Transactional In-Memory Data Grid

Kim-Thomas Möller, Marc-Florian Müller, Michael Sonnenfroh, Michael Schöttner

Kim-Thomas.Moeller@uni-duesseldorf.de

Heinrich-Heine-Universität Düsseldorf, 40225 Düsseldorf, Germany

Existing grid middleware typically supports only explicit data transfer between distributed nodes, e.g. GridFTP and GridMPI. We believe that future grid and cloud systems will benefit from complementing traditional message-passing with memory sharing techniques also known as in-memory data grids. The problem we address is the complex management of distributed shared states in distributed and parallel applications. We aim at simplifying the development of grid applications by abstracting from access location, replica management, data consistency, and fault tolerance. As a proof of concept, we are implementing an object sharing service (OSS) as part of the EU-funded XtreemOS project, which implements a Linux-based grid operating system (see http://www.xtreemos.eu).

OSS automatically replicates objects to improve reliability and performance. The software architecture we propose interweaves concepts of distributed caches, distributed shared memory, and peer-to-peer systems. Given that synchronization characteristics of shared data are application dependent, we have designed OSS to supports different consistency models within one distributed application.

Currently, we are implementing an extended transactional memory for wide area networks [1]. The main motivation here is that speculative transactions allow convenient and consistent access while at the same time relieving programmers from complicated lock management. As a demonstrator within the XtreemOS project we use OSS and the transaction-based data-sharing as a basic building block for providing a distributed platform for Wissenheim a multi-user 3D virtual world providing edutainment and entertainment content . In order to offer a platform for Wissenheim, we address scalability with a high number of nodes, increased network latency and bandwidth constraints. The virtual world scenes are rather static, such that bandwidth is a minor problem.

More challenging is network latency depending on the consistency requirements and the number of participating nodes. Before we present how to apply peer-to-peer technologies here, we first analyze the specific characteristics of Wissenheim. In Wissenheim we can accept higher latencies than single shooter online games which can tolerate around 150 ms max. Secondly, it has been observed that players form groups typically within equal time zones. Thus the distance among nodes and also the network latency among the majority of a group are limited. Like other projects, we divide the world into several consistency domains, e.g. islands or places [5]. In most cases this reduces the numbers of nodes involved in game state synchronization. But of course one could imagine the worst case where all online users meet at one place.

To improve scalability of speculative transactions, we have implemented a super-peer-based overlay network structure (based on the network latency metric) moving transaction validation to super-peer nodes. Peers can mask validation-latency by starting a depending transaction before the previous commit is finished. Here we take the risk of cascading aborts instead of idling, but limit the dependency chain allowing one or two pending commits.

Furthermore, OSS supports local commits: If none of the modified objects is replicated, the node can just commit without requiring a network synchronization. Other optimizations include the support of read-only transactions that may request specific object versions to obtain a consistent view even if there would be a conflict with an updating transaction. This is useful for objects which are updated and read periodically. For example, the game engine updates avatar positions once per second with a read-only transaction, and between update events use dead reckoning to interpolate or extrapolate avatar positions.

Instead of using a reliable overlay multicast, super-peers and responsible nodes provide transaction history buffers that allow other nodes to re-synchronize themselves in case of missed transactions. But of course other consistency models may show up requiring some kind of reliable multicast like for example provided by SCRIBE [2]. To speed up data search we use a CAN-based distributed hash table (DHT) implemented among super peers [6].

Fault tolerance is achieved by keeping a defined number of replicas spread over the overlay network. On the one hand, replicas shall be kept near interested clients for performance reasons, e.g. nodes controlling avatars seeing each other. On the other hand, replicas need to be geographically spread over the network in order to be able to mask network partitions. In order to allow local commits, we have implemented backup replicas which are valid but cannot be accessed directly by remote nodes. Anyway, in case of severe errors which cannot be masked by replication and other recovery mechanisms we rely on the grid checkpointing facility of XtreemOS which periodically checkpoints distributed and parallel applications.

We are encouraged by our preliminary experiments both with respect to ease of development and reasonable efficiency. Preliminary measurement results with synthetic measurements discussed below, show that transaction-based processing is not for free but can be efficient. The used testbed includes 16 nodes each with *2* AMD Opteron Dual Core CPUs, 1,8GHz, 2GB RAM and Debian Linux 64 (Kernel 2.6.24.3), and a Gigabit-Ethernet switched network.

Both figures evaluate two extreme situations: a worst case where all involved nodes increment the same shared variable and a best case where all nodes increment a private variable. The first will result in many transaction collisions whereas the latter will cause no collisions at all. Figure 2 shows transaction throughput for global commits whereas figure 3 shows the benefits of mixed commits (global and local commits). Local commits are only possible for the best case not for the worst case which requires global commits due to collisions. Global commits are serialized by a circulating token.

Figure 2 shows a much lower transaction throughput for the worst case caused by the collisions. However, also for the best case global commits require the token to be fetched over the network causing as expected a limited scalability. Currently, token requests are forwarded to the next probable token owner leading to expensive token races. In the super-peer network scenario the token is passed among super-peers only, but further optimizations are planned.

Figure 3 shows the benefits of local commits also improving the worst case, because a node may locally commit several transactions before a token request of another node arrives. The best case avoiding any collision scales almost linearly with the number of nodes.

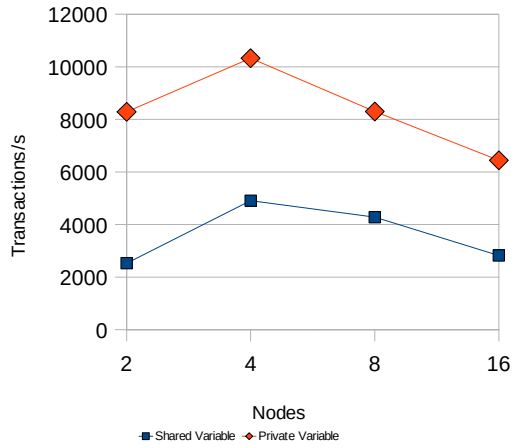

Figure 2, Transaction Throughput – Global Commit

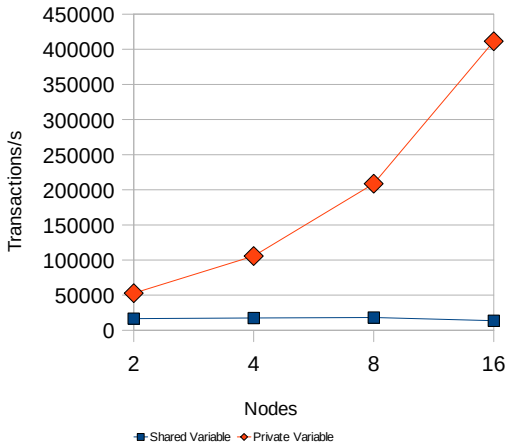

Figure 3, Transaction Throughput – Mixed Commit

We have measured an average page request time of 302µs and and an average token request time of 443µs. We are aware that wide area networks will dramatically reduce transaction throughput for global commits but we expect reasonable numbers.

Some commercial products implement in-memory data grids allowing location-transparent access to volatile data stored in volatile memory, e.g. Coherence (Oracle) and Gigaspaces. XAP (Gigaspaces). Most of these solutions are limited to clusters, but some support limited intra-cluster data sharing over wide area networks. But none of these systems support speculative transactions. Sharing volatile data in grid environments has also been proposed by JuxMem, interweaving concepts from distributed shared memory and peer-to-peer networks but also without transaction support [3].

We do not claim that a transactional in-memoy data grid can relieve programmers from all optimizations, but we want to study how far this concept can be pushed to ease the development of distributed and parallel applications. The growing interest in the field of transactional memory [4] makes us confident that, in the future, more transaction-based applications that can benefit of a distributed data sharing service like proposed will emerge. This ongoing work is funded by the European Commission (EC, Contract no FP6-033576). An early demo video of Wissenheim running on top of OSS is available at http://coconucos.cs.uni-duesseldorf.de/forschung/xtreemos/oss2.avi

References:

[1] M.-F. Müller, K.-T. Möller, M. Sonnenfroh, M. Schöttner, “Transactional Data Sharing in Grids”, in Proc. of the International Conference on Parallel and Distributed Computing and Systems, Orlando, USA, 2008.

[2] M. Castro, P. Druschel, A.-M. Kermarrec and A. Rowtorm, “SCRIBE: A large-scale and decentralized application-level multicast infrastructure”, IEEE Journal on selected areas in communications, Vol. 20, No. 8, Oct. 2002.

[3] J.-F. Deverge G. Antoniu and S. Monnet. “How to bring together fault tolerance and data consistency to enable grid data sharing. Concurrency and Computation: Practice and Experience, 2006.

[4] James Larus and Christos Kozyrakis, “Transactional Memory”, Communications of the ACM, Vol. 51, Issue 7, 2008.

[5] B. Knutsson, H. Lu, W. Xu, B. Hopkins, “Peer-to-peer support for massively multiplayer games”, IEEE Computer and Communications Societies, Volume 1, Issue , 7-11 March 2004.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, “A Scalable Content-Addressable Network”, Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, San Diego, USA, 2001.