SUMMARY:

Marcos Aguilera, Arif Merchant, Mehul Shah, Alistair Veitch, Christos Karamanolis

Sinfonia: a new paradigm for building scalable distributed systems

In *SOSP'07,* October 14-17, 2007.

DATE: February 22, 2010

This paper describes Sinfonia, a set of infrastructure services for building distributed applications in a reliable, highly connected environment. Sinfonia presents two key components - a memory-centric system architecture (as opposed to traditional message-passing architectures), and a 'microtransaction' operation which provides ACI (optionally D) properties to replicated data. The aversion to message passing is unusual - the complexities of properly developing multi-threaded shared data structures has pushed newer systems back towards a shared-nothing architecture (i.e. Barrelfish, among others). Data is addressed via node-unique address spaces, allowing applications to manually ensure optimal data placement within the system. It is unclear how much control applications have over things like data placement at replicas, or how certain node failure scenarios are presented to applications. The µtransaction utilizes an optimized two phase commit protocol (with user applications acting as the coordinator) to ensure atomicity, while primary-copy replication is used to provide availability. The low-level nature of Sinfonia means features like caching and load balancing are left to the applications.

The paper validates the concept by implementing both Sinfonia and two user services on top of it; a file system (SinfoniaFS) and an ordered multicast system (SinfoniaGCS). One major result of their implementation is the reduction in effort and code size necessary to implement user services compared to their traditional counterparts (although LinuxNFS and Spread provide many features not found in SinfoniaFS or SinfoniaGCS). Testing is also performed to validate both the performance and scalability of Sinfonia. While the results certainly show that the Sinfonia architecture is capable of providing similar performance results with substantially improved scalability, it is difficult to see what performance gains arose from using Sinfonia versus the implementation choices made in SinfoniaFS and SinfoniaGCS.

SUMMARIZED BY: Robert Robinson