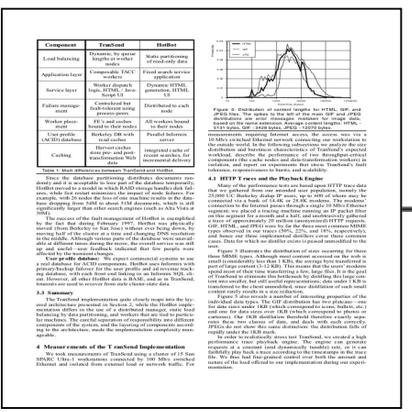
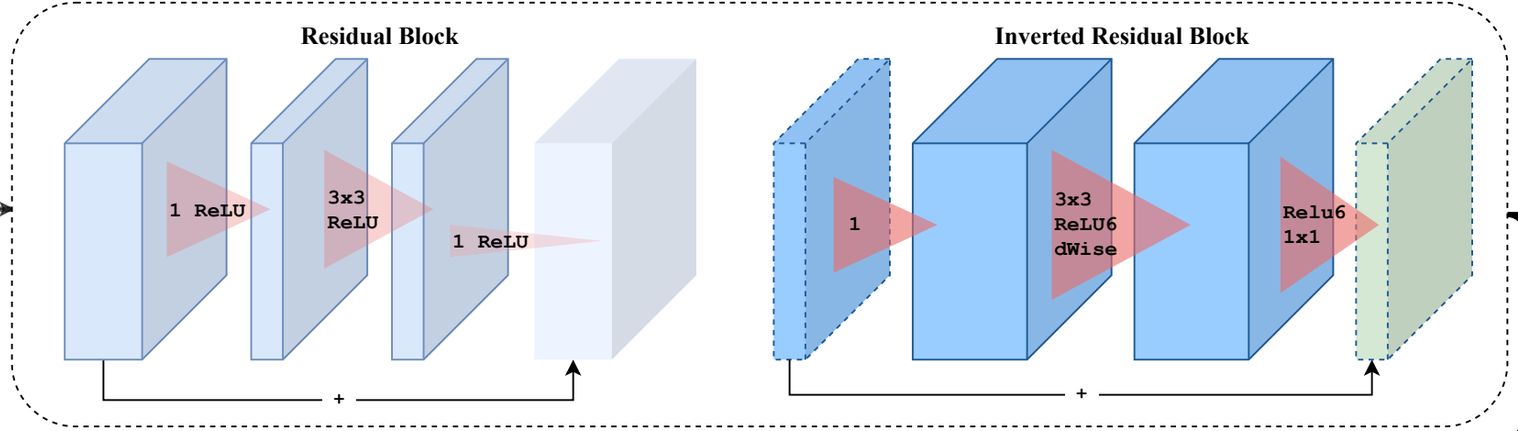
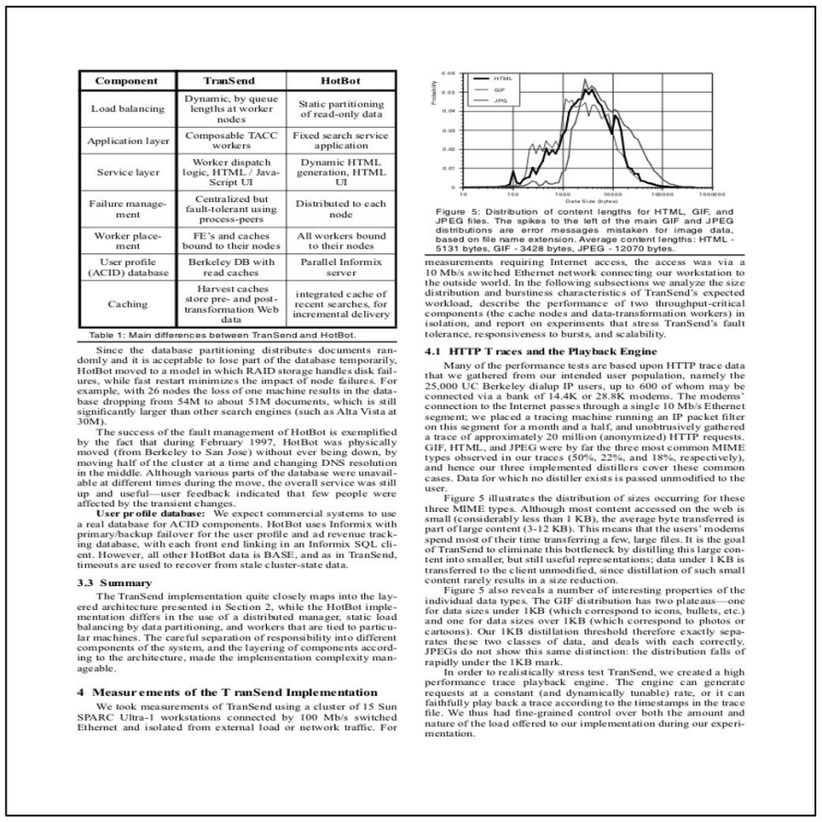


Image



EfficientNet Base Model



Component	TranSend	HotBot
Load balancing	Dynamic, by queue lengths at worker nodes	Static partitioning of read-only data
Application layer	Composable TACC workers	Fixed search service application
Service layer	Worker dispatch logic, HTML / Java-Script UI	Dynamic HTML generation, HTML UI
Failure management	Centralized but fault-tolerant using process-peers	Distributed to each node
Worker placement	FE's and caches bound to their nodes	All workers bound to their nodes
User profile (ACID) database	Berkeley DB with read caches	Parallel Informix server
Caching	Harvest caches store pre- and post-transformation Web data	integrated cache of recent searches, for incremental delivery

Table 1: Main differences between TranSend and HotBot.

Since the database partitioning distributes documents randomly and it is acceptable to lose part of the database temporarily, HotBot moved to a model in which RAID storage handles disk failures, while fast restart minimizes the impact of node failures. For example, with 26 nodes the loss of one machine results in the database dropping from 54M to about 51M documents, which is still significantly larger than other search engines (such as Alta Vista at 30M).

The success of the fault management of HotBot is exemplified by the fact that during February 1997, HotBot was physically moved (from Berkeley to San Jose) without ever being down, by moving half of the cluster at a time and changing DNS resolution in the middle. Although various parts of the database were unavailable at different times during the move, the overall service was still up and useful—user feedback indicated that few people were affected by the transient changes.

User profile database: We expect commercial systems to use a real database for ACID components. HotBot uses Informix with primary/backup failover for the user profile and ad revenue tracking database, with each front end linking in an Informix SQL client. However, all other HotBot data is BASE, and as in TranSend, timeouts are used to recover from stale cluster-state data.

3.3 Summary

The TranSend implementation quite closely maps into the layered architecture presented in Section 2, while the HotBot implementation differs in the use of a distributed manager, static load balancing by data partitioning, and workers that are tied to particular machines. The careful separation of responsibility into different components of the system, and the layering of components according to the architecture, made the implementation complexity manageable.

4 Measurements of the TranSend Implementation

We took measurements of TranSend using a cluster of 15 Sun SPARC Ultra-1 workstations connected by 100 Mb/s switched Ethernet and isolated from external load or network traffic. For

Figure 5: Distribution of content lengths for HTML, GIF, and JPEG files. The spikes to the left of the main GIF and JPEG distributions are error messages mistaken for image data, based on file name extension. Average content lengths: HTML - 5131 bytes, GIF - 3428 bytes, JPEG - 12070 bytes.

measurements requiring Internet access, the access was via a 10 Mb/s switched Ethernet network connecting our workstation to the outside world. In the following subsections we analyze the size distribution and burstiness characteristics of TranSend's expected workload, describe the performance of two throughput-critical components (the cache nodes and data-transformation workers) in isolation, and report on experiments that stress TranSend's fault tolerance, responsiveness to bursts, and scalability.

4.1 HTTP Traces and the Playback Engine

Many of the performance tests are based upon HTTP trace data that we gathered from our intended user population, namely the 25,000 UC Berkeley dialup IP users, up to 600 of whom may be connected via a bank of 14.4K or 28.8K modems. The modems' connection to the Internet passes through a single 10 Mb/s Ethernet segment; we placed a tracing machine running an IP packet filter on this segment for a month and a half, and unobtrusively gathered a trace of approximately 20 million (anonymized) HTTP requests. GIF, HTML, and JPEG were by far the three most common MIME types observed in our traces (50%, 22%, and 18%, respectively), and hence our three implemented distillers cover these common cases. Data for which no distiller exists is passed unmodified to the user.

Figure 5 illustrates the distribution of sizes occurring for these three MIME types. Although most content accessed on the web is small (considerably less than 1 KB), the average byte transferred is part of large content (3–12 KB). This means that the users' modems spend most of their time transferring a few, large files. It is the goal of TranSend to eliminate this bottleneck by distilling this large content into smaller, but still useful representations; data under 1 KB is transferred to the client unmodified, since distillation of such small content rarely results in a size reduction.

Figure 5 also reveals a number of interesting properties of the individual data types. The GIF distribution has two plateaus—one for data sizes under 1KB (which correspond to icons, bullets, etc.) and one for data sizes over 1KB (which correspond to photos or cartoons). The 1KB distillation threshold therefore exactly separates these two classes of data, and deals with each correctly. JPEGs do not show this same distinction: the distribution falls of rapidly under the 1KB mark.

In order to realistically stress test TranSend, we created a high performance trace playback engine. The engine can generate requests at a constant (and dynamically tunable) rate, or it can faithfully play back a trace according to the timestamps in the trace file. We thus had fine-grained control over both the amount and nature of the load offered to our implementation during our experimentation.

Sample Scanned-Document Image