**Abstract**. The road to exascale computing is well underway. Exascale systems that are slated for the end of this decade will include up to a million of compute nodes running about a billion threads of execution [1]. Ongoing efforts propose a radical change in the storage I/O stack [2,3]. Two key principles behind the design of these new systems are (1) asynchronous I/O and (2) transactional storage, both targeted at enabling high concurrency I/O capabilities. The implementation of these principles gives rise to new challenges that need to be addressed in order to satisfy the exascale requirements. In this project, we identify two of such new arising issues: transaction coordination and efficient I/O resource management at the network level.

## 1 Transaction Coordination

New Exascale I/O stacks will expose transactional capabilities to user applications without imposing any constraints in the serializability of operations [4]. For example, the Fast Forward IOD layer implements timestamp-ordering-based multi-version concurrency control [5] by requiring every call to be associated with a transaction ID [6]. The decision of how versions of a particular object interact among each other (eg. how new versions of an object are derived from committed transactions) is left to the application. Conceptually, a traditional transactional system can be broken into three orthogonal pieces [7]: concurrency, reliability and replication control sub-components. Thus, from this point of view, new I/O storage stacks such as FastForward's provide distributed concurrency control primitives on top of which reliability and replication mechanisms can be built. In this project we will look at the alternatives for providing transaction coordination (reliability control). Initially, we don't plan to deal with replication (eg. for high-availability or fault-tolerance), although we will keep it in our radar as we make progress in the transaction coordination front.

There is a large volume of work on the topic of transaction coordination [8], mainly in the context of relational databases, providing strong, serializable consistency. In our case, we are interested at looking at this from the point of view of scientific computing platforms, where the communication models are different (fast interconnects; collective I/O) and thus require to either adapt existing techniques [9] or even define new atomic commit protocols.

Recent work has analyzed alternative protocols that provide weaker isolation levels [10] in a distributed setting [11], which could also be applied "as is" in a scientific computing scenario, or might require to be redefined. The principal issue behind picking the right type of coordination lies in determining the application requirements with respect to degrees of isolation, which is not an easy task [12]. We will work with Sandia, LANL and associated co-design centers to learn more about the requirements of scientific applications. We will also empirically test by implementing several atomic commit protocol alternatives and executing existing scientific workloads on them, with the goal of gaining insight into the isolation semantics required by simulation/analysis jobs and the associated trade-offs.

Another area of interesting work is the analysis of the compromise between placing the I/O nodes in the same network where compute nodes reside (eg. infiniband), against having them on a slower, external communication channel (eg. 100Gbps ethernet). In this case, we have to identify the main differences among the two settings and determine the type of transaction coordination required on each, with the goal of optimizing performance.

## 2 Efficient I/O Resource Management Through SDN-based Quality of Service

Having asynchronous and concurrent capabilities at the I/O layer results in having tens or hundreds of jobs running in tandem. However, I/O resources are finite and thus many different tasks will eventually end up competing for them. Since not all applications have the same priority, this resource contention will negatively impact the performance of time-sensitive applications[1]. Since over-provisioning at Exascale will be too costly (mainly due to the energy requirements that this would imply), efficiently managing the resources of the I/O layer will be fundamental to achieving the required performance.

Current research in the area of Software Defined Networking (SDN) [13] is looking at the problem of QoS in datacenters [14]. Many of the design principles behind SDN can be transfered to high-performance interconnects [15]. Achieving QoS in an HPC setting is a challenging task for which we have identified two important problems: (1) devising QoS techniques suitable for the exa-scale requirements and (2) coordinating network changes in a consistent way.

Quality of service implemented in software-based controllers can potentially bring real-time computing to the network. In this project we plan to leverage the work done in our group [16] in the context of real-time systems and apply it to the new emerging field of SDN-based QoS. With such a solution, the entire software stack (from application to network, passing through memory and CPU) could get guaranteed performance, simplifying significantly the development of workload management.

A network operating under the SDN model runs a (logically) centralized controller that monitors the state of the network and reconfigures the switches dynamically. We anticipate the process of transitioning from one configuration to the next to be very similar to the one we have to solve for coordinating transactions at the exa-scale I/O layer. We plan to evaluate if the methods described in the previous section can be directly applied to the SDN-enabled HPC setting and, if needed, adapt or devise new ones.

[1]For example, consider the case where we want to have a suite of (highly parallel) analysis tasks completed for each checkpoint. Assuming a new checkpoint arrives every hour and the data of the current checkpoint only resides in the data staging area of the I/O layer (eg. the burst buffer in Fast Forward) for three hours before it is deleted (data staging areas are usually thought of having enough capacity to hold three checkpoints). The more analysis tasks have to be performed, the smaller the set of viable schedules that completes them all within three hours. Also, a lot of the analysis is based on a pipeline of intermediary results. If we treat all analysis tasks independently and merely fairly, they end up fighting each other for resources and the emerging schedule might not allow all tasks (or even all important ones) to complete.

## 3 References

[1] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir, “Toward Exascale Resilience,” *Int. J. High Perform. Comput. Appl.*, vol. 23, nov. 2009, pp. 374–388.

[2] Intel, “Scope statement for extreme-scale computing research and development (fast forward) storage and I/O,” dec. 2012.

[3] P. Braam, *The Exa-scale I/O initiative – EIOW*, 2012.

[4] E. Barton, “Lustre* - Fast Forward to Exascale,” mar. 2013.

[5] D.P. Reed, “Implementing atomic actions on decentralized data,” *ACM Transactions on Computer Systems*, vol. 1, feb. 1983, pp. 3–23.

[6] J. Bent, “Milestone 3.1 Initial Design - IOD,” mar. 2013.

[7] M.T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, Upper Saddle River, NJ, USA: Springer, 2011.

[8] Y.J. Al-Houmaily, “Atomic commit protocols, their integration, and their optimisations in distributed database systems,” *International Journal of Intelligent Information and Database Systems*, vol. 4, jan. 2010, pp. 373–412.

[9] J. Hursey, T. Naughton, G. Vallee, and R.L. Graham, “A Log-Scaling Fault Tolerant Agreement Algorithm for a Fault Tolerant MPI,” *Recent Advances in the Message Passing Interface*, Y. Cotronis, A. Danalis, D.S. Nikolopoulos, and J. Dongarra, eds., Springer Berlin Heidelberg, 2011, pp. 255–263.

[10] A. Adya, B. Liskov, and P. O’Neil, “Generalized isolation level definitions,” 2000, pp. 67–78.

[11] P. Bailis, A. Fekete, A. Ghodsi, J.M. Hellerstein, and I. Stoica, “HAT, not CAP: Highly Available Transactions,” *arXiv:1302.0309*, feb. 2013.

[12] P. Alvaro, N. Conway, J.M. Hellerstein, and W.R. Marczak, “Consistency analysis in Bloom: a CALM and collected approach,” 2011, pp. 249–260.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: enabling innovation in campus networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, mar. 2008, pp. 69–74.

[14] A.R. Curtis, J.C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: scaling flow management for high-performance networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 41, aug. 2011, pp. 254–265.

[15] G. Wellbrock, T. Wang, and O. Ishida, “New paradigms in optical communications and networks,” *IEEE Communications Magazine*, vol. 51, 2013, pp. 22–23.

[16] R. Pineiro, K. Ioannidou, S.A. Brandt, and C. Maltzahn, “RAD-FLOWS: Buffering for Predictable Communication,” 2011, pp. 23–33.