1. Application of NoSQL Database in Web Crawling

<u>Summary:</u>

Introduction: Web crawling is collecting, filtering and storing information in database from internet to design a search engine. So a database needed for this needs high storage capacity and less hardware cost. Scalability, performance and availability are main goals. The traditional relational databases stores data in 2D table as are bad choice when there is huge data. So many companies implement their own databases like BigTable of Google.

We will compare the relational (Sql) and non-relational (NoSQL) by using them to give solution for a certain meteorological BBS information collection system.

Web Crawling System:
Web crawler is a program which crawls the internets, reads the information and saves it in databases. It works by using the link relationship between webpages and expands to whole page. It uses spider for reading the web pages and saving URLs in URL databases, Controller to control the spider by deciding which URL to read from URL databases and page library to save the page data.

Meteorological BBS Information Collection System
Meteorological BBS information collection system filters and collects the posts of the representative meteorological BBS in Internet, including clud.weather.com.cn, www.cmabbs.com and so on. It provides a professional search engine database of meteorological information.
The posts are saved in text documents and then saved in databases.

First is the URL and the title of the post, next is “#floor” which means the start of the first floor, then “postby” means the person who post this floor, then time and content of this floor. If there are more floors, “#floor postby time content” will be repeated.

Data Characteristics
The structure of the post has the fixed URL and title, also has the unfixed floors. Each post has the different number of floors, so their structures are different. The system gets all the content of the post from the database according to ID, including all the floors. The structure design of the database needs to not only convenient to store the content from txt documents but also easy to query quickly.

Relational Database Solution
As this is huge data and it increases by day, SQL server 2005 was used for this. As SQL uses 2D table structure with no nested tables so they used two tables to store the data- tb_post and tb_postback, to store the posts and floors.
Table tb_post stores the fixed information of post, including URL and title. Table tb_postback stores the unfixed of the post, the floors. Each floor stores in tb_postback as a record with a foreign key postID point its post. When the system query the post by the ID, SQL clause is as follows:

Select * from tb_post where ID = strID; Select * from tb_postback where postID = strID order by time desc.

NoSQL Database Solution

NoSQL – Not Only SQL is another form of databases. It removes the constraint of SQL databases for high performance and scalability.
MongoDB is one of the famous NoSQL databases and it save the data in key value pairs. It uses BSON (Binary JSON) format to store the data in document format. MongoDB does not use schema model and so it have join free queries unlike relational databases. It has high performance, high availability (master- slave replica model: slave will automatically promoted to master in case master fails), easy scalability with automatic sharding. It also has rich query language.

First create database InfoDB. One document PostBar is added with title, URL and floor1. Floor1 has postby, time and content. If more posts are added then floors are added to the document.
The codes to create a document are as following:
1. Create a document by class BasicDBObject, add fields with simple data type:
BasicDBObject doc = new BasicDBObject();//create document
doc.put("URL","http://tieba.baidu.com/f?kz=78242.html");// add key/value pair to field URL
doc.put("title", "Dalian's weather forecast");// add key/value pair to field title
2.To the document type of field, first create a document by class BasicDBObject as embedded document. Read the post floor 1, fill the information to embedded document, then add the embedded document to field floor 1 in document.
BasicDBObject floor = new BasicDBObject();//floor create the embedded document floor
floor.put("postedby", author);// add the field postedby
floor.put("time", time);// add the field time
floor.put("content", content);// add the field content
doc.put("floor1" , floor);// add the embedded document floor as the field floor1 to document doc
3. Read each floor of the post replies circularly, repeat the code in step 2, add field floor i to document.
4.add document to collection:
baiducoll.insert(doc);
When the system query the post by the ID, Mongo query is InfoDB.baiducoll.find({ id : strID}).

Solution comparison
The relational database stores the data in table and that structure is fixed. So in a way it provides data consistency but since we are using two tables to store the data then it increases complexity of data query also. If the data is ever changing then NoSQL is better as it is schema free and we can store all the data in same document. Even if the format of data changes we can add new data in the documents. Also the performance, when size of post reaches hundred of thousands is better in case on NoSQL databases.

Conclusion
Relational database has multiple tables' storage with foreign key, sharp decline in query performance with huge amount of data, and vertical scalability with high cost. Compared to relational database, MongoDB supports schema-free, has great query performance with huge amount of data and provides easy horizontal scalability with low cost of hardware. It is more suitable for data storage in Web crawling.

2. Comparing NoSQL MongoDB to an SQL DB

<u>**Summary:**</u>

This paper compares SQL(SQL Server) and NoSQL(MongoDB) databases when working on modest amount of structural data. We already know that relational databases work better with normal amount of structured data while NoSQL databases works better on huge unstructured data. There was no comparison available on the working of NoSQL databases on modest amount of structured data which this paper tries to find out.

Introduction
Relational databases stores data in 2D tables and the data is always structured. The data are stored in multiple tables using primary keys and foreign keys. If we need to get data from multiple tables then we use join queries. The time to fetch data is directly proportional to number of tables used. On the other hand in non-relational databases there is no schema and no structure in the data. All the data can be stored under one document in a key value pair. The querying is simple but designing it can be difficult.
This paper compares the performance on relational and non-relational databases on modest amount of structured data on the basis od following three factors: insert speed, update speed, and select operation speed.

Related Work
There are many papers comparing various NoSQL databases but there are almost none comparing NoSQL to SQL databases. MongoDB saves the data in key value pairs. User can save all the values in same document in case of one to one or one to many relationships. In MongoDB either user can save all the document in one collection or make different collection and save only references of one in another. But unlike relational databases the non-relational does not have predefined queries. So user need to come up with their own queries when getting data from various collections.
For example: MapReduce in which user gives specific instructions during bot Map and Reduce phase.

Experiments
Three tables/collections – User, Department and project were made to run the experiment. The department can have many projects and projects can have many users. An manager which is an user can have many users working under them. A department can have many managers and many projects.
100 operations of all the kind of queries were run on both MongoDB and SQL server.

Result
<u>Insert</u>: The insert time is almost same for both MongoDB and SQL
<u>Update</u>: MongoDB took more time to update when working on non-indexed fields. On the other hand when the updates were made on indexed fields or primary keys then MongoDB fared better than SQL. It may be this performance gap is due to MongoDB having a pre-built index on the primary key of the document which is faster than SQL Server's primary key clustered index.
<u>Select</u>: MongoDB fared better with simple select queries but when the data from many tables were used and the queries became complex the performance of MongoDB deteriorated drastically as the amount of data increased.

Conclusion and Future Works
MongoDB works better for insert update and simple queries. While SQL works better when updating, querying non-key attributes and aggregate queries. MongoDB works great for large data. In future the writers would like to run MongoDB and SQL as a distributed database.