Distributed NoSQL Storage for Extreme-Scale System Services

Tonglin Li[1], Ioan Raicu[1,2]
[1]*Computer Science Department, Illinois Institute of Technology, Chicago, IL, USA*
[2]*MCS Division, Argonne National Laboratory, Lemont, IL, USA*

***Abstract*—Today with the rapidly accumulated data, data-driven applications are emerging in science and commercial areas. On both HPC systems and clouds the continuously widening performance gap between storage and computing resource prevents us from building scalable data-intensive systems. Distributed NoSQL storage systems are known for their ease of use and attractive performance and are increasingly used as building blocks of large scale applications on cloud or data centers. However there are not many works on bridging the performance gap on supercomputers with NoSQL data stores.**

This work presents a convergence of distributed NoSQL storage systems in clouds and supercomputers. It firstly presents ZHT, a dynamic scalable zero-hop distributed key-value store, that aims to be a building block of large scale systems on clouds and supercomputers. This work also presents several real systems that have adopted ZHT as well as other NoSQL systems, namely ZHT/Q (a Flexible QoS Fortified Distributed Key-Value Storage System for the Cloud), FREIDA-State (state management for scientific applications on cloud), WaggleDB (a Cloud-based interactive data infrastructure for sensor network applications), and Graph/Z (a key-value store based scalable graph processing system); all of these systems have been significantly simplified due to NoSQL storage systems, and have been shown scalable performance.

I. Introduction

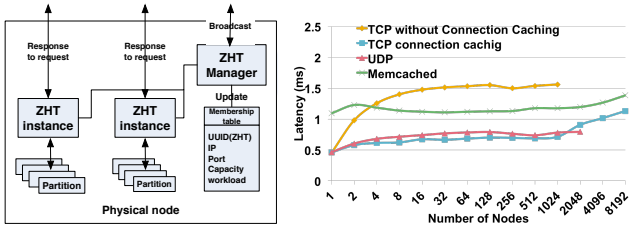
Today's science is generating datasets that are increasing exponentially in both complexity and volume, making their analysis, archival, and sharing one of the grand challenges of the 21st century. As supercomputers and data centers gain more parallelism at exponential rates, the storage infrastructure performance is increasing at a significantly lower rate. This implies that the data management and data flow between the storage and compute resources is becoming the new bottleneck for large-scale applications. The support for data intensive computing is critical to advancing modern science as storage systems have experienced a gap between capacity and bandwidth that increased more than 10-fold over the last decade. There is an emerging need for advanced techniques to manipulate, visualize and interpret large datasets. Many domains(e.g.astronomy, bioinformatics, and financial analysis) share these data management challenges, strengthening the potential impact from generic solutions.

Distributed NoSQL storage systems are known for their ease of use and attractive performance and are increasingly used as building blocks of large scale applications on cloud or data centers. However there are not many works on bridging the performance gap on supercomputers with NoSQL data stores. This work presents a convergence of distributed NoSQL storage systems in clouds and supercomputers. It firstly presents ZHT, a light-weight reliable persistent dynamic scalable zero-hop distributed key-value store, that aims to be a building block of large scale systems on both clouds and supercomputers. This work also presents several distributed systems that have adopted ZHT as well as other NoSQL systems, namely ZHT/Q, (a Flexible QoS Fortified Distributed Key-Value Storage System for the Cloud), FREIDA-State (state management for scientific applications on cloud), WaggleDB (a Cloud-based interactive data infrastructure for sensor network applications), and Graph/Z (a key-value store based scalable graph processing system); all of these systems have been significantly simplified due to NoSQL storage systems, and have been shown to outperform other leading systems by orders of magnitude in some cases. It is important to highlight that some of these systems are rooted in HPC systems from supercomputers, while others are rooted in clouds and ad-hoc distributed systems; through our work, we have shown how versatile NoSQL storage systems can be in such a variety of environments.

II. ZHT: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table

One of the major bottlenecks in current state-of-the-art storage systems is metadata management [1]. Metadata operations on most of parallel and distributed file systems can be inefficient at large scales. Our previous work [3] on a Blue Gene/P supercomputer with 16K-cores shows the various costs for file/directory creating (metadata operation of file systems) on GPFS. GPFS's metadata performance degrades rapidly under concurrent operations, reaching saturation at only 4 to 32 core scales (on a 160K-core machine). Ideal performance would have been constant at different scales, but we see the cost of these basic metadata operations (e.g. create file) growing exponentially, from tens of milliseconds on a single node (four-cores), to tens of seconds at 16K-core scales; at full machine scale of 160K-cores, we expect one file creation to take over two minutes for the many directory case, and over 10 minutes for the single directory case. Previous work shows these times to be even worse, putting the full system scale metadata operations in hours range, although GPFS might have been improved over the last several years. On a large scale HPC system, whether the time per metadata operation is minutes or hours, the conclusion is that the metadata management in GPFS does not have enough degree of distribution, and not enough emphasis was placed on avoiding lock contention.

HPC storage is not the only area that suffers the storage bottleneck. Similar with the HPC scenarios, distributed file system such as HDFS [4, 5] and cloud based distributed systems also have to face storage bottleneck. Furthermore, due to the dynamic nature of cloud applications, a suitable storage system needs to satisfy more requirements, such as being able to handle dynamic nodes join and leave on the



(a) ZHT server single node architecture. Each node run multiple ZHT instances with multiple partitions.

(b) Performance evaluation of ZHT and Memcached plotting latency vs. scale (1 to 8K nodes on the Blue Gene/P)

Fig. 1: ZHT server architecture and performance

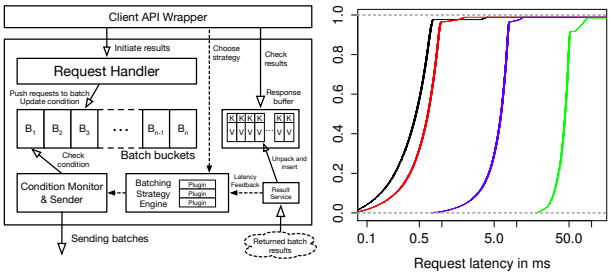
fly and the flexibility to run on different cloud instance types simultaneously.

As an initial attempt to meet these needs, we propose and build ZHT (zero-hop distributed hash table [6, 7], an instance of NoSQL database. ZHT has been tuned for the specific requirements of high-end computing (e.g. trustworthy/reliable hardware, fast networks, non-existent "churn", low latencies). ZHT aims to be a building block for future distributed systems, with the goal of delivering excellent availability, fault tolerance, high throughput, scalability, persistence, and low latencies. ZHT has several important features making it a better candidate than other distributed hash tables and key-value stores. Highlighted features include being light-weight, dynamically allowing nodes join and leave, fault tolerant through replication, handling failures gracefully, efficiently propagating events throughout the system, a customized consistent hashing mechanism. Unlike conventional key-value store, ZHT implemented new operations such as `append`, `compare_swap` and `state_change_callback` in addition to `insert`/`lookup`/`remove`. To provide ZHT a persistent back end, we also created a fast persistent single node data store that could be easily integrated and operated in lightweight Linux OS typically found on today's supercomputers as well as clouds. We have evaluated ZHT's performance under a variety of systems, ranging from a Linux cluster with 512-cores, to an IBM Blue Gene/P supercomputer with 160K-cores. Using micro-benchmarks, we scaled ZHT up to 32K-cores with latencies of only 1.1ms and 18M operations/sec throughput. We compared ZHT against two other systems, Cassandra and Memcached and found it to offer superior performance for the features and portability it supports, at large scales up to 16K-nodes. We also compared it to DynamoDB in the Amazon AWS Cloud, and found that ZHT offers significantly better performance and economic cost than DynamoDB. ZHT have been adopted in six real systems, namely FusionFS, IStore, MATRIX, Slurm++, Fabriq and Graph/Z. They have been implemented and evaluated at modest scales. 1) ZHT is used in the FusionFS distributed file system to deliver distributed meta-data management and data provenance capture/query. On a 512-nodes deployment at Los Alamos National Lab, FusionFS reached 509kops/s metadata performance. 2) ZHT is used in the IStore, an erasure coding enabled distributed object storage system, to manage chunk locations delivering more than 500 chunks/sec at 32-nodes scales. 3) ZHT is also used as a building block to MATRIX, a distributed task scheduling system, delivering 13k jobs/sec throughput at 4K-core scales. 4) Slurm++, a distributed job launch system that avoids the centralized gateway nodes in Slurm. 5) ZHT is used in Fabriq, a distributed message queue, to store messages reliably, and load balance the resource requirements. 6) As a back end and building block, ZHT is used to build Graph/Z, a key-value store based scalable graph processing system, and enable it to efficiently handle large data sets that cannot fit in memory.

III. A Flexible QoS Fortified Distributed Key-Value Storage System for the Cloud

In the era of Big Data and Cloud, distributed key-value stores are increasingly used as building blocks of large scale applications. Comparing to traditional relational databases, key-value stores are particularly compelling due to their low latency and excellent scalability. Many big companies, such as Facebook and Amazon, run multiple different applications and services on top of a single key-value store deployment to reduce the deployment and maintenance complexity as well as economic cost. However every application has its own performance requirement but most of current key-value store systems are designed to serve every application request equally. This design works well when single application accesses the key-value store, but it is not as good for the emerging concurrent multi-application scenario. In this work we present ZHT/Q, a flexible QoS Fortified distributed key-value storage system for Clouds. It enhances a high performance key/value store with flexible QoS (Quality of Service) properties such that both configurable latency and high aggregated throughput can be achieved. It satisfies different applications' latency requirements with QoS while improves the overall performance through dynamic and adaptive request batching mechanisms. The system QoS provides guaranteed and best-effort service on latency for different scenarios. It also watches the performance change and dynamically adjusts the batching strategy to alleviate performance degradation upon traffic.

We design the new system based on ZHT and propose to add a proxy layer (fig.2a) for dynamic batching mechanism on the client side instead of server side. The client proxy works on each client, collects and batches the requests that share a same destination server and sends to the server. The destination server unpacks the batch with a parser, executes the requests sequentially, packs the return status (including lookup results) in a batch and send back. This keeps the communication and storage layers of key-value store unchanged. The experiment results show that our new system delivers up to 28 times higher throughput than the base solution while more than 99% of requests' latency requirements are satisfied (fig.2b).



(a) Client side batcher architecture.

(b) Batch latency CDF (4 workloads)

Fig. 2: ZHT server architecture and performance

IV. FRIEDA-State: Scalable State Management for Scientific Applications on Cloud

Cloud is as an emerging platform and increasingly attractive to scientists due to its flexibility and convenience. But

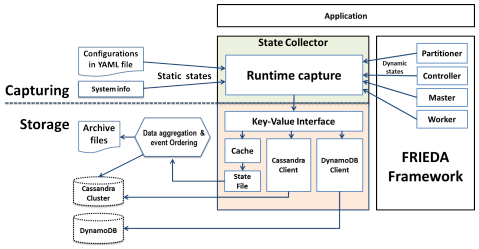


Fig. 3: FRIEDA-State system architecture. State collector captures static states from static and dynamic sources. Captured states are encapsulated into the form of key-value pairs and pushed to one of three storage solutions as selected by the user.

cloud environments are typically transient. Virtual machine instances are terminated after applications complete execution. Users cannot leave data and/or revisit the resource setup to diagnose discrepancies. In the cloud environment, users have the responsibility to capture everything before the virtual machines are shutdown. Big data scientific run-time systems [8] need to track every step of the scientific process, data access and environment for lineage, reconstruction, validity and reproducibility purposes. It is important to know the environment in which the applications run (e.g., floating point operations could give different results on different machines). Provenance tools have tracked workflow and data lineage at various levels, monitoring tools have been developed to monitor real-time system changes. These systems provide methods to collect, aggregate, and query monitoring data. However, this data is often insufficient for reproduction since they do not capture human knowledge. Furthermore, state management in cloud environments needs to tackle additional challenges due to its characteristics. First, the transient nature of the environments makes it important to capture metadata and state at various levels. Second, the performance and reliability characteristics of virtual machines is important to consider in the design of the collection system. Finally, different clock drifting rates on physical machines make it hard to have a unified time view for the end-user to rebuild meaningful semantics.

In this work, we present FRIEDA-State, a state management system for cloud environments 3. We use the term state to represent the metadata from both execution framework and applications. FRIEDA-State addresses the transient nature, performance concerns and clock drifting issue in its design. FRIEDA-State is currently implemented atop of FRIEDA, a data management and execution framework for cloud environments, which supports a high-throughput and data-intensive scientific applications,. We present a key-value based collection system to manage state in dynamic transient environments. We design and implement a vector clock based event-ordering mechanism to address the clock drifting issue. FRIEDA-State collects static and dynamic state data. Static state data is the information that doesnt change when the system is running (e.g., CPU/Memory info, environment variables and software stack information). Dynamic state data, on the other hand, changes during application running, such as the information on details of the input file that is processed, the time taken for a machine to finish execution or failure of jobs.

V. WaggleDB: A Dynamically Scalable Cloud Data Infrastructure for Sensor Networks

The last several years have seen a raise in the use of sensors, actuators and their networks for sensing, monitoring and interacting with the environment. There is a proliferation of small, cheap and robust sensors for measuring various physical, chemical and biological characteristics of the environment that open up novel and reliable methods for monitoring qualities ranging from the geophysical variables, soil conditions, air and water quality monitoring to growth and decay of vegetation. Structured deployments, such as the global network of flux towers, are being augmented by innovative use of personal mobile devices (e.g., such as use of cell phones to detect earthquakes), use of data from social networks, and even citizen science. In other words, rather than construct a single instrument comprised of millions of sensors, a "virtual instrument" might comprise dynamic, potentially ad hoc groups of sensors capable of operating independently but also capable of being combined to answer targeted questions. Projects organized around this approach represent important areas ranging from ocean sciences, ecology, urban construction and research, to hydrology. This calls for an infrastructure able to collect, store, query, and process data set from sensor networks. The design and development of such infrastructure faces several challenges. The first group of challenges reflects the need to interact with and administer the sensors remotely. The sensors may be deployed in inaccessible places and have only intermittent network connectivity due to power conservation and other factors. This requires communication protocols that can withstand unreliable networks as well as an administrative interface to sensor controller. Further, the system has to be scalable, i.e., capable of ultimately dealing with potentially large numbers of data producing sensors. It also needs to be able to organize many different data types efficiently. And finally, it also needs to scale in the number of queries and processing requests. In this work we present a set of protocols and a cloud-based data store called WaggleDB that address those challenges (fig.4). The system efficiently aggregates and stores data from sensor networks and enables the users to query those data sets. It address the challenges above with a scalable multi-tier architecture, which is designed in such way that each tier can be scaled by adding more independent resources provisioned on-demand in the cloud.

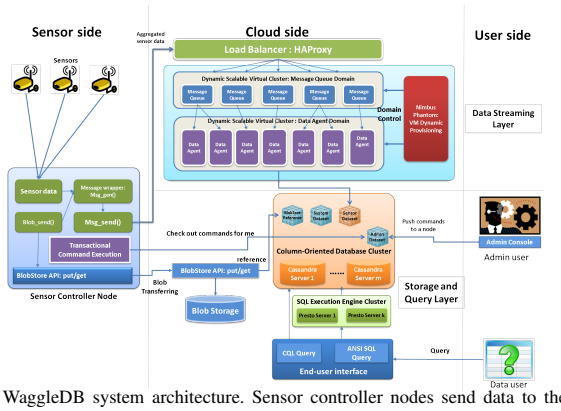


Fig. 4: WaggleDB system architecture. Sensor controller nodes send data to the cloud storage. Load balancer forwards client requests to data streaming layer. Nimbus Phantom controls dynamic scaling of queue servers and data agent servers.

VI. Graph/z: A Key-Value Store Based Scalable Graph Processing System

With the advancement of social networks, online gaming and scientific applications such as geospatial systems and bioinformatics, graph data has been used ubiquitously. There have been works on work flow systems and data streaming management systems attempted to handle structured big data

sets from scientific and commercial applications, which are typically stored in file systems(such as Hadoop HDFS and FusionFS), SQL databases(such as Oracle and DB2) or Column Family databases (such as Hadoop Hive and Cassandra). Data mining, machine learning and security management techniques are also widely used to extract the value from these big data sets. However it is not easy to fully reveal and utilize the scientific and commercial value from the continuously increasing graph data sets. It's even more challenging when moving these works to clouds. The traditional relational database has been used and dominated for many years, and it also works well for a long time. Graph related query is tremendously slow on the traditional relational database. An ideal solution for this problem is to replace the traditional data infrastructure with a graph-centric model, including storage and computing, thus to better serve graph-based applications in terms of performance and programmability.

Pregel [9] is a Bulk Synchronous Parallel model based distributed graph processing system developed by Google. It inspires couple of similar variation projects, such as Giraph and GraphLab, now known as Pregel-like systems. However Pregel-like systems have some limitations. First, they only work on in-memory data and don't accept new data as soon as data loading is finished. This limits their use especially when the dataset can't fit in memory. Second, the master node coordinates both synchronization barriers and checkpointing for fault tolerance, which makes it a significant bottleneck. We design and implement a new graph processing system Graph/Z with ZHT as a building block. Graph/Z can be considered as another Pregel-liked graph processing system, but it inherits some important features from ZHT, a distributed key-value store, which differentiate Graph/Z from other systems. ZHT is a zero-hop distributed key-value store featured with high scalability, persistency and fault tolerance. By leveraging ZHT's persistency, Graph/Z can run with a much larger working dataset.

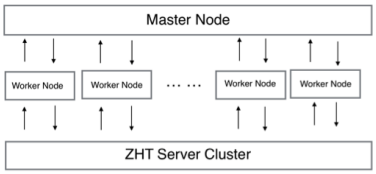


Fig. 5: Graph/Z architecture

The increment of loading time is almost linear because each worker node only need to load its local vertexes and don't need to communicate with a remote node. Due to the load balance feature of ZHT, the amount of work that each worker needs to do are basically equal. PageRank algorithm uses all the vertexes and edges in every superstep. Thus this is a good algorithm to test data locality and load balance. When running on 8 nodes, the system achieves the highest performance, and then it decreases greatly on 16 nodes. This is mainly because the average work load on each node is too small and relatively more cross-node communication is involved due to large scale.

VII. Conclusion

On different platforms, ranging from conventional cluster, super computers to multiple clouds, we have shown that NoSQL data storage systems exhibit great potential to be an excellent building block of large scale distributed systems, such as job schedulers, data streaming systems and file systems.

We believe that NoSQL data storage could transform the architecture of future higher level storage systems in HPC and clouds, and open the door to a broader class of applications that would have not normally been tractable. Furthermore, the concepts, data-structures, algorithms, and implementations that underpin these ideas in resource management at the largest scales, can be applied to emerging paradigms, such as High-Performance Computing, Cloud Computing and Many-Task Computing.

Based on these projects, I have 11 peer-reviewed publications [3, 6, 7, 10–17] on journals and conferences. These contributions have also lead to 13 additional [18–30] publications that are not directly authored by me and have been used as a building block towards more complex distributed systems.

References

[1] J. Liu and Y. Chen, "Fast data analysis with integrated statistical metadata in scientific datasets," in *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pp. 1–8, Sept 2013.
[2] J. Liu, Y. Zhuang, and Y. Chen, "Hierarchical collective i/o scheduling for high-performance computing," *Big Data Research*, vol. 2, no. 3, pp. 117 – 126, 2015. Big Data, Analytics, and High-Performance Computing.
[3] D. Zhao, Z. Zhang, X. Zhou, T. Li, K. Wang, D. Kimpe, P. Carns, R. Ross, and I. Raicu, "Fusionfs: Towards supporting data-intensive scientific applications on extreme-scale high-performance computing systems," in *Big Data, 2014 IEEE International Conference on*.
[4] X. Yang, Y. Yin, H. Jin, and X.-H. Sun, "Scaler: Scalable parallel file write in hdfs," in *Proc. of IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 203–211, IEEE, 2014.
[5] N. Liu, X. Yang, X.-H. Sun, J. Jenkins, and R. Ross, "Yarnsim: Simulating hadoop yarn," 2015.
[6] 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," IPDPS '13.
[7] T. Li, R. Verma, X. Duan, H. Jin, and I. Raicu, "Exploring distributed hash tables in highend computing," *SIGMETRICS Performance Evaluation Review*, 2011.
[8] K. Feng, Y. Yin, C. Chen, H. Eslami, X.-H. Sun, Y. Chen, R. Thakur, and W. Gropp, "Runtime system design of decoupled execution paradigm for data-intensive high-end computing," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1–1, IEEE, sep 2013.
[9] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, "Pregel: A system for large-scale graph processing," SIGMOD '10, 2010.
[10] T. Li, I. Raicu, and L. Ramakrishnan, "Scalable state management for scientific applications in the cloud," BigData Congress '14.
[11] T. Li, X. Zhou, K. Wang, D. Zhao, I. Sadooghi, Z. Zhang, and I. Raicu, "A convergence of key-value storage systems from clouds to supercomputers," *Concurr. Comput. : Pract. Exper.(CCPE)*, 2015.
[12] T. Li, K. Keahey, R. Sankaran, P. Beckman, and I. Raicu, "A cloud-based interactive data infrastructure for sensor networks," IEEE/ACM Supercomputing/SC'14.
[13] T. Li, K. Keahey, K. Wang, D. Zhao, and I. Raicu, "A dynamically scalable cloud data infrastructure for sensor networks," ACM ScienceCloud 15.
[14] T. Li, C. Ma, J. Li, X. Zhou, K. Wang, D. Zhao, I. Sadooghi, and I. Raicu, "GRAPH/Z: A key-value store based scalable graph processing system," Cluster'15.
[15] K. Wang, X. Zhou, T. Li, M. Lang, and I. Raicu, "Optimizing load balancing and data-locality with data-aware scheduling," IEEE BigData'14.
[16] I. Sadooghi, J. Hernandez Martin, T. Li, K. Brandstatter, Y. Zhao, K. Maheshwari, T. Pais Pitta de Lacerda Ruivo, S. Timm, G. Garzoglio, and I. Raicu, "Understanding the performance and potential of cloud computing for scientific applications," 2015.
[17] K. Wang, N. Liu, I. Sadooghi, X. Yang, X. Zhou, T. Li, M. Lang, X.-H. Sun, and I. Raicu, "Overcoming Hadoop scaling limitations through distributed task execution," IEEE Cluster'15, 2015.
[18] I. Sadooghi, S. Palur, A. Anthony, I. Kapur, K. Belagodu, P. Purandare, K. Ramamurty, K. Wang, and I. Raicu, "Achieving efficient distributed scheduling with message queues in the cloud for many-task computing and high-performance computing," CCGrid'14, 2014.
[19] K. Wang, A. Kulkarni, M. Lang, D. Arnold, and I. Raicu, "Using simulation to explore distributed key-value stores for extreme-scale system services," SC '13, 2013.
[20] K. Wang, X. Zhou, H. Chen, M. Lang, and I. Raicu, "Next generation job management systems for extreme-scale ensemble computing," HPDC '14, 2014.
[21] D. Zhao, C. Shou, T. Malik, and I. Raicu, "Distributed data provenance for large-scale data-intensive computing," CLUSTER, 2013.
[22] C. Shou, D. Zhao, T. Malik, and I. Raicu, "Towards a provenance-aware a distributed file system," TaPP13, 2013.
[23] K. Wang, A. Rajendran, K. Brandstatter, Z. Zhang, and I. Raicu, "Paving the road to exascale with many-task computing," SC'12 Poster, 2012.
[24] I. Raicu, I. T. Foster, and P. Beckman, "Making a case for distributed file systems at exascale," LSAP '11, 2011.
[25] D. Zhao, D. Zhang, K. Wang, and I. Raicu, "Exploring reliability of exascale systems through simulations," HPC '13, pp. 1:1–1:9, 2013.
[26] K. Wang, K. Brandstatter, and I. Raicu, "Simmatrix: Simulator for manytask computing execution fabric at exascales," HPC, 2013.
[27] D. Patel, F. Khasib, I. Sadooghi, and I. Raicu, "Towards In-Order and Exactly-Once Delivery using Hierarchical Distributed Message Queues," SCRAMBL'14, 2014.
[28] D. Zhao, K. Qiao, and I. Raicu, "Hycache+: Towards scalable high-performance caching middleware for parallel file systems," 2014.
[29] D. Zhao, J. Yin, K. Qiao, and I. Raicu, "Virtual chunks: On supporting random accesses to scientific data in compressible storage systems," IEEE BigData'14, 2014.
[30] A. Rajendran and I. Raicu, "Matrix: Many-task computing execution fabric for extreme scales," 2013.