D^{2}T: Doubly Distributed Transactions for High Performance and Distributed Computing

Jai Dayal[1], Jay Lofstead[2], Karsten Schwan[1], Ron Oldfield[2]
[1]CERCS, Georgia Institute of Technology, [2]CSRI, Sandia National Labs.

ABSTRACT

The Doubly Distributed Transaction (D2T) protocol offers a mechanism for a collection of clients and a separate collection of servers to orchestrate an action with semantics inspired by database ACID-transactions. Our previous work showed good potential, but suffered from known limitations due to the client and server side split coordination. The initial performance of this approach was acceptable and showed potential for good scalability. However, the communication bottleneck at scale between the client side coordination and the server side coordination would be unworkable. In this poster, we address these limitations and introduce three new technologies: 1) A client-side coordinator only optimization; 2) Data storage requirements and evaluation for a example transaction aware system; and 3) a metadata system requirements and evaluation for supporting transaction control over entries. Additionally, we were able to find many redundant or unneeded messages in the first version of the protocol that we removed without reducing the offered resilience. Experimental results show that with our refinements, we can attain the same level of transactional guarantees with a measurable decrease in overhead costs.

1. INTRODUCTION

As scientific applications scale towards exascale, they will incorporate more complex models that have previously only been run as separate applications. For example, in fusion science, simulation of the edge of the plasma [2] and the interior of the plasma [5] are currently separate simulations. To have a more complete, accurate model for a fusion reactor, these components will need to be tightly coupled to share the effects between the two models. The CESM climate model [4] is similar in that it incorporates atmosphere, ocean, land surface, sea ice, and land ice through a coupling engine to manage the interactions between each of these different systems yielding a more accurate model of global climate. In most cases, these and other scientific applications are part of larger offline workflows that process the output written to storage in phases that ultimately yields insights into the phenomena being studied. Current work to enable these coupling and workflow scenarios are all focused on the data issues to resolve resolution and mesh mismatches, time scale mismatches, and simply making data available through data staging techniques. In most of these cases, each of the components are run using a separate execution space for fault isolation and to aid in scalability.

Our initial implementation employed separate coordination systems for the client side and the server side. This has several limitations. First, the server side must listen for messages as well as process them, perhaps in complex ways. Using two different mechanisms such that one is capable of discovering an external message, such as RPC style systems like the NSSI system, and another like MPI for coordination among the server processes imposes a disconnect and introduces delays as the there needs to be some coordination between the threads that are listening for messages coming over different APIs and paradigms (RPC vs. MPI messaging). Second, the single point of communication between the client side and server side is all but guaranteed to incur a bottleneck that will prevent the level of scalability we desire. Third, the server side processes must be made aware of the transaction protocol to operate properly.

We addressed these problems by: 1) Merging into the subcoordinators on the client side the server affected; 2) Eliminating the need to have multiple messaging standards to support server-side functionality; 3) Abstracted the protocol to eliminate the need of the server to fully incorporate the transaction protocol affording integrating a variety of off-the-shelf products; and 4) Use message piggybacking to reduce the volume of messages without reducing resilience.

The D2T protocol aims to offer full ACID-style guarantees for the encapsulated operations such as data movement or system reconfiguration for fault tolerance and load balancing. While this is certainly possible with adequate hardware particularly to support the durability guarantee, this initial work shows that such a system can be built that supports most of the ACID-style guarantees with current hardware, what the performance impact will be, application coding implications, and begin to address the scalability challenges so that it is applicable for exascale-sized platforms. More specifically, D2T can address both the code coupling/online workflow/data staging as well as the fault tolerance/system reconfiguration scenarios.

While it is true that for many high transaction volume environments, ACID-style transactions can lead to scalability problems catalyzing the explosive growth of NoSQL style data stores. The D2T techniques are intended for a different environment than BASE properties can support. In our scenario, we are focused on supporting online workflows and other high performance and distributed computing opera-

tions. For these scenarios, eventual consistency is not always sufficient. Throughput in the online workflow is only possible with guarantees about data completeness and correctness. The BASE properties are insufficient for maintaining this throughput.

2. RELATED WORK

The concept of distributed transactions has been around for decades. We are extending this technology to address distributed clients working in concert. While this is not critical for the core database area, is crucial for HPC applications given the massively parallel nature of the modern HPC environment. ZooKeeper [1] and other Paxos [3] implementations have a superficial similarity to our D2T protocol in that they provide the consistency and synchronization mechanisms for messaging to a collection of servers from a distributed set of sources. Under the hood, Paxos is solely 1xN with an eventual consistency model. The inherent assumption that an update or insert originates from a single source limits the applicability of the protocol for this environment. GridFTP and Sinfonia offer related functionality, but either do not offer the same levels of guarantees or is limited to a 1xN semantics inappropriate for the HPC environment.

3. THE D^{2}T PROTOCOL

The D^{2}T protocol provides the necessary extensions to traditional distributed transactions to afford extensions to distributed clients as well as distributed servers, hence the doubly distributed transactions. (Figure included in final poster.)

In this edition of the protocol, we identified that several messages were unneeded. First, in the first version of the protocol, we had messages that were used to signify the ending and beginning of phases. After evaluating the protocol, we could see that the message ending one phase and a separate message starting the next phase was redundant; now the ending of one phase is implicit with the start of another. Resiliency is not lost here because if not all clients agree to move to the next phase, we can consider this to be a failure.

In the first version of the protocol presented last year, we relied on a server side coordinator to gather the votes for each server and inform the client coordinator of the collective vote (i.e., abort or pre-commit). We noticed that this was not needed because each client using a server can determine if the operation completed successfully. For example, if the clients are writing to files, the clients themselves can reread the contents to verify correctness. Additionally, many services return error or success codes. With this insight, servers do not need to vote amongst themselves and can be completely agnostic to the transactional protocol which makes integration of the protocol into other services easier.

4. EVALUATION

These tests are performed on the RedSky Sun blade system at Sandia. Our protocol is implemented solely with MPI as we no longer require the protocol to communicate across application boundaries.

We conduct three sets of experiments. The first set of experiments are microbenchmarks that capture the overheads associated with executing our optimized protocol. We also compare these results with the results from the previous edition to quantify the improvements with our optimized protocol. The results show that both pruning the volume of messages and removing the need communicate across application boundaries during the protocol drastically lower the time needed to execute the protocol (results included in the final poster).

The next sets of experiments make use of the protocol for real applications. In the first application, transactions are used to provide clients with transactional guarantees when performing operations on a metadata service. The second application uses the transaction protocol when operating on a distributed datastore that also aims to provide clients with durability by replicating the data. In both of these examples, we also introduce failures and show that the transaction protocol can inform the clients of these failures, and that overheads induced only minor overheads to the applications (Results and figures included in final poster).

5. CONCLUSION

In this paper we expanded upon our original implementation of D^{2}T and made several optimizations that will allow for improved scalability and greater applicability to a wider array of services. Our results and integration with real examples validate our approach and show our techniques allow us to get better scalability without loss of resilience. Our current and future work is integrating our transaction protocol into a runtime management system for online analytics.

6. ACKNOWLEDGEMENTS

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

7. REFERENCES

[1] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. Zookeeper: wait-free coordination for internet-scale systems. In *USENIX ATC*, USENIXATC'10, Berkeley, CA, USA, 2010. USENIX Association.

[2] S. Ku, C. S. Chang, M. Adams, E. D. Azevedo, Y. Chen, P. Diamond, L. Greengard, T. S. Hahm, Z. Lin, S. Parker, H. Weitzner, P. Worley, and D. Zorin. Core and edge full-f ITG turbulence with self-consistent neoclassical and mean flow dynamics using a real geometry particle code XGC1. In *Proceedings of the 22th International Conference on Plasma Physics and Controlled Nuclear Fusion Research*, number IAEA-CN-165/TH/P8-40, Geneva, Switzerland, 2008.

[3] L. Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16, May 1998.

[4] NCAR and UCAR. Community earth system model. http://www.cesm.ucar.edu/models/cesm1.0, 2012.

[5] W. X. Wang, Z. Lin, W. M. Tang, and W. W. Lee. Gyro-Kinetic simulation of global turbulent transport properties in tokamak experiments. *Physics of Plasmas*, 13(9), 2006.