Exploring the Network Scale-out in Virtualized Servers

Hangwei Qian
VMware, Inc.

Chenghua Cao
University of California Irvine

Li Liu
Marvell Semiconductor Inc.

Hualong Zu
University of Southern California

Qixin Wang
University of Southern California

Menghui Li
Peking University, China

Tao Lin
VMware, Inc.

Abstract—Nowadays, servers are popularly virtualized and highly consolidated, requiring much more network capacity than ever before. While scaling up the capacity of the network interface controllers (NICs) is inarguably important, we focus on the alternative and explore how to scale out the network capacity by aggregating multiple NICs. In this paper, we discuss the various requirements and problems of the NIC aggregation in virtualized servers and propose packet scheduling and load balance algorithms taking into consideration of the heterogeneity of NIC capacity and different priorities of the hosted VMs.

I. INTRODUCTION

Virtualization enables efficient sharing of the server resources multiplexed among the hosted applications, and plays a critical role in the cloud computing. With virtualization, larger servers can obtain the economy of scale by packing many virtual machines (VMs) into a single server to increase the resource utilization and reduce the cost. For example, it is not unusual nowadays that servers are configured with 32 cores, 128 GB memory.

While the hypervisors, like VMware ESX, Microsoft Hyper-V, and Xen, offer sophisticated mechanisms to manage the CPU and memory resources, the network is becoming more and more a pain. Among many issues, the most fundamental is the network capacity limitation, which can be caused by bandwidth consuming services, such as video streaming [23, 24, 25]. When a server gets stuck by the network bottleneck, its CPU and memory resources could be wasted. Following factors exacerbate this problem:

High server consolidation. The trend to concentrate the computing resources is giving rise to the high server consolidation. According to [29], there could be more than one hundred VMs packed on a single server in practical deployments. It is very challenging to provide enough connectivity for the large amount of network demand of the consolidated VMs.

Popular adoption of converged storage solutions. Nowadays, converged storage techniques, like network attached storage (NAS) and storage area network (SAN), are commonly employed to provide file-level and block-level storage services at the data centers. In the converged storage solutions, all the data is stored remotely at the shared storage infrastructures, making the system very sensitive to the network performance.

Increasing management traffic. In virtualized data centers, the management traffic grows significantly, due to the administrative operations, like VM migrations for load balance [17] and the network memory for handling memory overload [40], etc., consuming large portion of network bandwidth of the servers.

In order to increase the server network capacity, generally two options are available: scale-up and scale-out. In the scale-up solution, servers are configured with more powerful NICs. For example, many servers nowadays are equipped with 10-Gbit NICs to handle the network traffic. Alternatively, we can scale out the server network capacity by aggregating many NICs in servers, which is referred to as NIC aggregation. While the scale-up solution is inarguably important, the scale-out solution also has many benefits. First and for most, it is more advantageous economically. For example, nowadays, a Gigabit NIC costs about 20$, while a 10-Gbit NIC usually costs more than about 400$ at Amazon. Furthermore, scale-out solution can provide higher network reliability, since the server only loses a portion of connectivity in the failure of a NIC in scale-out solution, while it would be totally disconnected with scale-up solution. Note that, these two methods are not mutually exclusive since one can scale out multiple more powerful NICs to achieve even larger network capacity.

In this paper, we explore the scale-out solution in the virtualized servers. While some existing techniques, like the link aggregation [10], also provide some mechanisms to utilize multiple links or NICs to achieve higher capacity and reliability, they either have many limitations, or are not applicable in our environment (Section II has more detailed discussion).

Moreover, virtualization brings new challenges to the effective NIC aggregation. For example, in most hypervisors, VMs are connected through virtual switches to the NICs in a server, which may connect to multiple physical switches and networks, making the NIC aggregation much more complex, as shown at Section III. More importantly, packets from the high-priority VMs should be less affected by the congestion conditions on the NICs, and existing techniques lack effective mechanisms to deal with this issue.

In summary, this paper explores the network scale-out in virtualized servers. It investigates the VM interconnection in servers with various network configurations, identifies the requirements and challenges of NIC aggregation and proposes...
packet scheduling and load balance solutions.

II. BACKGROUND

Virtualization is an enabling technique for cloud computing. Many work focuses on resource management considering the CPU and memory issues, such as [33, 44, 31], while we address the network problems. In particular, different from the works that consider the network and performance issues in the distributed cloud environment like [30, 32], we only focus on the network scale-out for a single server.

In this section, we first explain the interconnection of VMs in a virtualized server. Then we discuss some existing techniques that also attempt to make use of multiple links, NICs and paths to increase the network capacity and reliability. We outline their limitations and identify their differences from the NIC aggregation addressed in this paper.

A. VM Interconnection

In a virtualized server, each virtual machine is connected to the virtual switches through the virtual NIC (vNIC), forming internal virtual networks. The virtual switches is also connected to the physical NICs, through which they are bridged to the external physical networks, as shown in figure 1.

Unlike physical switches that connect the servers to external networks, virtual switches are implemented as softwares. The virtual switches function as layer-2 devices that forward packets (or frames) among ports according to the destination MAC addresses.

B. Link Aggregation

Link aggregation allows one or more links to be aggregated together to form a Link Aggregation Group (LAG), which is treated as a single link [16, 10]. It can be applied between switches and switches, between switches and stations (servers or routers), and between stations and stations. The benefits of link aggregation are increased link capacity and high link availability, which is similar with the NIC aggregation addressed in our work. One special type of link aggregation is the NIC teaming, which is only applied between the servers and switches and is supported by many hypervisors, like ESX and hyper-v.

However, there are some limitations of the link aggregation. For example, it does not support aggregation among more than two systems, while a server may be connected to more than one physical switches for high availability. Also, it requires that all links in a Link Aggregation Group operate at the same data rate and thus can not aggregate NICs with different capacities. In addition, the link aggregation is unaware of the traffic coming from the upper layer, and thus can not achieve the flexible packet scheduling in the virtualized servers, e.g., scheduling traffic based on the priorities of the VMs.

C. IP Network Multipathing

IP network multipathing (IPMP) is a technique developed by Solaris to provide fault-tolerance and spread load among multiple NICs [9, 8]. In IPMP, each NIC is assigned a "test" IP address and the NIC group is allocated with a virtual IP address that is exposed to the applications. IPMP could detect NIC failures through the "test" IPs and failover the traffic to the NICs alive.

While IPMP is able to offer similar benefits, it is hard to be applied in the virtualized servers. In most hypervisors, VMs are connected through virtual switches with layer-2 functionalities [14, 29], therefore, binding techniques purely based on the IP addresses of NICs are inherently inappropriate in virtualized servers.

D. Multipath TCP

Multipath TCP is a technique to spread the traffic across multiple paths between the sender and the receiver to increase the resource usage and traffic reliability [41]. Although the goal of multipath TCP is similar, it focuses on utilizing the multiple paths spanning many links at transport layer and is thus orthogonal to the problem addressed in this work.

III. METHODOLOGY

Many challenges exist for effective NIC aggregation in virtualized servers. Firstly, it should support NIC heterogeneity. For instance, a server may be configured with both Gigabit NICs and 10-Gbit NICs and the NIC aggregation should be aware of the difference of the NIC capacities in the packet scheduling and load balance.

Moreover, NIC aggregation should support priority-based packets scheduling and load balance among the aggregated NICs. In the multi-tenant data centers, the VMs hosted in a server could belong to different customers who pay differently
to the infrastructure providers based on their specific quality of service (QoS) requirements. In order to satisfy different QoS requirements, the NIC aggregation should transmit the packets based on the different priorities of the network traffic.

In addition, the internal virtual network, multiple switch interconnections and multihoming of the servers make the NIC aggregation much more complex.

In this section, we discuss all these challenges and explain our proposed NIC aggregation mechanisms.

A. Virtual Network Architecture

As mentioned before, servers often have various interconnection configurations. For example, a server may connect to more than one physical switches to increase the redundancy of connectivity. Also, in multihomed data centers, a server may connect to multiple networks to support traffic engineering. In addition, some cloud customers prefer to have their VMs stay in separate VLANs for security and other reasons, making the interconnection more complex.

The interconnection in Figure 2 reflects these options. As shown in the figure, the server is connected to network 1 through the physical switch 1 and 2 for reliability. Meanwhile, the server is also multihomed to network 2 through physical switch 3. The VMs and NICs in network 1 are connected through the virtual switch 1, while the virtual switch 2 connects the VMs and NICs in network 2. Finally, all the VMs are divided into three VLANs according to the management policies. Note that, multiple virtual switches can be used to connect the VMs and NICs in a specific network, e.g., network 1 could use more virtual switches to connect the VMs and NICs in stead of only using virtual switch 1 as shown in the figure. A single virtual switch can aggregate and distribute traffic among the NICs more effectively in a network since it is aware of all the VMs and NICs in the network, while it may suffer scalability issue when the number of VMs and NICs is large, especially when we need to add additional features for priority-based packet scheduling as discussed shortly. In our ongoing work, we are evaluating the overhead of the software-based virtual switches to make better interconnection decisions.

The best position for traffic aggregation and distribution among the NICs is at the virtual switches. One advantage of the software implementation of the virtual switches is that we can easily add additional features upon the basic layer-2 functionalities. One important feature added to the virtual switch is the traffic identification among the VMs to support priority-based packet scheduling and load balance (note the load balance component in the figure). In order to do that, one could simply put the VM priority information in each packet to facilitate the packet scheduling. However, this would change the existing layer-2 protocols. In our proposed architecture, we associate each port of the virtual switch with the priority of the connected VM. This association is initialized whenever an VM is connected to the port and is updated when the VM is migrated or its priority is changed. When distributing the traffic among the NICs, the priority information of port where the packets come from is employed. More detailed discussion is available at Section III-B.

In addition, all the NICs connected to the virtual switches are initialized and aggregated to share the traffic load exiting or entering the servers. The virtual switches identify and mark the uplinks connecting the NICs and distribute the outbound traffic among them according to the packet scheduling mechanisms discussed bellow.

B. Outbound Traffic Balance

When VMs in a server communicate with each other, the traffic is limited within the internal virtual networks. However, if the VMs want to interact with the external entities, like a web server, then their traffic is spread among the NICs connected to the same virtual switches in the same network. One problem is: at what granularity to schedule the packets among the NICs? For best load balance, we can schedule the traffic packet by packet among the NICs. However, this would cause the mis-ordering of the packets belonging to the same "conversation", and further make the L4 packets (like TCP) out of order at the receiver side, which incurs non-trivial overhead [11]. To make the trade-off, the load balance component schedules the traffic on the granularity of "conversation", same with link aggregation [7, 10, 16]. One common implementation is to use the L3 hashes (i.e. based on the source and destination IP addresses) so that traffic for the same IP addresses always go through the same NIC. Another option is based on the L4 flows (i.e. TCP port), which can achieve better load balance but has higher scheduling overhead. In our ongoing work, we evaluate both options to quantify the load balance effect and overhead.

One main task is to develop an effective priority-aware scheduling policy for the load balance component, which consists of two parts: packet selection and assignment. The packet selection decides which packets to transmit among all
the packets arrived from the VMs, while the packet assignment focuses on to which NIC to forward each selected packet. While there are some existing efforts to address the network packet scheduling issue, they either have their limitations, or are different from our context. For example, multiple packet scheduling methods exist for link aggregation, such as the round-robin, XOR of MAC addresses [11] and hashing algorithms [5]. But these algorithms can not be applied in our context to balance the load among NICs since the capacities of the NICs are not equivalent in our case. Moreover, the packets from different VMs are treated equally in link aggregation while we need to consider the different priorities of the packets. Meanwhile, there are other efforts on the network I/O scheduling problem, like [28]. However, the problem is different from ours since we need to balance the network load among multiple NICs while they focus on the I/O performance impact of VMM scheduling.

**Packet selection.** In the prioritized packet scheduling, we should give more preference to packets with higher priority and yet stave off the starvation of the packets from VMs with low priorities. Meanwhile, the packet selection should be efficient enough so as not to degrade the throughput. In our packet selection, we maintain a packet queue for each priority level (associated with a non-negative numerical number) and packets with the same priority are put in the same queue based on the first-come-first-serve policy. We consider two packet selection policies bellow.

The first policy is called *probabilistic packet selection*. In this policy, the probability that the packet is selected from a specific queue is proportional to the associated priority. Whenever the load balance component transmits a packet, it generates a random to decide from which queue to select the packet. For the packets in a queue, they are selected in the order of their arrival time. One question is, what if the selected queue is empty? One way is simply to generate another random number and redo the selection again until a non-empty queue is selected. However, since generating random numbers is not trivial in terms of computational cost, this method could potentially have high overhead and degrade the transmission throughput. Another way is to scan the queues in the decreasing order of their priorities and pick up a packet from the first non-empty queue. This way gives additional credits to the packets with higher priorities. In addition, we can exclude the empty queues when selecting the queues, which incurs additional overhead since it requires to monitor the queue status. We are evaluating these options in our ongoing work.

Finally, in order to further reduce the overhead of generating the random numbers, we can do the packet selection in batch mode, i.e., picking up \( k \) packets a time, where \( k \) is an adjustable parameter. For example, after deciding to pick up packets from a specific queue, we grab \( \text{MIN}\{k, \text{queue length}\} \) packets from this queue. In this way, we can reduce the number of random number generations by approximately a factor of \( k \).

The second policy is dubbed *two-phase packet selection*. In stead of selecting packet one by one (or \( k \) by \( k \)) as in the probabilistic packet selection, we firstly calculate the overall throughput of all the NICs, say \( R \) packets per second, and try to pick up \( R \) packets with each queue contributing certain portion of packets proportional to their priorities, say \( R_1, R_2, \ldots, R_q \) (\( q \) is the number of queues), so that \( \sum_{i=1}^{q} R_i = R \). If the length of queue \( i \) is less than \( R_i \), then the difference is aggregated as the additional credits, denoted by variable \( C \).

After the first phase, the \( C \) is the number of packets that need to be selected additionally. The selection of these \( C \) packets is conducted in the second phase, in which we can just pick up \( C \) packets by simply going through the queues in the decreasing order of the priorities.

The two-phase packet selection policy gets rid of the expensive random number generation and thus has low computational cost. However, it depends on the capacity estimation of the NICs. In order not to congest the NICs, it is preferable to be conservative when calculating the overall capacity of the NICs.

**Packet assignment.** As mentioned above, the load balance component assigns the packets among the NICs at the granularity of “conversation”. Thus, when assigning the packets, we only care about the first packet in each “conversation”. In order to get the size of each “conversation” to better balance the load among the NICs, we approximate it by inferring from the packets that have already arrived at the virtual switches.

Given the size of the associated “conversation” of each packet and the capacity of the NICs, we need to find effective packet assignment policies to minimize the load imbalance among the NICs. Again, the assignment algorithms should be efficient enough in order not to degrade the network throughput. We investigate two packet assignment policies: *least-loaded first-fit packet assignment* and *probabilistic packet assignment*.

In the *least-loaded first-fit packet assignment*, NICs are ordered by their load status and the packet is assigned to the least-loaded NIC that can accommodate the packet. A data structure like min heap would achieve \( O(\log n) \) computational complexity. The load here can be the idle network capacity or network capacity utilization. When using the idle network capacity as the metric, the more powerful NICs are preferred, causing the concentration of the traffic on the more powerful NICs. For example, Gigabit NICs would not be assigned packets unless the 10-gbit NICs are at least 90% utilized. If the powerful NICs fail, then the impact of the existing conversations are large. The advantage is that the NICs with the largest idle capacity are always at the top of the heap to accommodate the packets and there is no need to try other NICs, thus reducing the overhead (actually it becomes least-loaded assignment algorithm).

Alternatively, if the capacity utilization is applied as the load metrics, then the packets are more easily assigned to the less capable NICs, increasing the connection reliability. The disadvantage is that the NIC at the top of the heap can not always accommodate the packet. For example, a Gigabit NIC with 80% utilized would be tried before a 10-gbit NIC with
90% utilized. If the size of the conversation of the packet is more than 200M, then the assignment policy would fail at the Gigabit NIC and needs to try more times, thus increase the scheduling overhead.

In the probabilistic packet assignment policy, the probability that an NIC is assigned the packet is proportional to the its capacity. The load balance component generates a random number to choose an NIC. If the NIC is able to accommodate the packet, then it is assigned to this NIC. Otherwise, it need to generate a random number again until an NIC with enough idle network capacity is selected.

Similar with the packet selection, we can also make the packet assignment in the batch mode, i.e., making a decision for every $k$ packets. In this way, the size would be the overall size of the $k$ packets.

C. Inbound Traffic Balance

The inbound traffic firstly arrive at the physical switches, which is configured to balance the load among the links of the connected NICs. Then the virtual switches forward the packets to the vNICs of the VMs with the right MAC addresses. In order to support heterogeneity of the NICs, the physical switch implements the packet assignment policies discussed above. Also, when the inbound traffic arrive at the virtual switches, the virtual switches can apply the priority-based packet selection policies discussed above to deliver the packets to the right VMs according to the destination MAC addresses.

It is easy for the virtual switches to apply the priority-based packet selection policies since they are implemented as a software within the server. But it would be hard to implement the packet assignment policies at the physical switches if they are customized layer-2 switch fabrics. Fortunately, in modern data centers, the servers are usually connected to the top of the rack (ToR) switches [19, 20, 27], many of which are equipped with CPU and memory components, like Cisco Catalyst 4900M [1] and 4948E [4] switch, HP ProCurve Switch 5400zl, 3500yl, and 6200yl Series [6] and Extreme Networks Summit X650 Series [12], etc., and thus are able to integrate the packet assignment policies discussed above.

Another difference is that the load balance of the inbound traffic can only be pursued among the NICs connected to the same physical switch (like the physical switch 1 in figure 2), while for the outbound traffic, it is conducted among the NICs connected to the same virtual switch (like the virtual switch 1 in figure 2). Since the load balance policies of the inbound and outbound traffic are applied by different entities and among different set of NICs, it makes the packet scheduling among the NICs much more challenging. Implementing and evaluating the various policies discussed above for packet scheduling are the main task of our ongoing work.

IV. DISCUSSION

In this section, we discuss some important issues for the NIC aggregation.

A. Scheduling Overhead

The priority-based packet scheduling policies are critical to satisfy the quality of service (QoS) requirements of different customers. However, they may cost non-trivial CPU and memory resources, and even incur additional delays for the packet transmission. Thus, we need to evaluate extensively the resource overhead and transmission delay of the various packet scheduling algorithms and make a good trade-off between the customer QoS satisfaction and system performance.

B. NIC Limitation

The NIC aggregation requires the servers to have multiple NICs installed at the chassis to share the network traffic from the consolidated VMs, thus it is limited by the number of NICs in the servers. While it is critical to increase the number of Peripheral Component Interconnect (PCI) slots so as to support more NICs in a server, now there is parallel effort to increase the number of ports for each NIC. For example, Intel offers dual-port and quad-port Gigabit Ethernet adapters to better support the high consolidation of the servers [13].

Meanwhile, many hypervisor can support many NICs in a server. For example, VMware ESX can support 32 NICs per server [15] and Xen can have 16 NICs for each server [3].

C. Switch Port Density

Since the number of cables connecting the servers to the physical switches would increase significantly, it requires the ToR switches to have high port density. At current practice, a standard ToR switch usually contains only 48 Gigabit ports for downlinks and 4 10-Gigabit ports for uplinks [27]. However, ToR switches with much higher port density are becoming available. For example, HP ProCurve Switch 5412zl can support 288 Gigabit ports [6], and Cisco Catalyst 6513 switch offers up to 576 ports [2].

V. RELATED WORK

Cloud computing received a lot of attention recently. Unlike other works that focus on the CPU, memory and storage, like [33, 44, 31] and [34, 18], we focus on the network issues since network becomes more and more a problem nowadays. There exist some work addressing the network I/O scheduling issues, like [28, 21]. This work differentiate from them in that we explore how to aggregate many NICs to scale out the network capacity in virtualized servers and study the packet scheduling and load balance among the NICs taking into consideration of the different priorities of the hosted VMs. Some other works focus on the general network problem, like [43, 26, 42], which is out of the scope of this work.

Some existing techniques, like link aggregation [11], have many limitations and can not be applied in our environment, as discussed in Section II. Also, since we try to minimize the overhead of the packet scheduling, we do not consider highly complex and sophisticated algorithms and mechanisms, like [45, 35, 36] and [37, 39, 38].
VI. CONCLUSION AND FUTURE WORK

In this paper, we investigate how to scale out the network capacity in virtualized servers through the aggregation of multiple NICs. Different from the link aggregation, we support NICs with various capacities. More importantly, we consider the different priorities of the hosted VMs and propose priority-based packet scheduling load balance algorithms. Finally, we discuss some issues, like the NIC limitation and switch port density, etc., for the NIC aggregation. In the ongoing work, we are implementing the various packet scheduling algorithms proposed in this paper and evaluating the system extensively.

REFERENCES

[30] Hangwei Qian and Qian Lv. Proximity-aware cloud selection and virtual machine allocation in iaas cloud platforms. In The International Workshop on Internet-


