A SCALABLE, TRANSACTIONAL DATA STORE FOR WEB 2.0 SERVICES

IMPLEMENTING WIKIPEDIA WITH SCALARIS

Web 2.0 – the Internet as an information society platform supporting business, recreation and knowledge exchange – initiated a business revolution. Service providers offer Internet services for shopping (Amazon, eBay), online banking, information (Google, Flickr, Wikipedia), social networking (MySpace, Facebook), and recreation (Second Life, online games). In our information society, Web 2.0 services are no longer a thing that is just nice to have, but customers today depend on their continuous availability, regardless of time and space.

How to cope with such strong demands, especially in case of interactive community services that cannot be simply replicated? All users access the same Wikipedia, meet in the same Second Life environment and want to discuss with others via Twitter. Even the shortest interruption, caused by system downtime or network partitioning may cause huge losses in reputation and revenue. Web 2.0 services are not just an added value, but they must be dependable. Apart from 24/7 **availability**, providers face another challenge: they must, for a good user experience, be able to respond within milliseconds to incoming requests, regardless whether thousands or millions of concurrent requests are currently being served. Indeed, **scalability** is a key challenge. Any scalable service, to be affordable, somehow requires the system to be self managing (see sidebar).

Availability is the proportion of time a system is in a functioning condition. More formally, availability is a ratio of the expected value of the uptime of a system to the aggregate of the expected values of up and down time. Availability is often specified in a logarithmic unit called “nines”, which corresponds roughly to a number of nines following the decimal point. “Six nines”, for example, denote an availability of 0.999999, allowing a maximum downtime of 31 seconds per year.

Scalability refers to the capability of a system to increase the total throughput under an increased load when resources are added. A scalable database management system is one that can be upgraded to process more transactions by adding new processors, devices and storage, and which can be upgraded easily and transparently without service interrupt.

Self Management refers to the ability of a system to adjust to changing operating conditions and requirements without human intervention at runtime. Self Management includes self configuration, self healing and self tuning.

Our **Scalaris** system, described below, provides a comprehensive solution for self managing, scalable data management. In our opinion, Scalaris and similar systems will be an important core service of future **Cloud Computing** environments.

As a common key aspect, all Web 2.0 services have to deal with concurrent data updates. Typical examples are checking the availability of products and their prices, purchasing items and putting them into virtual shopping carts, and updating the state in multi-player online games. Clearly, many of these data operations have to be atomic, consistent, isolated and durable (so-called ACID properties). Traditional centralized database systems are ill-suited for this task, sooner or later they become a bottleneck for business workflow. Rather, a scalable, transactional data store like Scalaris is what is needed.

SCALARIS KEY/VALUE STORE

As part of the EU funded SELFMAN[1] project we set out to build a distributed key/value store capable of serving thousands or even millions of concurrent data accesses per second. Providing strong data consistency in the face of node crashes and hefty concurrent write accesses was one of our major goals.

With our **Scalaris** system, we do not attempt to replace current database management systems with their general, full-fledged SQL interfaces. Instead our target is to support transactional Web 2.0 services like those needed for Internet shopping, banking, or multi-player online games. Our system consists of three layers:

- At the bottom, an enhanced structured overlay network, with logarithmic routing performance, provides the basis for storing and retrieving keys and their corresponding values. In contrast to many other overlays, our implementation stores the keys in lexicographical order. Lexicographical ordering instead of random hashing enables control of data placement which is necessary for low latency access in multi-datacenter environments.

- The middle layer implements data replication. It enhances the availability of data even under harsh conditions such as node crashes and physical network failures.

[1] SELFMAN is a specific targeted research project funded in the 6th framework programme of the EU under contract no. 34084.

- The top layer provides transactional support for strong data consistency in the face of concurrent data operations. It uses a fast consensus protocol with low communication overhead that has been optimally embedded into the structured overlay.

Together, these three layers provide a distributed key/value store as a scalable and highly available service which is an important building block for Web 2.0 applications.

WIKIPEDIA ON SCALARIS

As a challenging benchmark for Scalaris, we implemented the core of Wikipedia, the "free encyclopedia, that anyone can edit". Wikipedia runs on three sites. The main one in Tampa is organized in three layers, the proxy server layer, the web server layer, and the MySQL database layer. The proxy layer serves as a cache for recent requests, and the web server layer runs the application logic and issues requests to the data base layer. Wikipedia handles about 50,000 requests per second, from which 48,000 are cache hits in the proxy server layer and 2,000 are processed by the data base layer. The proxy and the web server layers are embarrassingly parallel and therefore trivial to scale. From a scalability point of view, only the data base layer is challenging.

Our implementation uses Scalaris to replace the data base layer. This enables us to run Wikipedia on geographically distributed sites and to scale to almost any number of hosts. It inherits all the favorable properties of Scalaris, such as scalability and self management.

The Wikipedia on Scalaris is fast. Using eight servers it executes 2,500 transactions per second. All operations are performed within transactions to guarantee data consistency and replica synchronization. Adding more computers improves the performance almost linearly. The public Wikipedia, in contrast, employs ten servers to execute the 2,000 requests per second on the large master/slave MySQL database in Tampa.

SELF-MANAGEMENT

For many Web 2.0 services, the total cost-of-ownership is dominated by the costs needed for personnel to maintain and optimize the service. Scalaris greatly reduces the operation cost with its built-in self* properties:

- **Self healing**: Scalaris continuously monitors the hosts it is running on. When it detects a node crash, it immediately repairs the overlay network and the database. Management tasks such as adding or removing hosts require minimal human intervention.

- **Self tuning:** Scalaris monitors the nodes' workload and autonomously moves items to distribute the load evenly over the system to improve the response time of the system. When deploying Scalaris over multiple data-centers, these algorithms are used to place frequently accessed items nearby the users.

In traditional database systems these operations require human interference which is error prone and costly. With Scalaris the same number of system administrators can operate much larger installations than with legacy databases.

SUMMARY

Scalaris provides a scalable and self managing transactional key-value store. We have implemented Wikipedia using Scalaris. Its scalability and self* capabilities were demonstrated in the IEEE Scalable Computing Challenge 2008, where Scalaris won the 1st prize (see plaque).

Compared to other data services, Scalaris has significantly lower operating costs. Scalaris and similar systems will be an important building block for Web 2.0 services and future Cloud Computing environments.

ADDITIONAL INFORMATION

- For the EU project Selfman see http://www.ist-selfman.org

- The Scalaris code is open source. It is available at http://code.google.com/p/scalaris/. Additional information (papers, videos) can be found at http://www.zib.de and http://www.onscale.de.