Eddie Gurnee
2015-04-13
COSC 571

NoSQL

The NoSQL movement marks the rise of data storage paradigms that differ from what has been the most historically prevalent: relational databases. Relational databases have many strengths in storing well structured data, as well as maintaining that the data storage transactions are all ACID compliant (Atomicity, Consistency, Isolation, Durability); however, with the ever changing way data is generated, how that data should be represented, and how much of that data should be stored, the traditional format of structured storage for information can no longer guarantee efficiency in storing and retrieving information. The 'No' in NoSQL is now taken to stand for 'Not only'; these data storage systems are designed to deal with the ever changing environment of information, in an adaptive and distributed manner.

NoSQL today exists in many forms, attempting to solve many of the big data problems that simply cannot be solved in a relational database manner. Simply too much data being retrieved by social media, major corporations, and governments, along with the semi- or un-structured manner this data is received, leads relational database solutions to fall behind.

A major strength of NoSQL lies in its support for concurrent connections and distributed processing; NoSQL systems were initially designed and created for the distributed environments[1] that were becoming most common. From day one, NoSQL systems have been designed with distributed computing in mind, aiming for a horizontal scalability in system growth with data being store among multiple nodes. This horizontal scalability is what allows NoSQL systems to combat the two of the major problems with relational databases: as massive amounts of data enters the database system the system will not need to be restructured with a stronger centralized processing power to contend with the additional data, simply adding additional storage nodes to hold the data is needed[1], secondly, as the types of data that needs to be stored changes and evolves a relational database would need to rewrite the entire schema in order for those changes to be supported, while NoSQL systems can handle the changes in data structure without hassle[2].

The term 'NoSQL' can be attributed to any data storage system that is not strictly a relational database model, but several features exist among most NoSQL models that allow for the distributed processing environment. Sharding is the main tool for providing horizontal scalability, each node in the system contains a piece of data and performs operations on that individual unit of data[1]. While sharding is an exceptionally powerful method for allowing horizontal growth of the system, sharding also leads to the probability of faults increasing; to combat those potential faults, much of the data should be redundantly stored. This data replication, where nodes can store multiple copies of data, helps improve system reliability and prevent system failures[1] and is often done in one of two ways: master/slave, where one node node holds most up-to-date info and slaves get data from that, or master-master, where all nodes can hold data that is computed and passed around.

With the master-master model of redundant storage, if the updates are performed asynchronously, which is advised for maximum use of the distributed model, there can exist an issue with data consistency where the user can read information that is not the most up-

to-date at a given moment; most NoSQL use the eventually consistency model and because of that there is an 'inconsistency window' with outdated data, though the data will become consistent eventually. Alternatively, the monotonic reads model of consistency states if a value was read by a client, then the further reads will never return older values[1]; this can be used with the 'read your writes' model which guarantees that the client always reads the data that it wrote earlier, so at the very least the client is never reading older data than it has already read or written. Finally the immediate consistency model guarantees that as soon as the data modification operation has finished, all clients can immediately see the updated data[1], allowing for the guarantee of consistency afforded by relational database models.

Beyond the features that bind NoSQL systems together, these systems do come in very distinct flavors. A key-value database stores data as key-value pairs, eschewing the relational model all together, stores data as a single collection which may have many different fields for every record[1]. A document oriented database is, at a basic level, a type of key-value database, with the crucial difference being that the data is stored in a self describing manner, the metadata explaining the data is store along with the data itself[1]. Column database systems store tuples that are a representation of the individual data: it stores a column name (that is unique), the content of the data, and a timestamp of the associated date when the data was most recently updated[2], this timestamp is critical for assisting in maintaining consistency among the distributions of data. Lastly, a graph database stores data as entities and stores all of the relationships between the entities as edges in the form of direct links[2].

The document oriented database should be of particular interest to any JavaScript developer; many document oriented databases take their root element of a 'document' from JSON[2]. In fact, CouchDB, a major document oriented database system, uses explicit JSON documents as their arbitrary unit of storage[2]; this allows for very direct communication between the database and the programs that interact with it, the database representation of the data is identical to the program runtime implementation of the data. This allows system developers to worry less about the conversion of data to proper database structure, and guarantees that the data a developer needs to use is the same as the data that is being stored.

NoSQL is the current forefront of database theory, attempting to build systems that not only can handle the data challenges that the world faces today, but in a way that allows the system to be able to adapt and expand to the challenges that the world could and will be facing tomorrow. Research into more abstract and simultaneously efficient ways to store data, not only NoSQL (or NoNoSQL), is needed for the technology for tomorrow; with the world requiring faster connection to the data they seek, as well as a desire for ever more data to be accessible, creating systems that can support these requests with some semblance of efficiency is mandatory.

References:

Kuznetsov, S. D. and Poskonin, A. V. NoSQL Data Management Systems. *Programming and Computer Software,* Pleiades Publishing, Ltd., 2014. 323–332. [1]

Redmond, Eric, Jim R. Wilson, and Jacquelyn Carter. *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. Pragmatic Bookshelf, 2012. [2]