An Analysis of Dynamic Flow Scheduling for Data Center Networks

An analysis of the paper "Hedera: Dynamic Flow Scheduling for Data Center Networks" [1]

Submitted by: Richard A. Kramer, Oregon State University

Abstract

This paper provides an analysis of the article "Hedera: Dynamic Flow Scheduling for Data Center Networks" [1] and the related subject matter ("Hedera" and/or "Hedera's" henceforth). Written in 2010, "Hedera's" goal is to optimize dynamic flow scheduling in data centers.

In short, "Hedera" presents:

> "[A] scalable, dynamic flow scheduling system that adaptively schedules a multi-stage switching fabric to efficiently utilize aggregate network resources"

(*see* "Hedera" at Abstract).

This paper provides a detailed analysis of "Hedera" in the following sections:

1. **The Problems to be Solved**
2. **The Proposed Solutions**
3. **Strengths of the Proposed Solutions / Effectiveness**
4. **Weaknesses of the Proposed Solutions / Additional Considerations**
5. **Conclusion**

Overall, "Hedera" proves effective based on solid simulation and testbed results for a given set of assumptions and constraints (e.g. large flows, commodity switches, packet transmission dynamics, etc.). The solid simulation results indicate that "Hedera's" performance is 96% of optimal (e.g. as compared to non-blocking throughput) and a 113% improvement over ECMP static hashing [2]. Further, "Hedera's" testbed results outperformed ECMP in a wide variety of test cases. Thus, "Hedera" appears to be effective in (1) paving the way for improved dynamic flow scheduling for data center networks and (2) opening up new possibilities for additional research to obtained even better dynamic flow scheduling performance within data center networks.

1. The Problems to be Solved

"Hedera" provides an analysis of the *problems* confronting "Dynamic Flow Scheduling for Data Center Networks" (*id.* at Title). The problems confronting data center networks include:

1. Data center designers have no way of knowing how data center network demand and workloads will vary over time, thus designers need a dynamic solution that can adapt over time.
2. The data center network system must operate using commercially available commodity system components without requiring protocols and/or software changes.
3. Inter-rack network bottlenecks caused by virtualization technology [3] including separate physical servers used to multiplex customers across multiple machines make it difficult to ensure the virtualization instances will run on the same physical rack.

The above problems are compounded by the fact that multi-rooted tree like networks structures provide many paths between host pairs, yet have decreasing aggregate bandwidth when moving up to the top of the hierarchical tree structure. At the time "Hedera" was written, ECMP with static hashing [2] was a prevalent means to route data flows, yet ECMP with static hashing resulting in "collisions overwhelmed switch buffers", thus depleted network system performance (*see* "Hedera" at Section 1).

"Hedera" addresses the problems noted above by collecting flow information, dynamically computing non-conflicting paths for the data flows, and then programming commodity switches to reroute the traffic according to the newly computed non-conflicting paths.

2. The Proposed Solutions

"Hedera" proposes solutions (e.g. algorithms) to optimize data flow network performance. Specifically "Hedera" proposes two algorithms to provide dynamic flow scheduling improvements:

1. Global First Fit ("GFF").
2. Simulated Annealing ("SA").

The proposed solutions are targeted for implementation on commodity switches and unmodified hosts (e.g. off-the-shelf components).

"Hedera's" proposed solutions are targeted at improving network performance within the data center network *by providing optimized/improved dynamic flow schedules to the switch fabric, to further compliment ECMP* (*see id.* at Section 2.2). More specifically, "Hedera" performs the following tasks:

1. Detects large flows / Estimate the natural demand for large flows within the system.
2. Computes "good" non-conflicting paths for the large flows.
3. Installs the new computed "good" paths to accommodate the large flows within the switch fabric / instructs the switches to reroute.

"Hedera" provides an in-depth analysis of the abovementioned GFF and SA algorithms to accomplish the objective of optimized data flow network performance. An overview of the GFF and SA algorithms is as follows:

Global First Fit ("GFF"): As the name indicates, when a new flow is detected, the GFF algorithm *globally* searches for the *first fitting* path that can accommodate the new flow and then reserves the capacity within the system to accommodate the new flow. As a result, the system must maintain a record of the reserved capacity of every link within the network and release the reserved capacity when the flow expires.

Simulated Annealing ("SA"): Annealing is the process of heating a material (e.g. adding energy) such as metal and then allowing the material to slowly cool (e.g. the decrementing or decreasing of temperature). As the *"simulated annealing"* name implies, "Hedera" *simulates annealing* as follows:

1. The analogous initial annealing heating "energy" ("E") is equated *to the total exceeded network capacity over all links*.
2. The analogous annealing decrementing / decreasing of "temperature" ("T") is equated *to the number of iterations that the SA algorithm "for loop" is executed.*
3. During each iteration of the SA "for loop" (e.g. decrease in temperature), the neighboring "state" ("s", mappings of destination hosts to core switches) available capacity is compared to the current selected state, seeking the lowest "energy". For each iteration, when a lower neighboring energy value and state ("e_N" and "S_N") is seen, the algorithm stores the better neighboring energy value and state as the "best" lowest energy and state ("e_B" and "s_B").
4. Whereas for the next iteration and assignment of the state "s" to neighboring state "s_n" is further determined by a probabilistic based function "P" and a randomizer, seeking a "reasonable" best case.

Thus SA (as compared to GFF) is an algorithm to iteratively seek, *not just the "first fit[ting]" link that can accommodate the large flow, but also a link that is reasonably best suited to accommodate the large flow. I find SA to be an important improvement over GFF for this very reason.*

3. Strengths of the Proposed Solutions / Effectiveness

In my opinion, "Hedera" proves effective related to a number of important factors to improve dynamic flow scheduling within data center networks. The areas that I found that "Hedera" proves effective include:

1. Ease of Implementation
2. Favorable Simulation Results
3. Favorable Testbed Results
4. Ability to Optimize Algorithms Based on Need

Ease of Implementation: "Hedera's" GFF and SA algorithms are very simple to implement and therefore allow for implementation on an FPGA which is critical to real world success. In fact, "Hedera" was implemented using the commercially available "NetFPGA 4-port GigE PCI card switch[]" [4] as shown below which includes a XILINX Vertex II FPGA.

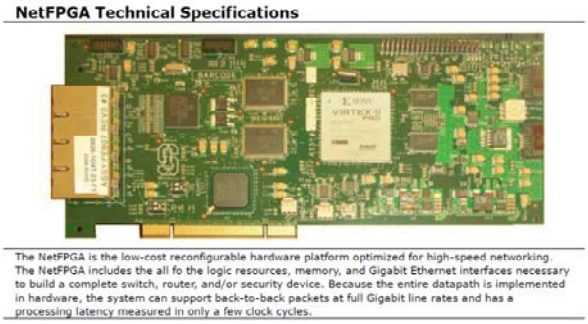

Figure 1: NetFPGA Technical Specifications [4]

Favorable Simulation Results: To prove its effectiveness, "Hedera" provides simulation results for the implementation of GFF and SA on an 8,192 host data center. The simulation delivered 96% of optimal performance and a 113% improvement over static load balancing methods such as ECMP static hashing [2] (*see id.* at "Hedera" Abstract). The simulation results included a well thought-out strategy to cover many scenarios for network traffic including TCP slow start and AIMD (*see* "Hedera" Section 5.5 and "Glossary of Terms Used" below).

Ability to Optimize Algorithms Based on Need: Further, when comparing GFF to SA, a number of advantages and disadvantages between the two algorithms were recognized, thus allowing for system design choices based on overall system needs. The advantage and disadvantages for GFF as compared to SA are as follows:

Global First Fit ("GFF"):

- **Fast processing time:** With GFF, flows can be rerouted quicker when the following equation is true:

  $Process\_Time_{(GFF)}$ [a function of $(k/2)^2$] < $Process\_Time_{(SA)}$ [a function of f_{ave}]

  Where "k" is the number of switch ports and f_{ave} is the average number of flows.

Thus in a system with a large number of average flows and "k" is comparatively low per the above equation, GFF will process faster.

As discussed below a disadvantage of the GFF algorithm is that GFF finds the first path available to accommodate a large data flow versus a path that may be more optimal (*see* discussion "Reasonably best suited path versus first path" below).

Simulated Annealing ("SA"):

- **Reasonably best suited path versus first path:** The SA algorithm is an iterative process and does not seek to find the absolute best link, *but rather a link that is reasonably best suited to accommodate the large flow. Finding the reasonably best suited path for the large flow is an improvement as compared to GFF because GFF simply assigns for first path that can accommodate the flow, thus not necessarily the reasonable best suited path.*

- **Slower processing time:** The SA algorithm is an iterative process using flow demand data from the prior states in order to predict the reasonably best suited future state data path(s). As a result, the time to process SA is dependent on the number of average flows to a given host. Because of this relationship, *SA generally takes longer to process as compared to GFF* (*see* "$Process\_Time_{(GFF)}$" above).

Each of the above factors are important because they allow for system design choices when designing a data center based on the types of services proved.

Favorable Testbed Results: To proves "Hedera's" effectiveness, "Hedera" also provides testbed results based on using 16 open socket hosts and a non-blocking 48-port gigabit Ethernet switch. The switches are based on the abovementioned NetFPGA platform, which further implements OpenFlow [5]. A wide range of traffic patterns were tested on the testbed including a full data shuffle to simulate MapReduce/HADOOP operations [6].

From this, "Hedera" provided superior performance over ECMP in all cases, typically by a significant margin (*see* "Hedera" at Fig. 9).

4. Weaknesses of the Proposed Solutions / Additional Considerations

While I found "Hedera" to be highly effective in providing dynamic flow scheduling for data center networks, because the subject matter is broad with an infinitely wide solution set, I did understandably find a number of weaknesses (*see* Section "Weaknesses" below) and areas to expand upon (*see* Section "Additional Considerations", below).

Weaknesses: "Hedera" restricts its analysis and/or accepts a number of assumptions that may not be accurate in the real world. These assumptions are important to consider when designing real-world systems. Specifically, "Hedera's" limitations and/or assumptions include the following factors to consider when designing real world systems:

1. "Hedera" adopts the use of "cheap edge switches" with many vertical layers versus more expensive horizontally wide architectures using more expensive routers (*see* "Hedera" at Section 2).
2. Hedera only supports "large flows" (*see id.* at Section 3) that exceed a threshold of "100 Mbps… 10% of each host's 1GigE link" (*see id.* at Sections 3.1). Further only a RTT of 100 µs is mentioned in "Hedera".
3. Bandwidth is assumed to be limited only by the sender's / receiver's maximum NIC maximum capacity (*see id.* at Section 4.2).

Further, the effectiveness of "Hedera" related to a number of real world factors were not evaluated, including:

1. There was no modeling of individual packets (and thus packet loss and retransmission timeouts). As a result, "Hedera" assumes that TCP Reno and New Reno [7][8][9] would perform worse, but it is not known how much "worse".
2. Inter-flow dynamics were not simulated, however a number of scenarios such as a data shuffle were tested on the testbed.

While I believe a number of the above noted factors are reasonable compromises to allow for a manageable solution set to be evaluated, on the other hand the omission of fully evaluating inter-flow dynamics between hosts appears to be a potentially problematic omission. I believe the omission of fully evaluating inter-flow dynamics is important because "Hedera's" performance could potentially be impacted by various protocols, collision schemes, and applications / services (both present and future).

Additional Considerations: From "Hedera", a number of improvements are brought to light as possible extensions, including:

First, the "Hedera" simulation and testbed scenarios each modeled large flows, whereas the future state of the flow was unknown to the GFF and SA algorithms. *One possible extension* to "Hedera's" algorithms may be to characterize the various applications / services provided by a data center (for example, build a characteristic load demand table for each application based on the historical demand for a given application / service), thus the next "time tick" for the flow would be predicatively known rather than unknown.

From this, given the future predictive state of the flow, an opportunity appears to exist to improve "Hedera" to support optimization of dynamic flow scheduling based on the above mentioned prediction data. For example, information may be predicatively known for a given application *on how large the flow will be* (both time duration and accumulative data size), thus allowing for more accurate scheduling reservations to be made.

Second, it would seem that the evaluation and better simulation of inter-flows would be an area worth researching further.

Third, it would seem useful to test the scalability of "Hedera" against other flow sizes in addition to 1 GigE and a 10% threshold, with the variables being both the flow's upper bandwidth limit and the flow's threshold limit to qualify the flow as a "large flow". As the flow size decreases, it would be of interest to evaluate the effectiveness of "Hedera" as network overhead

plays a potentially larger role in relation to the overall flow size.

Conclusion

In all, I found "Hedera" to be very insightful and compressive paper related to the subject of dynamic flow scheduling for data center networks. "Hedera's" analysis covers the spectrum of defining the problem, identifying potential solutions, and then qualifying the potential solutions both through simulation as well as on an actual test bed. Subsequently, I found that "Hedera" had many strengths and few weaknesses.

References

[1] Mohammad Al-Fares, et al. Hedera: Dynamic Flow Scheduling for Data Center Networks. NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation, 2010.

[2] Zhiruo Cao, Zheng Wang, and Ellen W. Zegura. Performance of hashing based schemes for internet load balancing. INFOCOM, pages 332–341, 2000

[3] Cisco Systems Inc. Cisco Data Center Infrastructure 2.5 Design Guide. Cisco Validated Design I. December 6, 2007

[4] NetFPGA Technical Specifications. URL: http://netfpga.org/1G_specs.html. Last visited August 17, 2015.

[5] OpenFlow.org, What is OpenFlow? URL: http://archive.openflow.org/wp/learnmore/. Last visited August 24, 2015.

[6] Jeffery Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Google, Inc., 2004.

[7] S. Floyd, T. Henderson. The New Reno Modification to TCP's Fast Recovery Algorithm. RFC 2582, 1999.

[8] Kevin Fall and Sally Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. Computer Communication Review, July 1996.

[9] Stevens, W., Allman, M. and V. Paxson. TCP Congestion Control. RFC 2581, April 1999.

Glossary of Terms Used

AIMD: Additive-Increase / Multiplicative-Decrease

ECMP: Equal-Cost Multi-Path

Bijective: Bijective function or one-to-one correspondence is a function between the elements of two sets, where every element of one set is paired with exactly one element of the other set, and every element of the other set is paired with exactly one element of the first set.

MapReduce / HADOOP: MapReduce is a programming model and an associated implementation for processing and generating large data sets [6].

NetFPGA: The NetFPGA is the low-cost reconfigurable hardware platform optimized for high-speed networking. The NetFPGA includes all logic resources, memory, and Gigabit Ethernet interfaces necessary to build a complete switch, router, and/or security device. Because the entire data path is implemented in hardware, the system can support back-to-back packets at full Gigabit line rates and has a processing latency measured in only a few clock cycles [4].

New Reno: A TCP/IP congestion control and avoidance mechanism. New Reno improves upon TCP Reno (*see* "TCP Reno" below) by adding the ability to detect multiple packet losses and thus it is much more efficient in the event of multiple packet losses. [7]

OpenFlow: OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks. OpenFlow is added as a feature to commercial Ethernet switches, routers and wireless access points – and provides a standardized hook to allow researchers to run experiments, without requiring vendors to expose the internal workings of their network devices. OpenFlow is currently being implemented by major vendors, with OpenFlow-enabled switches now commercially available [5].

RTT: Round Trip Time

Static Hashing (ECMP): A scheme of hashing the IP destination modulo the outgoing links "N" expresses as: H (Destination IP Address) = Destination IP Address mod N [2].

TCP Reno: A TCP/IP congestion control and avoidance mechanism that uses the basic principle of slow starts and a coarse grain re-transmit time and adds additional intelligence so that lost packets are detected early and that the pipeline is not emptied every time a packet is lost [8][9].