is a powerful information dissemination and fault-recovery mechanism, it does not share the locality awareness property which facilitates scalable wide-area operation. Specifically, the soft-state model does not differentiate between distant and nearby listeners. As a result, a naive application of soft-state can substantially increase bandwidth utilization and lead to non-scalable operation.

For example, Tapestry uses soft-state for the DOLR maintenance protocol between neighbor nodes on the overlay. To verify that neighboring overlay nodes on outgoing links are still available and reachable, Tapestry sends periodic probe messages along each outgoing link. Depending on the choice of polling period, this mechanism permits a node to quickly detect and react to faults of its links to neighbors. Because entries in higher level Tapestry routing tables (*i.e.*, routing prefixes that are longer) generally point to more distant nodes, a naive application of this mechanism might result in wide-area probe floods. Instead, Tapestry scales the probe period to decrease in proportion to the level of the route entry. The net effect is that links between distant neighbors are probed with longer periods, reducing the bandwidth load across the network. There is a tradeoff, however, in terms of the time to detect a fault.

3 Discussion and Future Directions

In this paper, we have made several observations regarding scalability of distributed infrastructures using decentralized object location and routing. We introduce *locality awareness* as an important metric and use it to evaluate the use of four design techniques: proximity in route selection, distribution of responsibility, data replication and soft-state.

We have shown how each of these techniques can be combined with locality awareness to yield improvements in wide-area scalability and performance. Overall, we believe that the use of such locality aware mechanisms will help provide true scalability to wide-area systems. However, there are still many open questions regarding the interactions between these techniques. For example in Tapestry, efficient, correct locality aware route selection requires consistent, up-to-date routing tables. These tables are kept consistent using a soft-state protocol. Therefore, increasing the soft-state period to reduce bandwidth requirements can affect the ability to perform efficient routing. It is clear that that further exploration of the design space is necessary.

Furthermore, while systems such as Chord, CAN, Pastry and Tapestry show the potential for scalability, they are only network infrastructures. Thus, an equal focus must be placed on wide-area network applications. They too must use locality aware mechanisms in their own design in addition to leveraging the mechanisms provided by DOLRs.

References

[1] ALBERT, R., JEONG, H., AND BARABASI, A.-L. Error and attack tolerance of complex networks. *Nature 406* (July 2000), 378–382.

[2] DEERING, S. *Host Extensions for IP Multicasting*. SRI International, Menlo Park, CA, Aug 1989. RFC-1112.

[3] HILDRUM, K., KUBIATOWICZ, J. D., RAO, S., AND ZHAO, B. Y. Distributed object location in a dynamic network. In *Proceedings of SPAA* (Winnipeg, Canada, August 2002), ACM.

[4] KARGER, D., AND RUHL, M. Find nearest neighbors in growth-restricted metrics. In *ACM Symposium on Theory of Computing* (Montral, Canada, 2002).

[5] MAHER, M. P., AND PERKINS, C. Session Announcement Protocol: Version 2. IETF Internet Draft, November 1998. draft-ietf-mmusic-sap-v2-00.txt.

[6] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SCHENKER, S. A scalable content-addressable network. In *Proceedings of SIGCOMM* (August 2001).

[7] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM Middleware 2001* (November 2001).

[8] ROWSTRON, A., KERMARREC, A.-M., DRUSCHEL, P., AND CASTRO, M. SCRIBE: The design of a large-scale event notification infrastructure. In *Proceedings of NGC 2001* (November 2001).

[9] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM* (August 2001).

[10] ZHAO, B. Y., DUAN, Y., HUANG, L., JOSEPH, A., AND KUBIATOWICZ, J. Brocade: Landmark routing on overlay networks. In *Proceedings of 1st International Workshop on Peer-to-Peer Systems* (March 2002).

[11] ZHAO, B. Y., KUBIATOWICZ, J. D., AND JOSEPH, A. D. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, Computer Science Division, April 2001.

[12] ZHUANG, S. Q., ZHAO, B. Y., JOSEPH, A. D., KATZ, R. H., AND KUBIATOWICZ, J. D. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV* (June 2001).