ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

Live Migration of Delta Compressed Virtual Machines using MPTCP

[1] Anu V.R., [2] Dr. Elizabeth Sherly
[1] Research Scholar, [2] Professor
[1] M.G .University Kerala, [2] IIITMK, Kerala

Abstract - **Since explained by Clark in 2005, live migration become an inevitable standard feature of hypervisors. Performance optimization process of live migration is an active area of research due to its significance in data centers. Major criteria to improve performance optimization is to reduce the amount of data transferred while migration. To reduce amount of data transferred in migration we use delta compression method .The algorithm used for the same is LZ4; because of better compression ratio and faster decompression rate. To utilize multiple paths simultaneously we use MPTCP transmission network instead of conventional TCP. From the experiments we evaluated that delta compression with LZ4 method reduce the amount of data transferred effectively and reduce down time and total migration time during live migration. Utilizing sub flow feature of MPTCP allow to use multiple paths to the destination server, further reduce total migration time. So the method discussed here is a two-fold solution for the performance optimization issue of live migration.**

Keywords- **Cloud computing, Delta compression, Live migration, Virtualization, TCP.**

I. INTRODUCTION

With the introduction of virtualization technology, many organization are deploying their services on virtual machines (VMs) which are hosted by physical servers in cloud data centres. Most of these services demands support from low latency networking and 24 x 7 services without any interruption. As the policy of cloud data centres to users are 'pay as to go', as the demand varies, data centres can scale up or down the number of VMs which are active for the hosted services. Server consolidation is another high lighting feature of cloud data centre in which the VMs in different servers are consolidated into fewer number of servers to maintain proper power management and power efficiency. Modern data centres are performing fault tolerance management and hardware maintenance without interrupting the online services provided by the VMs. Live migration is the key feature of virtualization technology used to implement all the above mentioned functions of data centres. Live migration transfers CPU , memory and storage states of VM from source server to destination ,ideally without any down time which keeps the migration process as ' live migration'. Total migration time, down time, bandwidth allotted, page dirty rate are the major factors which affects the performance of live migration. Many researchers are providing various strategies to optimize the live migration performance.

Performance optimization of live migration will be improved by reducing the major factors like total migration time, down time, amount of data transferred and page dirtying rate .In these factors the change of page dirtying rate cannot be predicted and controlled externally ,it depends on the nature of application or service running on the VM. By managing other factors, performance improvement of live migration can be done broadly in two ways was such as 1) reduce the amount of data transferred between the servers 2) utilize more number of paths to transfer the data quickly between the servers. The goal of this paper is to improve the VM live migration performance within the available network bandwidth by utilizing more paths simultaneously with reduced amount of data transfer between the servers. Here we utilize delta compression method for reduced data transfer and the multiple path feature of MPTCP protocol for effective utilization of bandwidth [1].

This paper is organized as follows. Section II describes motivation for this work, review and related work. Section III details literature review on Delta compression and MPTCP. Section IV describes the background knowledge about basic system. Section V provides mathematical model of the system. Section VI explains our proposed system model and implementation. Section VII describes the experimental setup and section VIII presents detailed analysis.

II. MOTIVATION

Majority of applications and services hosted on cloud data centres demands uninterrupted service without any

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

latency. So it has to be ensured that there is no performance degradation or resource unavailability even at the time of hardware maintenance like server consolidation, load balancing or server management .Live migration is the only choices which maintain ideally no downtime and very low performance degradation .Hence improving live migration performance is a mandatory requirement in data centres. In this work we contributed two levels of performance improvement in live migration, initially by reducing the amount of data transferred during migration by introducing improved delta compression and encoding scheme on VM snapshots and perform data transfer between servers using MPTCP.

When TCP/IP was designed, host had only a single interface and routers or gateways were equipped with several physical interfaces. Today most devices have more than one interfaces, especially mobile phones came up with both 3G/4G network and Wi-Fi add more multi homed devices on internet. But now a days also 95 % of total internet traffic uses TCP and it demands a manual switch over between different network interfaces and it affect the performance of services running on the host. TCP services has to stop and re-establish during this switch over [2].

Many solutions were put forward by the internet research community to tackle the issue such as shim6, host identity protocol (HIP), some transport layer protocol solutions like extensions to TCP were introduced. SCTP (stream control transmission Protocol) was another option to introduce to handle the situation and even though implemented in several operating systems, it is now limited to specific applications because it demands modifications to services when it is used and also middle boxes got identification issue while using this. But again SCTP cannot support simultaneous use of multiple paths [4, 5].

Instead of TCP protocol here we use MPTCP protocol, so multiple path selection or less traffic path selection is possible which will reduce the migration time. Conventional TCP protocol is a connection oriented three way handshaking protocol which establish the connection between two nodes with tuples such as (IP_{source}, TCP port $_{source}$, $IP_{destination}$, TCP port $_{destination}$)[6].So if any of the value is missing in this tuple there is a high chance to lose the connection. But with the introduction of sub flow management in MPTCP even if the master connection is lost it can maintain communication with additional sub flows. While migration is progressing, due to varying dirty rate the amount of date transferred will be varied .But this variation can be effectively managed with the introduction of this two level optimization. Overall complexity will not be increased by the introduction of this method because performance improvement is done in two different levels.

In Delta compression method, instead of transferring the state of VMs during migration, current and previous versions of VM snapshots are compared and only the change will send to the destination. This change is called delta [7, 8]. This delta is encoded with an encoding algorithms at source server and sent to the destination. For encoding, here we use LZ4 algorithm which gives better compression ratio and high speed of compression. After migration, decompression is performed and updated VMs are regenerated at destination side. Since only delta is migrated, a much reduced amount of data is transferred from source to destination. Additionally we transfer delta using MPTCP protocol, so simultaneously multiple paths or less traffic paths can be used for data transfer. This reduces total migration time and which improves live migration performance.

III. LITERATURE REVIEW

MPTCP (an extension of TCP) is an advance communication protocol formalized in RFC 6824[14], standardised by IETF and used in cloud datacentres for effective implementation of fat tree topologies. MPTCP gives provision to change the endpoints of the connection and have capability to add middle boxes in to on going communication.[2].MPTCP enables fat tree topologies effectively which TCP cannot utilize with same cost as conventional TCP.[1] MPTCP is easily deployable with ECMP(Equal Cost Multi Path) switches. MPTCP can be utilized instead of TCP when there is path diversity and provides seamless mobility, bandwidth aggregation. Since data flow is split over several path MPTCP provides better confidentiality and difficult for the attacker to reconstruct the whole connection flow. MPTCP provides lower response time by utilizing multiple paths for data transfer and hence congested paths can be avoided .One of the great advantage of MPTCP is backward compatibility, applications running in TCP can run in MPTCP without any change. [1] Provides performance improvement of MPTCP, which involves the path selection procedure and congestion control mechanism of MPTCP. [2] Put light on embedding middle boxes into the internet architecture with MPTCP. Experiments show that connection redirection, connection migration and virtual machine migration are efficient and perform well with MPTCP. [3] Discussed the high level design of MPTCP with compatibility goals. This paper describes implementation of MPTCP using Open-stack.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

Demonstrated how different mechanisms at network and transport layer can significantly improve the performance of live migration with MPTCP and also explains the use of MPTCP in bandwidth aggregation. [13] Proved the capability of MPTCP for data transfer in mobile devices using wireless communication in 3G/4G networks. MPTCP utilize network resources effectively, [14] details design and congestion control for MPTCP.MPTCP never harm other traffic and always chose a network with less traffic and help to control congestion effectively.[4] tries to implement multipath TCP in today's Internet with ordinary middle boxes such as NAT and firewalls and provides solutions for limited receiver buffer between multiple flows.

P. Sv¨ard, J. Tordsson [7], proposes a page reordering technique that reduces the amount of transmitted data by sending the memory pages in reverse order of usage frequency to reduce re-transfer rate. Xiang Song [15] proposes a method for Parallelizing live VM migration operations, includes applying data parallelism and pipeline parallelism to most primitive operations. P Svärd, B Hudzia [1, 7] propose a live migration algorithm where the VM in-memory state is compressed during transfer and reordering the packets while transferring.

IV. BACKGROUND

a. Live Migration with Open stack: Live migration solves the problem of migrating virtual machines from one server to another without affecting the services provided by guest applications and the approach first demonstrated by Clark [3].Even though the major strategies for migration includes pre-copy or post-copy methods, the typical migration process implements the following steps. Initially all pages are marked dirty and transfer the whole memory state of guest from source VM to destination. Then iteratively transfer all dirty pages from source to the destination. Once the number of dirty pages are below certain threshold value, stop the execution of VM at source and transfer all state information of CPU, Memory and storage as fast as possible to destination and synchronize all the VM image(s) on guest at destination. Immediately after successful VM migration at destination, broadcast an Ethernet packet to announce new location of NIC(s). Furthermore, the whole migration process is completely transparent to the services running in the migrated VM and do not need to be migration-aware in any manner. In all migration strategies the whole VMs or states of VMs will be transferred from source to destination .But the major concern is that, since VMs are rich in size, while progressing migration process, huge amount of data need to be transferred between the servers. An optimization method required to be implemented for reducing the amount of data transferred at the time of migration. Here we use delta compression and encoding algorithm.

There are many tools available to implement Live VM migration in data centres [21]. Open stack is one of the popular open source cloud platforms which is based on several services like nova, swift, glance, horizon which exchanges messages through AMPQ (Active Message Queuing Protocol) .Neutron is the service which provides network as service in open stack. Here we utilize Neutron along with KVM hypervisor and QEMU for live migration. The detailed procedure for live migration is as follows. First request is received by controller for migration via nova compute API. Nova conductor verifies availability of storage for live migrations. Then compatibility of transfer between hosts are checked using AMPQ RPC call. Nova compute generate instance directory and create empty instance disks for live migration [16].An additional delta compression module is added at nova compute module. After preparing instances for migration at source side, delta compression of instance is done and encoded with LZ4 algorithm and sent to the destination.

b. MPTCP

Major aim of introduction of MPTCP is to realize resource pooling across the network to increase the resilience of connectivity between different paths and increase the efficiency of resource usage and thus improve throughput [17]. MPTCP can handle multiple paths between two end points with following compatibility goals,

- The performance of MPTCP should not be poor as any of a single path flow on the best route.
- The capacity taken by a multi path flow should not be more than a single path flow using that route
- To reduce congestion multipath flow should avoid most congested routes.

MPTCP service model is in order, byte oriented and reliable as that of TCP. The transport layer is divided into two sub layers for implementing MPTCP. The upper sub layer called semantic layer manages application oriented transport functions like connection establishment, packet reordering, congestion control etc. and operates in end to end.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

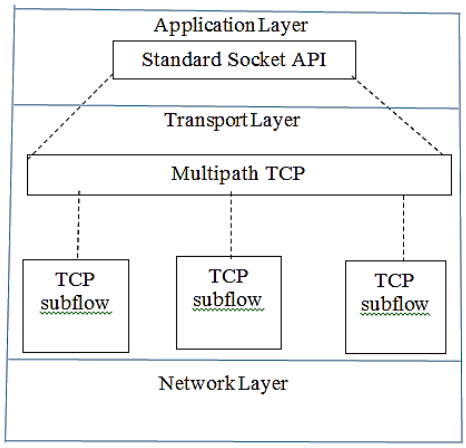

Fig 1: Protocol Architecture of MPTCP

While the lower sub layer called Flow endpoint layer collects a set of sub-flows that can be seen as one TCP flow. MPTCP have a shim layer represented in fig. 1 .The shim layer works between the application and the TCP stack which combines several TCP connections called sub-flows in MPTCP.A 5-tuple representation is used to characterize a sub-flow as (IP source, TCP port source, IP destination, TCP port destination) and assigned with a sub- flow id generated by MPTCP stack [2]. To perform packet reordering on several sub-flows, MPTCP add a global sequence number (GSN) shared among sub-flows and exchanged through TCP options. To map the multiple TCP sub- flows in MPTCP, the GSN is mapped with Sub-flow Sequence Numbers (SSN) through Data Sequence Signal (DSS).

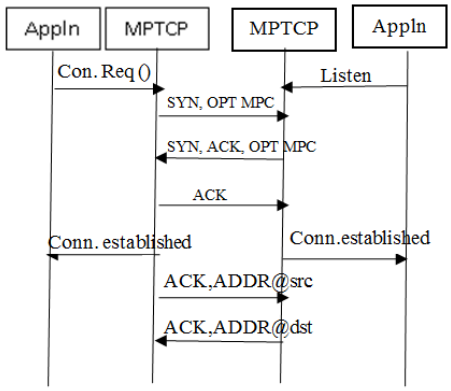

Fig 2: MPTCP Connection establishment

Fig 2. Narrates the MPTCP connection establishment between two nodes. The connection is established through TCP socket interface through connect system call. This first TCP connection is called the master connection.

A random key is generated along with this process and this key is hashed later and used to authenticate additional sub-flows [3]. It is same as the three way handshake connection mechanism in TCP. To manage additional sub-flow, the host can open a new sub-flow as soon as the acknowledgement signal for DSS is received.

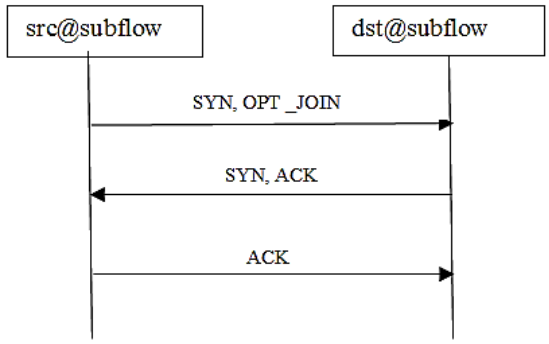

Fig 3: MPTCP Sub-flow initiation

Both client and server can create sub-flows either through a new connection or announce a token (IP, port) that the peer can connect to. Strategies are not yet standardised for the sub- flow connection opening/closing and it as per local policies. Due to the presence of NAT, it is wiser to initiate sub- flows from client side.
Since additional sub-flows are also there, many features have to be taken care of while MPTCP is implemented like congestion control, flow control, acknowledgement management, last packet recovery, failure management, data distribution over paths, managing out of sequence data arrival, path management. In our optimization method we are interested only in congestion control and data distribution over paths features.

Congestion control: MPTCP allows more sub-flows for improving throughput, robustness and thus utilizing networking resources more effectively than conventional TCP .But TCP fairness in the sub- flow paths are always a controversial issue. Sharing the same resources by the sub-flows creates bottleneck in network path in real scenario [4]. Buffering policies, single resource pooling principle are some methods suggested to manage congestion control. There is no standardized algorithms to manage congestion control effectively in MPTCP. From different references it is understood that developing local congestion control algorithms is optimum as per application requirements. Sender regulates its throughput according to the available network. In each sub-flow level a congestion window is maintained, by changing the

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

window size as per the requirement will minimize the congestion and improves the network performance. At the receiving end there is a global receiving window for established sub- flows. Four different algorithms have been proposed by [17] to manage congestion control by adjusting window size are Uncoupled, Fully coupled, Linked Increase, and RTT compensator. [16] Tries to give a solution for congestion control by shifting quantity of traffic from more congested paths to less congested one according to the local knowledge on network resources and congestion status. A fair an efficient traffic shifting always strives to equalize the extent of congestion on all available paths namely "Congestion Equality Principle".

c. Delta compression: Major bottleneck with migration algorithms are the time taken to transfer pages from source to destination through the network due to network congestion and down time during migration, even GB Ethernet is slower than RAM or disk access. This delay leads to poor migration throughput and ultimately ended in long down time and interruption in services. Shortening downtime is one of the key optimization factor in live migration process. Lot of research has been carrying out to reduce the downtime considerably low during migration process, and effective utilization of bandwidth. The amount of data transferred during the downtime can be effectively reduced by delta encoding method. Instead of transferring all the memory pages, only the dirty pages are transferred to the destination server. Since the difference between previous page and current modified page (delta) are only transferred, the amount of data transferred is too small as compare with other strategies. P Svärd, B Hudzia [7, 8] incorporated delta compression method in live migration process for CPU or memory intensive virtual servers. In his method, initially all the RAM pages are marked as dirty and sent to the destination. Thereafter, only the modified pages are iteratively transferred. If a longer time an iteration takes, the more number of pages get dirtied during the process and has to retransmit to destination. When the remaining amount of pages are below a threshold value, VM suspended, avoid more memory writes and transfer all remaining pages (down time). But longer the downtime, more amount of memory get dirtied and need to be retransferred. Compression reduces the size of VM by transferring only the dirty pages between the servers at the time of migration. Transfer time is one of the prime factor in which the performance of VM migration measures. The transfer time can be calculated as the ratio between the amount of RAM remaining and the bandwidth allotted. Since compressed delta pages are only transferred, the transfer time is improved much as compared with transfer time in normal live migration .This helps to improve the total migration time also.

V. MATHEMATICAL MODEL

Mathematical model of Delta compression: The data transmitted in round i can be calculated as:

$$D_i = \begin{cases} V_m, & if\ i = 0 \\ K.T_{i-1}, & Otherwise \end{cases} \quad (1)$$

Let K be the page dirtying rate and R be the memory transmission rate .Let α is the ratio between dirty page rate with memory transfer rate and it is small at initial time.

$$\alpha = {}^{K}/_{R} \quad (2)$$

Let T_i be the elapsed time at each round

$$T_i = \frac{K-T_{i-1}}{R} = \frac{V_m.K^i}{R^{i+1}} \quad (3)$$

Combining (1), (2) and (3), we have the network traffic during the round *i.*

$$N_i = K.\frac{V_m}{R}.\alpha^{i-1} = V_m.\alpha^i \quad (4)$$

The relation between network traffic and bandwidth available is

$$N \propto 1/B$$

From the (4) improving the utilization of bandwidth the following scenarios are favorable. Either page dirtying rate (K) has to be reduced, or memory transfer rate (R) has to be increased or V_m size has to be reduced [13]. The page dirtying rate depends up on the change or updation happened to VM during running and it cannot be controlled externally. Memory transfer rate is set initially and cannot be change dynamically as per each requirement. But a considerable positive change can be made in the case of VM size, while migrated from one server to another. Instead of sending the whole VM snapshots repeatedly, only the difference in the present and previous snapshots are being sent. But the delta page has the same size as that of memory page. In most of the applications a specific part of VM's RAM is constantly dirtied and this fact leads to delta compression is an effective mechanism to reduce the amount of data transferred through VM migration. In [7] P Svärd, B

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

Hudzia used RLE method for compression of delta pages. The idea behind RLE data compression is to replace the *n* occurrence of data item *d* with single pair *nd*. The *n* consecutive occurrence of a data item are called a run length of *n*.

For delta encoding at source end and decoding at destination end, the difference between current page and previous page is required. At destination side, it is not difficult because the sources' previous version of VM page is the destinations' current version of VM page [20] .But in the source side the pages are continuously overwritten and difficult to get the delta page. To overcome the difficulty, the pages in the cache are used at source side to take the delta difference. The pages from cache is effective only for similar hot page set. If different pages are used or memory intensive operations are taking place during migration, this caching mechanism will not be a feasible solution for providing previous VM pages for taking delta difference [23, 24]. In existing system, for page replacement in cache two way set associative mechanism is used.

Mathematically the performance of live migration with delta compression is compared with live migration without compression in the following way. Let t_{nd} be total migration time without compression, t_{tr} is the transfer time.

$$t_{nd} = t_{tr} \quad (1)$$

If the previous page required for delta compression is available in cache (cache hit) then total migration time t_{hit} is,

$$t_{hit} = t_{ct} + t_{cpt} + t_{chtr} + t_{cdt} \quad (2)$$

Where t_{cpt} is compression time required in cache hit, t_{chtr} is the cache hit transfer time and t_{cdt} is the cache hit decompression time at destination.

If the required previous page delta compression is not available in cache then cache miss occurs and total migration time t_{miss} is,

$$t_{miss} = t_{ct} + t_{tr} \quad (3)$$

Let Pg be the total number of pages and n is the total number of iterations and h is cache hit ratio.

Total migration time = $\sum_1^n Pg\ t_{nd}$

$$= \sum_1^n Pg_{hit}\ t_{hit} + \sum_1^n Pg_{miss}\ t_{miss}$$

$$= n((t_{hit}.h) + (1-h)\, t_{miss}) \quad (4)$$

By comparing total migration time without compression and with compression both in cache hit and cache miss, total time gain with delta compression over non compression is,

$$t_{gain} = ((1-h)(t_{miss}-t_{nd})-h\,(t_{nd}-t_{hit})) \quad (5)$$

Mathematical model of MPTCP: This model describes the congestion control and sub- flow management of MPTCP. It works with congestion equality principle which states that a fair and efficient traffic shifting implies that every flow strives to equalize the extent of congestion that it perceives on all its available paths [4]. In a network let S be set of links with capacity c= (c_s, s∈ S), which are shared by a set of F flows. A path p ∈ P is defined as the subset $S_p \subseteq S$. The relationship between F and P can be represented with a matrix R, where $r_{s,p} = 1$, where s ∈S_p and $r_{s,p} = 0$ otherwise. Each flow f ∈ F is associated with a subset of path $P_f \subseteq P$. This relationship can also be represented by a matrix called flow matrix T, where $t_{f,p} = 1$ if p ∈ P_f and $t_{f,p} = 0$ otherwise. Let $v_{f,p}$ be the rate of flow f on path p and total rate of flow is the summation of each flow [25]. Let the total rate of flow $U_f = \sum_{p \in Pt} t_{f,p}$.

Then utility can be defined as $D_s(c_s)$.So in congestion control we try to determine appropriate rates for the flows so that it maximize the total utility subject to link capacity constraints by varying total rate of flow U_f.

Corollary to Congestion Equality Principle: In the above mathematical model if every flow strives to equalize the extent of congestion that it perceives on all its available paths by means of shifting traffic, then network resources will be fairly and efficiently shared by all the flows.

Managing congestion control is a significant task in MPTCP network to reduce any delay between the source and destination nodes. So supporting to Congestion Equality Principle the above mentioned corollary point out that shifting of traffic from overloaded path to other paths is better to be in equal rate which improves more efficiency than any other policies.

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

VI. SYSTEM MODEL

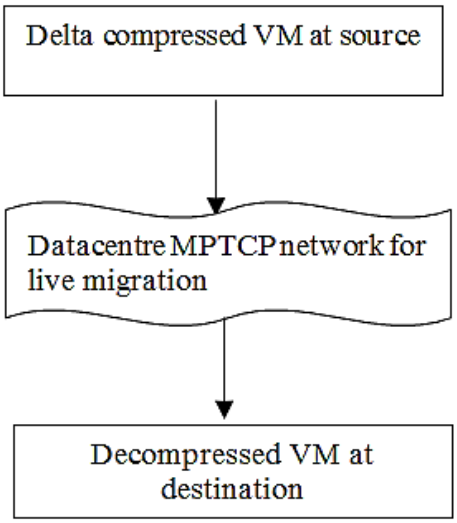

Fig 4: system architecture

The basic algorithm of LZ4 is Lempel Zev Welch algorithm and here each string is encoded as a token using a dictionary. The most important design principle behind LZ4 is its simplicity. The dictionary is created dynamically in the compression process and no need to transfer it with the encoded message for decompression at other end [8, 20].It improves overall performance by reducing total amount of data transferred while migration. At decompression of message at destination the dictionary is created dynamically. For compressing long data streams multiple blocks are required. For these multiple blocks a common header is there to handle its content. A compressed block in LZ4 composed of sequences. A sequence is a group of literals (constants which are not compressed bytes), followed by a match copy and each sequence is starts with a token. Token is an 8 but value separated into two 4 bit fields. So it range is from 0 to 15.In this 15 bits ,the first high 4 bits of token is used to represent length of literals to follow and if that value is zero means there is no literal. If it is 15, then we need to add some more bytes to indicate the full length. Each additional byte then represent a value from 0 to 255, which is added to the previous value to produce a total length. When the byte value is 255, another byte is output. There can be any number of bytes following a token. There is no "size limit"[22].

LZ4 delta compression algorithm at source side,

1. If the previous page is in cache, then
    a. Delta page is created for the current page
    b. Delta page is compressed with LZ4
    c. LZ4 page is transferred to destination
2. If the page is not in the cache, then
    a. Update cache with the page
    b. Perform strep 1.

LZ4 delta decompression algorithm at destination side,
At destination side, search for the encoded page
If found
Decode LZ4
Recreate delta page
Recreate original page from delta page

VII. EXPERIMENTAL SETUP

Experimenting with MPTCP is a challenging task under real scenario .Receiver/sender buffer capacity, window management, RTT management, congestion control and shifting traffic while in congestion are challenging in experimental set up. This section describes different phases performed to achieve MPTCP test bed implementation. The objective is to evaluate MPTCP performance over TCP while performing delta compression live migration. Experimental set up for performing live migration contains three nodes with open stack cluster and use traffic generator to congest the network to analyse the performance of nodes through MPTCP at congestion control. The cluster nodes are installed with Ubuntu 14.04 LTS 64 bit server with MPTCP version 0.89.The open stack cluster run Nova compute at each node .Every node is equipped with a dual port gigabit Ethernet card where one interface for open stack management and other is for VM communication. The hypervisor used is KVM QEMU (version 2.0.0) and also include libvirt 1.2.2. We use iperf and netperf-wrapper to congest the network. The TCP sub-flow management of MPTCP is used for managing the network traffic. We use the congestion control mechanisms available in MPTCP to shift traffic from congested paths to less traffic path .In the experiment network a 5 port gigabit switch is used.

In our experiment we add additional delta compression and decompression module into each compute node in the network. While performing live migration, VM is delta compressed by the compression module and sent through the MPTCP network to destination where it is

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

decompressed by decompression module. Here we use LZ4 delta algorithm for compression and decompression.

VIII. EVALUATION

a) Live migration under MPTCP: In this scenario, initially we migrate a VM which runs OS and basic applications with a total size of 2GB.Progressing the migration process we transferred more VMs with bigger size and analyse downtime and total migration time .For 2 GB VM, when TCP is used in the network, migration TMT is 25 s and downtime is 210ms and when MPTCP is used, TMT is 15s and downtime is 210ms (same in both cases).While using LZ4 algorithm for encoding and decoding compression ratio is 0.79.

b) Delta compression live migration under MPTCP: Here we conducted live migration of VMs through delta compression. Delta compression is a method for storing data in the form of changes between versions called 'delta 'instead of the full data sets, so the amount of data transferred is reduced. This improves total migration time (TMT).For that we include the delta compression module at source server and decompression module at destination node. Before transferring VMs through MPTCP network; it is delta compressed and encoded with LZ4 algorithm. As compared with other encoding algorithms LZ4 algorithm shows better compression ratio. Using MPTCP the 'delta' packets took various simultaneous paths to reach the destination, which also improves overall performance of live migration. From our experiments it is clear that when MPTCP have eight or more number of sub flows, it effectively manage congestion control and show considerable improvement in total migration time.

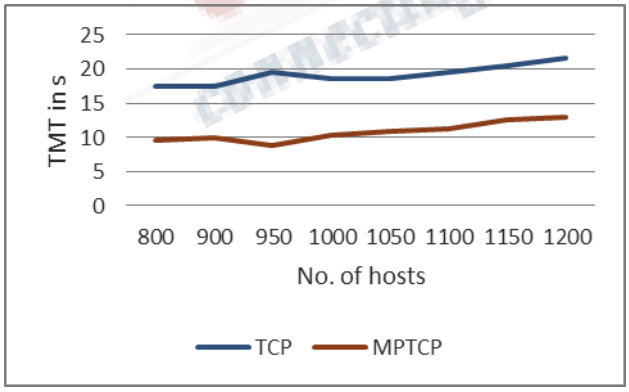

Fig.5: comparison of TCP vs. MPTCP in terms of total migration time

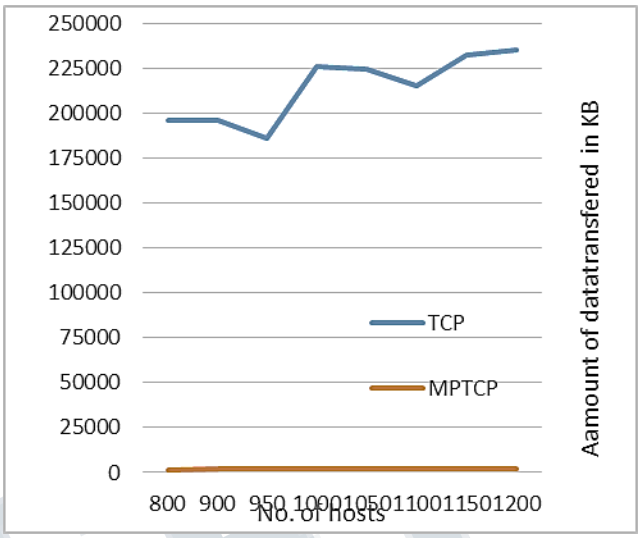

Fig.6: comparison of MPTCP vs. TCP in terms of amount of data transferred.

c) Compare the performance of LZ4 with XBRLE: In [] P Svard & B.Hudzia described about a delta compressed live migration technique in TCP network where XBRLE was used. Since the VM memory pages are in the binary form, so it is easy to compute the delta page by taking the difference between the current and previous versions of a page. In XRBLE method delta page is computed by applying XOR between current and previous versions of a page. Size of delta page is same as that of page try to transfer from source to destination. So to reduce the amount of data transferred, delta page size is reduced using RLE algorithm. The reverse of this process produce the current version of page at destination side.

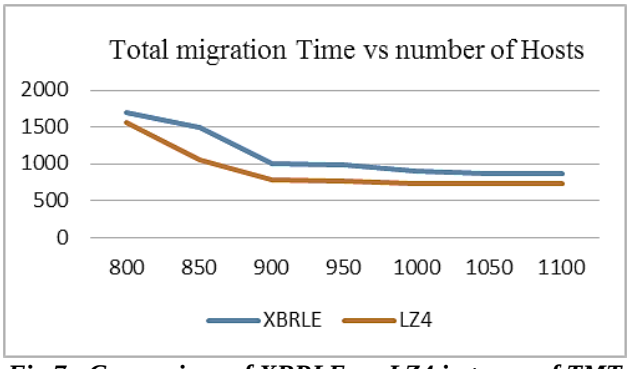

Fig 7: Comparison of XBRLE vs. LZ4 in terms of TMT

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

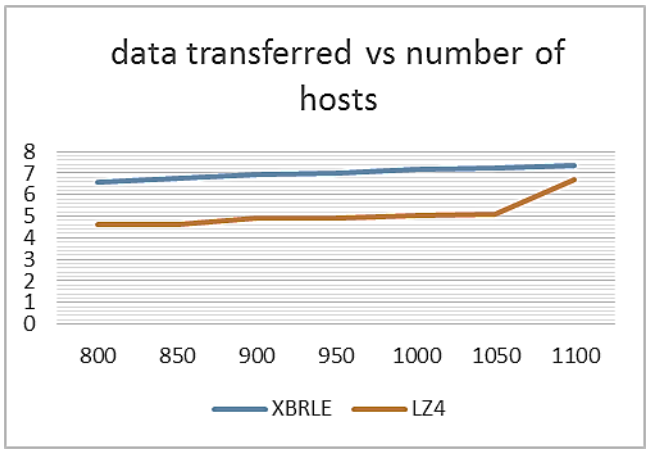

Fig 8: Comparison of XBRLE vs. LZ4 in terms data transferred.

Since the user input is unpredictable, most of the time it gives worst input to RLE algorithm which decreases the efficiency of compression process thus lead to performance degradation in live migration. In the memory intensive VM migration cases RLE algorithm generates the output data which is 2 times more than the size of input data and it is due to the fewer amount of runs in the source file. In majority of situation we cannot predict the dirty bit change rate and if it is too high RLE is not an effective solution for compression. To overcome these disadvantages, here we use LZ4 algorithm which is dictionary based compression technique and independent of nature of input. LZ4 shows high speed of compression and better compression ratio than XBRLE method. Better compression ratio of LZ4 improves downtime and total migration time (TMT). As compared with compression, decompression is faster in LZ4.

CONCLUSION

In this paper we introduce a new approach for live migration by combining delta compression with MPTCP protocol. We have implemented and evaluated MPTCP mechanism with LZ4 compression algorithm. This method has more advantages as compared with conventional live migration strategies because it reduce total migration time (TMT)-a major performance metric of live migration - effectively. With delta compression and MPTCP the amount of data transferred is reduced in double rate and this drastically reduce total migration time.

REFERENCES

[1] Raiciu, Costin, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. "How hard can it be? Designing and implementing a deployable multipath TCP." In Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, pp. 29-29, 2012.

[2] Bonaventure, Olivier. Multipath TCP: An annotated bibliography. Technical report,, < https:// github. Com/ obonaventure /mptcp - bib, April 2015.

[3] Arzani, Behnaz, Alexander Gurney, Shuotian Cheng, Roch Guerin, and Boon Thau Loo. "Impact of path characteristics and scheduling policies on MPTCP performance." In Advanced Information Networking and Applications Workshops (WAINA), IEEE 2014 28th International Conference on, pp. 743-748, 2014.

[4] Cao, Yu, Mingwei Xu, and Xiaoming Fu. "Delay-based congestion control for multipath TCP." In Network Protocols (ICNP), 2012 20th IEEE International Conference on, pp. 1-10. 2012.

[5] Alizadeh, Mohammad, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. "Data center tcp (dctcp)." In ACM SIGCOMM computer communication review, ACM ,vol. 40, no. 4, pp. 63-74, 2010.

[6] Raiciu, C., Pluntke, C., Barre, S., Greenhalgh, A., Wischik, D. and Handley, M.,October. Data center networking with multipath TCP. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (p. 10). ACM. 2010.

[7] Svärd, P., Hudzia, B., Tordsson, J. and Elmroth, E., Evaluation of delta compression techniques for efficient live migration of large virtual

machines. ACM Sigplan Notices, 46(7), pp.111-120. 2011.

[8] David Salomon "Dictionary Methods" in Data Compression: The Complete Reference Fourth Edition, Springer, pp. 141-259, 2007.

[9] Kwon, S.J., Kim, S.H., Kim, H.J. and Kim, J.S., LZ4m: A fast compression algorithm for in-memory data. In Consumer Electronics (ICCE), 2017 IEEE International Conference pp. 420-423, 2017, January.

[10] Kane, J. and Yang Compression speed enhancements to lzo for multi-core systems. In Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium pp. 108-115,2012, October.

[11] Wischik, D., Raiciu, C., Greenhalgh, A. and Handley, M., March. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In NSDI ,Vol. 11, pp. 8-8, 2011

[12] Bonaventure, O., Handley, M. and Raiciu, C., An overview of Multipath TCP. ; Login: 37(5), p.17, 2012.

[13] Nakasan, C., Ichikawa, K., Iida, H. and Uthayopas, P., A simple multipath OpenFlow controller using topology-based algorithm for multipath TCP. Concurrency and Computation: Practice and Experience, 29(13), 2017.

[14] Coudron, M. and Secci, S., An implementation of Multipath TCP in ns3. Computer Networks, 116, pp.1-11.2017.

[15] Peng, Q., Walid, A., Hwang, J. and Low, S.H., Multipath TCP: Analysis, design, and implementation. IEEE/ACM Transactions on Networking, 24(1), pp.596-609, 2016.

[16] Chen, Y., Wu, X. and Yang, X., 2011. MAPS: Adaptive path selection for multipath transport protocols in the Internet. Duke Univ., Durham, NC, USA, TR-2011-09.

[17] Chihani, B. and Collange, D. Simulation-based study of MPTCP (Multipath TCP). arXiv preprint arXiv:1112.4742, 2011.

[18] Bartík, M., Ubik, S. and Kubalik, P, December. Lz4 compression algorithm on fpga. In Electronics, Circuits, and Systems (ICECS), 2015 IEEE International Conference on pp. 179-182, 2015.

[19] Collet, Y.: "RealTime Data Compression: Development blog on compression algorithms". [Online]. Available: tinyurl.com/qc9yve4

[20] Leelipushpam, P.G.J. and Sharmila, J. Live VM migration techniques in cloud environment—a survey. In Information & Communication Technologies (ICT), 2013 IEEE Conference pp. 408-413, 2013, April.

[21] Waghulde, R., Gurjar, H., Dholakia, V. and Bhole, G.P., New Data Compression Algorithm and its Comparative Study with Existing Techniques. International Journal of Computer Applications, pp.102-107, 2014.

[22] Fowler, J.E., March. Delta Encoding of Virtual-Machine Memory in the Dynamic Analysis of Malware. In Data Compression Conference (DCC), pp. 592-592, 2016.

[23] Deshpande, U. and Keahey, K., Traffic-sensitive live migration of virtual machines. Future Generation Computer Systems, 72, pp.118-128, 2017.

[24] Chihani, B. and Denis, C., 2011. A Multipath TCP model for ns-3 simulator. arXiv preprint arXiv:1112.1932.

[25] Barré, S., Paasch, C. and Bonaventure, O, Multipath TCP: from theory to practice. NETWORKING 2011, pp.444-457,2011