Passive Network Performance Estimation for Large-Scale, Data-Intensive Computing

Jinoh Kim, *Member*, *IEEE*, Abhishek Chandra, *Member*, *IEEE*, and Jon B. Weissman, *Senior Member*, *IEEE*

Abstract—Distributed computing applications are increasingly utilizing distributed data sources. However, the unpredictable cost of data access in large-scale computing infrastructures can lead to severe performance bottlenecks. Providing predictability in data access is, thus, essential to accommodate the large set of newly emerging large-scale, data-intensive computing applications. In this regard, accurate estimation of network performance is crucial to meeting the performance goals of such applications. Passive estimation based on past measurements is attractive for its relatively small overhead compared to relying on explicit probing. In this paper, we take a *passive approach* for network performance estimation. Our approach is different from existing passive techniques that rely either on past direct measurements of pairs of nodes or on topological similarities. Instead, we exploit secondhand measurements collected by other nodes without any topological restrictions. In this paper, we present Overlay Passive Estimation of Network performance (OPEN), a scalable framework providing end-to-end network performance estimation based on secondhand measurements, and discuss how OPEN achieves cost-effective estimation in a large-scale infrastructure. Our extensive experimental results show that OPEN estimation can be applicable for replica and resource selections commonly used in distributed computing.

Index Terms—Network performance estimation, secondhand estimation, data-intensive computing, replica selection, resource selection.

1 Introduction

In the distributed computing domain, demands on data have significantly increased over the past few years, and importantly, such applications are increasingly utilizing *distributed* data sources. For example, projects such as LHC [1] in high energy physics and SkyServer [2] in astronomy produce petabytes of new data every year, and researchers over the world access the data, often distributed, for their own experiments. An example of such applications is data-intensive scientific workflows [3], [4]. For such data-intensive tasks, data access cost is a significant factor in their execution performance in addition to the computation cost. Hence, it is essential to consider data access cost in launching data-intensive computing applications.

Large-scale computing infrastructures such as grids [5], [6] and desktop grids [7], [8], [9] are attractive due to their scalability and cost effectiveness. However, the loosely coupled nature of many of these platforms often makes them unpredictable in their resource availability and performance, particularly in terms of data access. Despite their rich set of computational resources, the unpredictable nature of large-scale computing platforms makes it hard to deploy such data-intensive applications or limits the size of data access making them inefficient to deploy. Thus, providing predictability in data access is a vital requirement for enabling such data-intensive tasks on large-scale systems. The goal of this work is to successfully execute data-intensive computing applications on unpredictable but appealing large-scale systems.

A key requirement for achieving data access predictability is the ability to estimate network performance for data transfer, so that computation tasks can take advantage of the estimation in their deployment or data source selection. In other words, network performance estimation can provide a helpful guide to run data-intensive tasks in such unpredictable infrastructures having a high degree of variability in terms of data access.

Active probing can be an option for estimation, but is unscalable and expensive in using back-to-back measurement packets. Passive estimation is attractive for its relatively small overhead, and thus could be desirable for many networked applications that do not require an extremely high degree of accuracy such as that needed by network-level applications like network planning. For example, a substantial number of networked applications, such as Web server selection and peer selection for file sharing, rely on *ranking*. According to a peer-to-peer measurement study in [10], the second placed peer performance is only 73 percent of the best peer performance. This significant gap implies that some degree of estimation inaccuracy would be tolerable for such ranking-based applications. A potential problem of passive estimation is that it can suffer from estimation failure due to the unavailability of past measurements. This problem can be mitigated by sharing measurements among nodes; thus, a node can estimate performance even against a server it has never contacted. In previous work [11], [12], however, the sharing was restricted to specific underlying topologies such as a local network, limiting scalability. In this work, we

• *The authors are with the Department of Computer Science and Engineering, University of Minnesota, EECS Building, 200 Union Street SE, Minneapolis, MN 55455.*
E-mail: {jinohkim, chandra, jon}@cs.umn.edu.

Manuscript received 4 Mar. 2010; revised 26 July 2010; accepted 23 Sept. 2010; published online 8 Nov. 2010.
Recommended for acceptance by J. Zhang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2010-03-0143.
Digital Object Identifier no. 10.1109/TPDS.2010.201.

present a novel approach enabling nodes to utilize past measurement information with no reliance on topological similarities, so as to minimize blind spots in the system and to reduce uncertainty in data access.

To realize this goal, there are two important challenges. The first challenge is the *characterization* of a node in terms of its data access capability to enable it to utilize others' measurements for its own estimation. This characterization is key for topology-independent utilization of secondhand measurements. The other important challenge is how to facilitate local measurements to be globally available to other nodes in the system for system-wide sharing. Any server-based techniques for storing global information are limited by well-known problems of scalability and fault tolerance. At the other end of the spectrum is flooding-based dissemination, which while fully distributed, has high network overhead. In this paper, we present *Overlay Passive Estimation of Network performance (OPEN)*, a scalable framework for end-to-end network performance estimation. OPEN provides a correlation-based secondhand estimation with empirical node characterization (proposed in our previous work [13]) and *proactive dissemination* of measurements with limited overhead by diverse optimizations.

Our key contributions can be summarized as follows:

- We present the OPEN framework that performs passive network performance estimation based on secondhand downloading information. OPEN is highly scalable, fully distributed, and topologically neutral.
- To enable cost-effective information sharing, we introduce two optimization techniques in addition to a probabilistic approach, both of which separately disseminate measurement information based on its "criticality," i.e., how important it is to share in the system. We show that these optimization techniques dramatically reduce dissemination overhead without significant performance loss.
- We evaluate OPEN with two selection problems common in distributed computing, *resource selection* and *replica selection*. To emulate a large-scale system, we collected download traces for 10 months in PlanetLab [14]. Using data sizes contained in GridFTP workloads [15], the results show that OPEN consistently outperforms selection techniques based on statistical pairwise estimations as well as random and latency-based selections in diverse experimental settings.
- For extensive evaluation, we present additional experimental results for the OPEN framework with S^3 data traces from HP Lab [16] and live experimental results with Montage [17], a toolkit for astronomical research, conducted on PlanetLab.

2 Distributed Computing Model

We consider a large-scale infrastructure for distributed computing. The system consists of compute nodes that provide computational resources for executing application jobs, and data nodes that store data objects required for computation. In our context, data *objects* can be files, database records, or any other data representations. We assume that both compute nodes and data nodes are connected in an overlay structure without any assumption of centralized entities for scalability. We do not assume any specific type of organization for the overlay, but assume that the overlay provides basic data access functionalities including *search*, *store*, and *retrieve*. In addition, each node in the system can be a compute node, data node, or both.

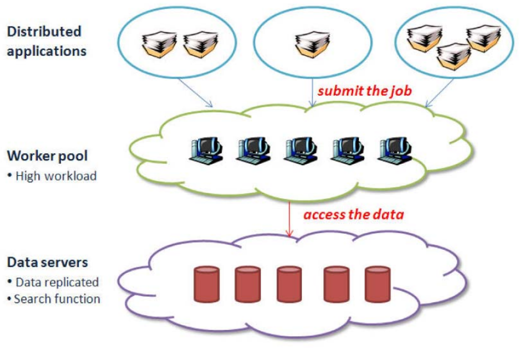

Fig. 1. Distributed computing model.

Fig. 1 illustrates the distributed computing model we consider. In the worker pool, computational resources are provided to run applications, while data nodes serve data objects accessed by the compute nodes. Distributed applications share the computational resources by submitting their jobs. In this model, we focus on two selection problems common in the distributed computing domain.

- *Replica selection*. The data object is replicated in multiple data nodes geographically dispersed, and the compute node needs to select a replica to download. The goal of this selection is to identify a replica server having minimal data access cost from the compute node.
- *Resource selection*. For job allocation, one or more compute nodes should be chosen from a list of computational resources. In this context, the job requires accessing data for task completion. The goal of this selection is to identify a compute node which can access the data server with minimal data access cost.

Hence, the *cost of data access* is a vital factor for both selection problems. While the computation cost is also important for overall task performance, our focus in this paper is on the communication cost. The question is *how to estimate the data access cost accurately and cheaply for the selection problems described above.* In the next section, we discuss estimation techniques based on past firsthand or secondhand measurement information and the benefits of secondhand measurement-based estimation.

3 Secondhand Estimation

We classify estimation techniques into several categories, based on the *degree of measurement sharing* for their estimation: *pair-level*, *domain-level*, and *system-wide*, in addition to non-sharing model, as summarized in Table 1.

Pair-level sharing only utilizes the direct (*firsthand*) measurements made by a *specific pair* of nodes for their network path estimation. Many statistical or time-series forecasting techniques, such as exponential moving average, belong to this class. Previous studies, such as [19], showed the

TABLE 1
Degree of Measurement Sharing

Degree	Non-sharing	Pair-level	Domain-level	System-wide
Approach	On-demand measurement	Statistical estimation Time-series forecast	Sharing in a LAN Sharing in a domain	Sharing in a system
System/ Technique	Pathchar [18] Packet pairs [20] bprobe/cprobe [22]	NWS [19] HB prediction [21]	SPAND [11] Webmapper [12]	OPEN

high accuracy of these techniques, but this class requires $O(n^2)$ measurements for estimation between all pairs.

In contrast, some estimation techniques enable nodes to utilize indirect (*secondhand*) measurements provided by other nodes for their own estimation. In domain-level sharing, past measurements in a domain (e.g., a single network or logical group of nodes) are shared between nodes belonging to the same domain. In SPAND [11], nodes in a single network share past measurements for Web server selection. Webmapper [12] shares passive measurements to select a Web server based on a logical group clustered by IP prefixes. By sharing the measurements in a domain, it is possible to estimate performance if any node in the domain has communicated with the server. Again, however, the sharing is restricted to the domain. In addition, the underlying assumption of existing techniques belonging to this class is that the nodes in a domain have closely similar characteristics in network access. If this is not the case, sharing measurements without considering node characteristics may cause inaccuracy in estimation.

Unlike the above two classes of sharing, system-wide sharing, we propose in this work, has no constraints on sharing measurements across the system. In other words, if any measurement against a server is available in the system, any other node can utilize that information for its own estimation to that server. Thus, it is possible to perform any-pair estimation with $O(n)$ measurements. Since it does not rely on topological similarities, node characterization is essential to utilize others' experiences. In addition, efficient sharing is also a key for this approach. Before discussing how OPEN realizes those key functions, we briefly describe the rationale for secondhand estimation in large-scale infrastructures.

3.1 Why Secondhand Estimation?

While firsthand-based estimation is likely to be more accurate than secondhand-based estimation, it is unlikely that all nodes will have firsthand observations to all servers (a worst case of $O(n^2)$ total measurements in the system if all workers are also data servers). Thus, there would be no estimates available for node pairs that lack direct measurements.

Fig. 2 compares the potential estimation failure rates of a pairwise firsthand estimation technique to that of a system-wide secondhand estimation approach (OPEN),[1] caused by a shortage of existing relevant measurements. This result is obtained through a trace-driven simulation, where we tested 100,000 estimations in two systems with sizes $n = 100$ and $n = 1{,}000$.[2] We assume there are no measurements at all in the beginning, and one random pairwise measurement is recorded at each time instant. As can be seen from the figure, the failure rates decrease as more measurements are added over time. In particular, we observe that OPEN dramatically diminishes the failure rates over time by using secondhand measurements for estimation. In contrast, the pairwise firsthand technique suffers from significant failure rates; the system with $n = 1{,}000$ has over 90 percent failure, even at the end of the simulation. This is because the probability that a node has any measurements to a server goes down as the system size grows. Given that a large-scale system can consist of tens of thousands nodes, the pairwise approach must ensure, in the worst case, that $O(n^2)$ measurements exist, which could require active probes to fill in the gaps due to insufficient firsthand observations; or it may suffer from high failure rates due to a lack of sufficient measurements. Therefore, the secondhand approach should be beneficial in terms of both scalability and overhead.

Again, domain-level sharing also performs secondhand estimation, but relies on topological similarity. Our intention is to design a framework to enable secondhand estimation without any topological constraints, as described in the next section.

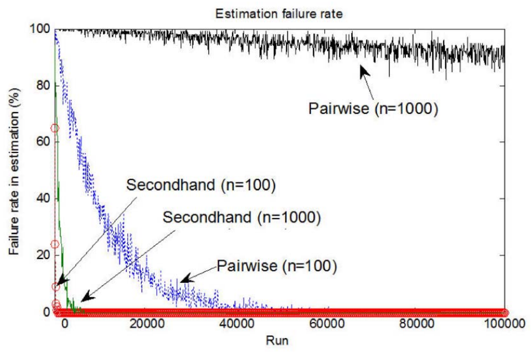

Fig. 2. Hit rate of relevant measurements.

4 The OPEN Framework

In our previous work [13], we proposed an estimation technique utilizing measurements observed in the neighbor nodes in an overlay network. While working well in a small setting, the estimation technique can suffer from a shortage of measurements in a large-scale environment, significantly degrading performance. In this work, we relax the constraint on measurement sharing. The OPEN framework we present in this paper enables system-wide sharing of measurements without both underlying and overlay topological restrictions. OPEN provides end-to-end network

1. OPEN uses dissemination of secondhand measurements, as will be discussed in more detail in the next section.
2. We will present details of the trace and our methodology in Section 5.1.

performance based on shared secondhand measurements, unlike the previous work providing the expected downloading time for a specific data object.

The OPEN framework consists of two core mechanisms: *secondhand estimation* of end-to-end network performance and *proactive dissemination* of observed measurements. Since secondhand estimation is based on our previous work in [13] and is not our main contribution in this paper, we omit here but Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.201, briefly provides our secondhand estimation method. In this work, we more focus on the dissemination part.

For secondhand estimation, it is necessary that secondhand measurements are globally visible, so that any other nodes can make their own estimations by referring to the shared measurements. OPEN facilitates dissemination of measurements in a proactive manner to minimize estimation failures due to shortage of relevant measures. OPEN mainly relies on *probabilistic dissemination* taking advantage of cost effectiveness and fault resilience. Many optimizations can be possible for probabilistic dissemination, as did in [23], [24], [25], [26], [27], our intention is not to make further optimization for dissemination protocols, but to provide insights for application-oriented optimizations for efficient dissemination, particularly for the OPEN framework. In this section, we introduce two techniques that can reduce dissemination overheads in our secondhand-based estimation framework.

4.1 Selective Eager Dissemination

Intuitively, disseminating collectively based on time-out, rather than individually at once, can reduce dissemination overhead, and we observed that such periodic dissemination significantly reduces dissemination overhead in terms of the number of exchanged messages. Although periodic dissemination can make a significant enhancement for dissemination, one shortcoming would be the propagation delay due to its periodicity in dissemination. Some applications need to spread critical information more quickly. For example, we may want to disseminate the secondhand measurement if we have no information about *that* server in the measurement yet, in order to reduce the potential miss rates in estimation. To handle this, we consider *selective eager dissemination*, which disseminates hot information quickly without delay, while cold information is delivered periodically. In other words, only critical information is *eagerly* propagated to the system in this technique.

Algorithm 1 illustrates the procedure of selective eager dissemination. Function **initiate** is performed by a source node when a new measurement is obtained by actual downloading, and the source node determines if the new measurement is worth being distributed *eagerly*. Based on the decision, the measurement is either forwarded to neighbors at once (if *is_eager(m) == true*) or stored in the list for periodic dissemination (if *is_eager(m) == false*). A receiving node performs a similar function: it forwards the information immediately if it is hot; otherwise, it is moved to the periodic forwarding list, as seen in the **receive** function. Each node performs periodic dissemination when the periodic timer expires by the **time-out** function. The internal functions can be defined on the local state, as perceived by the initiating or receiving node.

Algorithm 1 Selective eager dissemination
```
1: initiate(message $m$):
2: if is_eager($m$) == true then
3:     forward($m$);
4: else
5:     forwardList.append($m$);
6: end if
7: receive(message $m$):
8: if message $\notin$ historyList then
9:     historyList.append($m$);
10:    if is_eager($m$) == true then
11:        forward($m$);
12:    else
13:        forwardList.append($m$);
14:    end if
15: end if
16: timeout():
17: forward(forwardList);
18: forwardList $\leftarrow \emptyset$
19: forward(message_array $m[]$):
20: $N \leftarrow$ neighbor nodes;
21: for all $n \in N$ do
22:     if random() $\leq p$ then
23:         send $m[]$ to $n$;
24:     end if
25: end for
```

This technique allows any disseminated information to be eventually disseminated to the entire active nodes in the system within a finite time, if the underlying probabilistic dissemination protocol is *reliable*. In our definition, a dissemination protocol is *reliable*, if there exists no active node in the system that does not receive the disseminated information within a given time constraint. We provide the related definition and propositions in Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.201.

Hot information can be determined in several ways, such as by using repetitive counters, timestamps, statistical deviations, or any combination of these techniques. However, determining hot information is application specific. In this work, we use a *threshold*, such that if the number of measurements for a server is below this threshold, then the server-specific measurement is more eagerly distributed. For example, if a measurement is "below threshold," then a node would forward it without any delay; otherwise, the measurement is regarded as cold. Thus, it is lazily forwarded after the given periodic interval expires.

4.2 Selective Deferral and Release

Another optimization technique we introduce is selective dissemination based on *deferral and release conditions*, which define whether new information can be deferred (for its dissemination) or released (to the system). If a "deferral" decision is made for some new information, the source node does not emit it into the system until the corresponding "release" condition is met. Thus, the deferral condition tests if new information is critical, while the release condition retests if deferred information is critical based on the passage of time. In this technique, any deferred information will either be disseminated if it becomes important later or discarded when it becomes stale. In contrast, selective eager dissemination ultimately forwards all information.

The basic idea of this technique is to distribute a newly collected measurement only if it offers unique information different from past measurements. For example, suppose node A makes an estimation of 100 KB/s for end-to-end throughput to node B based on past shared measurements. Now assume node A just downloaded a data object from B with 100 KB/s throughput. Then, node A may not want to disseminate such redundant information to others (*deferral*). However, this cold information can be changed to hot as more measurements are collected in the system. Continuing with the above example, suppose node A later sees its estimation to B with newly collected information to be significantly different from its own past measurement. For example, for a new measurement of 10 KB/s, node A may want to tell other nodes about the deferred experience (*release*).

Algorithm 2 illustrates details of the selective deferral and release technique. As in Algorithm 1, a node performs **initiate** when it obtains a new measurement, while non-source nodes perform **receive** when they receive dissemination messages from neighbors. If the measured information is hot to the system (i.e., *deferral_cond(m) == false*), it is immediately disseminated; otherwise, it is put in the deferred list, as seen in **initiate**. As before, these functions can be defined on the local node state. Any receiving node stores new information and simply forwards it if it has not seen the information before, as shown in the **receive** function. In both **initiate** and **receive**, a release test follows after new information is forwarded. This checks whether any prior deferred information is now hot and can be distributed, as shown in **release_test**. Although not shown explicitly in the algorithm, deferred messages will be purged, based on their age.

Algorithm 2 Selective deferral and release

```
1: initiate(message $m$):
2: if deferral_cond($m$) == true then
3:     deferredList.append($m$);
4: else
5:     forward($m$);
6:     release_test($m$);
7: end if
8: receive(message $m$):
9: if message $\notin$ historyList then
10:     historyList.append($m$);
11:     forward($m$);
12:     release_test($m$);
13: end if
14: release_test(message $m$):
15: $D \leftarrow$ deferred messages to the same server as $m$ from deferredList;
16: for all $d \in D$ do
17:     if release_cond($m$) == true then
18:         forward($d$);
19:         deferredList.delete(d);
20:     end if
21: end for
```

In this work, we define a deferral condition and a release condition based on the difference between new measurement and current estimation derived from prior measurements. Again, defining these conditions is application specific. Nonetheless, our definitions here are based on the *property of estimate proportionality*, provided in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.201. According to the property, the ratio between two estimates in a node is equal to the ratio between two estimates in any other node if it estimated with the identical measures. As can be seen below, the deferral and release conditions are established based on a ratio between the median estimate and the new measure. Thus, it would be possible to filter redundant information for estimation by deferral (or to restore essential information by release).

To define a deferral condition, let us suppose that $observed$ is a newly measured throughput to a specific server, and $expected$ is the estimated throughput to that server based on past measurements. The deferral condition is then defined as follows:

$$\text{Deferral condition}: \frac{|observed - expected|}{observed} < \tau_1.$$

If this condition is true, we defer dissemination for the given information. Thus, $\tau_1 = 0$ means no information will be deferred, whereas any arbitrary large value of τ_1 (e.g., $\tau_1 = 100$) may defer most of the newly collected measurements.

The release condition is similarly defined with the deferral condition by comparing deferred measurement ($deferred$) and current estimation ($expected$), as follows:

$$\text{Release condition}: \frac{|deferred - expected|}{deferred} \geq \tau_2.$$

Since $expected$ is the estimated throughput with all past relevant measurements, it can be different from the estimated value computed in the deferral phase. By this condition, if the deferred measurement has distinct information from the current estimation, it begins to be disseminated.

Defining τ-values may depend on application or system requirements. In the evaluation section, we examine how τ-values impact performance and dissemination overhead.

Unlike selective eager dissemination, it has no guarantee about eventual dissemination, unless the release condition happens for each disseminated information. In other words, some deferred information will be omitted in dissemination due to the dissatisfaction of the release condition. Hence, there may be a potential possibility of information loss. Nonetheless, we established our deferral and release conditions based on the *property of estimate proportionality* (in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.201.), and hence, we believe that our technique in this paper could minimize the effect of such loose synchronization.

5 Evaluation

5.1 Experimental Setup

For evaluation, we compare our OPEN-estimation-based selection (OPEN) with a diverse set of selection techniques, based on trace-based simulation with our 10-month data traces collected from PlanetLab [14].[3] We assume overlay systems with random topology, in which each node has two to eight neighbors (d = 2-8) by default. We considered two selection applications, replica selection and resource selection, common in distributed computing. For replica selection, we use r for replication factor, and for resource selection, we use c for the number of candidate resources in question. The selection techniques include *random selection*

3. The traces are accessible at http://ridge.cs.umn.edu/pltraces.html.

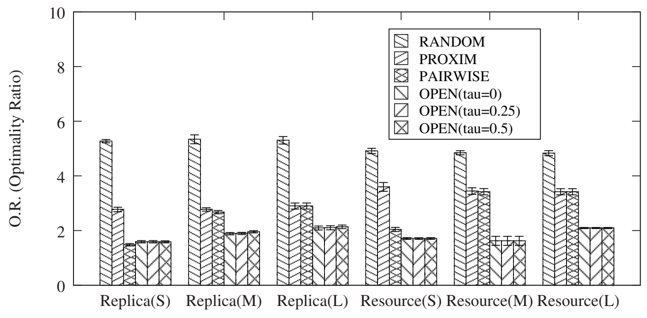

Fig. 3. Performance comparison.

(RANDOM) that randomly selects a node, and *latency-based selection* (PROXIM) that finds a client-server pair with the smallest RTT. In addition, we consider selection based on several *pairwise estimation* techniques that use only firsthand measurements. These techniques include statistical mean, median, exponential smoothing, and last value; we choose the *best* one of this group and call it PAIRWISE. For example, if selection by median yielded the best result at a round, we take that result (by median) as PAIRWISE for that round.

Unlike RANDOM and PROXIM, the other selection techniques can suffer from estimation failures due to shortage of relevant measurements.[4] To avoid meaningless estimation values from impacting the selection algorithm, we use the PAIRWISE and OPEN estimation techniques only if at least *half* of the measurements required for estimation are available, based on our observation that performance gets degraded if we perform selection with less than half; otherwise, we assume that the selection using these techniques falls back on latency-based selection (*fallback*). Thus, *fallback ratio* refers to the fraction of fallback out of the entire selections.

To compare the different selection algorithms, we mainly use the metric *Optimality Ratio* (OR) [10], where optimal is an oracle-based algorithm that chooses the best from a given set of trace data for each selection with a priori knowledge. Thus, $\mathrm{OR} = 1$ means that the selection technique chooses optimal. Since we used *mixed* data sets in simulation as mentioned, relative comparison is also more meaningful than providing absolute download times. We also examine *overhead* of dissemination. For this, we evaluate *number of messages* generated for dissemination of measurements to share in the system. The *normalized* number of messages refers to the average number of messages per round at each node.

5.2 Experimental Results

In this section, we report our core results. Additional results are available in Appendix C, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.201. Fig. 3 shows the results for both replica selection ($r = 8$) and resource selections ($c = 8$) for small ($S; n \approx 250$), medium ($M; n \approx 1{,}000$), and large ($L; n \approx 10{,}000$) systems. As expected, PROXIM works better than random choices. However, PAIRWISE does not work much better than PROXIM, except for the small system. This is because, as discussed in Section 3.1, there is a high probability that the

4. PROXIM does not fail since the trace data include latency information.

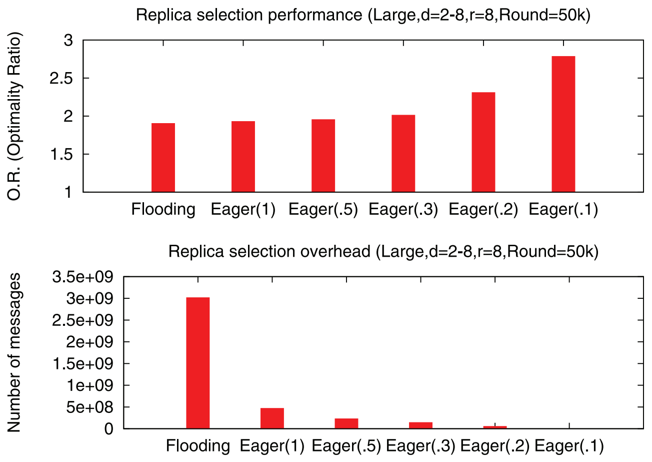

Fig. 4. Performance with selective eager dissemination.

pairwise techniques fail to see relevant measurements in their estimations, and hence will fall back to PROXIM. In replica selection, the fallback ratio to PROXIM is 15 percent in the small system, but it increases to 95 percent in the medium system. In the large system, PAIRWISE fallback ratio reaches almost 100 percent, indicating that no pairwise estimation was made due to lack of pair-level measurements. In contrast, OPEN falls back to PROXIM 0.5 percent in the small system, 2 percent in the medium system, and 18 percent in the large system. *This result emphasizes again why secondhand estimation is attractive for large-scale systems.* Fallback ratios for PAIRWISE are slightly greater in resource selection, while OPEN shows similar fallback ratios in both replica and resource selections. In the large system, OPEN requires more rounds to collect measurements for each server. This slightly affects performance in the large system.

In the figure, we can see that OPEN outperforms all the other selection techniques in both replica and resource selections. We set up three configurations for OPEN: OPEN($\tau = 0$) which disables selective deferral and release, OPEN($\tau = 0.25$) and OPEN($\tau = 0.5$) which enable selective deferral and release. In this experiment, we use $\tau = \tau_1 = \tau_2$ to make deferral and release decisions. Smaller τ would make deferral decision less likely, whereas greater τ tends to aggressively defer dissemination of new measurements. Even with deferring dissemination, we can see little performance loss in the figure. Although not shown in the figure, we observed that a substantial number of measurements were deferred with the selective deferral and release. For example, in replica selection, 28 percent of measurements were in the deferred list for $\tau = 0.25$ at the end of the simulation, while it was 56 percent for $\tau = 0.5$.

Next, we present some experimental results for selective eager dissemination with diverse dissemination probabilities. For evaluation, we examine *Eager* with periodic interval 1,000 and eager threshold 2, and *Flooding* for comparison. With *Eager*, any new measurements will be eagerly forwarded if the node has seen fewer than two measurements for the corresponding server, while others will be periodically disseminated with the interval. In addition, we gave different dissemination probability for each *Eager*; for example, *Eager(0.5)* stands for selective eager dissemination with a probability of 0.5. Fig. 4 shows the results. We can see that *Eager(0.3)* yields fairly comparable results to flooding. In this case, the overhead is only 5 percent of *Flooding*. Even in the case of *Eager(1.0)*, it reduces the number of messages to 15 percent of the *Flooding* without performance loss.

5.3 Discussion

An important question is whether the overheads of dissemination might swamp the gains. Although random selection yielded poor and unstable results, it did not create any additional cost for the purpose of estimation. However, any selection based on estimations would incur extra load and traffic which may affect user data access. For example, for selective deferral and release with gossip probability $p = 0.3$ and selective deferral and release parameters $\tau_1 = \tau_2 = 0.25$, we observed that each node created 1.15 MB additional traffic on average to share 50,000 measurements representing 50,000 distinct downloads over time.[5] In the same setting, Spruce [28] requires around 3 GB traffic per node for all-pair "single" measurements in a 10,000-node system (based on 300 KB per measurement that is the traffic requirement on average in Spruce). Given the rich availability of peer-to-peer bandwidth, and the time frame for sharing 50,000 distinct downloads, the OPEN overhead is likely to have minor impact on the results. In addition, dissemination messages can be piggybacked over other system messages to reduce the number of extra messages, e.g., periodic neighbor heartbeats needed for system health.

Another issue would be "information inequality" due to different joining times or imperfect probabilistic dissemination. This may result in different decisions even for the same event at each node. In the selective eager dissemination technique, each node makes its own eager or periodic forwarding decision. Similarly, in the selective deferral and release technique, the source node makes a decision whether new information is distributed immediately or not. Those decisions rely on local information, and thus can be biased. For example, source node S is long lived and can make a deferral decision because it has redundant information, but any recently joined node may suffer from estimation failure due to lack of relevant information which should have been available if S released it. This information inequality can be mitigated by downloading shared measurements from parent nodes at joining times.

6 Related Work

A system with similar goals to ours is NWS [19], [29]. NWS maintains network sensors measuring network performance by periodic probing and statistical predictions based on probing results. Thus, it requires $O(n^2)$ communications for all-pair probing, where n is the number of sensors. To reduce the probing overhead, the sensors are organized in a hierarchical structure by configuring logical clusters called cliques, and end-to-end network performance between two nodes lying different clusters is determined by measured network performance between the sensors in those clusters. While accurate for sensor nodes, other nodes estimation may be inaccurate in non-clustered distributed environments. In addition, all-pair probing is still expensive even with a hierarchical design, particularly in large-scale systems this paper considers. OPEN enables scalable, low-cost estimation by sharing secondhand measurements, i.e., without relying on all-pair probing.

iPlane [30] is an infrastructure-based service which provides prediction of end-to-end network performance in the Internet. For the prediction service, iPlane measures segment paths chosen based on the Internet topology, and the end-to-end path property is inferred from the collected measurements. While providing fairly accurate estimation, it requires explicit probing between additionally deployed probing sensors. Its offspring iPlane Nano [31] improves scalability by compacting the network topology information, but limits the prediction capability to latency and loss rate estimation and relies on distributed sensors for explicit probing.

Gossip-based dissemination is scalable and resilient to failure, so it is used in many areas, such as wireless ad hoc networks [26], [27] and large-scale networks [23], [24], [25], [32], [33], for diverse purposes like routing, data dissemination, membership management, and so forth. Haas et al. [26] proposed a set of gossip protocols with several parameters, including forwarding probability, number of hops, and neighbor size, to improve ad hoc routing protocols, many of which are based on flooding. Although gossip techniques can significantly reduce the dissemination overhead with negligible information loss, we consider application-level semantics with respect to information criticality for further reduction of dissemination overhead in this paper.

7 Conclusion

Large-scale distributed systems are attractive for many distributed computing applications by providing a scalable, cost-effective infrastructure. However, a primary challenge is the unpredictability of data access which can lead to significant performance bottlenecks. Providing predictability in data access is needed to accommodate the large set of data-intensive computing applications. To this end, we have designed a framework called OPEN which offers end-to-end network performance estimation based on secondhand measurements observed other nodes in the system. To share secondhand measurements, OPEN proactively distributes newly collected measurements by a probabilistic dissemination technique. We presented our extensive experimental results that show resource and replica selections with OPEN consistently outperform selection techniques based on statistical pairwise estimations as well as latency-based selection. We also show that OPEN can dramatically reduce dissemination overhead to share secondhand measurements without significant performance loss by proposed optimization techniques such as selective deferral and release.

We envision that the OPEN framework can collaborate with other services for distributed computing. For instance, our framework can work with perfSONAR [34], currently based on direct measurements for network performance estimation (for replica selection), to enable cost-effective services in large-scale grid systems.

5. We calculated with 40 bytes for one dissemination message (including TCP header) with a message identifier (4 bytes), client and server identifiers (8 bytes), a distance to server (2 bytes), a throughput (2 bytes), a download power (2 bytes), and a timestamp (4 bytes).

ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers for their constructive comments. This work was supported in part by US National Science Foundation grant CNS-0643505 and IIS-0916425. Appendices, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.201, for additional details and extended experimental results are also available from Digital Library with the electronic version of the paper.

REFERENCES

[1] "LHC: Large Hadron Collider," http://lhc.web.cern.ch/lhc/, 2011.

[2] "Sloan Digital Sky Survey/Skyserver," http://cas.sdss.org/dr7/en/, 2011.

[3] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, and M.-H. Su, "Characterization of Scientific Workflows," *Proc. Third Workshop Workflows in Support of Large-Scale Science (WORKS '08)*, 2008.

[4] E. Deelman and A. Chervenak, "Data Management Challenges of Data-Intensive Scientific Workflows," *Proc. Eighth IEEE Int'l Symp. Cluster Computing and the Grid (CCGRID '08)*, pp. 687-692, 2008.

[5] D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience," *Concurrency - Practice and Experience*, vol. 17, nos. 2-4, pp. 323-356, 2005.

[6] "The Globus Alliance," http://www.globus.org/, 2011.

[7] D. Kondo, A.A. Chien, and H. Casanova, "Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids," *Proc. ACM/IEEE Conf. Supercomputing (SC '04)*, 2004.

[8] V. Lo, D. Zappala, D. Zhou, Y. Liu, and S. Zhao, "Cluster Computing on the Fly: P2p Scheduling of Idle Cycles in the Internet," *Proc. IEEE Fourth Int'l Conf. Peer-to-Peer Systems*, pp. 227-236, 2004.

[9] D.P. Anderson and G. Fedak, "The Computational and Storage Potential of Volunteer Computing," *Proc. Sixth IEEE Int'l Symp. Cluster Computing and the Grid (CCGRID '06)*, pp. 73-80, 2006.

[10] T.S.E. Ng, Y. hua Chu, S.G. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems," *Proc. IEEE INFOCOM*, pp. 2199-2209, 2003.

[11] S. Seshan, M. Stemm, and R.H. Katz, "SPAND: Shared Passive Network Performance Discovery," *Proc. USENIX Symp. Internet Technologies and Systems*, pp. 135-146, Dec. 1997.

[12] M. Andrews, B. Shepherd, A. Srinivasan, P. Winkler, and F. Zane, "Clustering and Server Selection Using Passive Monitoring," *Proc. IEEE INFOCOM*, pp. 1717-1725, 2002.

[13] J. Kim, A. Chandra, and J.B. Weissman, "Using Data Accessibility for Resource Selection in Large-Scale Distributed Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 20, no. 6, pp. 788-801, June 2009.

[14] "PlanetLab," http://www.planet-lab.org, 2011.

[15] N. Kourtellis, L. Prieto, A. Iamnitchi, G. Zarrate, and D. Fraser, "Data Transfers in the Grid: Workload Analysis of Globus Gridftp," *Proc. Int'l Workshop Data-Aware Distributed Computing (DADC '08)*, pp. 29-38, 2008.

[16] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee, "S3: A Scalable Sensing Service for Monitoring Large Networked Systems," *Proc. SIGCOMM Workshop Internet Network Management (INM '06)*, pp. 71-76, 2006.

[17] G.B. Berriman, A.C. Laity, J.C. Good, J.C. Jacob, D.S. Katz, E. Deelman, G. Singh, M.-H. Su, and T.A. Prince, "Montage: The Architecture and Scientific Applications of a National Virtual Observatory Service for Computing Astronomical Image Mosaics," *Proc. Earth Sciences Technology Conf.*, 2006.

[18] A.B. Downey, "Using Pathchar to Estimate Internet Link Characteristics," *Proc. ACM SIGCOMM '99*, pp. 241-250, 1999.

[19] R. Wolski, N. Spring, and J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing," *J. Future Generation Computing Systems*, vol. 15, pp. 757-768, 1999.

[20] S. Keshav, "Packet-Pair Flow Control," *IEEE/ACM Trans. Networking*, 1995.

[21] Q. He, C. Dovrolis, and M. Ammar, "On the Predictability of Large Transfer Tcp Throughput," *Proc. ACM SIGCOMM '05*, pp. 145-156, 2005.

[22] R.L. Carter and M. Crovella, "Server Selection Using Dynamic Path Characterization in Wide-Area Networks," *Proc. IEEE INFOCOM*, pp. 1014-1021, 1997.

[23] A.-M. Kermarrec, L. Massoulié, and A.J. Ganesh, "Probabilistic Reliable Dissemination in Large-Scale Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 3, pp. 248-258, Mar. 2003.

[24] S. Voulgaris and M. van Steen, "Hybrid Dissemination: Adding Determinism to Probabilistic Multicasting in Large-Scale p2p Systems," *Proc. ACM/IFIP/USENIX Int'l Conf. Middleware*, pp. 389-409, 2007.

[25] M. Deshpande, B. Xing, I. Lazardis, B. Hore, N. Venkatasubramanian, and S. Mehrotra, "Crew: A Gossip-Based Flash-Dissemination System," *Proc. 26th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06)*, p. 45, 2006.

[26] Z.J. Haas, J.Y. Halpern, and L. Li, "Gossip-Based Ad Hoc Routing," *IEEE/ACM Trans. Networking*, vol. 14, no. 3, pp. 479-491, June 2006.

[27] P. Kyasanur, R. Choudhury, and I. Gupta, "Smart Gossip: An Adaptive Gossip-Based Broadcasting Service for Sensor Networks," *Proc. IEEE Int'l Conf. Mobile Adhoc and Sensor Systems*, pp. 91-100, 2006.

[28] J. Strauss, D. Katabi, and F. Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools," *Proc. Third ACM SIGCOMM Conf. Internet Measurement (IMC '03)*, pp. 39-44, 2003.

[29] R. Wolski, "Experiences with Predicting Resource Performance Online in Computational Grid Settings," *SIGMETRICS Performance Evaluation Rev.*, vol. 30, no. 4, pp. 41-49, 2003.

[30] H.V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An Information Plane for Distributed Services," *Proc. Seventh USENIX Symp. Operating Systems Design and Implementation (OSDI '06)*, 2006.

[31] H.V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "Iplane Nano: Path Prediction for Peer-to-Peer Applications," *Proc. Sixth USENIX Symp. Networked Systems Design and Implementation (NSDI '09)*, pp. 137-152, 2009.

[32] I. Gupta, A.-M. Kermarrec, and A.J. Ganesh, "Efficient and Adaptive Epidemic-Style Protocols for Reliable and Scalable Multicast," *IEEE Trans. Parallel and Distributed Systems*, vol. 17, no. 7, pp. 593-605, July 2006.

[33] M. jang Lin and K. Marzullo, "Directional Gossip: Gossip in a Wide Area Network," *Proc. European Dependable Computing Conf.*, pp. 364-379, 1999.

[34] "perfsonar," http://www.perfSONAR.net/, 2011.

Jinoh Kim received the BE and MS degrees in computer science and engineering from Inha University, Korea, in 1991 and 1994, respectively, and the PhD degree in computer science from the University of Minnesota, Twin Cities, in 2010. From 1991 to 2005, he was a researcher at ETRI, Korea, and participated in various nation-wide research projects, including ATM system/network management, IP over ATM, network security, and policy-based security management. He is currently a postdoctoral fellow at the Lawrence Berkeley National Laboratory. His research interests are in the areas of distributed computing, energy-proportional computing, distributed systems, and network security, measurement, and management. He is a member of the IEEE and the IEEE Computer Society.

Abhishek Chandra received the BTech degree in computer science and engineering from IIT Kanpur, and the MS and PhD degrees in computer science from the University of Massachusetts Amherst. He is an associate professor in the Department of Computer Science and Engineering at the University of Minnesota. His research interests are in the areas of operating systems, distributed systems, and computer networks. He is a recipient of the US National Science Foundation (NSF) CAREER Award, his PhD dissertation titled "Resource Allocation for Self-Managing Servers" was nominated for the ACM Dissertation Award, and he has been a coauthor on two Best Paper/Student Paper Awards. He is a member of the IEEE and the IEEE Computer Society.

Jon B. Weissman received the BS degree from Carnegie-Mellon University in 1984, and the MS and PhD degrees from the University of Virginia in 1989 and 1995, respectively, all in computer science. He is a leading researcher in the area of high performance distributed computing. His involvement dates back to the influential Legion project at the University of Virginia during his PhD. He is currently an associate professor of computer science at the University of Minnesota where he leads the Distributed Computing Systems Group. His current research interests are in grid/cloud computing, distributed systems, high performance computing, resource management, reliability, and e-science applications. He works primarily at the boundary between applications and systems. He is a senior member of the IEEE and the IEEE Computer Society, and awardee of the US National Science Foundation (NSF) CAREER Award (1995).