Hypertable: The Storage Infrastructure behind Rediffmail - one of the World's Largest Email Services

Doug Judd
CEO, Hypertable Inc.

Introduction

Rediff.com India (Nasdaq: REDF) is one of India's top Internet portals, providing email, search, news, entertainment, and shopping services to India and the global Indian community. With over 100 million registered users, Rediff.com is one of the largest Indian Internet portals and one of the top email providers worldwide.

Rediff.com's popular email service, Rediffmail, has experienced steady growth ever since it was launched in 1998. Architected for high availability, Rediffmail is a geo-distributed application served out of multiple datacenters. By 2011, the request load generated by the application had begun to overwhelm the underlying storage system, causing sluggish responses. Metadata updates in connection with mail list management turned out to be the primary culprit, generating over 75% of the storage system request volume. In late 2011, the Rediffmail engineering team re-architected the system on top of Hypertable, solving the mail list management problem and easing the associated sluggish responses.

Current Architecture

The Rediffmail data set is split into horizontal "shards", with each shard managed in multiple data centers. Within each datacenter there is a large cluster of front-end webservers that connect directly to a set of NAS[1] appliances used for storing and retrieving mail message bodies. The front-end web servers are also connected, through a load balancer, to two large Hypertable clusters for storing and retrieving mail metadata. The two Hypertable clusters contain replicated copies of the metadata and are configured as a primary and secondary. Updates are sent to both clusters, but the primary cluster handles all of the read traffic. The application is configured to send read traffic to the secondary cluster when maintenance or upgrades are performed on the primary. The following diagram provides a high-level overview of the system architecture.

[1] Network-attached Storage

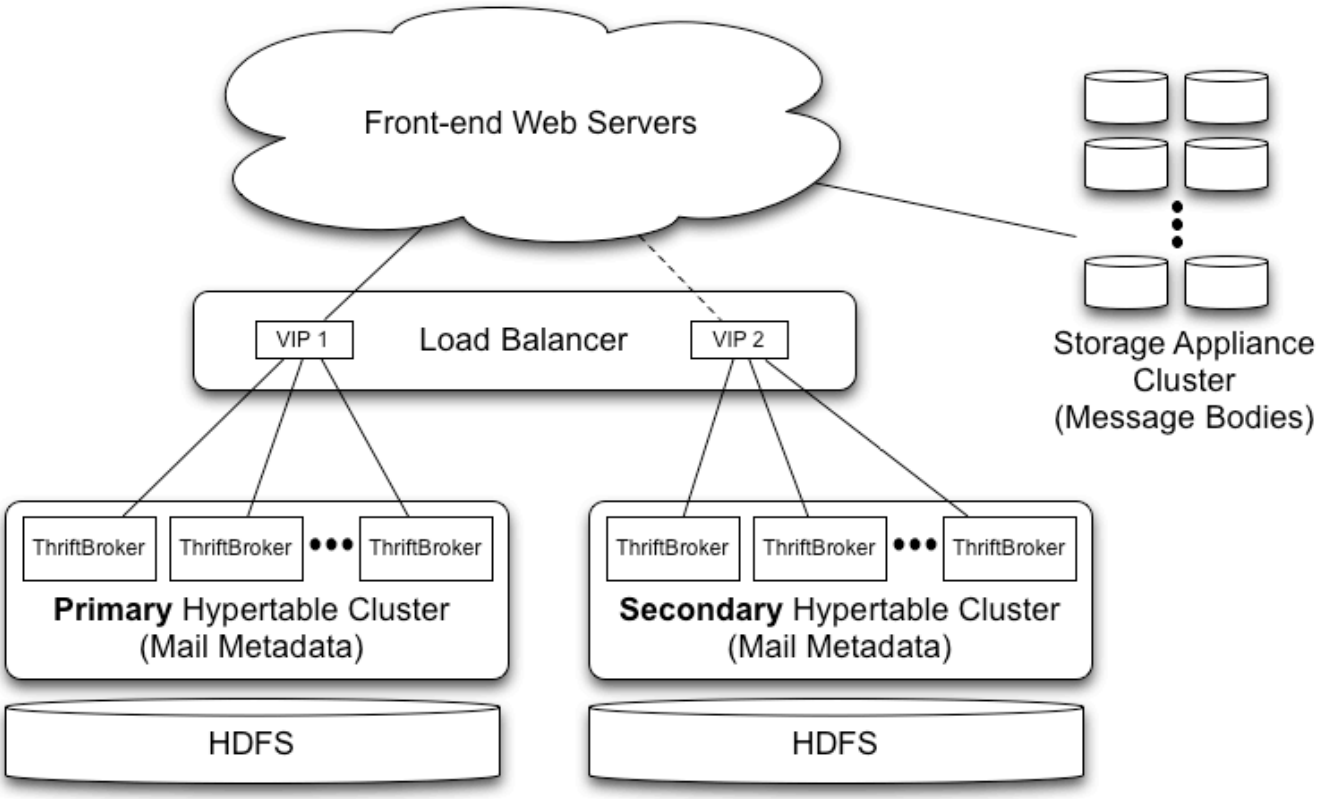

Figure 1. Rediffmail Architecture

How It Works

An email generally consists of two parts, a mail header and mail body. Mail headers generally include sender, receiver, mail attachment details etc. Both email parts are considered to be static information, which never changes. However, an email can go through multiple state changes based on the actions taken (read, reply, forward, etc) by the user on the mail. The mail header information and actions are stored in a table in Hypertable created with the following Hypertable Query Language (HQL) statement:

```
CREATE TABLE 'mail_metadata' (
  'mail_headers' MAX_VERSIONS=N,
  'mail_actions' MAX_VERSIONS=N,
);
```

The row key represents the unique identity of a particular inbox. One of the requirements is to restrict the versions of mail header information so as to be able to identify the latest state of the mail after the user performs an action with an email. This is achieved through the use of the MAX_VERSIONS option. The column qualifier for both of these columns is the unique mail message ID. The following table summarizes the key format for the *mail_headers* column:

Row key	Unique Mailbox ID
Column qualifier	Message ID
Value	Header details
Timestamp	Mail create timestamp

Table 1. Key Format for mail_headers column

The header value consists of fields that contain the mail header information and the path to the mail message body on the Network attached Storage. It is generally written once on mail delivery and read many times. Value based regex queries are used to enable search functionality within mail metadata. The following illustrates what one of the cells in the *mail_headers* column might look like:

Row: `anurag.jalota@rediffmail.com`
Column: `mail_headers:1906593928.10833.62518.f6-145-153.mail`
Value: `\t0\t\tL\tInbox\t0\t\t1\t\tF\tnoreply@lockerz.com\tS\tNew Lockerz friend request from Malik\tT\tanurag.jalota@rediffmail.com\tC\t\tA\t\t1\t\t2\t\tP\t /Storage_72142/9164921/410437/Vyei5jb20ATmV3IExvY2tlcnogZnJpZW5kIHJlcXVlc3QgZn JvbSBTaGFhbi.1906593928.10833.62518.f6-145-153.mail\t2\t\t3\t\tN\t\tR\tnoreply@lockerz.com\tX\t <noreply@lockerz.com>\tD\t1293406071\t3\t`

The *mail_actions* column is used to capture all actions that a user performs on a particular email. The following table summarizes the key format for this column:

Row key	Unique Mailbox ID
Column qualifier	Message ID
Value	Action details
Timestamp	Mail action timestamp

Table 2. Key Format for mail_actions column

The *mail_actions* column contains an indicator of the action taken and includes the path to the mail body (if any) on the Network attached Storage and the action that was performed. The following illustrates what one of the cells in the *mail_actions* column might look like:

Row: `anurag.jalota@rediffmail.com`
Column: `mail_actions:1986302861.5887.9615.f6-145-167.mail`
Value:
`/Storage_72842/7252532/23532092/TlJlWrBNeVBhZ2UASGFwcHkgYmlydGhkYXkgU29udQ.1986302861. 5887.9615.f6-145-167.mail`

Updates

Updates are sent using a process running on each web server, which immediately writes the updates to both the primary and secondary Hypertable clusters. If one of the clusters is offline, then the updates are queued until the cluster comes back online, at which time, the updates are processed by clearing the queue. Figure 2 shows write statistics for a single machine over a one-week period.

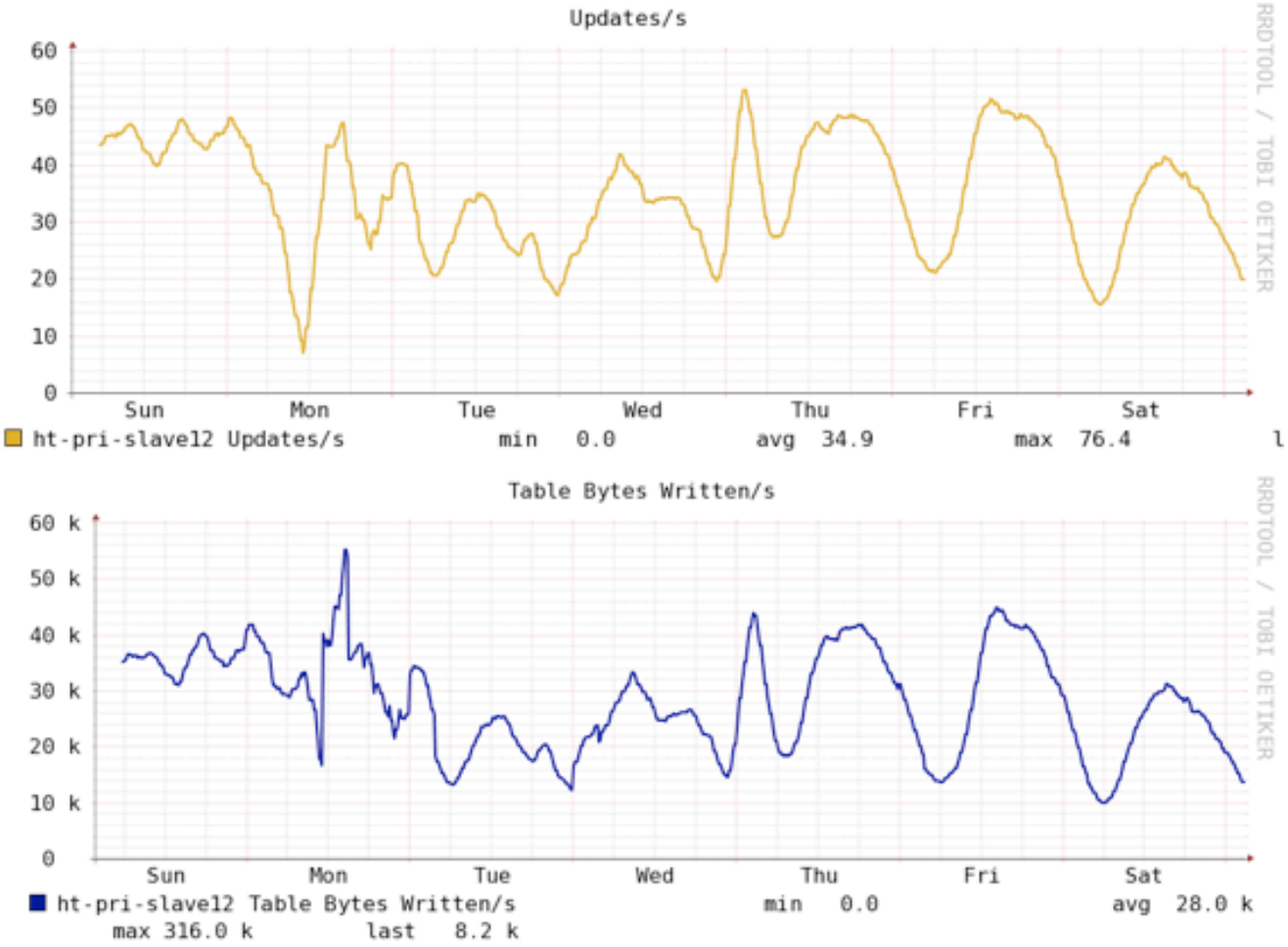

Figure 2. Single machine write statistics

Queries

Queries are normally sent to the primary Hypertable cluster. If the primary cluster is offline for maintenance or upgrades, then queries are sent to the secondary. There are three scenarios in which queries are normally issued to Hypertable:

1. **Mail Listing.** Provides a listing of all the mails in the user's mailbox. The listing generally includes sender, subject, timestamp, last action taken. This results in fetching information from both the columns.

2. **Mail Search.** Based on the search string, gives a listing of all the mails qualifying the search. Again the listing includes sender, subject, timestamp, last action taken. This results in a REGEX query on the value of header column and also getting actions from action columns

3. **Mail Sync.** Enables syncing of mails from other mail access subsystems like mobile device apps, IMAP, pop3 etc. All the actions and headers posted since the previous sync (timestamp) are fetched from both columns.

Figure 3 shows read statistics for a single machine over a one-week period.

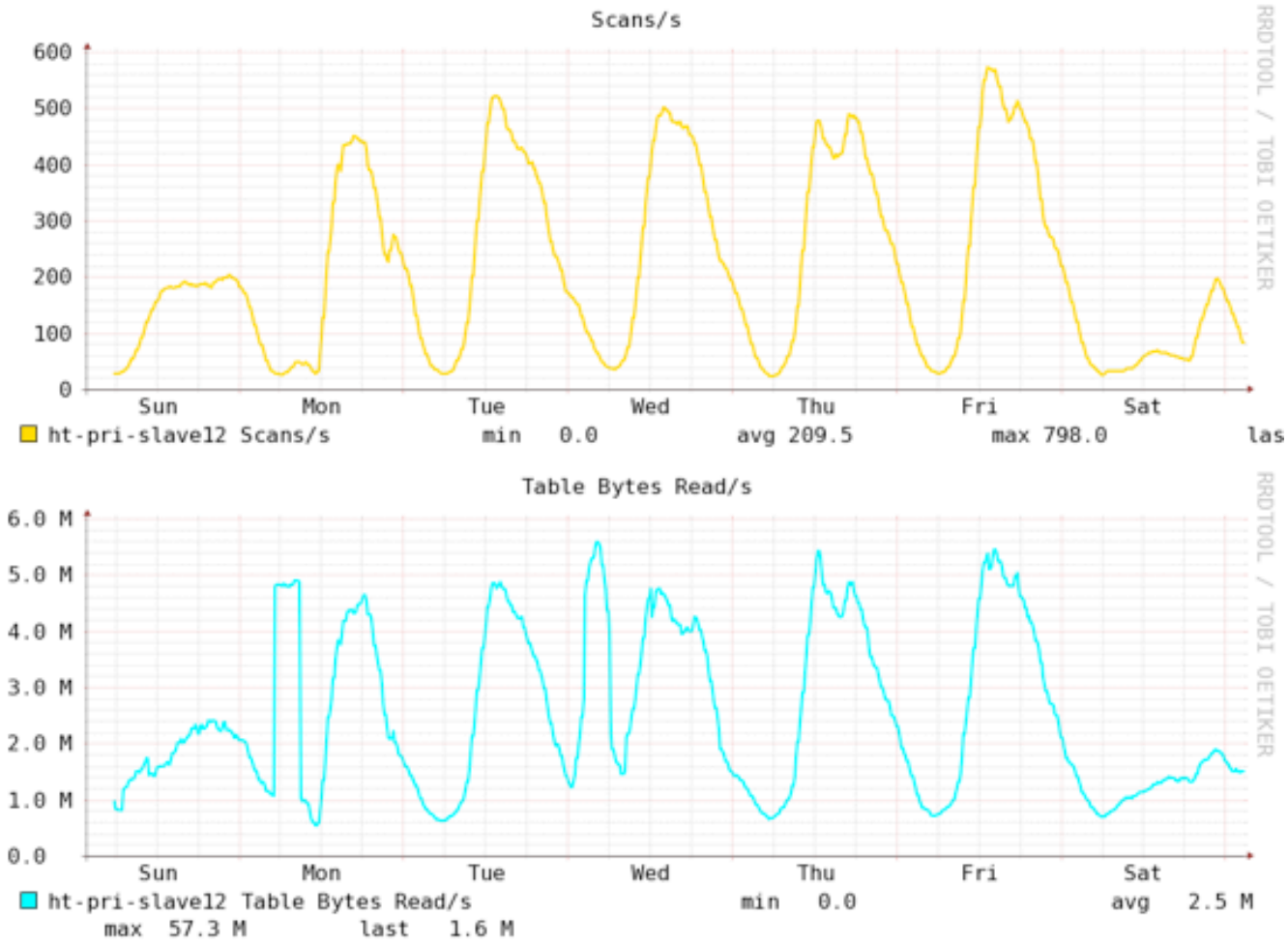

Figure 3. Single machine read statistics

Figure 4 shows caching statistics for a single machine over a one-week period.

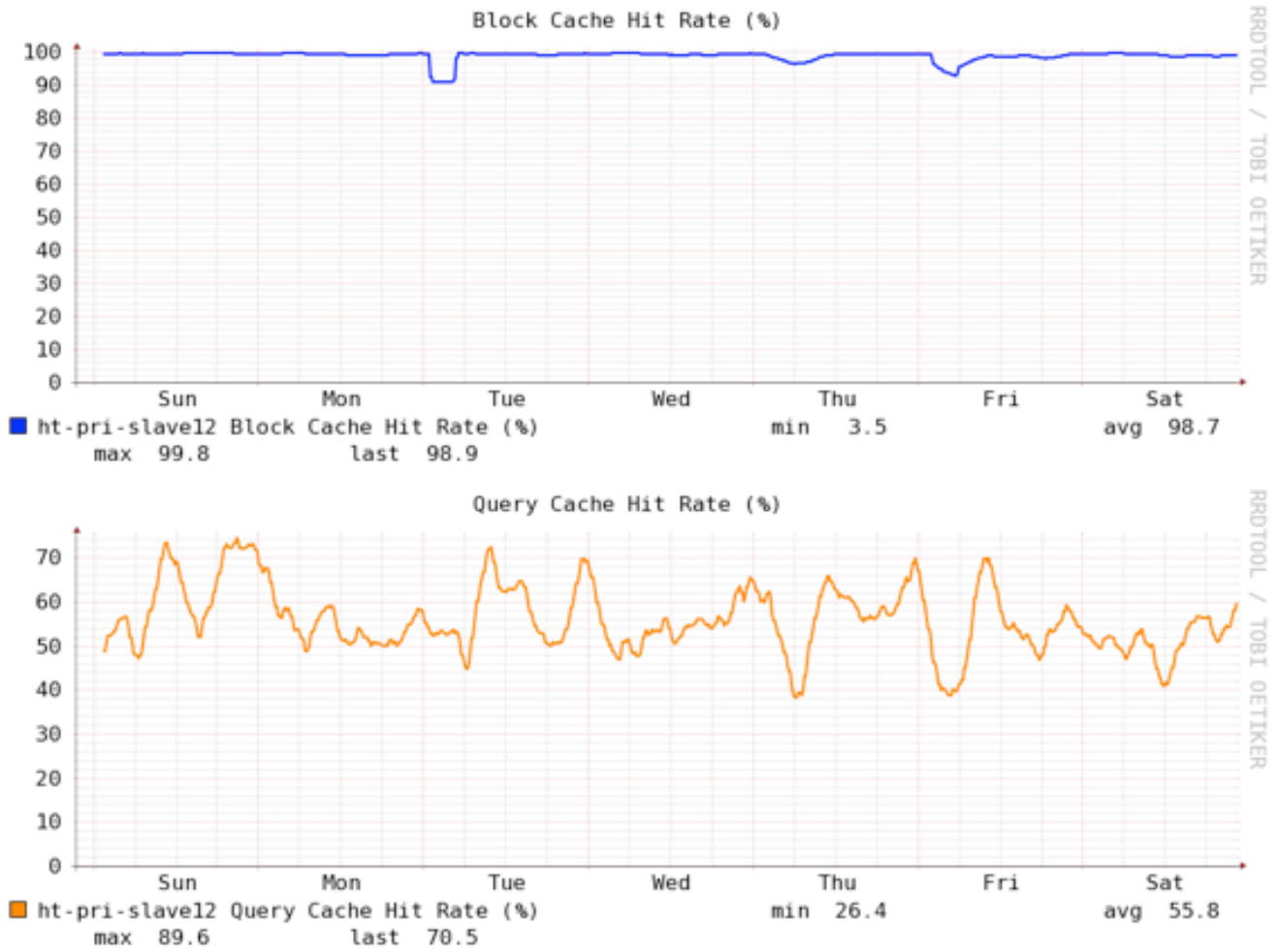

Figure 4. Single machine caching statistics

Figure 5 shows the latency graph for one 24-hours worth of activity on the cluster.

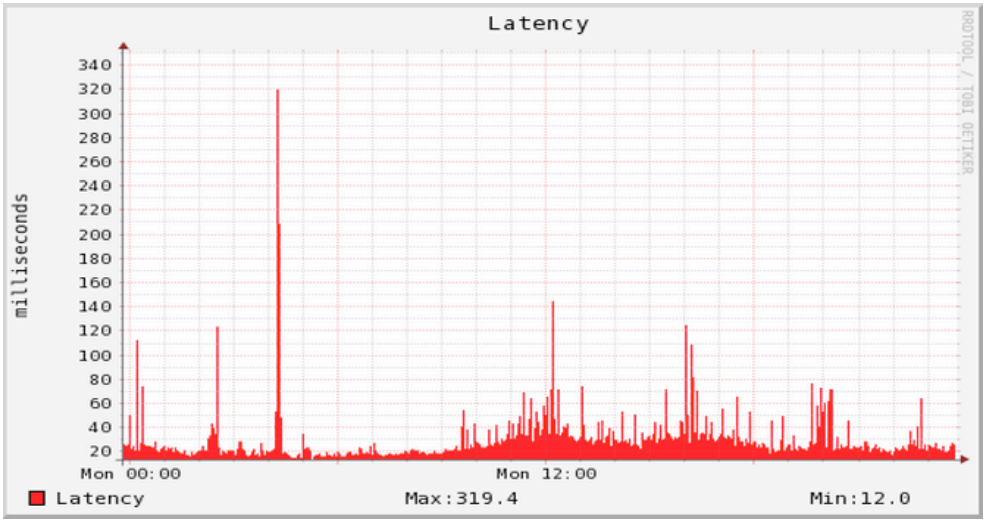

Figure 5. Latency graph for 24-hour period on Reliance cluster

General System Statistics

Each cluster consists of a mix of server hardware due to the fact that machines were provisioned over a period of time. The newer machines have 24GB of RAM, 4 CPU cores, and three 300GB SAS drives mounted JBOD[2]. The older machines have 16GB of RAM, 8 CPU cores, and four 250GB SATA drive configured RAID 0 using a hardware RAID controller.

The newer machines are running CentOS 6.2, while the older machines are running CentOS 5.4. Each cluster is running Cloudera's CDH3u3 version of Hadoop and Hypertable version 0.9.5.6. Figure 6 shows the dstat output of one of the slave machines during off-peak load and Figure 7 shows the dstat output of one of the slave machines during peak load.

----total-cpu-usage----						-dsk/total-		-net/total-		---paging--		---system--	
usr	sys	idl	wai	hiq	siq	read	writ	recv	send	in	out	int	csw
10	2	87	1	0	1	1163k	1698k	0	0	3145B	3304B	4037	5231
3	2	94	2	0	0	0	64k	28M	1304k	0	0	3268	2836
1	0	98	0	0	0	0	168k	57k	1452k	0	0	751	777
1	0	99	0	0	0	0	84k	57k	667k	0	0	878	1094
3	0	96	0	0	0	4096B	200k	126k	987k	0	0	1195	1559
2	1	97	1	0	0	0	32k	72k	599k	0	0	871	1294
1	0	95	4	0	0	0	948k	77k	545k	0	0	847	1024
0	0	99	1	0	0	0	12k	61k	300k	0	0	778	774
3	0	96	0	0	0	0	4096B	644k	920k	0	0	1730	2613
1	0	99	0	0	0	0	56k	64k	245k	0	0	1016	1238
1	0	98	1	0	0	0	40k	114k	231k	0	0	787	922
0	0	98	1	0	0	0	104k	83k	222k	0	0	968	1128

Figure 6. Single machine dstat output (off-peak)

----total-cpu-usage----						-dsk/total-		-net/total-		---paging--		---system--	
usr	sys	idl	wai	hiq	siq	read	writ	recv	send	in	out	int	csw
10	2	87	1	0	1	1133k	1902k	0	0	3744B	3934B	3929	5056
12	3	84	1	0	1	1552k	20k	4935k	10M	0	0	7777	10k
24	4	67	2	0	2	2856k	156k	11M	13M	0	0	13k	16k
15	2	80	1	0	1	2456k	236k	5523k	11M	0	0	8449	9527
17	2	79	1	0	1	1172k	20k	4258k	7736k	0	0	8229	10k
9	2	87	1	0	1	300k	24k	4291k	6673k	0	0	6607	8386
11	3	85	0	0	1	700k	0	4315k	8668k	0	0	7242	10k
20	2	76	1	0	1	384k	60k	7067k	10M	0	0	8748	10k
25	2	70	1	0	1	928k	456k	2098k	6912k	0	0	7279	8376
28	3	65	2	0	2	8656k	20k	3068k	11M	0	0	8925	12k
25	2	70	1	0	1	1784k	40k	3262k	7129k	0	0	7375	9209
12	3	83	1	0	1	708k	0	3853k	8410k	0	0	7716	9905

Figure 7. Single machine dstat output (peak)

[2] Just a Bunch Of Drives.

Future Architecture

The Rediffmail engineering team is currently planning a future refinement of the architecture. The new architecture will consist of a Hypertable cluster that acts as a cache of mail metadata and body snippets. Requests that require data that is older than cache TTL[3] will fall through to a Hypertable cluster that contains the full metadata history and the storage appliance cluster that contains the full message body history. Figure 8 illustrates this new architecture.

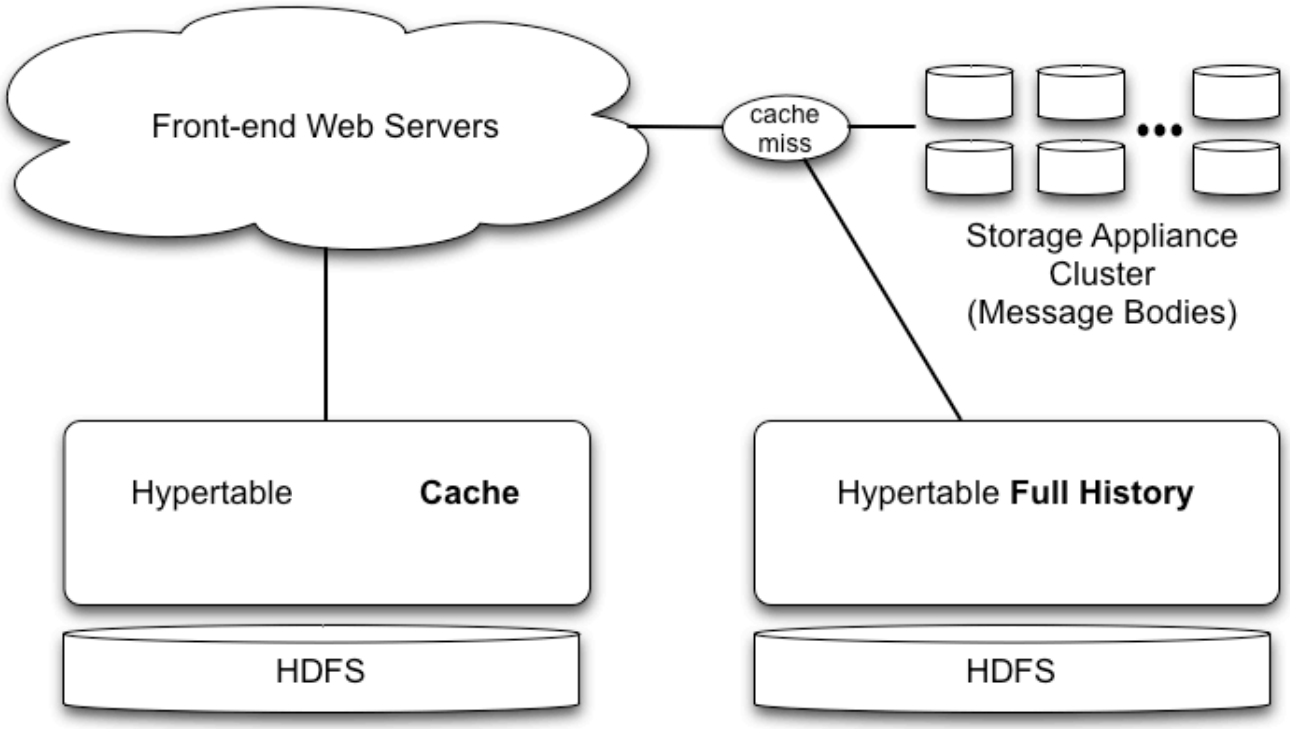

Figure 8. Future Rediffmail architecture

Summary

The Rediffmail email service is an example of a modern, high-load Internet service that has been successfully built on top of Hypertable. The heavy demand that this service has placed on Hypertable has helped to stabilize Hypertable by uncovering a number of issues that have since been addressed. This kind of deployment experience has been invaluable in helping to make Hypertable a rock solid, high performance, scalable database backend capable of supporting high-traffic online services. We thank Rediff.com and the entire Rediffmail engineering team for choosing Hypertable and for all of the valuable feedback they have provided to us.

[3] TTL: Time To Live