ZHT: A Zero-hop Distributed Hash Table for High-end Computing Systems

Tonglin Li[1], Antonio Perez de Tejada[1], Kevin Brandstatter[1], Zhao Zhang[2] , Ioan Raicu[1,3]

[1]Department of Computer Science, Illinois Institute of Technology

[2]Department of Computer Science, University of Chicago

[3]Mathematics and Computer Science Division, Argonne National Laboratory

Abstract

One critical component of future file systems for high-end computing is meta-data management. This work presents ZHT, a zero-hop distributed hash table, which has been tuned for the requirements of HEC systems. ZHT aims to be a building block for future distributed file systems to implement distributed metadata management. The goals are delivering availability, fault tolerance, high throughput, and low latencies. ZHT has some important properties, such as being light-weight, fault tolerant using replication and persistence. We have evaluated ZHT's performance under a variety of systems, ranging from a Linux cluster to an IBM BlueGene/P supercomputer. We scaled ZHT up to 16K processes and achieved 4M operations/sec throughput. Latencies have scaled similarly well, with sub-milliseconds latencies at 4K-core scales. We compared ZHT against other systems and found it offers superior performance for the features and portability it supports.

Experiment setup

Hardware

- IBM Blue Gene/P supercomputer
- 1024 nodes
- 2GB RAM/node
- 4096 cores in total

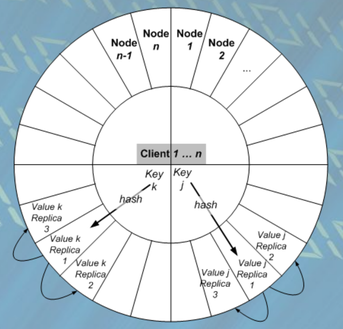
Software

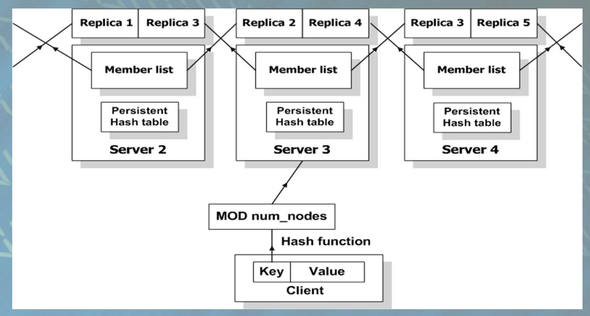
- OS: ZeptOS
- Batch execution system: Cobalt
- Persistent hash table: NoVoHT
- Data serialization: Google protocol buffers

Experiment

Each client creates a long list of key-value pairs, here we set the length of key is 15 byte and length of value is 132 bytes. Clients firstly sequentially send all these key-value pairs ZHT API, ZHT will decide which server to send. Secondly clients send the same list of keys as lookup parameter to servers; finally send remove request with the same list of keys.

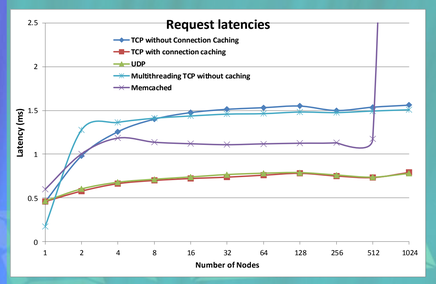
Architecture and design

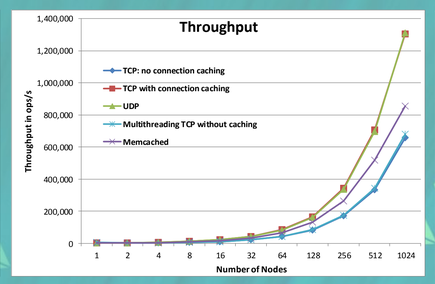


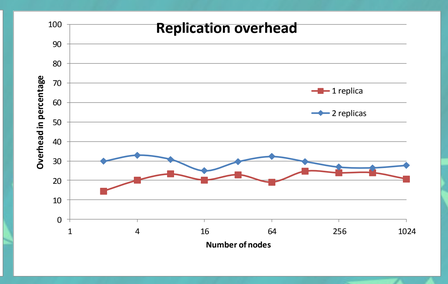


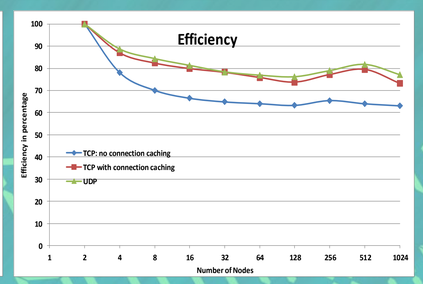
- **Assumptions:** reliable hardware, fast networks, non-existent "churn", low latencies, and scientific computing data-access patterns.
- **Solution:** a light-weighted and high performance DHT for metadata management.
- **Design goal:** excellent availability, fault tolerance, high throughput, and low latencies.

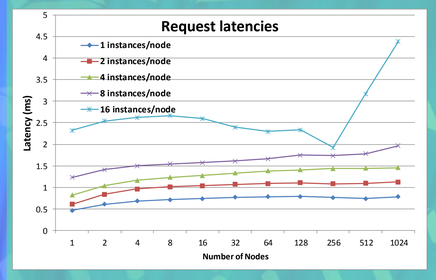
Performance evaluation

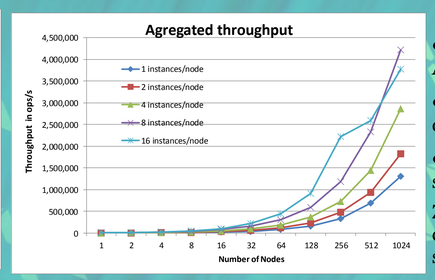












- More than 4M operations/sec aggregated throughput with 16K ZHT instances on 1024 nodes. As low as 0.78ms latency on 1024 nodes scale.
- UDP shows a better scalability and reliability. TCP can be as fast as UDP when using connection caching.
- The performance differences among three basic operations (insert, lookup and remove) are very small.

ZHT uses a direct 0-hop algorithm and that the majority of the overhead comes from network communication, it is not expected that the time per operation to increase significantly with larger scales.

Conclusion

ZHT optimized for high-end computing systems is architected and implemented as a foundation in the development of fault-tolerant, high-performance, and scalable storage systems.

We performed an extensive performance evaluation of ZHT on a modest scale up to 1K nodes and 16K instances on an IBM BlueGene/P. We achieved more than 4M operations/sec throughput. The latency is as low as 0.78ms at 1K node scale. We hope to extend the performance evaluation to significantly larger scales, as the machine we tested on has 40K nodes.

We believe that ZHT could transform the architecture of future storage systems in HEC, and open the door to a much broader class of applications that would have normally not 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 new emerging paradigms, such as Cloud Computing.

Related work and acknowledgements

G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels. “Dynamo: Amazon’s Highly Available Key-Value Store.” SIGOPS Operating Systems Review, 2007

B. Fitzpatrick. “Distributed caching with Memcached.” Linux Journal, 2004(124):5, 2004

Z. Zhang, A. Espinosa, K. Iskra, I. Raicu, I. Foster, M. Wilde. “Design and Evaluation of a Collective I/O Model for Loosely-coupled Petascale Programming”, IEEE MTAGS 2008

I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, B. Clifford. “Toward Loosely Coupled Programming on Petascale Systems,” IEEE SC 2008

This work is supported in part by the U.S. Dept. of Energy under Contract DE-AC02-06CH11357, as well as the National Science Foundation grant NSF-0937060 CIF-72 and NSF-1054974.

Future work

Broadcast primitive: transmit the key/value pair over the edges of the spanning tree with the goal to distribute the key/value pair to all the caches.

Data Indexing: When dealing with massive data collections, one challenge is indexing the material to support re-use and analysis.

Distributed Metadata Management: FusionFS will use ZHT to implement the distributed metadata management.