Multi-Core HTB for Bandwidth Sharing

Chengjun Jia[†] Zhe Fu[†] Xiaohe Hu[†] Shui Cao[*] Liang Wang[*] Jun Li[†] [◊]
[†] Tsinghua University [*] Huawei Technologies
[◊] Beijing National Research Center for Information Science and Technology

ABSTRACT

Rate limiting with bandwidth-sharing is important and widely used in various scenarios such as multi-tenant cloud. We propose a new rate limiting architecture which fully utilizes the parallel computing capabilities on the multi-core platforms. With *Bandwidth Allocator* allocating the bandwidth to *Rate Limiter*s, we expand HTB into mHTB, which could provide scalable and flexible rate limiting on multi-core platforms.

CCS CONCEPTS

• **Computer systems organization** → **Multicore architectures**;

KEYWORDS

Multicore, Bandwidth Sharing, Run To Completion

ACM Reference Format:
Chengjun Jia[†] Zhe Fu[†] Xiaohe Hu[†] Shui Cao[*] Liang Wang[*] Jun Li[†] [◊] [†] Tsinghua University [*] Huawei Technologies
[◊] Beijing National Research Center for Information Science and Technology . 2018. Multi-Core HTB for Bandwidth Sharing. In *ANCS '18: Symposium on Architectures for Networking and Communications Systems, July 23–24, 2018, Ithaca, NY, USA.* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/3230718.3232111

1 INTRODUCTION

In cloud networks, it is important to control the rate of different classes of network flows and make full use of bandwidth resources for the providers in the meanwhile. It is similar with the tradeoff between QoS (Quality of Service) and link-sharing in traditional networks. Hierarchical link sharing algorithms, such as CBQ[5], H-PFQ[4] and HFS[10], can help.

However, 10Gbps/40Gbps NICs are common in datacenters at present and 100Gbps NICs will be large-scale deployed in the future. Single 2-3GHz CPU core is not competent for the high bandwidth[3], and thus network processing with multi-core platforms becomes prevailing. With the usage of lock-free FIFOs, HTB (Hierarchical Token Buckets) [1], a certain H-PFQ algorithm, turns into a 2-stage pipeline on the multi-core platform, with throughput improving from 0.5Gbps to 2Gbps[6].

Apart from the pipeline design, RTC (Run To Completion) can take full advantage of the performance of multi-core platforms meanwhile. We can use multiple rate limiters for performance improvement, while constraints exist between rate limiters for the shared bandwidth. In the other word, there is a resource sharing problem among multiple rate limiters. The typical methods to allocate the resource are:

1. **Uniform Allocation**. A central unified scheduler allocates the resources to each node. For example, the shared bandwidth of parent node is assigned to the chidren node by WFQ with priority in HTB.
2. **Distributed Competition**. With the design of nodes' competition strategy, the expected allocation scheme can be achieved. For example, TCP traffic control, bandwidth allocation in cloud [7] [8] and the *qos_sched* of DPDK[2] are designed in this way.

Although distributed competition is easy to expand, it is inflexbile to implement customized traffic control strategies and to ensure the effectiveness of bandwidth allocation. In this paper, we extend the rate limiting method to multi-core platform with uniform allocation (scalability due to lock-free design) and realize mHTB (multi-core HTB) by decomposing HTB tree structure in the allocator (flexibility due to hierarchical sharing). We elaborate our design in next section.

2 DESIGN

As shown in Figure 1, our design has two kinds of modules:

1. *Rate Limiter*: Reorganizing the sequence the received packets and limiting the traffic rate.
2. *Bandwidth Allocator*: Collecting the information from the *Rate limiters* and delivering the configuration.

To achieve the basic features of HTB, mHTB works as following:

ANCS '18, July 23–24, 2018, Ithaca, NY, USA

ACM ISBN 978-1-4503-5902-3/18/07…$15.00
https://doi.org/10.1145/3230718.3232111

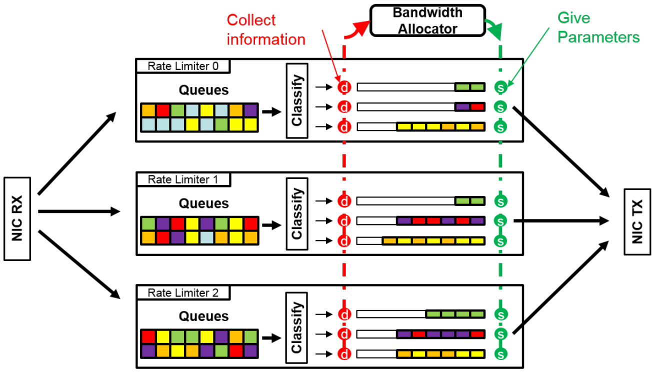

Figure 1: The architecture of multi-core rate limiting

(1) *Rate Limiter* handles traffic with DRR[9].
(2) *Rate Limiter* maintains a sensor to get the actual network traffic rate with EWMA.
(3) *Bandwidth Allocator* issues commands to some *Rate Limiter*s based on the information of the past, of the allocation scheme and of *Rate limiter*s.

We assume four parameters for each node in the HTB tree for the calculation of bandwidth allocation:

(1) D (Demand rate): Actual network traffic rate.
(2) S (Supply rate): The allocation of bandwidth.
(3) AR (Assured Rate): The guaranteed bandwidth.
(4) CR (Ceil Rate): The upper bandwidth limit.

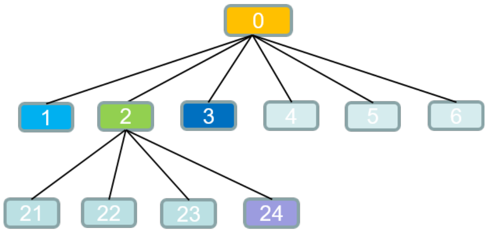

Figure 2: The HTB tree example

With the HTB tree in Figure 2 as an example, define $A = \{21, 22, 23, 24\}, B = \{1, 3, 4, 5, 6\}$. The principle of HTB could be expressed by linear programming of LP1. In LP1, D_i, AR_i and CR_i ($\forall i \in A \cup B$) are known and S_i can be solved easily.

$$\begin{aligned} max \quad & \sum_{i \in A \cup B} S_i && \text{(LP1)} \\ \text{s.t.} \quad & \min\{D_i, AR_i\} \le S_i \le CR_i, \quad \forall i \in B && (1) \\ & \min\{\sum_{i \in A} D_i, AR_2\} \le \sum_{i \in A} S_i \le CR_2 && (2) \\ & \min\{\sum_{i \in A \cup B} D_i, AR_0\} \le \sum_{i \in A \cup B} S_i \le CR_0 && (3) \end{aligned}$$

method	1 core	2 cores	3 cores
HTB	4.2Mpps	7.0Mpps	6.5Mpps
qos_sched	-	7.4Mpps	7.4Mpps
mHTB	7.4Mpps	14.4Mpps	14.4Mpps

Table 1: The throughput of different methods

method	1 Level	2 Levels	3 Levels
HTB with 1 core	7.5Mpps	5.5Mpps	4.2Mpps
HTB with 2 cores	14.4Mpps	7.4Mpps	5.9Mpps
mHTB with 2 cores	14.4Mpps	14.4Mpps	14.4Mpps

Table 2: The throughput for different HTB levels

3 EVALUATION

We use two machines with CPU: Xeon E3-1241 v3 @ 3.50GHz and NIC: Intel Corporation Ethernet Controller X710 for 10GbE SFP+. One machine generated 64B packets of different destination IPs with Pktgen 3.4.2 while the other ran the rate limit program in DPDK stable 16.11.3.

To compare the performance, we use the *qos_sched* demo in DPDK and implement a naive HTB with spinlock. We test the throughput with different core numbers in Table-1 and with different HTB trees in Table-2. From the table, we can conclude that mHTB can achieve higher throughput.

For a three-level HTB tree with 1024 leaf nodes and 32 inner nodes, the complete calculation for LP1 needs 100K CPU cycles or 0.05ms for a 2GHz CPU, damaging the dynamic response of mHTB. If the traffic rate jumps in a large scale, packet loss may occur.

4 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel traffic control architecture which implements high performance sharing-enabled rate limiting on multi-core platforms. Preliminary experiments demonstrate that our method is capable in 10Gbps network environment and achieves more than 200% improvement compared to existing methods. Our future work includes: (1) evaluation on dynamic response time, (2) removing the sensor in *Rate Limiter* by concluding D from the average queue length, the number of drop packets or the idle time. (3) simplifing the calculation process of LP1.

REFERENCES

[1] http://luxik.cdi.cz/~devik/qos/htb/.
[2] http://dpdk.org.
[3] Soumya Antony, Simy Antony, AS Beegom, and MS Rajasree. 2012. HTB based packet scheduler for cloud computing. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*. ACM, 656–660.
[4] Jon CR Bennett and Hui Zhang. 1997. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking (TON)* 5, 5 (1997), 675–689.

[5] Sally Floyd and Van Jacobson. 1995. Link-sharing and resource management models for packet networks. *IEEE/ACM transactions on Networking* 3, 4 (1995), 365–386.

[6] Zheng Li, Nenghai Yu, and Zhuo Hao. 2010. A novel parallel traffic control mechanism for cloud computing. *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010* (2010), 376–382. https://doi.org/10.1109/CloudCom.2010.9

[7] Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C Snoeren. 2007. Cloud control with distributed rate limiting. *ACM SIGCOMM Computer Communication Review* 37, 4 (2007), 337–348.

[8] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival O Guedes. 2011. Gatekeeper: Supporting Bandwidth Guarantees for Multi-tenant Datacenter Networks.. In *WIOV*.

[9] Madhavapeddi Shreedhar and George Varghese. 1996. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on networking* 4, 3 (1996), 375–385.

[10] Ion Stoica, Hui Zhang, and TS Ng. 1997. *A hierarchical fair service curve algorithm for link-sharing, real-time and priority services*. Vol. 27. ACM.