;login:
THE MAGAZINE OF USENIX & SAGE
August 2001 • Volume 26 • Number 5

Special Focus Issue:Clustering

Guest Editor: Joseph L. Kaiser

inside:

USENIX & SAGE
The Advanced Computing Systems Association &
The System Administrators Guild

conference reports

This issue's reports are on the 3rd USENIX Symposium on Internet Technologies and Systems (USITS '01), OUR THANKS TO THE SUMMARIZERS:

Hari Balakrishnan for organizing and editing the reports.
Stergios Anastasiadis
Allen Miu
Anupam Rastogi
Alex C. Snoeren

3rd USENIX Symposium on Internet Technologies and Systems (USITS '01)

San Francisco, California

March 26-28, 2001

KEYNOTE

Interfaces Are Forever, or It's the Hole, Stupid

Scott Guthery, Mobile-Mind, Inc.

Summarized by Allen Miu

IT engineers are the civil engineers of the Internet. IT engineers build systems by bridging very different technologies. Therefore, technology providers must be very clever about building usable interfaces. However, building and dealing with interfaces are notoriously difficult tasks.

Technology providers often have a hard time predicting what the "killer apps" are. For example, Apple II was anticipated as a popular appliance for gaming and keeping recipes at home. However, the killer apps turned out to be home publishing and spreadsheets. The surprise highlights the fact that technology providers are often not the best application providers. Therefore, it is in the technology provider's best interest to create flexible interfaces that lay a versatile platform on which unanticipated killer applications may be created and evolve.

Even when there is a clear direction to build interfaces for creating powerful development platforms, subtle details can cause the greatest successes and failures. One such example is WAP (Wireless Access Protocol), which is an interface designed to bridge mobile telephony and the Internet. Thus far, the development effort has been focused on exposing the Internet to mobile-phone applications. Unfortunately, people have found it very difficult to implement traditional Internet applications such as Web browsers on the mobile phone for various reasons, such as the form factor of the mobile phone and limited bandwidth. However,

Scott Guthery

the application model will become much more powerful and interesting if we flip the interface and try to expose mobile telephony to the Internet. For example, imagine the possibilities of embedding a lightweight HTTP server and a Smartcard chip on the mobile phone. Instantly, the mobile phone becomes a mobile authenticator for the user to conduct transactions on the Internet.

After showing various examples illustrating the importance of interfaces, the talk concluded with an outlook for building future "killer" applications on mobile devices. The speaker metaphorically described mobile computers as remote controls for reality. We should concentrate on building new interfaces that exploit the inherent real-time, interactive capabilities provided by any mobile platform. In fact, many existing applications can already take advantage of the mobile platform. For example, instant text messaging is a long-dominant Internet application. It has been adopted by the GSM network and became one of the most popular applications running on today's cell phones. Online auctioning is another highly successful mobile extension of a popular Internet application. NTT has recently launched a system that under-

writes mobile online auctioning transactions for the cell phone users connected to their DoCoMo network. Like the Internet counterpart, the DoCoMo online auctioning system became a hugely popular application among cell phone users.

We are just beginning to provide "killer apps" for mobile devices. There are still numerous desktop applications that do not have mobile counterparts because we still lack the appropriate "interfaces" that allow engineers to create mobile extensions of these applications.

SESSION: FLEA MARKET

Summarized by Anupam Rastogi

The Age Penalty and Its Effect on Cache Performance

Edith Cohen, AT&T Labs – Research; Haim Kaplan, Tel-Aviv University

This paper addresses the issue of the cache-age penalty in wide area networks. This issue arises when there is a hierarchy of caches between the server and end users of content. Such hierarchies are becoming more common today, with proxy caching, reverse proxies, and Content Delivery Networks (CDNs) being increasingly deployed. Caches determine an expiration time for a cached copy by computing its freshness lifetime and its age. A copy becomes stale when its age exceeds its freshness lifetime, and it must be refreshed, even though it may not have been modified at the higher level. When we have hierarchies of caches, lower levels get data with positive age and, thus, a shorter time-to-live compared to what it would have been if the data had come directly from the origin server. This imposes a penalty, since the cached data would now become stale sooner. This is termed "the age penalty" in the paper.

The age penalty is measured by comparing the performance of a low-level cache that gets its data from another cache with the performance of the same cache if it

Tom Anderson, USITS '01Program Chair

received its data from the origin server. Simulations have been carried out to measure the impact of cache penalty on performance. Trace-based simulations are also used to measure the extent of age penalty for content served by content delivery networks and large caches. The age penalty has been shown to be significant in some cases.

The age penalty can be avoided by maintaining strong consistency between high-level caches and the origin server. But this is expensive and difficult to implement. The future work includes two possible approaches: source selection, where low-level caches can select where they forward a request on a miss, and rejuvenation, where pre-term validation of selected copies is used to decrease age.

Online Markets for Distributed Object Services: The MAJIC System

Lior Levy, Liad Blumrosen, and Noam Nisan, The Hebrew University of Jerusalem

Blumrosen described an infrastructure that performs online auctions for computer services over distributed-object systems. An implementation of such a system over Jini was also presented.

The motivation for the need for such services is that there are many distributed resources on the Internet which belong to different organizations. These organizations must have a motivation to share these resources. This is realized by having an infrastructure where services are paid for (economic paradigm). Examples of some such existing systems are Spawn, Popcorn, and SuperWeb.

The distributed-object paradigm entails that systems on the network encapsulate sharable resources in well-defined interfaces, which can be accessed using Remote Procedure Calls (RPC) / Remote Method Invocation (RMI). The distributed-object paradigm and the economic paradigm are combined to get the new infrastructure, where services are offered for a price, and "customers" can "buy" the service with the best combination of service parameters and price. A service marketplace functions as the object-request broker. For this, each service type has parameters defining the parameter space. Sellers provide quote functions for parameters, which give the price for providing a service for the given parameters. Buyers, or service users, employ a utility function, based on service parameters, which measures the utility of a service for the buyer. Buyers also provide a parameters search engine, which attempts to find the optimal parameters, given the quote functions.

The system functions as follows: the market holds current quotes from all sellers; when it receives a request from a buyer, it attempts to match the request with the best seller and choose the best parameters using the utility function and parameters search engine provided by the buyer.

It is claimed that such economic systems can also provide load balancing automatically if designed correctly. Also, the system architecture allows avoidance of inefficient allocations caused by untruthful sellers.

The MAJIC (Multi-Parameter Auctions for Jini Components) system was presented. MAJIC is built on top of Sun's Jini platform and implements the basic architecture of the system described above. Performance studies showed 15%

overhead per request due to the MAJIC system in a high-load scenario.

More information is available at : *http://www.cs.huji.ac.il/~majic*.

INVITED TALK

Search Engine Experience and Internals

Mike Burrows, Compaq Computer Corporation Systems Research Center

Summarized by Alex C. Snoeren

Mike Burrows gave two related short talks. The first described the internals of a general library he implemented for indexing text, which formed the basis of the search engine known as AltaVista. The second talk was a humorous retrospective on the issues encountered while deploying AltaVista.

He began the first talk by enumerating a set of goals he had in mind for a general purpose indexing library. He made special note that he did not originally have AltaVista in mind when designing the library. One of the key features of the library was its ability, unlike previous similar libraries, to support online updates, that is to continue to support queries during updates, but still provide reasonable update performance.

He described the flat-file storage mechanisms employed by the library, detailing the tricks necessary for rapid processing. In particular, he showed that by supporting a small number of basic operations, the library is able to support an arbitrary set of conjuncts and disjuncts while processing sequentially through the data file. Hence his library provides stream abstractions called Indexed Stream Readers (ISRs) which are powerful enough to provide full-search functionality. Avoiding random access provides enormous implementation efficiency, enabling him to fully utilize the deeply pipelined multi-stage Alpha architecture. To prove his point, he presented a single slide of a highly optimized Alpha assembly language which provided all the basic searching functionality.

After delving into the gory details of how online updates are supported by dividing the dataset into tiers of hash buckets, Burrows presented some (slightly outdated) performance metrics that showed that the search library was entirely CPU bound, and did not fully utilize the memory bus of the Alpha in use, hence additional performance gains could be achieved by adding processors.

Redundancy was the theme of the second talk, in which he described the actual implementation of AltaVista (as of a few years ago), which utilized multiple Alpha workstations as front ends, a few 4-10 CPU Alpha servers as back ends (the system was purposefully designed to showcase DEC's flagship big iron hardware, each of which was configured with 8GB of RAM and 150GB of disk space), and connected them with a FDDI switch. The back-end machines were clustered in groups of 4 to 10 machines, each of which shared their own copy of the index.

Burrows described the great pains taken to ensure failure-free operation of every major subsystem. The hard drives were managed by RAID controllers with spare disks; hot-spare workstations could automatically take over the IP addresses of downed front ends; a cold-spare FDDI switch was kept on-site at all times; each front end could dynamically select alternate back ends; and the entire site could failover to the backup site with a manual DNS change.

Burrows was quick to point out, however, that the seemingly impressive amount of replication was far from sufficient in practice. He wound through a comical tale of disasters large and small, ranging from hardware issues such as the sad truth about self-repairing RAID controllers (performance drops by a factor of two during reconstruction), unexpectedly high file-system corruption rates, and poorly designed interface cards whose pins were all too easy to bend, to software issues such as poor testing, design flaws, and operator error caused by poor interfaces. In addition, he pointed out that spammers and denial of service attacks became quite common as AltaVista grew, and he eventually began spending a great deal of time simply dealing with users abusing the system.

Burrows concluded by calling AltaVista a "success disaster." Generally never staffed by more than two people at a time, the search-engine load grew at a rate of 10-15% per week for over a year, a rate which they found extremely difficult to support. In retrospect, Burrows suggested that if he were to design AltaVista again from scratch, he would prefer to use lots of smaller machines as a back end instead of the small clusters of larger ones.

SESSION: ADAPTATION

Summarized by Stergios Anastasiadis

CANS: Composable Adaptive Network Services Infrastructure

Xiaodong Fu, Weisong Shi, Anatoly Akkerman, and Vijay Karamcheti, New York University

The CANS architecture injects application-specific components in the data path between applications and services. This allows seamless integration of services and devices in diverse networking environments.

The data path notion is extended to include application-specific functionality in the form of different components: drivers and service. The drivers are soft-state mobile code modules that apply operations to data streams passively. Besides the type model used, their efficient composition and reconfiguration requires adherence to restricted interfaces. Services, on the other hand, could be legacy components which use any standard Internet protocol. They could maintain persistent state and do not have to adhere to standard interfaces. Legacy applications can be integrated through an interception layer.

Dynamic changes in system characteristics are handled by three different modes of adaptation. Intracomponent adaptation occurs when services or drivers detect and adapt to environment change by themselves. Data path reconfiguration and error recovery include localized changes to the data path involving insertion, deletion, and reordering of drivers. Replanning is the response to large-scale system variations that require tearing down existing data paths and constructing new ones. The runtime overhead of the system is shown to be negligible.

Related information is available at the project Web site: *http://www.cs.nyu.edu/pdsg*.

Dynamic Host Configuration for Managing Mobility between Public and Private Networks

Allen Miu, MIT Laboratory for Computer Science; Paramvir Bahl, Microsoft Research

The CHOICE network provides authenticated users with high-speed wired or wireless access to the Internet. It supports secure, customized, and accountable services to possibly unknown customers, and it operates seamlessly as mobile clients move across different public and private networks. The underlying protocol is called Protocol for Authorization and Negotiation of Services (PANS).

In a CHOICE network, IP addresses are leased to potential clients through a standard DHCP server, while an authentication database is globally accessible through the Internet. The PANS Authorizer provides controlled access to the authentication service and determines a customized service policy based on the user's credentials. Once the user has been authenticated, the PANS Authorizer generates a session key that is distributed securely to both the PANS Client and the PANS Verifier. From then on, the PANS Client cryptographically tags every transmitted packet with the given session key and sets the PANS Verifier as the default gateway to access the Internet. The PANS Verifier enforces service policy by checking the tag of every transmitted packet and accounts for the client's resource usage by keeping a log of traffic generated by each user.

Mobility between public and private networks is managed by the PANS Autoconfiguration module, which offers service discovery, bootstrapping, protocol configuration, and key management. Configuration parameters are transmitted to the client modules using a beaconing technique that is based on lightweight periodic broadcasting. Multiple verifiers can be used for managing the active key set in order to provide fail-over operation and load balancing. Scalable key distribution is achieved by migrating keys on demand as clients roam between different subnets. Finally, several techniques are described for making denial of service (DoS) and hijacking attacks difficult and detectable.

Details can be found at the project Web site: *http://www.mschoice.com*.

SESSION: COOL HACKS

Summarized by Anupam Rastogi

Alpine: A User-Level Infrastructure for Network Protocol Development

David Ely, Stefan Savage, and David Wetherall, University of Washington

This work addresses the issue of making the task of modifying the network protocol code simpler and less tedious by moving the network stack to user space for development. The premise is that kernel development is a pain, the main factor being the time required for recompiling and rebooting. The networking protocols are currently in the kernel, thus making changes to the network code a pain, too.

There are previous applications like Entrapid, OSKIT, and X-Kernel which address similar issues, but these require changes to the kernel, the applications, or the networking stack. Alpine requires no changes to any of these.

In Alpine, the socket, TCP, and IP layers are moved into a library. A "faux Ethernet" layer is inserted below the IP layer. To the IP layer, it appears like a normal Ethernet driver, but it sends packets to the actual interface using raw sockets and receives using a packet capture library.

Ely also described various challenges in the implementation, involving virtualizing kernel services and virtualizing the system-call interface.

Alpine is shown to perform almost as well as the kernel in terms of throughput up to a bandwidth of around 10Mbps, after which the performance gap starts widening.

The major benefits of this infrastructure are easy source-level debugging of code and quick turnaround between revisions. Alpine can be a useful environment for class projects and application-specific protocol extensions.

Some of the limitations of the current version of Alpine are: it only works for TCP and UDP; the number of sockets usable by Alpine is limited to 100; fork() calls in the application code are not currently supported; and it requires root privileges.

More information about Alpine is available at *http://alpine.cs.washington.edu/*.

Measuring Client-Perceived Response Times on the WWW

Ramakrishnan Rajamony and Mootaz Elnozahy, IBM Austin Research Lab

An important factor that affects the success of a WWW service is the Client-Perceived Response Time (CPRT). If this time is high, the user may get bored and go away. CPRT is the gap between click time and view time, and is the sum of the network delay, the server processing time, and the rendering time taken by the browser. Quantitative information about response times can be important to businesses and Web sites, and may be used to determine whether an improved server

or network infrastructure is required. This paper presents a framework for measuring the actual response time perceived by customers as they access a Web service. The scheme uses HTML and JavaScript, which are supported by most browsers and load fast. An "instrumented" entry to a Web page causes embedded JavaScript within the downloaded page to execute on the client. The client-side script then notes the time at which a subsequent request is made and records it locally, permitting JavaScript downloaded along with the Web page response to compute the delta between the "click" time and the "fully loaded" time. The response time is then sent by the script to a predetermined record-keeping Web site, which can collect the data and process it.

The system has been implemented for the "Wondering Minstrels: Poem of the Day" Web site (*http://www.cs.rice.edu/~rrk/minstrels.html*). Assuming the attention time, i.e., the time after which the user gets bored, to be four seconds, around a sixth of the response times were found to be more than the attention time. The response times have further been studied with respect to accessing top level domains and time of the day.

The limitations of the outlined scheme are that response time can only be measured for an instrumented entry to a Web page — response times to MIME types other than objects embedded within HTML cannot be measured – and that it works only if JavaScript is enabled.

The overhead of instrumentation of Web pages is about 200 bytes per page and 2KBytes per site. The script itself is downloaded only once per site, and it never expires. It is claimed that the extra code has minimal effect on load time.

The advantages of the scheme over other related ones are that no changes to the server, browser, or proxies are required (only JavaScript support in the browser is required) and that the CPRT can be sent to any third party.