2013

Jumbo Frames or Not: That is the Question!

Pawan Prakash
Purdue University, pprakash@purdue.edu

Myungjin Lee
University of Edinburgh, myungjin.lee@ed.ac.uk

Y. Charlie Hu
Purdue University, ychu@purdue.edu

Ramana Rao Kompella
Purdue University, rkompella@purdue.edu

Twitter Inc.

See next page for additional authors

Report Number:
13-006

Jumbo Frames or Not: That is the Question!

Pawan Prakash[†], Myungjin Lee[‡], Y Charlie Hu[†], Ramana Rao Kompella[†], Jun Wang[ℓ] and Samya Dassarma[ℓ]
[†]Purdue University, [‡]University of Edinburgh, [ℓ]Twitter

***Abstract*—We focus on a simple question in this paper: Should data center network operators turn on Ethernet jumbo frames? While prior work supports using jumbo frames for their throughput and CPU benefits, it is not clear whether these results are directly applicable to data center networks, particularly since they were performed on older hardware and only focused on TCP performance and not application-level. Instead, in this paper, we evaluate the advantages of jumbo frames using modern hardware with features such as large send/receive offload, and with canonical data center applications such as MapReduce and tiered Web services. We find that the throughput and CPU utilization benefits still exist generally, although compared to prior studies, are significantly reduced. Based on these results, we conclude that data center network operators can safely turn jumbo frames on, despite a small side effect we discovered.**

I. Introduction

Networking is a field of constant flux. Network devices are continuously updated as new technology (e.g., 40/100Gbps Ethernet), new protocols (e.g., OpenFlow), and in general, new features are continuously introduced. Interestingly, not all features prove to be immediately impactful in practice: For some, it takes a significant amount of time to gain acceptance, while for others, technological changes may make them less relevant over time. In any case, the forces that govern wide deployment of any new feature in any environment are quite simple: If network operators find that their applications perform better with the new feature, or the new feature offers new management benefits, then they are likely to adopt it in their environment. Regardless of the specific choices made, network operators have the unenviable job of ensuring their networking infrastructure provide the best possible network performance to their applications at all times, by carefully and continuously evaluating which features and technology to embrace and which ones to phase out.

In this paper, we focus on one such feature, *Ethernet jumbo frames*, which are essentially Ethernet frames with size greater than 1500 bytes up to 9000 bytes. The key argument in support for jumbo frames has been mainly that they help reduce CPU overheads for TCP processing and achieve better network utilization and throughput since they lead to overall fewer packets. Unfortunately, despite the fact that jumbo frames were introduced almost 15 years ago, their deployment in the Internet is not widespread. Part of the reason is that larger frame sizes can cause additional delays for latency-sensitive traffic, as each frame has a high serialization delay [1]. A more serious reason, perhaps, is that it requires all ASes along an end-to-end path to upgrade their network to provide a larger MTU, which poses significant deployment challenges. Deployment challenges are slightly less aggravated in enterprise networks which are managed by a single authority. Nonetheless, jumbo frame deployment in enterprise networks has been relatively sparse, possibly because of the presence of several types of middleboxes (e.g., firewalls, load balancers) that are often incompatible with jumbo frames, even if most commodity forwarding devices (e.g., switch, routers) today are more or less compatible with jumbo frames.

The recent popularity of cloud computing platforms, both public as well as private, and data centers, however, has sparked a renewed interest in jumbo frames. These data center environments have all the same advantages as any enterprise networks that a single authority manages all components of an end-to-end path. In addition, many data center applications involve significant east-west traffic (server-to-server) within the data center in addition to the traditional north-south (client-to-server) traffic to and from the data center, which can benefit from the presence of jumbo frames. Many current data center operators are therefore seriously considering turning on jumbo frames in these environments, for the classical reasons of reducing packet processing overheads as well as TCP throughput improvement. Indeed, in some contexts, jumbo frames have already been turned on specific interfaces such as those used for VM migration using vMotion [2]. However, general deployment within a data center is still not widespread.

Before network operators can turn jumbo frames on, however, it is extremely important to ascertain whether there are any application-level benefits of jumbo frames and also determine whether they cause more harm than good for a canonical set of data center applications. Unfortunately, there exist no prior studies that explore whether jumbo frames offer any advantages in these specific scenarios. Most studies [3], [4] conducted in the past used outdated hardware that did not possess new features such as large-send offload (LSO), large-receive offload (LRO) which essentially make the software networking stack deal with large packets (almost 64KB) than typical Ethernet MTUs of 1500 bytes. Even for those relatively recent studies (e.g. [1]) conducted with modern hardware, they deal with latencies at the Internet scale and not intra data center such as less than a couple of milliseconds. Finally, none of the existing studies focus on data center applications, which makes it hard for operators to directly use these results.

To address this gap, in this paper, we conduct a detailed empirical study involving jumbo frames with canonical data center applications such as file transfer, MapReduce and 3-tiered Web applications. While our study is not the final word for all data centers and all scenarios, our study is timely as many data center network operators are thinking about turning on jumbo frames in their data centers. Indeed the study itself originated from such a requirement at a large web service provider's data center network with whom we spent a considerable amount of time understanding the needs of their environment and their best practices. Our study is complementary to all existing studies involving jumbo frames, but provides new results as we use modern hardware with features such as LSO and LRO, and is conducted with more relevant data center oriented applications, making our study more applicable to data center operators today.

Through our comprehensive evaluation of jumbo frames in our data center testbed consisting of 12 servers, we observe increased throughput and reduced CPU utilization across all tested applications. For instance, jumbo frames achieved 5.5–11% higher throughput for file transfer applications, and a small reduction (up to 4%) in job completion time for MapReduce applications, but virtually no benefit for Olio web service. While these results are far less impressive than reported by prior work, they still show that jumbo frames generally benefits TCP throughput. We also observed generally lower CPU utilization for these applications with jumbo frames, although the gains varied significantly depending on the application, and were not as impressive as previous works reported.

Somewhat surprisingly, however, we observed that response time of the web application has gone up, even if only slightly. Upon further investigation, we discovered that the Nagle's algorithm in TCP causes this effect; upon disabling Nagle (which some data center network operators, like the one we have interacted with, already do), we found that response times were back to normal. This points to an important issue regarding the interplay of Nagle's algorithm with latency sensitive algorithms that merits a more detailed treatment outside the scope of this paper.

Recommendations. Based on these results, we advocate that data center network operators can turn on jumbo frames for their intra data center traffic, since we could confirm most of the expected benefits (better throughput and reduced CPU utilization) even for data center applications. However, the benefits are not as pronounced as prior work reported. The only issue with jumbo frames we have found is that response time of the web server increased slightly because of the Nagle algorithm. Since Nagle can be turned off in the application itself using a simple socket option, data center operators may not need to worry about it too much. Since many data center operators (like the one we have interacted with) care about good response times for their web services, they may have disabled Nagle's algorithm already anyways, and hence may not be a problem.

II. Motivation

In this section, we provide the motivation for conducting a detailed empirical study on jumbo frames in modern data center environments. At a high level, there are primarily two reasons that necessitate deliberation about jumbo frames in modern networks. Firstly, most of the observations and results about jumbo frames are outdated; thereby past research conducted with obsolete devices creates disagreement in the community about the efficacy of jumbo frames. Another motivation for this study is the environmental shift. In the past, jumbo frames were mainly studied for wide area networks and specialized storage networks. Modern data center networks provide a new frontier where jumbo frames have not been thoroughly evaluated. In the following, we enlist some of the canonical properties (or myths) of jumbo frames that we wish to re-evaluate in data center setup. We also take a look at some key features of data center networks that make the use of jumbo frames conducive in this environment.

A. *Properties associated with jumbo frames*

P1: Increased throughput. The most well-known property of jumbo frame is the significant throughput gain that can be achieved with it. For instance, in [3], Feng *et al.* illustrated, in their micro-benchmark tests using Chelsio T110 10GE adapters, that with 9KB MTU, TCP achieves a throughput of up to 7.2 Gbps, but only 4.9 Gbps with 1,500 byte MTU. Similarly, in [4], significant throughput gain was reported when 8KB jumbo frames are used along side other TCP optimization mechanisms such as zero-copy and checksum offloading. In contrast, another study [5] reports that LSO (Large Segment Offload) achieves the most gain while LSO along with jumbo frames provides additional 8% improvement. Due to the mixed view about the benefits of jumbo frames, some engineers question the need for jumbo frames [6], [7] and some others advocate the adoption of jumbo frames in modern data centers [2].

Similar to previous works, we focus on evaluating the throughput gains. We, however, use several real data center applications and more typical traffic patterns. We also vary the size of data transfers and RTTs (from hundreds of microseconds to ten milliseconds) in the network and provide a fully rounded analysis of throughput gain with the use of jumbo frames.

P2: Increased delay. A large packet size basically means a large transmission time. The transmission time of a 9KB jumbo frame is approximately 6 times more than a 1500 byte Ethernet frame. Thus, jumbo frames lead to increased delay of packet transmission [8]. This property of jumbo frames may worsen the response times of real time applications such as VoIP, gaming, web services and HPC applications, which are severely impacted by end-to-end latency.

Higher transmission time for jumbo frames is a non-issue for modern faster network cards. For instance, serialization delay of a 9K frame on a GE network is less than the serialization delay of a 1500 byte packet on a 100 Mbps network. But unlike throughput, the studies that analyze the effect of using jumbo frame on the response times of these services have been very limited. In [9], Joubert *et al.* measure response times of memory-based web servers in terms of connection rate, but the results are only about jumbo frame-enabled cases, not the regular Ethernet frame case. Thus, they do not study the benefits of using jumbo frames over regular frames, let alone consider different frame sizes and fully explore the advantages and disadvantages of jumbo frames in these environments [10], [11]. We use a popular open source web service to study the impact of frame sizes on response time in modern data center networks.

P3: Reduced system overhead. One of the essential network properties is reduced per-packet processing overhead at end hosts and switches. Given the same amount of data, using jumbo frame generates fewer packets to be processed than the usual 1,500 byte MTU Ethernet frames. Thus, end hosts generate fewer interrupts and thus save CPU cycles. For instance, Chase *et al.* showed that 8KB jumbo frames reduced the CPU utilization by a factor of 3, compared to the standard Ethernet frames [4]. However, the benefit of jumbo frames seems to wither due to the wide adoption of LSO as the LSO which also significantly saves CPU utilization [12]. We want to revisit this property in case of various data center applications.

B. *Uniqueness of data center*

Modern data centers possess a lot of characteristics that make them feasible candidates to use jumbo frames. We highlight a few of those properties that show case the necessity of studying the impact of jumbo frames in data centers.

High bandwidth. The most distinctive aspect of modern data center networks is the availability of high bandwidth to the end hosts. Some are already deploying 10 Gbps Ethernet, while practically every network has at least 1 Gbps at the edge. Jumbo frames were designed keeping high bandwidth networks in mind and are well suited for use in data center networks. In addition to increasing throughput, the reduction of overheads can help accommodate more jobs at these servers.

Low latency. Nodes in a data center are closely located from each other in a geographically small region. Therefore, the end-to-end delay between any pair of servers is typically a few hundred microseconds but can go upto a few milliseconds during congestion [13]. The study of the influence of jumbo frames in such a low-latency environment is limited.

New applications. Given the high capacity and low latency characteristics, many new applications with different requirements started to nest in data centers. For instance, MapReduce tends to demand higher throughput while less caring about delay. On the other hand, low latency is key to meeting the response time requirement of latency-sensitive applications such as search, 3-tier web services and many HPC applications. Jumbo frames can be a desirable feature to bandwidth-hungry applications, but not to latency-sensitive applications. Moreover, all these services are hosted in the same data center. Thus, it is important to understand how the option influences both types of applications.

Single administrative authority. Data centers are mostly managed by a single organization. Multiple management domains (*i.e.,* different ISPs) were one of the main hurdles that prohibited the adoption of jumbo frames in the Internet. On the other hand, the single ownership of a data center makes enabling jumbo frames much simpler and straightforward.

C. *Other use cases of jumbo frames*

JumboGen [14] enables packet aggregation with jumbo frames at core networks while keeping edge networks unmodified. It reduces the overhead of processing a number of small packets and increase the core network utilization. However, it does not show what "real" benefits jumbo frames would create for today's data center applications. In the data center context, a few recent research papers consider jumbo frames as one of their system features. In [15], Rhoden *et al.* propose that enabling jumbo frames would be beneficial for block-level data transfer. Unfortunately, there is no quantitative evaluation on the benefits of jumbo frames. Storage Area Network (SAN) is one canonical field where jumbo frame has been heavily tested and used as a part of the iSCSI (SCSI over TCP) specification because 4KB or 8KB SCSI blocks cannot fit in a single 1,500-byte Ethernet frame. The performance of the iSCSI protocol that relies on jumbo frames was evaluated in several previous studies [16], [17], [18].

III. Methodology

In this section, we describe our testbed, the set of applications, and the performance metrics we use to evaluate the impact of jumbo frames.

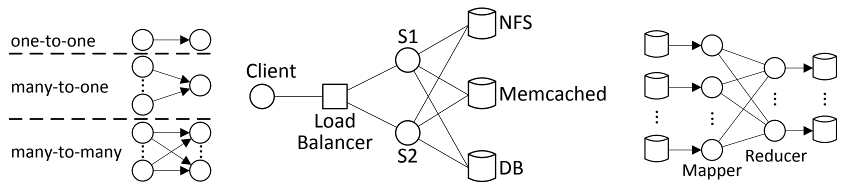
A. *Testbed Setup*

Our testbed consists of a rack of 12 servers connected via full duplex 1 GE links to a top of rack network switch. Each server has a quad-core Intel(R) Xeon(R) CPU X3430 (@2.40GHz) with no hyper-threading support. The L1 cache size is 256 KB, L2 cache size is 1024KB, and the L3 cache size is 8192KB. Each server has a total of 4GB RAM. Each server is connected with SATA drives with a rotational speed of 7200 RPM. The servers are running Linux 3.0.0-12 provided with Ubuntu 11.10 distribution.

Every host has an Intel Corporation 82574L Gigabit network card which provides jumbo frame support. Technologies like LSO (large segmentation offload), TSO (TCP segmentation offload), LRO (large receive offload), and interrupt coalescing in order to reduce packet processing overhead, are enabled. LSO eliminates the per-packet CPU overhead at the sender side and thus limits the benefits of jumbo frames. On the other hand, LRO (the inbound counterpart of LSO) does not totally eliminate the CPU overhead borne by incoming packets. All these features are supported by the network card we use. While one could explore how these options interplay with jumbo frames, it does not make sense to pursue this direction since these options are pretty much universally deployed.

B. *Data Center Applications*

We evaluate the performance of jumbo frames on three applications, file transfer application, Hadoop MapReduce application and Olio which is a tiered web service application. These applications vary in terms of their traffic patterns and system resource requirements. Figure 1 shows the traffic patterns of these applications. On one end of the spectrum, file transfer applications require mostly network resources and



(a) File transfer types (b) 3-tier web service (c) MapReduce

Fig. 1. Communication patterns of applications tested.

consume little computation resources. Hadoop MapReduce applications like terasort have high requirement for computation resources as well as network resources. At the other end of spectrum, Olio is representative of web service applications, which have many small HTTP request-response type of flows. We describe the details of these applications in below.

1) File Transfer: We use standard FTP clients and servers for experiments. We experiment with different traffic patterns ranging for one-to-one, many-to-one and many-to-many. In these applications, we are mainly concerned about the throughput achieved by using different frame sizes.

2) Hadoop: The Hadoop test setup consists of 13 nodes. We run Hadoop version 1.2.0 on these servers. One of the servers acts both as the namenode and the jobtracker. All other servers run datanode and tasktracker on them. Each server runs a maximum of 2 mappers and 2 reducers. The tasks are initiated in JVM running with a heap size of 512MB. The various configurations for Hadoop setup are similar to the one described in [19]. We run MapReduce terasort and grep applications on our testbed. In these applications, we are concerned about the completion time of a given job.

3) Olio: Olio is a web 2.0 toolkit to help evaluate the suitability, functionality and performance of web technologies [20]. Among many different implementations, we choose the binary kit for the RubyOnRails implementation. Our setup consists of 1 load balancer (running the nginx server), 2 web servers (running the thin servers), 1 SQL database server, 1 memcached server , 1 NFS server, and 1 faban client which is driving load against the web application. In Olio web service, we are interested in recording the response time and number of operations successfully completed using different frame sizes.

C. Performance Metrics

The metrics of interests are throughput, CPU cycles, number of instructions, instruction per cycle (IPC), and number of network interrupts. We choose the metrics because they are ones that system administrators in data centers most care about. We use the Linux perf tool for collecting various system-level metrics and counters *cpu-cycles* (cycle count when the CPU is not idle), *instructions*, and IPC (instructions per cycle). We record these counters system-wide with and without running the applications. The difference gives us an estimate of the counter values while executing applications. Similarly, the numbers of network transmit interrupts (NET_TX) and receive interrupts (NET_RX) are reported through `/proc/softirq`. In addition to system-level metrics, we also record various application level metrics which vary with different applications.

IV. Evaluation

In the experiments, we configure the network cards with 4 different MTU values: 1,500bytes, 4KB, 8KB and 9KB. While 1,500bytes is the default Ethernet MTU value (thus, we call this 1,500 byte packet Ethernet frame hereafter), 9KB is the widely accepted frame size for jumbo frames. We also find 4KB and 8KB as interesting choices for frame sizes as they can fit within one and two memory pages respectively.

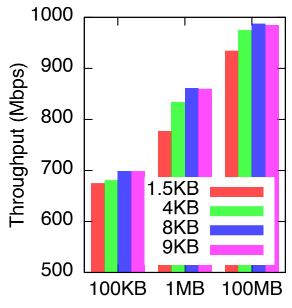
Jumbo frames typically reduce per-packet overhead both at the end host and at the network switches. Thus, expected microscopic benefits of jumbo frames are increased throughput and reduced number of CPU cycles and instructions for packet processing. To quantitatively evaluate such improvements, the first application we test is file transfer since jumbo frames are best known for its efficacy for these types of applications.

A. File Transfer Application

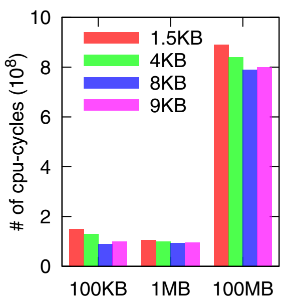
In order to evaluate the benefits of jumbo frames, we conduct a progressive study starting from one-to-one file transfer, to many-to-one file transfer and finally many-to-many transfer. In all three cases, we conduct experiments with traditional FTP (file transfer protocol) server/client with different file sizes: 100KB, 1MB and 100MB ([13] shows that the flow sizes in data centers can range from a few kilobytes to hundreds of megabytes). All the results are averaged over 5 runs.

1) One-to-one Transfer: This basic setup involves two hosts, where one host transfers data to the other. Figure 2(a) depicts the throughput achieved for different file transfer sizes. We observe that in case of 100KB file transfer, the throughput achieved with all the frame sizes are less than 700 Mbps. Because of the smaller file size to transfer, the flow is not able to increase its congestion window enough to fully utilize the link capacity. The throughput achieved by jumbo frames of sizes 8K and 9K is 700 Mbps which is almost 4% more than the throughput achieved by the 1.5K Ethernet frames (675 Mbps). The.throughput achieved by 4K frame lies between these two. To the best of our knowledge, none of the previous works have results that compare the throughput of regular and jumbo frames when a small sized flow is used.

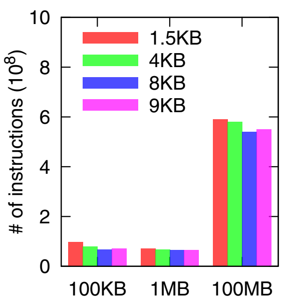
As the file size is increased, we observe higher throughput for all the frame sizes. With 1MB files (which is still not large enough for saturating the link capacity), the throughput with jumbo frames (8K/9K) is almost 10% more than with



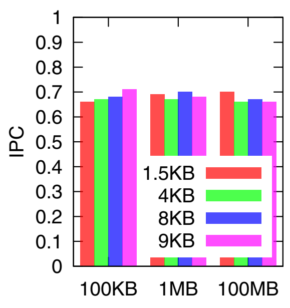
(a) Average throughput



(b) Average number of cpu cycles



(c) Average number of instructions



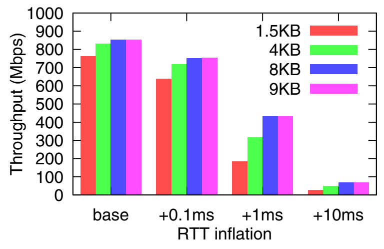
(d) IPC

Fig. 2. Results in one to one file transfer using FTP.

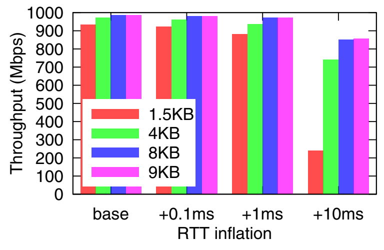
the 1.5K frames. The throughput obtained by jumbo frames in transferring 1MB files show better improvement than the standard Ethernet frames. It is because of the higher MSS value from jumbo frames that leads to larger initial congestion window (10 MSS [21]) and also a rapid increase in the congestion window. A file transfer size of 100MB leads to full utilization of the link capacity. At saturation, we see the best performance by all the frame sizes. The throughput achieved with jumbo frames is around 985 Mbps as compared to 934 Mbps with standard Ethernet frames. We observe that use of jumbo frames in one-to-one file transfer provides a 5.5% potential benefit over standard Ethernet frames even when utilizing full link capacity.

At the sender side, there is no significant difference in the system level statistics in case of regular and jumbo frames. This is expected as LSO eliminates the per-packet CPU overhead on the system (which the jumbo frames were designed to help with). The major difference is visible at the receiver side. Figure 2 highlights the number of CPU cycles used and instructions executed with different frame sizes. Looking at the graph corresponding to the 100MB file transfer (for better visibility, the graphs corresponding to 100KB has been scaled up by a factor of 100 while, the for 1MB it has been been scaled up by 10), we can observe that the total number of *CPU cycles* in case of jumbo frames (8K/9K) is 13% less and the number of *instructions* is also similarly low. It shows that for receiving the same amount of data, end host CPU spends fewer cycles and instructions when using jumbo frames. This saving can help other applications running on the same host, especially those that are CPU-bound. The IPC numbers depicted in Figure 2(d) show similar values for different frame sizes.

In previous work, Chase *et al.* showed that transferring data over jumbo frames reduces CPU utilization by a factor of 3 [4]. In our setup, we did not see huge improvements in CPU utilization. We see a slightly lower CPU usage in case of data transfer with jumbo frames. The average CPU usage reported for the period of data transfer over jumbo frame (9K) is *usr* = 0.08, *sys* = 0.94, *softirq* = 0.04. In case of Ethernet frames, the average CPU usage is *usr* = 0.13, *sys* = 1.77, *softirq* = 0.17. As we mentioned previously, modern network cards have



(a) Average throughput 1MB file transfer



(b) Average throughput 100MB file transfer

Fig. 3. Average throughput in file transfer application with inflated RTTs.

technologies like GSO, TSO, and interrupt coalescing, that help to reduce per packet CPU overhead. Hence we do not see as high CPU savings as observed in previous work.

One-to-one Transfer with inflated RTTs. The experiments conducted in the previous section have no interference from any other network traffic. The RTTs were of the order of 200 microseconds. In data centers, one would expect some queuing delays at the routers and switches. We used Linux traffic controller to artificially inflate the RTTs. Figure 3(a) demonstrates that for a small size file transfer (1MB), increase in RTT has a huge impact on the application throughput. When the RTT is inflated by 1ms, the throughput obtained with standard Ethernet frames drops well below 200 Mbps, but the throughput with jumbo frames are still around 400 Mbps (more than 200% compared to Ethernet frames).

With large size file transfer (100MB), we see that an RTT inflation of 10ms has a drastic impact on the throughput of Ethernet frames. It drops below 250 Mbps. The jumbo frames still obtain a throughput of more than 800 Mbps. We also tested out a 1GB file transfer with 10 ms RTT (result not shown) and found that the jumbo frame achieves 950 Mbps compared to 875 Mbps with regular frames. While 10ms may seem like a large value of RTT for intra data center environment, it is very common for inter data center data transfers. Hence having jumbo frames enabled in the network can speed up the process of taking data center backups which require transferring large amounts of data.

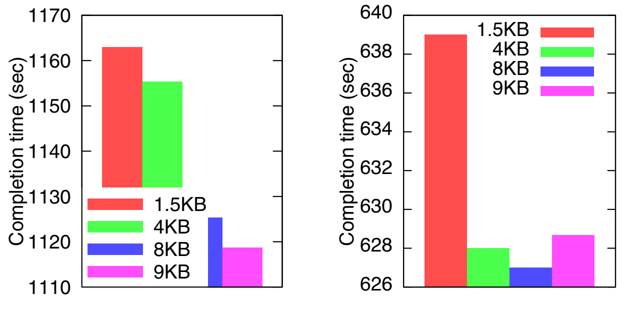
2) Many-to-one Transfer: Many-to-one data transfer is one of the more commonly occurring traffic pattern in data center networks (*e.g.,* barrier-synchronized workload in MapReduce). In our setup we have 12 nodes on a rack, with 11 of them sending data to 1 end host. We repeat experiments with FTP; each host is sending a file with the same x amount of data where x is 100KB, 1MB and 100MB. The throughput presented in Figure 4(a) is the aggregated throughput at the receiving node. We observe that regardless of file sizes, the throughput obtained by jumbo frames (8K/9K) is much higher than that of the regular frames. When the file size is small (100KB), the jumbo frames achieve almost 11% higher throughput.

In case of many-to-one file transfer of 1MB, using jumbo frames saturates the link capacity, whereas using 1.5K frames still has room for improvement. In fact, the throughput with jumbo frames is still 11% more than with 1.5K frames. The performance of 4K frame lies between the two. With 100MB file transfer, the throughput of using jumbo frames does not change, while the using Ethernet frames finally saturate the link. In this state, jumbo frames have 5.5% higher throughput compared to 1.5K frames.

Figure 4 also highlights the CPU resources used at the receiving host. The total number of *CPU-cycles* used is almost 17% lesser with jumbo frames while 22% fewer *instructions* are executed. As we observed in the one-to-one scenario, the IPC values are roughly the same across different frame sizes. The savings in CPU resources is a huge benefit of using jumbo frame; not only we get higher application throughput, we also save a lot of system resources useful for other operations.

3) Many-to-many Transfer: In many-to-many file transfer, half of the 12 nodes on the rack are transferring data to the other half as depicted in Figure 1(a). We repeat experiments with file sizes of 100KB, 1MB and 100MB. In our setup, all the senders send data to all the receivers resulting in 6 concurrent connections at each receiving node. We report the average throughput achieved by individual receivers.

Figure 5(a) demonstrates that the data transfer over jumbo frames achieves a higher throughput compared to over standard Ethernet frames. The results are similar to many-to-one file transfer experiments. For small files, jumbo frames have a 7% higher throughput. We also see that jumbo frames achieve link capacity faster than the 1.5K frames. At link saturation, the average throughput achieved by the receivers is 860 Mbps with jumbo frames and 830 Mbps with Ethernet frames. Jumbo frames also saves a lot of CPU resources compared to Ethernet frames. Figure 5(b) shows the average number of cycles used at each receiver. We observe that the use of jumbo frames saves more than 20% of CPU cycles at each receiver. The number of instructions is also reduced which leads to overall low CPU utilization at the receivers.



(a) Terasort with 100GB of data (b) Grep with 10GB of data

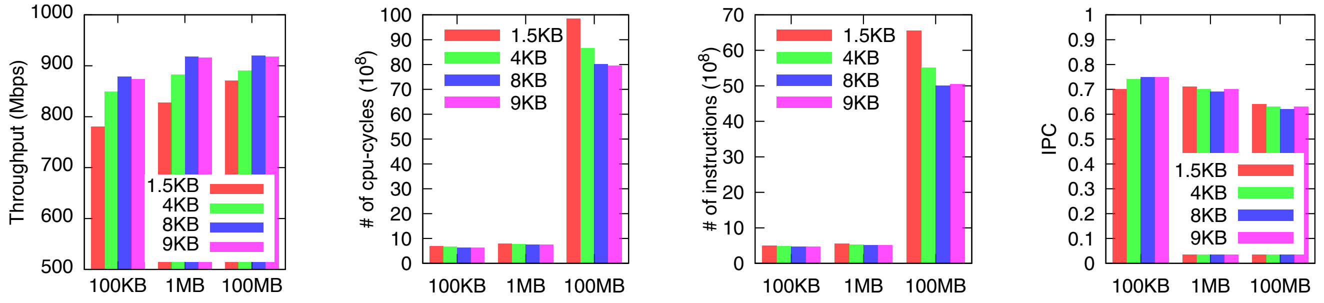
Fig. 6. Average completion time of Hadoop applications.

B. MapReduce applications

Next we evaluate the impact of jumbo frame on MapReduce applications, which are commonly used in data centers. We test two simple but popular MapReduce applications—terasort and grep—using Hadoop. The two applications have slightly different flavor. On one hand we have the Terasort has a significant networking component and involves shuffling large amounts of data, while Grep is mostly CPU bound with a small sized shuffle phase. We want to study the impact of jumbo frames on both classes of applications. All of the results reported are averaged over 3 runs.

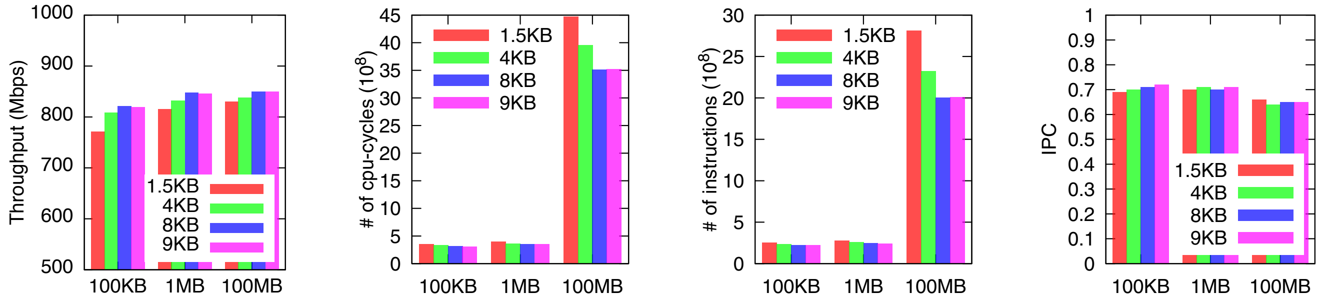
1) Hadoop terasort: We run the MapReduce terasort application on a 100 GB file generated using the Hadoop teragen application. As mentioned in previous section, Hadoop is run on 12 nodes with 2 mappers and 2 reducers running on each node. The task to sort 100 GB file is broken down into 744 map and 24 reduce tasks. Figure 6(a) shows the completion time of terasort with different frame sizes. We observe that the completion time with standard Ethernet frame is almost 4% more than the completion time with 9K jumbo frames. To reason about the difference in completion time, we analyze the shuffle phase of the reduce tasks. In case of standard Ethernet frames, the average shuffle phase lasted for a total of 440 seconds (it lasted for 434 seconds with 4K frames). With 9K jumbo frames, the average shuffle phase lasted for only 421 seconds. We see that use of jumbo frames results in a slightly smaller shuffle phase period which helps the terasort application to finish up faster.

Table I shows the results for terasort. Similar to our observations in file transfer application, we find that jumbo frames contribute in reducing the system overhead of running Hadoop applications. For example, 9K jumbo frames allow terasort to consume about 1% less CPU cycles than Ethernet frames. The number of network receive interrupts in case of 1.5K frames is also about 6% higher than in case of 9K jumbo frames. In terms of CPU utilization, we observe a marginal benefit with jumbo frames. The average CPU usage reported for jumbo frames is *usr* = 30.5, *sys* = 4.66, *softirq* = 0.62. In case of Ethernet frames, the average CPU usage is *usr* = 32.18, *sys* = 4.82, *softirq* = 0.72. Thus we see that a higher network throughput and lower resource utilization from using jumbo



(a) Average throughput (b) Average number of cpu cycles (c) Average number of instructions (d) IPC

Fig. 4. Micro-benchmark results in many to one file transfer using FTP.



(a) Average throughput (b) Average number of cpu cycles (c) Average number of instructions (d) IPC

Fig. 5. Micro-benchmark results in many to many file transfer using FTP.

The base count for cycles and instructions: $\times 10^{12}$

Fsize	Terasort			Grep		
	cyc	*ins*	IPC	*cyc*	*ins*	IPC
1500	3.221	3.577	1.110	2.660	5.940	2.231
4000	3.199	3.574	1.113	2.659	5.939	2.232
8000	3.190	3.558	1.115	2.658	5.936	2.235
9000	3.192	3.564	1.117	2.658	5.935	2.232

TABLE I
AVERAGE CPU CYCLES, INSTRUCTIONS AND IPC VALUES FOR DIFFERENT HADOOP APPLICATIONS

frames result in almost 40 seconds difference in the completion time of the terasort application.

2) Hadoop grep: We run the MapReduce grep application on a 10GB file. This is a small scale job consisting of 96 mappers and 24 reducers. We want to study the impact of jumbo frames on MapReduce applications which may not have a large networking component but is CPU intensive. Figure 6(b) highlights the completion time of grep with different frame sizes. We see that the completion time under jumbo frames is about 1.5% shorter than with standard Ethernet frames. The gain is smaller as compared to the terasort application (which has a significant network load).

In the grep application, the number of CPU cycles used by jumbo frames (8K) is 0.07% lower than used by standard Ethernet frames. The number of network receive interrupts is about 5% lower. In terms of CPU utilization, we observe a marginal benefit with jumbo frames. The average CPU usage reported for jumbo frames is *usr* = 35.5, *sys* = 0.24, *softirq* = 0.12. In case of Ethernet frames, the average CPU usage is *usr* = 36.1, *sys* = 0.28, *softirq* = 0.15.

C. Olio web application

Olio is an open source reference architecture supported by apache to evaluate various web2.0 technologies. It showcases various components used in social web sites. In our setup (Figure 1(b)), we have one faban driver [22] which drives load against the application setup. We have two web servers running instances of thin servers. The requests to the web servers are routed through a load balancer running nginx server. The driver emulates different kinds of client operations like login, accessing home page, adding events, etc. An operation involves loading one entire web page (which consists of multiple HTTP requests to complete the page). The inter-arrival times between operations are chosen from a negative exponential distribution with a mean of 5 seconds. The inter arrival time can be understood as the think time between operations. To fulfill client requests, the web servers access an SQL database server, a memcached server and an NFS server. We disabled Nagle's algorithm on all these systems (details discussed below).

Faban is configured to drive load against the application servers for a certain amount of time (in our case we set it to 150 seconds of ramp up time and 400 seconds of steady time). At the end of a run, faban outputs the average number of operations per second completed by the web servers. In addition to

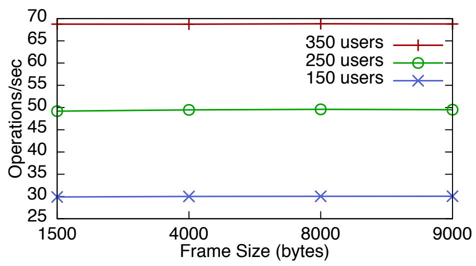


Fig. 7. Average number of operations per second for Olio.

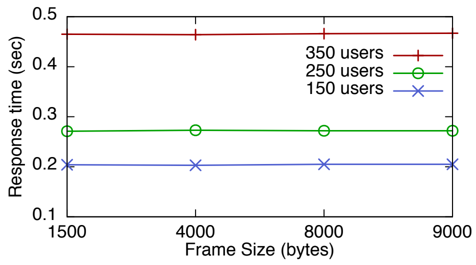


Fig. 8. Average response time for Olio with Nagle disabled.

Component	Ethernet Frame			Jumbo Frame		
	Avg	Max	Min	Avg	Max	Min
LoadBal	8	14	4	7	11	4
Web-1	28	49	12	27	44	11
Web-2	28	48	12	27	45	11
Database	2	6	1	2	6	0.5
Memcached	1	3.5	0	1	2.5	0
NFS	0.75	4	0	0.5	3	0

TABLE II
CPU UTILIZATION (IN %) AT DIFFERENT COMPONENTS OF OLIO.

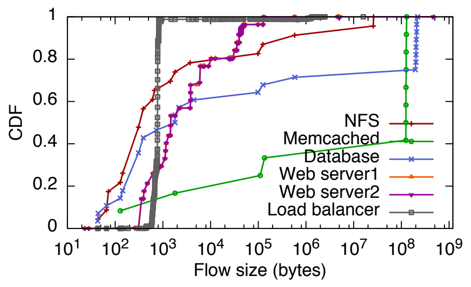


Fig. 10. CDF of flow size distribution at different components of olio.

the number of operations, it also outputs the average response time for various types of operations. We progressively increase the load on the web servers till the response times of difference operations are within the acceptable limit as prescribed with the olio application. In our setup, we found that a load of 250 concurrent users achieves this saturation point. Note that actual number of loaded users emulated by olio is about 100× the number of concurrent users.

Figure 7 depicts the average number of operations per second completed by the web servers with different Ethernet frame sizes. The average is taken over 3 runs for each frame size. We can see that the number of operations per second in case of different Ethernet frame sizes remains similar. The network workload generated by olio operations are mostly of HTTP request-response type. The average flow size for most flows is less than 1KB (Figure 10). In these kinds of requests, there is not much to choose between standard Ethernet or jumbo frames. Even when we have relatively lower or higher number of concurrent users (150 or 350), we see that the number of operations per second stays the same for different frame sizes. The average response time in case of different frame sizes also remains similar as shown in Figure 8.

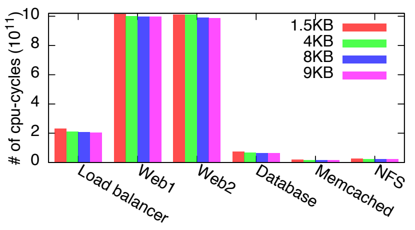
Along with application level metrics, we also record the system level metrics at different components. Figure 9 shows the micro-benchmark results with the olio web application. At a high level we observe that jumbo frames reduce the system overheads slightly for most of the web application components. For instance we can see that for web servers (which are the most loaded component for olio), jumbo frames (8K/9K) save about 3.25% of the cpu cycles during the experimental period compared to the Ethernet frames. For other components also, we observe similar savings in terms of the number of cycles and instructions. The IPC values stay the same for all the Ethernet frame sizes. We see that even in olio, the savings of system resources are similar to other applications like file transfer and Hadoop.

In terms of CPU utilization, jumbo frame (9K) has a marginal benefit over the Ethernet frames. Table II highlights the average, maximum and minimum value of the total cpu used (including usr, sys etc.) at different components of the olio web application. We observe that use of jumbo frames save about 1-2% of total CPU at most the components. This is consistent with our observations with previous applications.

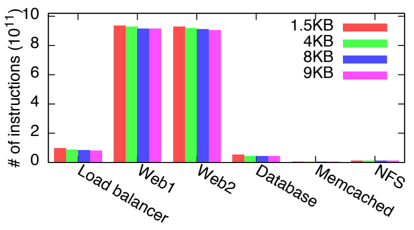
Nagle's algorithm and response time. Nagle's algorithm is known to increase the response time for small requests [23], [24]. As Nagle's algorithm holds the application data (less than the size of MSS) in the TCP buffer until it receives an acknowledgement, it causes more delay in case of jumbo frames. We conducted our Olio experiments with Nagle enabled at first, and saw that the response times with jumbo frames were higher than standard Ethernet frames in Figure 11. Then we disabled Nagle on our system and repeated the same experiments. We make two interesting observations with the new setup. First, the response time decreased for all frame sizes and with different client load. For example, the average response time for 250 users decreased from around 500 ms to 270 ms. Second, the response times became similar for different frame sizes. A more detailed study of the interplay between Nagle's algorithm and responsiveness, however, is outside the scope of this paper.

D. Summary and discussion

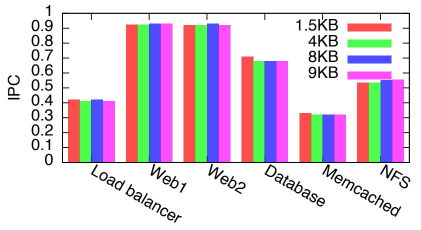
In this section, we studied various properties (introduced in Section III) of jumbo frames in the data center environment. Using the file transfer application, we discovered that the use of jumbo frame leads to increased throughput (P1). Though the gain is not as high as reported in some of the previous work, we still see about 6% improvement in throughput at full link capacity. The gains are higher in case of smaller file transfers, or when we have higher network RTTs. In fact, when



(a) Average number of cpu cycles



(b) Average number of instructions



(c) IPC

Fig. 9. Results at each component of the olio web application.

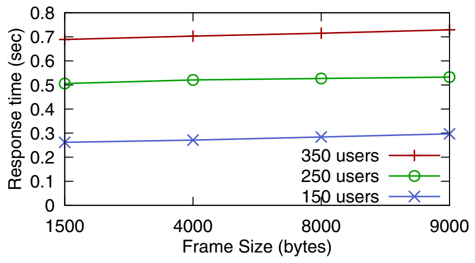


Fig. 11. Average response time of olio with Nagle enabled.

the network latencies are between 1–10ms ([13] reports that in data centers with excessive queuing, delays could be as high as 1–14 ms), the throughput with jumbo frames could be 2-4 times higher. In the Hadoop terasort application, we saw a 3% decrease in completion time with jumbo frames. It shows that jumbo frames are well suited for modern applications with high network requirements. It can increase the application throughput and help them to finish faster. The benefits are even higher in congested networks with higher latencies.

We also studied the delay introduced by jumbo frames using a tiered web service application (P2). When we experimented with default settings we found that the response times corresponding to jumbo frames were higher than regular frames. However, further investigation reveals that the difference was due to Nagle's algorithm which is known to increase response time for small requests. So for this application, we disabled Nagle's algorithm and observe that the response times of the web requests become similar with regular and jumbo frames. The number of operations per second was also similar in both cases. Our experiments show that different frame sizes have similar performance for this class of applications. For all of the above applications, we observe lower resource utilization (P3) when using jumbo frames for data transfer. The savings may not be as high as observed in previous work. But with all the optimizations in modern network cards, we still see a saving of about 1-2% in CPU utilization.

V. Conclusion

In this paper, we revisit Ethernet jumbo frames in the context of data center networks. We empirically evaluate the impact of jumbo frames on a set of canonical data center applications. Our evaluations show that jumbo frames are advantageous to applications like file transfer and Hadoop MapReduce. For a tiered web service, jumbo frames can lead to increase in response time with Nagle algorithm enabled on the system. But with Nagle disabled, it performs as well as with regular Ethernet frames. All of the above observations highlight that turning on jumbo frames can be beneficial for data centers in general.

References

[1] D. Murray, T. Koziniec, K. Lee, and M. Dixon, "Large mtus and internet performance." in *HPSR'12*.
[2] L. Macvittie, "Supersizing the data center: Big data and jumbo frames," http://cloudcomputing.sys-con.com/node/2170934, Feb. 2012.
[3] W. Feng, P. Balaji, C. Baron, L. N. Bhuyan, and D. K. Panda, "Performance characterization of a 10-gigabit ethernet toe," in *HOTI'05*.
[4] J. Chase, A. Gallatin, and K. Yocum, "End-system optimizations for high-speed tcp," *IEEE Communications Magazine*, 2000.
[5] S. Makineni and R. Iyer, "Architectural characterization of tcp/ip packet processing on the pentium® m microprocessor," in *IEEE HPCA*, 2004.
[6] D. Gentry, "Requiem for jumbo frames," http://codingrelic.geekhold.com/2011/12/requiem-for-jumbo-frames.html, Dec. 2011.
[7] "Jumbo frames comparison testing with ip storage and vmotion," http://www.boche.net/blog/index.php/2011/01/24/jumbo-frames-comparison-testing-with-ip-storage-and-vmotion/.
[8] "Ethernet jumbo frames," http://www.ethernetalliance.org/wp-content/uploads/2011/10/EA-Ethernet-Jumbo-Frames-v0-1.pdf, Nov. 2009.
[9] P. Joubert, R. B. King, R. Neves, M. Russinovich, and J. M. Tracey, "High-performance memory-based web servers: Kernel and user-space performance," in *USENIX ATC*, 2001.
[10] "Talking about ethernet jumbo frames," http://community.calix.com/t5/Calix-Community-Blog/, Sep. 2011.
[11] "The promise and peril of jumbo frames," http://www.codinghorror.com/blog/2009/03/the-promise-and-peril-of-jumbo-frames.html, Mar. 2009.
[12] D. Freimuth, E. Hu, J. LaVoie, R. Mraz, E. Nahum, P. Pradhan, and J. Tracey, "Server network scalability and tcp offload," in *ATC'05*.
[13] M. Alizadeh, A. Greenberg, D. A. Maltz *et al.*, "Data Center TCP (DCTCP)," in *SIGCOMM*, 2010.
[14] D. Salyers, Y. Jiang, A. Striegel, and C. Poellabauer, "Jumbogen: dynamic jumbo frame generation for network performance scalability," *ACM SIGCOMM Computer Communications Review (CCR)*, Oct. 2007.
[15] B. Rhoden, K. Klues, D. Zhu, and E. Brewer, "Improving per-node efficiency in the datacenter with new os abstractions," in *SOCC'11*.
[16] D. Grunwald, "A performance analysis of the iscsi protocol," in *MSST2003*.
[17] W. Y. H. Wang, H. N. Yeo, Y. L. Zhu, T. C. Chong, T. Y. Chai, L. Zhou, and J. Bitwas, "Design and development of ethernet-based storage area network protocol," *COMCOM*, 2006.
[18] H. Xiong, R. Kanagavelu, Y. Zhu, and K. L. Yong, "An iscsi design and implementation," in *MSST 2004*, 2004.
[19] "Hadoop cluster setup," http://hadoop.apache.org/.
[20] "Apache olio," http://incubator.apache.org/olio/.
[21] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, "An argument for increasing tcp's initial congestion window," *ACM SIGCOMM CCR*, 2010.
[22] "Faban," http://faban.org/.
[23] "Nagle's algorithm is not friendly towards small requests," http://blogs.msdn.com/.
[24] G. Minshall, Y. Saito, J. C. Mogul, and B. Verghese, "Application performance pitfalls and tcp s nagle algorithm," *SIGMETRICS*, 2000.