In this study, we present LWDLS, a lightweight data location service designed for Exascale storage systems (storage systems with order of 10^18 bytes) and global distributed storage systems (Large storage systems with physically distributed storage locations). LWDLS provides a search-based data location solution, and enables free data placement, movement, and replication. In LWDLS, probe and prune protocols are introduced that reduce topology search algorithm (TSA) is presented that achieves higher search efficiency than pure flooding search while delivering comparable search speed and coverage to pure flooding search.

LWDLS is lightweight and scalable in terms of incorporating low overhead, high search efficiency, no global state, and avoiding periodic messages. LWDLS is fully distributed and can be used in nondeterministic storage systems and in deterministic storage systems to deal with cases where search is needed.

Extensive simulations modeling large-scale HPC (High-Performance Computing) storage environments provide representative performance outcomes. Performance results validate the benefit of adding LWDLS into existing search algorithms and neighbor. Results show that LWDLS is able to locate data efficiently with low cost of state maintenance in arbitrary network environments. Through these simulations, we demonstrate the effectiveness of protocols and search algorithm of LWDLS.

Probe and Prune Protocols

Figure 6. Two searching strategies: (a): Prune. (b): Probe.

Figure 7. Probing operation based on the ratio of the length of the probing signal to a triangle. (a): Server A initiates a flooding search, and two relevant messages are generated on the graph. (b): Server A initiates a flooding search, and two relevant messages are generated on the graph. (c): Server A initiates a flooding search, and two relevant messages are generated on the graph. (d): Server A initiates a flooding search, and two relevant messages are generated on the graph. (e): Server A initiates a flooding search, and two relevant messages are generated on the graph. (f): Server A initiates a flooding search, and two relevant messages are generated on the graph. (g): Server A initiates a flooding search, and two relevant messages are generated on the graph. (h): Server A initiates a flooding search, and two relevant messages are generated on the graph. (i): Server A initiates a flooding search, and two relevant messages are generated on the graph. (j): Server A initiates a flooding search, and two relevant messages are generated on the graph. (k): Server A initiates a flooding search, and two relevant messages are generated on the graph. (l): Server A initiates a flooding search, and two relevant messages are generated on the graph. (m): Server A initiates a flooding search, and two relevant messages are generated on the graph. (n): Server A initiates a flooding search, and two relevant messages are generated on the graph. (o): Server A initiates a flooding search, and two relevant messages are generated on the graph. (p): Server A initiates a flooding search, and two relevant messages are generated on the graph. (q): Server A initiates a flooding search, and two relevant messages are generated on the graph. (r): Server A initiates a flooding search, and two relevant messages are generated on the graph. (s): Server A initiates a flooding search, and two relevant messages are generated on the graph. (t): Server A initiates a flooding search, and two relevant messages are generated on the graph. (u): Server A initiates a flooding search, and two relevant messages are generated on the graph. (v): Server A initiates a flooding search, and two relevant messages are generated on the graph. (w): Server A initiates a flooding search, and two relevant messages are generated on the graph. (x): Server A initiates a flooding search, and two relevant messages are generated on the graph. (y): Server A initiates a flooding search, and two relevant messages are generated on the graph. (z): Server A initiates a flooding search, and two relevant messages are generated on the graph.

Figure 8. Diagram of states transition through searching algorithms.

Figure 9. Heuristic algorithm simulation.

Figure 10. Comparison of search performance with LWDLS.

Conclusions

- LWDLS is a search-based data location service, enables free data placement, movement, and replication.
- LWDLS is able to locate data efficiently in nondeterministic Exascale storage systems.
- LWDLS is lightweight, efficient, and scalable in terms of avoiding global state, periodic messages, or the limitations on locations of data.
- With the prune and prune protocol and search algorithm, LWDLS is able to address problems of topology mismatch and inefficient search performance of pure flooding search algorithms.
- LWDLS provides higher search efficiency than pure flooding search while delivering comparable search speed and search coverage.
- The effectiveness of LWDLS has been tested through extensive simulations modeling large-scale HPC storage environments.

Acknowledgements

This work was supported in part by the United States Department of Energy under Contract DE-AC19-19LS01370. This work was supported in part by the National Science Foundation under grants 1208195 and 1816918 and providing timely support.