abstract

In distributed systems, importance about data communication technologies with distributed databases that processing asynchronously with user's business operation is increasing day by day. And an architecture with high reliability and scalability is demanded because amount of data and services are increasing recently. These systems are expected to complete their requests without loosing them when any system failure has occurs.

Application servers use databases for processing, the following architectures are already proposed:

- RDB based: all application servers use only one relational database
- KVS based: each application servers use their own databases which are constructed by Key-Value stores

RDB based architectures, which send data / receive data with queue asynchronously, have the following two issues:

- fault-tolerance: the queuing DB server may become a SPOF (Single Point of Failure), because there is only one instance in one system, so if it fails, whole system will stop.
- scalability: the queuing DB server may become a bottleneck for performance, because all application servers send data to the one instance, so the number of requests exceed the capacity of queuing DB server.

The goals of this survey are as follows:

- to provide a new architecture which can solve those problems, it is based with KVS(Key Value Store).
- to proof the effectiveness of this new architecture (KVS based) comparing with the current architecture (RDB based).

In the rest part of this paper, section 2 gives our survey on distributed systems and distributed databases (some variation of RDB and KVS based systems).

In section 3, we proposed the new architecture (KVS based) which consists of several application servers merged with queuing server and a database server.

We define a concrete application and provide a model to select measure points.

In section 4, we reveal the advances of KVS based architecture through experiments.

We found some unexpected results, so we inspect those results and identify causes in section 5.

In section 6, we discuss about adaptations for three systems with the following different features:

- which requires a large amount of transactions
- which don't need queuing DB server because of low amount of update, frequency, and conflicts
- which don't have to need special database realization

Finally, we conclude with several issues about our architecture.

- effectiveness
- adaptation
- capability for another applications with different database (KVS)

In conclusion, KVS based architecture is suitable about distributed systems from the viewpoints of fault tolerance and scalability.

section 2

We survey about distributed systems and centralized system to give a proof which KVS based architecture is prefer to RDB based architecture.

- distributed systems: at another place, cooperative each other (KVS based)
- centralized systems: at only one place, intensively (RDB based)

Following systems are surveyed:

- Client-Server
- Peer to Peer
- Grid computing
- Three-tier architecture

We survey about database especially because if database broke, system will break.
Following database (KVS) are surveyed:

- Oracle NoSQL (Master -Replica)
- HBase (Manager and Region)
- Cassandra (no special nodes)

We selected Cassandra as our new architecture.

section 3

We propose a new architecture and explain its features to give a model in order to clarify parameters.

We consider following issues as goals when we provide services with systems:

- maximize the availability by minimization range of impacts when some failures occur
- maximize the availability by dynamic system updates

- minimization of response time when number of response and amount of data increase

We select "stock price alert system". This system's features are follows:

- which receive data from data resource system when some notifiable movements in the stock market occurs
- which deliver data to users who wants to get new information about stock market within 20 seconds

We make several models about "stock price alert system", and perform some experiments by measuring functions and performance both RDB based architecture and KVS based architecture.

- normal performance (with no fault)
- processing and recovery when some fault occurs
- availability and degradation

section 4

We change following parameters and measured performance:

- number of delivered messages (10, 50, 100, 150, 200)
- number of application servers (1, 2, 3, 4)

We didn't change database schema, data size (request and response) and message path.

As the consequence of experiments, we find that KVS based architecture has following advantages to RDB based architectures:

- TPS is higher in whole
- Scalability is linear
- when database fault occurs, connection change is faster
- degradation is smaller
- CPU utilization in application servers are higher, and that in database server is smaller

section 5

We found following consideration points to build other systems on KVS based architecture.

- issues not appeared
  - memory leak
  - degradation

- issues with few influence

  - number of errors detected when some database failures occur
  - amount of memory utilization

- immaterial issues because of law frequency
  - RDB based sometimes gives higher performance than KVS temporary under low system loads

As for the performance of application servers, KVS requires higher than RDB because CPU bottlenecks move from a database server to application servers.

section 6

We consider about following systems to apply KVS based on architecture:

- which requires a large amount of transactions
  - which can apply same mechanism about queuing except initialization and methods for achieving redundant configuration

- which don't need queing DB server because of low amount of update, frequency, and conflicts
  - which can be adapted with KVS
  - which can't be adapted with KVS if RDB function is effective and there are much analyze dimensions

- which don't have to need special database realization
  - which can be adapted with KVS because of independence from database functions

section 7

We found following future works:

- control configuration and definition about access points from applications
- study on operations on recovery from database faults
- availability about all services
- study strategy about mechanism of another parameters

As for conclusion, KVS based architecture is suitable about distributed systems from the viewpoints of fault tolerance and scalability.
In the future, this can be adapted to various system developments with database with further knowledge of issues on performance bottlenecks, strategy for resource usages in order to maximize performance.