Exploring Trade-offs in Transactional Parallel Data Movement
Ivo Jimenez, Carlos Maltzahn (UCSC); Jai Dayal (Georgia Institute of Technology); Jay Lofstead (Sandia National Labs);

The Road to Exascale

Exascale systems that are slated for the end of this decade will include up to a million compute nodes running about a billion execution threads. In this scenario, traditional methods that ameliorate I/O bottlenecks do not work anymore. I/O Staging \(^1\) proposes designating a portion of the nodes to manage I/O.

The Need for Transactions

Transferring a checkpoint or analysis output to the staging area (or from the staging area to long-term storage) is challenging, even at current petaflop scales. Transactions provide a framework in which users can easily reason about data movement across the I/O stack.

The Challenge

Traditionally, transactional systems assume that requests are initiated from a single client, and that each client's transaction are relatively independent of each other. HPC workloads don't fit these assumptions since all clients work in unison producing simulation output. A user would like to observe atomic and durable transfers across the I/O stack.

I/O stack requirements

In order to solve the multi-client scenario, recent work \(^3\)–\(^4\) proposes abstracting the storage with basic concurrency control capabilities and thus allow clients to manage isolation semantics. One way this can be achieved is by having storage servers that implement:

1. Multi-versioning concurrency control.
2. Object visibility control.

I/O stack requirements

In order to solve the multi-client scenario, recent work \(^3\)–\(^4\) proposes abstracting the storage with basic concurrency control capabilities and thus allow clients to manage isolation semantics. One way this can be achieved is by having storage servers that implement:

1. Multi-versioning concurrency control.
2. Object visibility control.

Consensus Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Fault Model</th>
<th>Blocking</th>
<th>Async Replication</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>NBTA</td>
<td>none</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2PC</td>
<td>fail-stop</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3PC</td>
<td>fail-stop</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Paxos</td>
<td>fail-recover</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1. Several consensus protocols and their features. The NBTA protocol is a variation of the Highly Available Transactions \(^5\) formalization, providing Read Committed isolation guarantees.

Our goal is to explore the trade-offs across the transaction coordination spectrum, identifying precisely where overheads are at and thus provide a toolkit for scientists to allow them to pick the most appropriate alternative for their workloads.

Preliminary Evaluation

Related Work

- The DOE’s Fast Forward Storage and I/O project is implementing transactional features into a next-generation stack. The FastForward protocol used to implement transactions is similar to the NBTA protocol referenced here.
- Many proposals for fault-tolerance \(^6\) in HPC make use of consensus protocols to identify faulty processes. Our work is complementary to these efforts.

---

\(^{1}\) Liu et al., On the Role of Burst Buffers in Leadership-class Storage Systems. MSST ’12. http://dx.doi.org/10.1109/MSST.2012.6232369
\(^{2}\) Lofstead et al., Adaptable, metadata rich IO methods for portable high performance IO. IPDPS ’09. http://dx.doi.org/10.1109/IPDPS.2009.5161052
\(^{3}\) Lofstead et al., D2T: Doubly Distributed Transactions for High Performance and Distributed Computing. CLUSTER ’12. http://dx.doi.org/10.1109/CLUSTER.2012.79
\(^{4}\) DOE Extreme-Scale Technology Acceleration. FastForward https://asc.llnl.gov/fastforward/