conference reports

IN THIS ISSUE

WORLDS '06: 3rd USENIX Workshop on Real, Large Distributed Systems

Seattle, Washington
November 5, 2006

Summarized by Iulia Ion

MEASUREMENT AND MONITORING

Geolocalization on the Internet through Constraint Satisfaction

Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer, Cornell University

Bernard Wong presented a new method of determining the location of Internet hosts, based on constraint satisfaction. With existing techniques such as static databases, users have to get precise city location data from the database and constantly update this information. The authors propose a dynamic approach that uses latency information to determine the location of nodes. The main challenge is caused by network congestion, which might introduce extra delay and therefore makes it difficult to locate node positions.

Bernard presented the Octant system, which extracts constraints based on network measurements, anchors them to the globe, and makes a latency-to-distance relationship. Given the basic assumption that there is a strong correlation between latency and distance, Octant shows the likeliness that a node is a particular distance away. The data is afterward formalized as positive and negative constraints. Given different landmarks, the speaker proposes taking the intersection of the constraints. To deal with overaggressive constraints, different weights can be assigned. Therefore, results have different degrees of confidence, which increases exponentially as the latency reduces linearly. An additional technique used in Octant to address indirect routing and its effects on constraint construction involves piecewise localization, wherein each node is localized depending on another. Some other results-improving techniques are congestion estimation (reducing or increasing the actual latency as measured) and removing regions where people are unlikely to live, such as oceans, rural areas, and deserts.

To evaluate Octant, the authors collected traceroute data among 51 PlanetLab nodes and compared the results to previous geolocalization techniques. GeoLim proved to be the next best system. Octant is able to extract more aggressive constraints with lower error rates. The use of geographic and demographic constraints can further reduce the size of the estimated target region. A demo can be viewed at https://www.cs.cornell.edu/~bwong/octant/query.html.

Audience questions addressed the following: (1) How would Octant perform on DSL nodes of PlanetLab? Bernard answered that although currently Octant uses traceroutes to get the latency information, different measurement techniques could also be used to fit with the PlanetLab DSL nodes model. (2) How well does Octant work outside the United States? Bernard answered that the system has not yet been evaluated. The results would depend on the concentration of nodes in the region (e.g., in Europe it would work better than in Australia). Octant pins down the intermediate nodes in series, but pinning down the first router depends on the previous ones. (3) How accurately are you able to pin down intermediate routers toward the end? Bernard replied that it is difficult to measure accuracy. (4) How often do the final regions end up being disconnected parts? Answer: It happens a lot.

■ ***A Platform for Unobtrusive Measurements on PlanetLab***

Rob Sherwood and Neil Spring, University of Maryland

The talk was given by Rob Sherwood. Rob started by explaining the need for measurements and the benefit that these would bring to many applications such as performance optimization, overlay construction, and network diagnosis. The grand challenge would be to record a day in the life of the Internet. The problems encountered in making such measurements are mainly due to firewalls, abuse reports, and limited bandwidth. Abuse reports occur because exceptional measurement traffic is often considered suspicious and prevents one from measuring everything.

The measurement platform used was Sidecar, which works by injecting probes into normal traffic. Rob presented the tool, how it works, and a couple of quick examples. Basically, Sidecar tracks connections by recording data as it passes by. Neither side knows that there is any sort of probing going on. Sidecar can traverse NATs and firewalls. Probes are retransmissions and require no end-point support. The authors can modify probes for specific measurements such as reducing TTLs, sending probes in trains, and adding IP options.

Rob explained that most often the abuse reports were caused by the application logs. The lesson learned is that when doing traffic generation, one must pay attention to what abuse records are generated. Other problems were caused by clock irregularities (clocks would change rate and jump backward), causal packets reordering, and lag in I/O Systems Calls. The Artrat tool can be used to try to decide from the receiver side where the bottleneck is. The technique uses the IP timestamp option with ICMP echo to measure queuing delay.

Rob was asked whether the measurement platform requires symmetric routes. He answered no. Sidecar doesn't require symmetric routes, but it does assume that the Sidecar listening machine is on both the forward and the reverse path. In their experiments, they simply ran Sidecar on one of the end-hosts to accomplish this.

■ ***ConfiDNS: Leveraging Scale and History to Improve DNS Security***

Lindsey Poole and Vivek S. Pai, Princeton University

The talk was given by Lindsey Poole.

Although cooperative DNS resolver systems, such as CoDNS, have demonstrated improved reliability and performance over standard approaches, their security has been weaker, since any corruption or misbehavior of a single resolver can easily propagate. The authors addressed this weakness in a new system called ConfiDNS, which augments the cooperative lookup process with configurable policies that utilize multisite agreement and per-site lookup histories.

The threat model used focuses on client-side attacks because they are easier to carry out and harder to catch. The advantage of the technique is that there is no need to change the server infrastructure. An incrementally deployable client-side solution can be carried out. The authors evaluated the system and proved that ConfiDNS can provide better security than CoDNS and local resolvers, while retaining the other benefits of CoDNS, such as incremental deployability, improved performance, and higher reliability.

Lindsey was asked whether it would be more useful to source-route DNS lookups from each client and avoid local nameservers entirely. The reasons for not using this approach would be a large increase of lookup traffic to potentially constrained nameservers, and failure to defend against adversaries operating on the local network. ConfiDNS can use encrypted peer traffic to avoid local adversaries, and the peer traffic can be absorbed at peer resolvers, mitigating the impact on remote nameservers. The final observation was that only 10% of failures are client-side, and 30% are server-side.

MANAGING SCARCE RESOURCES

■ ***Resource Management for Global Public Computing: Many Policies Are Better Than (N)one***

Evangelos Kotsovinos, Deutsche Telekom Laboratories; Iulia Ion, International University in Germany; Tim Harris, Microsoft Research Cambridge

The talk was given by Evangelos Kotsovinos.

Management responsibility in global public computing systems is distributed among different individuals and organizations. Such stakeholders can be network administrators, server owners, or infrastructural authorities. Unfortunately, high-level resource management facilities are often absent from public computing systems. Whereas federation is crucial for scalability and cost-efficiency, it introduces important resource management challenges related to expressing policies and managing policy overlaps. Usually, there are different users asking for resources on a server. The challenge is reaching a decision given the different policies and interests of different stakeholders.

Evangelos explained that these overlaps occur because different stakeholders may have different views on how server resources are to be apportioned. The authors proposed a practical system that allows the different stakeholders to independently express federated policies. The Role Based Resource Management system (RBRM) provides mechanisms for resolving potential constraint overlaps automatically and reaches decentralized decisions.

Role-based resource management policies are defined using a Web interface and consist of the following elements: role declarations (a group of users to which common policies apply), role entry conditions, constraint definitions (reservation or usage limitation on a resource that applies to all members of a role), and constraint relationships. When more than one constraint is applicable to a user request for a certain resource there is an overlap; the system needs to determine how much of the resource can be made available to the user. The system supports advanced pattern-matches for existing constraints, together with variable bindings, and generates a replacement constraint for the ones that overlap.

When a user requests resources from a server, resource allocation is determined based on the policies deployed on the server, resource availability, and user properties and credentials. Depending on these inputs, the system allocates the requested resources, denies access, or starts negotiation.

The authors demonstrated experimentally on the XenoServers platform that the system scales gracefully and introduces only a very low performance overhead. RBRM is suitable for operating in realistically large and complex settings. Furthermore, it has been able to express a large set of real-world policies that users of existing platforms have requested.

The addressed questions were: (1) How would the system work for servers or services as opposed to users who do it for themselves? (2) Give us some examples of where it would be useful to negotiate. Would the user be happy with less? Evangelos answered that users will ask for a much higher allocation than they actually need. In such cases they might reconsider it. (3) Do you have a model for online negotiation? Can you do RBRM recursively? The answer was yes, it's supported by the framework. (4) Suppose I gave all the bandwidth away and somebody else comes. Is there a way I can renegotiate resource allocation for running sessions? Evangelos answered that, with the current model, once resources are allocated, they cannot be revoked for the lifetime of the session.

Evangelos invited everyone to attend the demo session scheduled later that day where the authors would present a demo of the running system. Further information can be obtained at http://www.xenoservers.net/.

■ *Optimizing Grid Site Manager Performance with Virtual Machines*

Ludmila Cherkasova and Diwaker Gupta, Hewlett-Packard Labs; Eygene Ryabinkin, Roman Kurakin, and Vladimir Dobretsov, Russian Research Center "Kurchatov Institute"; Amin Vahdat, University of California, San Diego

The talk was given by Lucy Cherkasova and dealt with analysis of Grid workload for the past year from a Tier-2 Resource Center at the RRC Kurchatov Institute (Moscow, Russia). The analysis revealed that a large fraction of Grid jobs have low CPU utilization.

Virtualization can add many desirable properties to Grid computing, such as customized environments, QoS provisioning, and policy-based resource allocation. Additionally, the authors sought to justify an economic and performance incentive to move to a VM-based architecture. Using a simulation model, they showed that a half-size infrastructure augmented with four VMs per node can process 99% of the load executed by the original system in RRC Kurchatov Institute.

The authors described a prototype design built on top of the Xen VMM. The goal of the developed prototype is to integrate VMs in the Grid workflow. The prototype offers a clear separation between mechanisms and policies, was deployed for testing, and was integrated with the Grid workflow in the Kurchatov Institute.

Future work involves getting data from the running system, investigating how often migration is needed, and addressing scalability issues. The authors plan to determine the best migration policies and develop a management suite for enterprise applications.

These follow-up questions were asked: (1) To what extent are the presented resource usage figures representative of global public computing platforms in general? While the data we have are quite representative (spanning more than a year), this is the first Grid workload study of a single data center. We do not have enough publicly available data from other data centers to draw general conclusions.

(2) What fraction of resources are used by short versus long jobs? Study has shown that 20% of overall CPU usage is consumed by the jobs that run less than one day (which represent 92% of all the jobs). Jobs that run around 3 days (representing 4% of all the Grid jobs) are responsible for 42% of overall CPU usage.

For further information check out http://www.hpl.hp.com/personal/Lucy_Cherkasova/projects/grid-vm.html.

PLANETPANEL

■ *Learning from PlanetLab*

Thomas Anderson, University of Washington; Timothy Roscoe, Intel Research Berkeley

The session was started by Thomas Anderson.

Although PlanetLab has been enormously successful in fostering distributed system research, it is not as successful as it could be. Thomas looked at nine important reasons why PlanetLab is not yet the platform for huge distributed systems. PlanetLab is not viral, as is BitTorrent, where people are contributing resources to the system. There are no people contributing resources in order to get access to the system. PlanetLab has enduring limitations. In terms of scalability, although the number of participants is increasing, the number of online nodes remains constant.

Thomas's hypothesis is that not enough has been learned from past experience. He describes nine decisions that have been crucial to PlanetLab's success but which, he argues, should be rethought now that PlanetLab is successful. His points are:

1. Centralizing trust: PLC is a trusted intermediary between node owners and node users. Thomas argues that a single point of trust is unsustainable and that trust should be explicit and flexible. Each site should be able to select its own PLC.
2. Centralizing resource control: PlanetLab Central controls resource allocation. Site administrators have very limited control over what runs on their site, or which jobs get which resources. Thomas argues for better incentives for management.
3. Decentralizing management: By design, PlanetLab provides minimal services to users, which has not worked in practice. The authors argue that encouraging community contributions is inconsistent with centralized trust/control. Instead, we need a set of initial versions of services to demonstrate that the API is complete.
4. Treating bandwidth as free: PlanetLab does not charge users for bandwidth. The authors believe that the lack of accounting offers perverse incentives. Instead, accurate fine-grained cost accounting, visible to applications, is needed.
5. Providing only best effort: PlanetLab provides no resource reservations or resource predictability. There are no limits on the number of jobs that run on each node. The authors' view is that power users crowd out everyone else and that there is room for much better short-term/long-term schedulers.
6. Using Linux as the execution environment: Thomas argues that Linux is the wrong API for distributed systems. The audience argued for the advantages of Linux and gave as examples the existence of top, personal folders, and ssh agents.
7. Distributing OS services: PlanetLab is a distributed OS with few distributed services.
8. Evolving the API: PlanetLab was designed to evolve.
9. Focusing on the machine room: PlanetLab focused on large machines. Thomas brings up the issue of running PlanetLab on PDAs and other small devices.

Thomas concludes that for PlanetLab or GENI to thrive requires large-scale community involvement in defining and improving the platform.

■ *The Lessons of PlanetLab*

Thomas Anderson, University of Washington; Marc Fiuczynski, Princeton University; Michael Freedman, New York University; Rob Ricci, University of Utah

Michael Freedman talked about the problem of misaligned incentives and pointed out that the success of PlanetLab is largely judged by the number of nodes and of slices, not by impact and result.

Another problem with PlanetLab is that slices cannot specify policies. The proposed solution is to provide an ability to easily determine the current status of sessions. However, the concern is that virtualization and isolation of resources is not a panacea.

Michael also argued that decentralized trust/control exists and that sites rarely enforce their local AUPs, and only then in a haphazard manner; therefore he proposes an explicit method for expressing rules and policies.

Marc E. Fiuczynski brought again into discussion the lack of incentive for people to contribute with resources and argued that not enough resources are available, because people do not contribute. Again the discussion was brought to ssh forwarding, which got broken as PlanetLab evolved. The reason for this is that a degree of isolation between researchers was needed, but instead the result was resource isolation. Are there things that PLC could be doing differently to provide incentives or recognize input from users? The future envisages Private PlanetLab (MyPLC) where you can bring up your own private PlanetLab in 30 minutes. MyPLC lets you have complete control over software and API, allowing you to implement your own resource control and have several PlanetLab deployments. In such a context, Marc referenced the work on Role Based Resource Management and expressed his hope of using such a framework to do policy management in a scalable fashion.

Real challenges remain in specifying peering agreements between private PlanetLabs (e.g., what to do when one party is in violation of an agreement) and resource management and control (expressing federated resource management policies and managing conflicts). Finally, the session ended with a reference to the Prisoner's Dilemma: If I buy 10 boxes, what's the benefit to me?

TRENCHES

Summarized by Rik Farrow

■ *Towards Fingerpointing in the Emulab Dynamic Distributed System*

Michael P. Kasick and Priya Narasimhan, Carnegie Mellon University; Kevin Atkinson and Jay Lepreau, University of Utah

Mike Kasick began by explaining that the number of errors produced while Emulab is used overloads operators. Emulab has 1300 users, 430 local nodes, and 740 remote nodes (including PlanetLab) and uses software created over five years comprising 490,000 lines of code, within many scripts. Emulab allows users to create virtual networks as well as load operating systems and applications on the PCs within Emulab.

Since the existing error-reporting system was deficient, they first built tblog, a set of scripts that logged all errors to a database, and included new functions that can be called from within scripts. tblog performs call-chain analysis to determine which of a cascade of error messages contains information about the event that caused the failure to occur. In the previous system, operators would have to examine several email messages and collect context from other log files to determine the triggering event.

Tblog improved the situation, but it did not solve the root problem. Existing scripts produced opaque messages, designed to be human readable but not machine parsible. Their second approach, tbreport, focuses on producing structured error messages that are easily parsible by scripts because they include consistent error types and sufficient context, and always propagate the primary errors, avoiding "me too" error messages. Mike provided some examples of how tbreport helped to improve error analysis through live use on Emulab. David Anderson of Carnegie Mellon asked, "What five-page document should I read now to avoid the problems you found in Emulab?" Mike answered that there isn't such a document, but he advises people to modularize code. When people write code, they focus on the success case and don't includes labels in code so you can grep through it searching for failure points. Also, use RPC mechanisms to do global finger-pointing.

■ *Data Management for Internet-Scale Single-Sign-On*

Sharon E. Perl, Google Inc.; Margo Seltzer, Harvard University and Oracle Corporation

Sharon Perl described her experiences while building a unified login system for Google Accounts that supports both Gmail and AdSense. Google began life without any notion of customer accounts, but work began in April 2002 to create a very simple backend database. Version 2 used the same API, but it replaced the database with a replicated Berkeley DB–based system. Sharon pointed out that "at Google's scale, even rare events happen often." They needed consistency, automated failover, and low latency.

Sharon knew about Paxos, a consensus algorithm designed so that all nodes in a distributed system will agree on a value. After a Google tech talk by Margo Seltzer, Sharon became aware that Berkeley DB could do replicated backups and already provided the simple key-to-value support needed, so she decided to make a Berkeley DB system that worked like Paxos. The production single sign-on system relies on a single master system which holds a lease that allows the master to commit changes. If there is a timeout by the master leaseholder (on the order of seconds), replicants vote to select a new master. The actual data gets replicated across several datacenters. Locking depends on the Chubby Lock mechanism (see the relevant paper in OSDI '06).

A lively Q&A session followed. One person wondered how they would know if they had multiple masters at some point, and Sharon answered that they could look at timestamps in logs and see that sometimes there were two masters. Another person asked how clock skew would affect lease timeouts. Sharon answered that someone within Google who understood hardware came up with a safe timeout value (which is on the order of several seconds, as I found out later). Someone asked about server replacements. Sharon answered that reboots are okay, as no state gets lost, but losing a disk would be a problem.

- ***A Distributed File System for a Wide-Area High Performance Computing Infrastructure***

Edward Walker, University of Texas at Austin

Ed Walker works in the Texas Advanced Computing Lab and uses NSF TeraGrid, a national high-performance computing infrastructure for performing large-scale engineering and scientific problems. TeraGrid currently uses GPFS crossmounts for supporting remote file sharing. But because of operating systems issues, not all sites can use IBM's GPFS, and in a survey of users in 2005, scp was cited as the most important data management tool.

Ed pointed out that many desktops are becoming computation science–capable and that the majority of links within TeraGrid participants have less than 2% utilization, so that bandwidth can be used. He then described XUFS, a userspace overlay that hooks file system calls by interposing a shared object before libc to get transparent file system redirection. XUFS has goals of location transparency (since laptops and even desktops move easily), performance, and private name space, but not file sharing, as his research has shown that scientific computing files are rarely shared (umask of 077). XUFS aggressively uses local caches, and it also performs write-on-close to sync up locally made changes with the remote copy.

In performance testing, XUFS does as well as or better than GPFS in most cases (the exception being smaller files). XUFS has a command-line tool for flushing the local write cache in case of a client crash, and automatic recovery in case of host crashes or network outages. Gunnar Sirer asked about the lack of support for file sharing. Ed answered that out of nearly 2000 GPFS users, only one had changed the default permissions to all group read permissions within directories.