ISSN:0975-9646

Cloud Computing and an End-System Redundancy Elimination Service for Enterprises

Ganapathi Avabrath,
Asst. Professor,Impact College of Engineering & Applied Sciences, Bangalore

Abstract-In many enterprises today, WAN optimizers are being deployed in order to eliminate redundancy in network traffic and reduce WAN access costs. In this paper, we present the design and implementation of ERE, an alternate approach where redundancy elimination (RE) is provided as an end system service. Unlike middleboxes, such an approach benefits both end-to-end encrypted traffic as well as traffic on last-hop wireless links to mobile devices.
ERE needs to be fast, adaptive and parsimonious in memory usage in order to opportunistically leverage resources on end hosts. Thus, we design a new fingerprinting scheme called SampleByte that is much faster than Rabin fingerprinting while delivering similar compression gains. Unlike Rabin fingerprinting, SampleByte can also adapt its CPU usage depending on server load. Further,we introduce optimizations to reduce server memory footprint by 33-75% compared to prior approaches.Using several terabytes of network traffic traces from 11 enterprise sites, testbed experiments and a pilot deployment,we show that ERE delivers 26%bandwidth savings on average, processes payloads at speeds of 1.5-4Gbps, reduces end-to-end latencies by up to 30%, and translates bandwidth savings into equivalent energy savingson mobile smartphones.

Key words: cloud computing, redundancy, latency, encryption

1. DEFINING THE CLOUD COMPUTING

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS).[a] Some vendors use terms such as IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) to describe their products, but we eschew these because accepted definitions for them still vary widely. The line between "low-level" infrastructure and a higher-level "platform" is not crisp. We believe the two are more alike than different, and we consider them together. Similarly, the related term "grid computing," from the high-performance computing community, suggests protocols to offer shared computation and storage over long distances, but those protocols did not lead to a software environment that grew beyond its community.
Cloud computing, the long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about overprovisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or underprovisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1,000 servers for one hour costs no more than using one server for 1,000 hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT.

2. INTRODUCTION

With the advent of globalization, networked services have a global audience, both in the consumer and enterprise spaces. For example, a large corporation today may have branch offices at dozens of cities around the globe. In such a setting, the corporation’s IT admins and network planners face a dilemma. On the one hand, they could concentrate IT servers at a small number of locations. This might lower administration costs, but increase network costs and latency due to the resultant increase in WAN traffic. On the other hand, servers could be located closer to clients; however, this would increase operational costs.
This paper arises from the quest to have the best of both worlds, specifically, having the operational benefits of centralization along with the performance benefits of distribution. In recent years, protocol-independent redundancy elimination, or simply RE has helped bridge the gap by making WAN communication more efficient through elimination of redundancy in traffic. Such compression is typically applied at the IP or TCP layers, for instance, using a pair of middleboxes placed at either end of a WAN link connecting a corporation’s data center and a branch office. Each box caches payloads from flows that traverse the link, irrespective of the application or protocol. When one box detects chunks of data that match entries in its cache (by computing “fingerprints”of incoming data and matching them against cached data), it encodes matches using tokens. The box at the far end reconstructs original data using its own cache and the tokens. This approach has seen increasing deployment in “WAN optimizers”.

3. MOTIVATION

In exploring an end-point based RE service, one of the main issues we hope to address is whether such a service can offer bandwidth savings approaching that of WAN optimizers. To motivate the likely benefits of an end-point based RE service, we briefly review two key findings from our earlier study [8] of an IP-layer WAN Optimizer.
First, we seek to identify the origins of redundancy. Specifically, we classify the contribution of redundant byte matches to bandwidth savings as either *intra-host*(current

and matched packet in cache have identical source-destination IP addresses) or *inter-host* (current and matched packets differ in at least one of source or destination IP addresses). We were limited to a 250MB cache size given the large amount of meta-data necessary for this analysis, though we saw similar compression savings for cache sizes up to 2GB. Surprisingly, our study revealed that *over 75% of savings were from intra-host matches*. This implies that a pure end-to-end solution could potentially deliver a significant share of the savings obtained by an IP WAN optimizer, since the contribution due to inter-host matches is small. However, this finding holds good only if end systems operate with similar (large) cache sizes as middleboxes, which is impractical. This brings us to the second key finding.

4 Design Goals

ERE is designed to optimize data transfers in the direction from servers in a remote data center to clients in the enterprise, since this captures a majority of enterprise traffic. We now list five design goals for ERE—the first two design goals are shared to some extent by prior RE approaches, but the latter three are unique to ERE.

1. **Transparent operation:** For ease of deploy-ability,the ERE service should require no changes to existing applications run within the data center or on clients.
2. **Fine-grained operation:** Prior work has shown that many enterprise network transfers involve just a few packets. To improve end-to-end latencies and provide bandwidth savings for such short flows, ERE must work at fine granularities, suppressing duplicate byte strings as small as 32-64B. This is similar to ,but different from earlier proposals for file-systems and Web caches where the sizes of redundanciesidentified are 2-4KB.
3. **Simple decoding at clients:** ERE's target client set includes battery- and CPU-constrained devices such as smart-phones. While working on fine granularities can help identify greater amounts of redundancy, it can also impose significant computation and decoding overhead, making the system impractical for these devices. Thus, a unique goal is to design algorithms that limit client overhead by *offloading* all compute-intensive actions to *servers*.
4. **Fast and adaptive encoding at servers:** ERE is designed to opportunistically leverage CPU resources on end hosts when they are not being used by other pplications.
   Thus, unlike commercial WAN optimizers and prior RE approaches [20], ERE must *adapt* its use of CPU based on server load.
5. **Limited memory footprint at servers and clients:**
   ERE relies on data caches to perform RE. However,memory on servers and clients could be limited and may be actively used by other applications. Thus, ERE must use as minimal memory on end-hosts as possible through the use of optimized data structures.

5 ERE Design

In this section, we describe how ERE's design meets the above goals.

ERE introduces RE modules into the network stacks of clients and remote servers. Since we wish to be transparent to applications, ERE could be implemented either at the IP-layer or at the socket layer (above TCP). As we argue , we believe that socket layer is the right place to implement ERE. Doing so offers key performance benefits over an IP-layer approach, and more importantly, shields EndRE from network-level events (e.g., packet losses and reordering), making it simpler to implement.

There are two sets of modules in ERE, those belonging on servers and those on clients. The server-side module is responsible for identifying redundancy in network data by comparing against a cache of prior data, and encoding the redundant data with shorter meta-data.The meta-data is essentially a set of <offset, length> tuples that are computed with respect to the client-side cache. The client-side module is trivially simple: it consists of a fixed-size circular FIFO log of packets and simple logic to decode the meta-data by "de-referencing" the offsets sent by the server. Thus, most of the complexity in ERE is mainly on the server side and we focus on that here. Identifying and removing redundancy is typically accomplished [20, 7] by the following two steps:

- *Fingerprinting:* Selecting a few "representative regions" for the current block of data handed down by application(s). We describe four fingerprinting algorithms that differ in the trade-off they impose between *computational overhead* on the server and the *effectiveness* of RE.
- *Matching and Encoding:* Once the representative regions are identified, we examine two approaches for identification of redundant content : (1) Identifying chunks of representative regions that repeat in full across data blocks, called Chunk-Match and (2) Identifying maximal matches around the representative regions that are repeated across data blocks, called Max- Match. These two approaches differ in the trade-off between the *memory overhead* imposed on the server and the *effectiveness* of RE.

6 Implementation

In this section, we discuss our implementation of ERE.

We start by discussing the benefits of implementing ERE at the socket layer above TCP.

6.1 Performance benefits

Bandwidth: In the socket-layer approach, RE can operate at the size of socket writes which are typically larger than IP layer MTUs. While Max-Match and Chunk-Match do not benefit from these larger sized writes since they operate at a granularity of 32 bytes, the large size helps produce higher savings if a compression algorithm like GZIP is *additionally* applied, as evaluated.

Latency: The socket-layer approach will result in fewer packets transiting between server and clients, as opposed to the IP layer approach which merely compresses packets without reducing their number. This is particularly useful in lowering completion times for short flows, as evaluated.

6.2 End-to-end benefits

Encryption: When using socket-layer RE, payload encrypted in SSL can be compressed before encryption,providing RE benefits to protocols such as HTTPS. Iplayer RE will leave SSL traffic uncompressed.

Cache Synchronization: Recall that both Max-Match and Chunk-Match require caches to be synchronized between clients and servers. One of the advantages of implementing ERE above TCP is that TCP ensures reliable in-order delivery, which can help with maintaining cache synchronization. However, there are still two issues that must be addressed.

First, multiple simultaneous TCP connections may be operating between a client and a server, resulting in ordering of data across connections not being preserved.

To account for this, we implement a simple sequence number-based *reordering mechanism*. Second, TCP connections may get reset in the middle of a transfer. Thus, packets written to the cache at the server end may not even reach the client, leading to cache inconsistency. One could take a *pessimistic* or *optimistic* approach to maintaining consistency in this situation. In the pessimistic approach, writes to the server cache are performed only after TCP ACKs for corresponding segments are received at the server. The server needs to monitor TCP state, detect ACKs, perform writes to its cache and notify the client to do the same. In the optimistic approach, the server writes to the cache but monitors TCP only for reset events. In case of connection reset (receipt of a TCP RST from client or a local TCP timeout), the server simply notifies the client of the last sequence number that was written to the cache for the corresponding TCP connection. It is then the client’s responsibility to detect any missing packets and recover these from the server. We adopt the *optimistic approach of cache writing* for two reasons: (1) Our redundancy analysis indicated that there is high temporal locality of matches; a pessimistic approach over a high bandwidth-delay product link can negatively impact compression savings; (2) The optimistic approach is easier to implement since only for reset events need to be monitored rather than every TCP ACK.

7. Conclusion

Using extensive traces of enterprise network traffic and testbed experiments, we show that our end-host based redundancy elimination service, EndRE, provides average bandwidth gains of 26% and, in conjunction with DOT, the savings approach that provided by a WAN optimizer.

Further, ERE achieves speeds of 1.5-4Gbps, provides latency savings of up to 30% and translates bandwidth savings into comparable energy savings on mobile smartphones. In order to achieve these benefits,EndRE utilizes memory and CPU resources of end systems.

For enterprise clients, we show that median memory requirements for ERE is only 60MB. At the server end, we design mechanisms for working with reduced memory and adapting to CPU load.

Thus, we have shown that the cleaner semantics of end-to-end redundancy removal can come with considerable performance benefits and low additional costs. This makes ERE a compelling alternative to middleboxbased approaches.

References

[1] Cisco Wide Area Application Acceleration Services. http://www.cisco.com/en/US/products/ps5680/Products Sub Category Home.html.

[2] Jenkins Hash. http://burtleburtle.net/bob/c/lookup3.c.

[3] Peribit Networks (Acquired by Juniper in 2005): WAN Optimization Solution. http://www.juniper.net/.

[4] Power Monitor, Monsoon Solutions. http://www.msoon.com/ powermonitor/powermonitor.html.

[5] Riverbed Networks: WAN Optimization. http://www.riverbed. com/solutions/optimize/.

[6] Windows Filtering Platform. http://msdn.microsoft.com/ en-us/library/aa366509(V.85).aspx.

[7] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination.In ACM SIGCOMM, Seattle, WA, Aug. 2008.

[8] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee. Redundant in Network Traffic: Findings and Implications. In ACM SIGMETRICS, Seattle, WA, June 2009.

[9] S. Annapureddy, M. J. Freedman, and D. Mazires. Shark: Scaling file servers via cooperative caching. In NSDI, 2005.

[10] M. Arlitt and C. Williamson. An analysis of tcp reset behavior on the internet. ACM CCR, 35(1), 2005.

[11] K. C. Barr and K. Asanovic. Energy-aware lossless data compression. IEEE Transactions on Computer Systems, 24(3):250–291, Aug 2006.

[12] F. Douglis and A. Iyengar. Application-specific delta-encoding via resemblance detection. In USENIX, 2003.