International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), Volume 5, Special Issue 2, May 2015)
International Conference on Advances in Computer and Communication Engineering (ACCE-2015)

A Cached Middleware to Improve I/O Performance using POSIX Interface

Jesna fathima[1], Dr Ambareesh S[2]
[1]PG Scholar, Dept of CSE, Vemana IT
[2]Associate Professor, Dept of CSE, Vemana IT
[1]jesnajaseem@gmail.com, [2]ambareeshs@vemanait.edu.in

***Abstract*--The performance of computing devices is increasing day by day, the storage devices are not capable to reach with this benchmark. The compute and storage resources are forming two parts of the shared network. This paper is presenting a storage middleware called HyCache+ to reduce the high bi-section bandwidth of the parallel computing systems. The middleware uses POSIX interface to the end users to swap the data with the high capacity network attached storage. The system uses a 2-layer scheduling approach. The HyCache+ can be deployed on the IBM BlueGene/P supercomputers for the performance comparison**.

***Keywords*-- Middleware, parallel file system, POSIX**

I. INTRODUCTION

There are a lot of challenges in the computation environment. One of the most important among them is the fastest development of the computational resources and the lack of better storage. This made a big gap between the nodes and the input output. The study of fusion science [8] shows that, it can produce an output of 3.5 TB per second in 2005, where as fastest storage system [11] in 2013 was able to deliver only 1.4TB per second output. The reason for such a big drop was that of the slowest development of storage resource.

With the advantage of high performance computing system, the gap of input output and computational nodes increased further more the important problem was that of allocation of the computational as well as the storage resources as two parts. They are providing a high bandwidth. This is supporting compute instance petascale applications, but not suitable for the data intensive petascale computing. The solution for the problem was to allocate the storage and compute resources together, so that it can provide a high concurrency level.

The paper presents HyCache + a middleware in the computational node to manage the heterogeneous storage device. For distributed file systems. The HyCache+ provides with a standard POSIX interface, this can be used for holding the hot data which will be usually the meta data and it can interchange with the cold data which was stored in the parallel file system. The newly implemented HyCache + can provide high performance as well as cost effective high capacity as shown in figure 1.

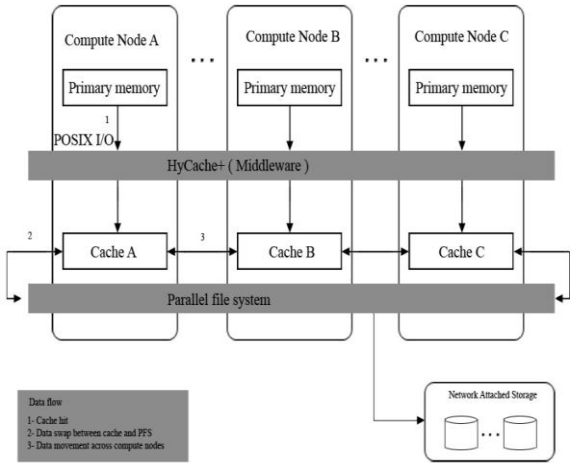

Figure 1: HyCache+ hierarchy

In HyCache the middleware for the distributed file system was implanted and this study was the basic for the HyCache +. The POSIX interface for the HyCache + was implemented using FUSE [3] in addition to HyCache+ , HyCache+ supports networked storage and the data reliability, as summarized in Table 1.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), Volume 5, Special Issue 2, May 2015)
International Conference on Advances in Computer and Communication Engineering (ACCE-2015)

TABLE I:
COMPARISON OF HYCACHE+ OVER HYCACHE

Feature	HyCache	HyCache+
NAS	No	Yes
Data transfer	local	Local & Remote
Metadata	Symbolic link	DHT
Scalability	1 node	1024 nodes
Replicas	1	Arbitary

In [13] data diffusion approach that acquire the compute and storage resources dynamically as well as replicates data in response to demand, HyCache + implements a two layer scheduling for improving the data locality of the cached data. The first layer schedules the jobs on available nodes to minimize the file size that needs to be transferred the functionality of the second layers is to allocate the data locations across different storage resources. The HyCache + middleware can be installed in the IBM BlueGene/P super computer for making a comparison between the two layer scheduling and the least recently used algorithm.

II. The Design And Implementation Of HyCache +

The HyCache + is implemented in a two nodes cluster as shown in Figure 2. The proposed job scheduled can be implemented in any of the nodes. In this design the job scheduler was deployed in the node 1.For the transfer of data that is either data read or write, HyCache+ uses a specific manner. The write operation uses the local node whereas the read can use either the local or remote node provided the remote node should be as close as possible.

In some scenarios the file to be written originates from a remote node, the job scheduler will schedule the write job in the system in which the global job scheduler is indented. Once the file is modified instead of sending back the data file, the metadata is send back by updating the file. This approach is used since the cost of modifying metadata is less than the original file.

The hot files contained in the local cache are swapped with the cold files in the storage server according to the caching algorithm.

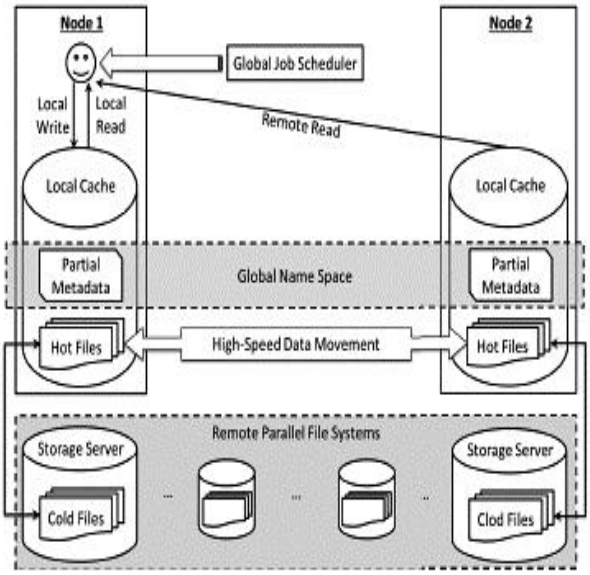

Figure 2:HyCache+ design

User interaction

The HyCache + should be implemented in such a way that it is completely transparent to the end users. The users are basically unaware of which files are stored in which physical nodes . This drawback can be compromised by providing a global view of the metadata of the file stored in different storage space.

POSIX is used as the interface for the middleware inorder to maintain the compatibility between operating systems. FUSE framework [3] to implement a fully functional file system in a user space program was leveraged in the HyCache+ sytems. The FUSE that supports the multithreaded programming has for context switches. The two switches between caller and virtual filesystem, then another two between LibFuse(the user space library) and the FUSE.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), Volume 5, Special Issue 2, May 2015)
International Conference on Advances in Computer and Communication Engineering (ACCE-2015)

Network protocols

The network protocols are enclosed within LibNap. It will basically support four protocols. They are TCP,UDP,MPI and UDT.TCP and UDP is selected depending on whether the channel is connection oriented or connectionless.MPI the message passing interface is a standardized and portable message passing system.UDT[9] is the user level protocol having high reliability. For the purpose of multithreading epoll is used. The drawback with epoll is that the message received are not arranged in the same order in which it was send. For determining the message a message_id is appended to each of message packet.

Metadata Management

The metadata management is an important clique in the HyCache+ system. The system is deployed by modifying the previous work ZHT[5].The distributed hash table in it contains the key value pair that can be either on the local node or remote node.DHT is used as an intermediate between the partial metadata and global namespace. On choosing an appropriate hash function, the key value pairs will be distributed across all the nodes and thus can achieve loadbalance. In addition to DHT some details could be included. It is owner id, modification time and so on. These can be included according to the space availability.DHT has some strict consistency rules, only the primary copy should be used for read and write. Replicas are needed to avoid data loss in case of node failures.

Data transfer

The client can only write to its local storage nodes. The write throughput is always flawless, all the writes are related to the local I/O throughput and neglects the overhead which are commonly seen in the other systems. If the client wants to write to file that is stored in any other nodes ,according to the rule, the updated file will not be sent back instead the metadata is updated and send. A priority queue to keep track of hot files is maintained in cache. The queue possess name of file and file size.

Fault tolerance

To achieve fault tolerance the primary method is to make a number of replicas or copy of the system.

So that when the primary copy fails one of the replicas can be restored to replace with the failed copy. The method has its own easiness of use and has less computational overhead. But has a drawback that it's having less storage efficiency. HyCache+ concentrates on three things to achieve data redundancy, synchronous replica, asynchronous replica and erasure coding. Asynchronous will give better throughput while s synchronous update is a costlier method.

III. Two Layer Scheduling Of The Cache

Job scheduling strategies

When the application needs to get the file from remote machine, a set of constraints should be considered to determine where the file is exactly. Let *M* be a machine from a set of machines, *A* be the application that is running, *F* is set of all files required. *Fx* , set files referenced by *Ax* ε *A*.

Pk,i is the file placed on *Mi and Rk,i* is the application on a particular machine. The general problem was to find *R* to minimize the total network running costs. Two constraints are considered here. The first one is that the file could be placed on exactly one node.

$$\sum_{Mi \epsilon M} Pk,i = 1, \forall\, Fk \,\epsilon\, F$$

The second constraint is that the job should be scheduled on exactly a single node. It satisfies the below equation

$$\sum_{Mi \epsilon M} Rk,i = 1, \forall\, Ak \,\epsilon\, A$$

Pk,j, Rk,i ϵ {0,1},$\forall k, i$

This implies that both the matrices should store only the binary values {0,1} to guarantee the constraints.

The algorithm to find the minimal network cost is given in Algorithm 1.The input for the job index was *a*, and it will return the machine index *b*.

Algorithm 1 Global scheduling

Input: *a* th job scheduled

Output: *b* th machine in which a th job is scheduled

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), Volume 5, Special Issue 2, May 2015)
International Conference on Advances in Computer and Communication Engineering (ACCE-2015)

```
1.function GlobalSchedule(a)
2.    MinCost← ∞
3.    b ← NULL
4.    for Mk ∈ M do
5.        Cost ←0
6.        for Fi ∈ F do
7.            find Mx such that Pi,x=1
8.            Cost ← Cost + Size(Fi)
9.        end for
10.   if Cost < MinCost then
11.       MinCost ← Cost
12.       b ← k
13.   end if
14. end for
15.return b
16.end function
```

In the algorithm for global scheduling a priori input is used so that we can evaluate the correctness of the procedure. The network cost of each job is less. The jobs are considered to be independent and hence can conclude that the overall network cost will be minimal.

IV. EVALUATION

The simulation can done in the IBM Bluegene/P supercomputer. It consists of 160k cores.

For the evaluation 1024 nodes which is 4096 cores can be used. Each experiments can be repeated 5 times until it become stable enough and the average of all was taken.

A.FUSE Overhead

To understand the FUSE overhead in HyCache+ the I/O performance between the RAM and simples FUSE mounted on the RAM is compared. The study shows FUSE+SSD file system could achieve 580 MB/sec aggregate bandwidth. The results suggest that FUSE could not compete with the kernel level file system in raw bandwidth primarily due to the overhead associated with file system in the user space, extra copies as well as additional context switching.

B. I/O Throughput

By implementing the middleware HighCache+ the throughput of the parallel file system was improved. Every single nodes its local cache size was set to 0.25 GB, 256 MB file is written on every client for the interchange of data between local cache and Parallel file systems. Figure 3 illustrates the aggregate throughput showing a performance drop at 90 second timestamp, when 15.75 GB data is finished on local cache.

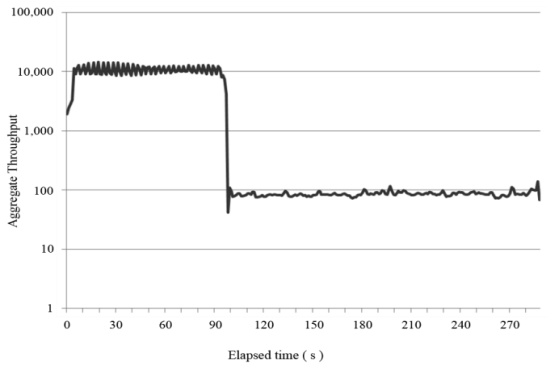

Figure 3. Throughput on BlueGene/P(256 cores)

C. Scalability

The simulation could be done using 1024 node(4096 cores).To achieve scalable performance the same can be repeated with 512 node(2048 cores) ,which may show a decrease in the number of cores used. The predicted throughput is cited in figure 4.

The results shows that HyCache+ gives better scalability for cached as well as remote data.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459 (Online), Volume 5, Special Issue 2, May 2015)
International Conference on Advances in Computer and Communication Engineering (ACCE-2015)

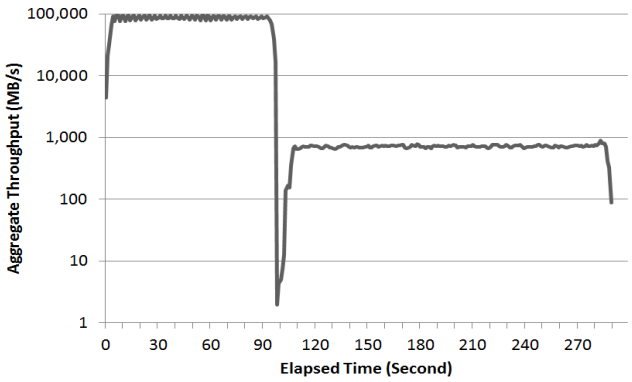

Figure 4. Throughput on BlueGene/P(512 nodes)

D. Fault tolerance

For tolerance was measured in terms of synchronous replication and asynchronous replication. The studies show that asynchronous replication gives an efficiency of 90% when compared to no replication method. While considering the consistency factor synchronous replication incurs high overhead, which will produces 68% throughput out of no replication throughput.

V. CONCLUSION

This paper presents a caching middleware called HyCache+ for improving the performance of the parallel file system. The file system was attached to network attached storage so that multiple clients can acess.The proposed system implements a two layer scheduling for reducing the network cost. The HyCache+ can include up to 4096 cores. The simulation for the work could be done in the IBM BlueGene/P supercomputer and the performance throughput as well as scalability was evaluated.

REFERENCES

[1] Y. Gu and R. L. Grossman. Udt: Udp-based data transfer for high-speed wide area networks. Comput. Netw., 51(7):1777–1799, May 2007.

[2] Google protocol buffers. http://code.google.com/p/protobuf/.

[3] FUSE Project. http://fuse.sourceforge.net.

[4] R. Fares, B. Romoser, Z. Zong, M. Nijim, and X. Qin. Performance evaluation of traditional caching policies on a large system with petabytes of data. In Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference

[5] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu. Zht: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table. In Proceedings of the 2013 IEEE 27th International Symposium on Parallel and Distributed Processing

[6] C.Ambühl and B. Weber. Parallel prefetching and caching is hard. In STACS, pages 211– 221, 2004

[7] R. Bellman. Dynamic programming treatment of the travelling salesman problem. J. ACM, 9(1):61–63, Jan.1962

[8] P. A. Freeman, D. L. Crawford, S. Kim, and J. L. Munoz. Cyberinfrastructure for science and engineering: Promises and challenges. Proceedings of the IEEE,

[9] S. Huang, Q. Wei, J. Chen, C. Chen, and D. Feng. Improving flash-based disk cache with lazy adaptive replacement. In Mass Storage Systems and Technologies (MSST), 2013 IEEE 29th Symposium on, pages 1–10, 2013.

[10] G. Calinescu and K. Qiao. Asymmetric topology control: Exact solutions and fast approximations. In IEEE International Conference on Computer Communications

[11] Titan. http://phys.org/news/2013-04-supercomputer-titan-world-fastest-storage.html.

[12] P. A. Freeman, D. L. Crawford, S. Kim, and J. L. Munoz. Cyberinfrastructure for science and engineering: Promises and challenges. Proceedings of the IEEE, 93(3):682–691, 2005.

[13] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, and M. Wilde. Falkon: a fast and light-weight task execution framework. In Proceedings of the 2007 ACM/IEEE conference on Supercomputing, SC '07, pages 43:1–43:12, 2007.

[14] P. Gonzalez-Ferez, J. Piernas, and T. Cortes. The ram enhanced disk cache project (redcap). In Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies, MSST'07, pages 251–256, 2007.

[15] S.Albers, N. Garg, and S. Leonardi. Minimizing stall time in single and parallel disk systems. In Proceedings of the thirtieth annual ACM symposium on Theory of computing, STOC '98, pages 454–462, 1998

[16] S.Albers, N. Garg, and S. Leonardi. Minimizing stall time in single and parallel disk systems. In Proceedings of the thirtieth annual ACM symposium on Theory of computing, STOC '98, pages 454–462, 1998

[17] R. Li, R. Guo, Z. Xu, and W. Feng. A prefetching model based on access popularity for geospatial data in a cluster-based caching system. Int. J. Geogr. Inf. Sci., 26(10):1831–1844, Oct.2012.

[18] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai. Efficient batched synchronization in dropbox-like cloud storage services. In Proceedings of the 14th International Middleware Conference, Middleware '13, Beijing, China, 2013.

[19] J. Lin, Q. Lu, X. Ding, Z. Zhang, X. Zhang, and P. Sadayappan. Enabling software management for multicore caches with a lightweight hardware support. In Proceedings of the 2009 ACM/IEEE conference on Supercomputing, SC '09, pages 14:1–14:12, 2009.

[20] R. Lohfert, J. Lu, and D. Zhao. Solving sql constraints by incremental translation to sat. In 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2008.

[21] A. Manzanares, X. Ruan, S. Yin, J. Xie, Z. Ding, Y. Tian, J. Majors, and X. Qin. Energy sefficient prefetching with buffer disks for cluster file systems. In Proceedings of the 2010 39th International Conference on Parallel Processing, ICPP '10, pages 404–413, 2010

[22] D. Meister, J. Kaiser, and A. Brinkmann. Block locality caching for data deduplication. In Proceedings of the 6th International Systems and Storage Conference, SYSTOR '13, pages 15:1–15:12, 2013.

[23] J. S. Plank, M. Blaum, and J. L. Hafner. Sd codes: Erasure codes designed for how storage systems really fail. In Proceedings of the 11th USENIX conference on File and Storage Technologies,FAST '13, 2013

[24] K. Qiao, F. Tao, L. Zhang, and Z. Li. A ga maintained by binary heap and transitive reduction for addressing psp. In Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on, pages 12–15, Oct 2010.

[25] I. Raicu. Many-task computing: bridging the gap between highthroughput computing and high-performance computing. PhD thesis, Chicago, IL, USA, 2009.

[26] I. Raicu, I. Foster, A. Szalay, and G. Turcu. Astroportal: A science gateway for large-scale astronomy data analysis. In TeraGrid Conference, 2006.