Hybrid Load Balancers for an Effective Multipath Switching System

M.Bharathi Rani [1], S.Suresh Babu[2]

[1]Student, Sri Mittapalli College of Engineering, Tummalapalem Guntur Dist, Andhra Pradesh, India

[2]Assistant Professor, Sri Mittapalli College of Engineering, Tummalapalem Guntur Dist, AndhraPradesh, India

Abstract: Switching is a core technology in a wide variety of communication networks, including the Internet, circuit-switched telephone networks and optical fiber transmission networks. Prior approaches only implemented plain switching between different servers using either packet-based solutions or flow-based hashing algorithms which are not an optimal solution. For better performance previously a load balancing method that implements a Flow Slice (FS) that cuts off each flow into flow slices at every intra flow intervals leading to a larger than a slicing threshold and balances the load on a finer granularity based on that threshold. Although effective it involves multiple hardware implementation costs. We propose to us the hardware driven flow slicing along with a software driven load balancer to reduce implementation costs. The process involves using Resin as the Load Balancer that uses an integrated HTTP proxy cache. This enables a web-tier machine cache results for the backend servers and load re transfer it along less stress paths before estimating the load of a server. A practical implementation of the proposed system validates our claim to deliver better performance using this hybrid architecture.

Index Terms: *Load-Balancing, Multipath Switching, Multistage Multiplane, Flow-Slice*

I. INTRODUCTION

Switching is a core technology in a wide variety of communication networks, including the Internet, circuit-switched telephone networks and optical fiber transmission networks. Program slicing is the computation of the set of programs statements, the program slice that may affect the values at some point of interest, referred to as a slicing criterion. Program slicing can be used in debugging to locate source of errors more easily. Other applications of slicing include software maintenance, optimization, program analysis, and information flow control. The last decade has been a time of rapid development for switching technology in the Internet, with terabit capacity backbone routers becoming commonplace. Slicing techniques have been seeing a rapid development since the original definition by Mark Weiser. At first, slicing was only static, i.e., applied on the source code with no other information than the source code.

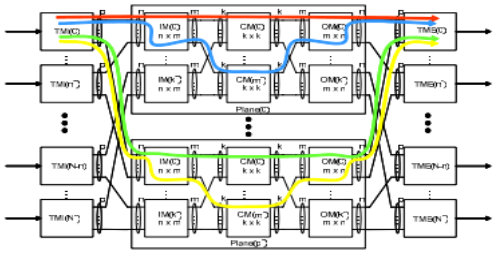

Figure 1. TrueWay Switch Architecture

One of the major as aspect for designing multipath switching in load balancing distribution with incoming traffic analysis across the switching process present in the network. We present a new scheme, namely *Flow-Slice* (FS) that perfectly achieves the three objectives defined above. Our idea is inspired from the observations on tens of broadly located Internet traces that the intra-flow packet intervals are often, say in 40%~50% percentages, larger than the delay upper bound at MPS which is calculated statistically. In this paper we present a technique Resin as a load balancer web server requires a two minimum of two configuration files backend servers and load balancing servers. Due to this web server takes load balancing.

II. BACKGROUND WORK

Large-scale data centers and cloud computing are driving the development of 10 Gb Ethernet switches that are dramatically changing price-performance tradeoffs for network equipment.

These advances are enabling high performance overlay networks that support novel network architectures and services. Previous packet-based solutions either suffer from delay penalties or lead to O (N^2) hardware complexity, hence do not scale to current trends. Flow-based hashing algorithms also perform badly due to the heavy-tailed flow-size distribution. Prior approaches only implemented plain switching between different servers which is not an optimal solution. Traditionally Proposes to implement Flow Slice (FS) that cuts off each flow into flow slices at every Intraflow interval larger than a slicing threshold and balances the load on a finer granularity. The FS scheme achieves comparative load-balancing performance to the optimal one. It also limits the probability of out-of-order packets to a negligible level at the cost of little hardware complexity. MPS internal switching paths needs to meet at least three objectives simultaneously:

- Uniform load sharing.
- Intraflow packet ordering.
- Low timing and hardware complexity.

We measure four performance metrics:

- average packet delay;
- packet loss rate;
- Intraflow packet out-of-order probability; and
- average per-flow delay jitter

This traffic analysis based switching system in the routers that can estimate the load faced by servers and implement routing procedures based on that estimation offer better performance than plain switching systems.

III. PROPOSED WORK

The distinction between "hardware" and "software" load balancers is no longer meaningful. A so-called "hardware" load balancer is a PC class CPU, network interfaces with packet processing capabilities, and some software to bind it all together. A "software" load balancer realized on a good server with modern NICs is the same like a high end hardware load balancer. High-end commercial hardware offerings like F5 or Citrix Netscaler offers:

1. A rich and deep feature set. Their solution is mature and can quickly handle all common needs and some uncommon ones as well.
2. Excellent statistics. Management types love statistics, and network techs realize that stats can be useful in troubleshooting too.

3. A single vendor to choke when something isn't working, i.e. support contract directly with the solution vendor.
4. Lower salary costs. The appliance mostly just works and managing one doesn't take that many hours.

But problem lies in the investment aspect which is quiet expensive. With software load balancers we don't get the opposite, what we get depends on the software we choose and how we go about it. That said, typically we'll see(side effects of software load balancer): Longer time to set up the initial solution. Especially if we need more than just load balancing, fx caching + content rewriting + HA, then setting up open source software takes more manhours. We build it, We own it. If A company sets up open source software load balancers within house techs, then we're 100% responsible for the solution our self. Documentation, upgrade path, disaster recovery etc will all need to be considered and perhaps be implemented by us. The differentiation isn't really on "hardware" versus "software". It is on "buy a proven technology stack as an appliance" versus "builds it yourself". But the key benefit lies in saving the expenditure. A software load balancer implementation contains Using Resin as the Load Balancer. Resin includes a Load Balance Servlet that can balance requests to backend servers. Because it is implemented as a Servlet, this configuration is the most flexible. A site might use 192.168.0.1 as the frontend load balancer, and send all requests for /foo to the backend host 192.168.0.10 and all requests to /bar to the backend host 192.168.0.11. Since Resin has an integrated HTTP proxy cache, the web-tier machine can cache results for the backend servers. Using Resin as the load balancing web server requires a minimum of two configuration files: one for the load balancing server, and one for the backend servers. The front configuration will dispatch to the backend servers, while the backend will actually serve the requests. The web-tier server does the load balancing.

IV. PERFORMANCE ANALYSIS

In this section we describe the following methods as follows:

Load Balancing:

Interflow packet order is natively preserved besetting slicing threshold to the delay upper bound at .Any two packets in the same flow slice cannot be disordered as they are dispatched to the same switching path where processing is guaranteed; and two packets in the same flow but different flow slices will be in order at departure, as the earlier packet will have depart from before the latter packet arrives. Due to the fewer number of active flow slices, the only additional overhead in, the hash table, can be kept rather small, , and placed on-chip to provide ultrafast access speed. This table size depends only on system line rate and will stay unchanged even if scales to more than thousand external ports, thus guarantees system scalability.

Multipath Switching: Through lay-aside Buffer Management module, all packets are virtually queued at the output according to the flow group and the priority class in a hierarchical manner. The output scheduler fetches packets to the output line using information provided by. Packets in the same flow will be virtually buffered in the same queue and scheduled in discipline. Hence, Intraflow packet departure orders hold their arriving orders at the multiplexer. Central-stage parallel switches adopt an output-queued model.

By Theorem, we derive packet delay bound at first stage. We then study delay at second-stage switches. Define native packet delay at stage m of an be delay experienced at stage m on the condition that all the preceding stages immediately send all arrival packets out without delay.

Multistage Multiplane Clos Switches: We consider the Multistage Multiplane Clos-network based switch by Chao et a . It is constructed of five stages of switch modules with top-level architecture similar to a external input/output ports. The first and last stages Clos are composed of input demultiplexers and output multiplexers, respectively, having similar internal structures as those in PPS. Stages 2-4 of M2Clos are constructed by parallel switching planes; however, each plane is no longer formed by a basic switch, but by a three-stage Clos Network to support large port count.

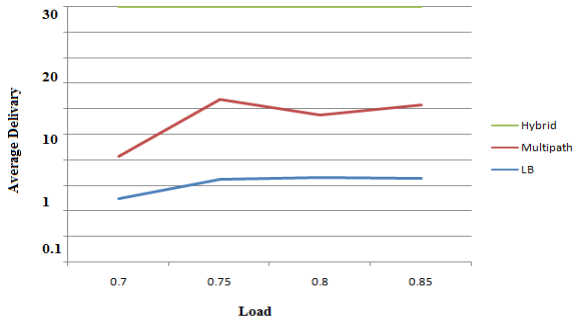

Fig 4: Average delivery with load balancing.

Inside each Clos Network, the first stage is composed by k identical Input Modules. Each IM is an packet switch, with each output link connected to a Central Module. Thus, there are a total of m identical in second stage of the Close networks.

V. CONCLUSION

Flow Slice is a novel load-balancing scheme, that is based on the fact that the Intraflow packet interval is often, say in 40-50 percent, larger than the delay upper bound at MPS. Due to three positive properties of flow slice, our scheme achieves good load-balancing uniformity with little hardware overhead and O(1) timing complexity. By calculating delay bounds at three popular MPSes, we show that when the slicing threshold is set to the smallest admissible value at 1-4 ms, the FS scheme can achieve optimal performance while keeping the Intraflow packet out-of-order probability negligible (below 10 6), given an internal speedup up to two. Our results are also validated through trace-driven prototype simulations under highly bursty traffic patterns. FS scheme is validated in switches without class-based queues. As QoS provisioning is also critical in switch designs, one of our future works will be studying FS performance under QoS conditions.

VI. REFERENCES

1)http://en.wikipedia.org/wiki/Program_slicing

2) Lei Shi1, Bin Liu1, Changhua Sun1, Zhengyu Yin1, Laxmi Bhuyan2" Flow-Slice: A Novel Load-Balancing Scheme for Multi-Path Switching Systems", *ANCS'07,* December 3–4, 2007, Orlando, Florida, USA.ACM 978-1-59593-945-6/07/0012.

3) Li, W. ; Bin Liu ; Xiaojun Wang," Flow Mapping In The Load Balancing Parallel Packet Switches", High Performance Switching And Routing, 2005. HPSR. 2005 Workshop On.

4) Bin Liu ; Changhua Sun ; Zhengyu Yin ; Laxmi Bhuyan ; Chao, H.J, "Load-Balancing Multipath Switching System with Flow Slice ", Computers, IEEE Transactions on (Volume:61 , Issue: 3), March 2012.

5) Smiljanic, A. "Performance Of Load Balancing Algorithms In Clos Packet Switches ",High Performance Switching And Routing, 2004. HPSR. 2004 Workshop On.