International Journal of Current Engineering and Technology
©2015INPRESSCO®, All Rights Reserved

E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

Research Article

Supercharging web applications with Dart and MongoDB

Shrikar Chonkar[†*], Siddaharth Suman[†], Shreyas Sawant[†] and Shikha Moondra[†]

[†]Department of Computer Engineering, Atharva College of Engineering, Marve Rd, Malad (West), Mumbai-95, Maharashtra, India

Accepted 12 Feb 2015, Available online 17 Feb2015, **Vol.5, No.1 (Feb 2015)**

Abstract

Dart is an open source, structured programming language for the web. The syntax of Dart closely resembles to that of Java, JavaScript and C#. It can be used in both the server and client environment, which makes it a sturdy bridge between a backend and a front-end. MongoDB is a new age database solution. Although it may not replace the relational databases as completely in near future, it has certain advantages over traditional databases which allow it to work seamlessly over the distributed storage like the cloud. Hence, Dart coupled with MongoDB can be a perfect formula for growing demand of faster and more scalable web applications.

Keywords: *Dart, MongoDB, JavaScript, NoSQL, Open Source, Mongo-Dart.*

1. Introduction

The World Wide Web was a very simple collection of static html pages for some time after it was launched in 1990. This could have sufficed for that time but now it is much more than a mere collection of simple static web pages. Today's web is much more complex and full of possibilities but it is not as fast as desktop applications.

The world today lives on complex web applications namely Gmail, Facebook, Twitter, YouTube, etc. We do more than just browse through plain text documents, we listen to music, watch movies, play games and what not. Crave for speed still exists today. The world wants the web to be faster to load and easier to access.

2. Javascript

JavaScript is a programming language most commonly used as part of developing web based applications. The JavaScript language provides the developers with the ability to make their pages dynamic.

JavaScript however, should not be confused with the Java programming language. First appeared in 1995, the language has evolved from just being a simple scripting language to be able to become basis for full-stack web application frameworks. Although it has been proved to work for the web developers for so many years, it has some of its own shortcomings.

As Paul Rubens says in his article, "JavaScript is often used in a way that was never intended: as a platform for the development of large Web applications that are hosted in the browser. If that had been its intended use rather than simply for adding simple dynamism to Web pages - it would almost certainly have been designed differently." (Paul Rubens, 2013)

JavaScript proved fast enough at the start for the simple things, but as it became a large part and demanded performance, it simply couldn't live up to it. This is when Google developed the V8 engine for JavaScript and embedded it into its Chrome browser in 2008. Since then, it has seen considerable improvements in performance and it seemed that JavaScript still had some hope left. Now, the web demands for much faster and efficient solutions since it is no longer just limited to desktops and laptops but also expanded towards smartphones as well and JavaScript kills battery life on embedded devices. When dealing with JavaScript, there are usually two problem areas found: Performance and Code Structure.

2.1 Performance

Performance wise, JavaScript is only good for small things. When used for making large applications, JavaScript is quite slow and thus impacts the user experience end.

2.2 Code Structure

JavaScript was not meant to be used as basis for large complex applications, thus it is a difficult language for large teams to develop in. This is because the language lacks structure. For a single developer, there is no problem, but for a team working on a big project, communicating the intent of the code becomes difficult.

*Corresponding author: **Shrikar Chonkar**; **Shikha Moondra** is working as Assistant Professor

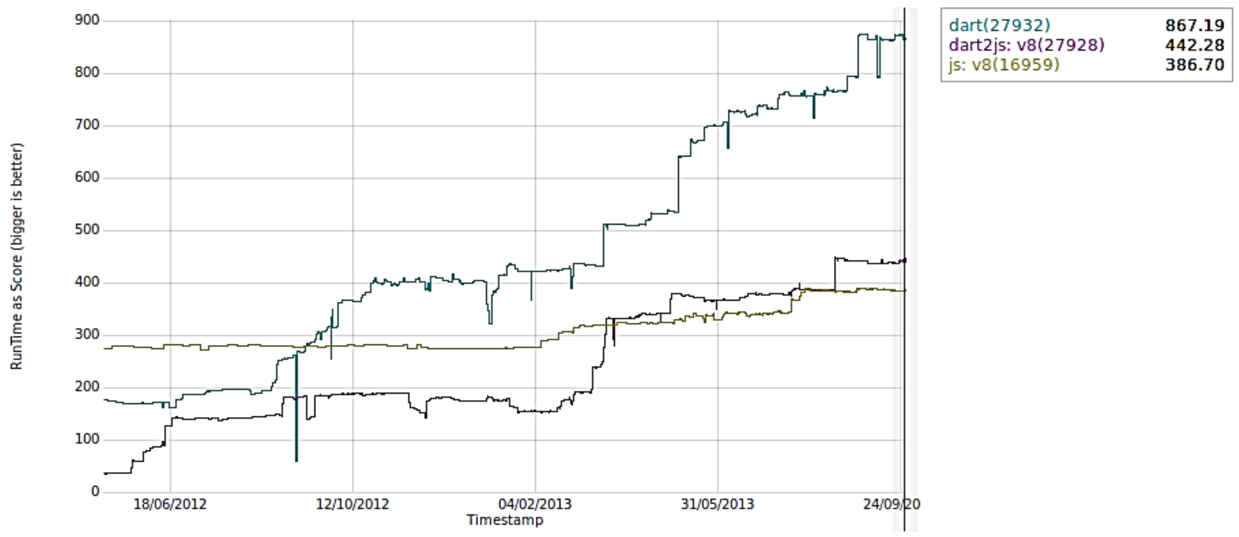

Fig.1 Performance of Dart, dart2js and JS (www.dartlang.org)

3. Dart

Dart, appeared in 2013, is an open source, structured, object oriented, programming language developed by Google for developing browser-based, complex web applications.

Google created Dart because even though JavaScript engines (V8) are becoming faster, web apps start and work very slowly. We expect dart to help in two main ways: better performance and better productivity. (Kathy Walrath and Seth Ladd, 2012)

3.1 Performance

Performance wise, JavaScript is only good for small things. When used for making large applications, JavaScript is quite slow and thus impacts the user experience end.

3.2 Productivity

Dart has a wide range of support libraries and packages which help in working with other developers and can easily reuse code from other projects.

Dart has familiar syntax code. Whether it is for a client-side developer experienced in JavaScript or for a server-side developer familiar with Java and C#, both will find the Dart code syntax very familiar. The creators of Dart have taken special care in making the language utterly easy to pick up.

To help with this developer diversity, Dart has an optional typing feature, which allows developers to specify absolutely no types (by using the var keyword, as in JavaScript), or use type annotations everywhere (such as String, int, Object), or use any mixture of the two approaches. This allows Dart's type system to bridge the gap between JavaScript's dynamic type system and Java's and C#'s static type system. (Chris Buckett, 2013)

Dart is suitable for building single page web applications which are the norm these days. Dart can run in the client as well in the server, hence the developers don't have to switch to different languages when dealing with client and server applications.

Therefore, Dart can act as a full-stack programming language.

When reading JavaScript, it is hard to point out the entry point code, whereas Dart has a single entry point called main() function. So, it is easier to find out where the code is going to start from and hence, more structure.

Dart is fairly new and it needs a browser which comes with a Dart VM to run it. Therefore, it runs on a special version of Google's open source Chromium browser called Dartium. To maintain backward compatibility with other browsers, Dart code compiles to JavaScript by use of the dart2js compiler. The thus obtained JavaScript code is claimed to be faster and lighter than conventional handwritten JavaScript code because of the tree-shaking technique. The tree-shaking technique "shakes" off unused code, thus decreasing the size of resultant JavaScript code.

Apart from these, Dart has many other features like: simple concurrency with Isolates, asynchronous tasks with Futures, Method Cascades, separate light-weight editor, static compilation with help of Type Annotation.

Keeping the evolution of the web applications in mind, it is always necessary to have a back-end that is able to withstand the growing possibilities and capabilities in web applications. Considering this, we require databases that are able to store large amounts of data effectively and provide a significantly high performance when reading-writing-updating the data.

4. Relational databases

Relational databases are the most commonly used and widely trusted type of databases till date. However these databases perform well only when it comes to small read and write operations.

As quoted by Buzz Moschetti in his blog, "RDBMS came into the picture in 1974, since then the business goals have changed and the pace of business has increased." (Buzz Moschetti, 2014) Along with the business goals, the technologies and programming languages that are using the databases have changed drastically.

During this change, the traditional relational databases faced many challenges. Especially in the web applications, that scale up and down constantly, it is difficult for relational databases to scale accordingly due to their rigid and schema based structure. Such structure takes away the ability of relational databases to store and query the dynamic user data.

This is where MongoDB comes in.

5. MongoDB

"MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling." (*www.docs.mongodb.org*) MongoDB is of the leading NoSQL databases that supports the schema-less data storage and portrays the ability of handling large amount of unstructured as well as structured data. Since the schema of this NoSQL database is not fixed, it is highly possible to change the structure of the data as per the scaling of the web application.

5.1 Data Structure

Each entry in the MongoDB database is essentially a key and value pair. Such a pair is called a 'document' and is equivalent to a tuple in relational databases. When compared to relational databases, key holds the same significance as a column. Such documents are very much similar to JSON objects and can include other documents, arrays or arrays of documents.

```
{
  name : "Stephen",
  id : 45,
  hobbies:["soccer", "reading"]
}
```

Fig.2 Example of a JSON object in MongoDB

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism. (*www.docs.mongodb.org*)

Each document is a part of a larger bunch of documents, called a 'collection'. The collections are similar to tables in a relational database. Lastly, number of collections together form a database and hence in order to access a record, one must first access the database, then the appropriate collection and finally the document.

5.2 Productivity

The productivity of NoSQL databases was designed while keeping their distributed usage in mind. To perform well in such environments, NoSQL databases use two technologies. Scaling and MapReduce.

The MongoDB supports automatic scaling and is known for its outward and horizontal scalability as a core feature.

Here scaling out refers to partitioning the data across different machines by adding additional commodity servers or by adding extra storage space. The document oriented data model of MongoDB makes it easier for the database to split up data across multiple servers. MongoDB automatically takes care of balancing data and load across a cluster, redistributing documents automatically and routing use requests to the correct machines. This allows developers to focus on programming the application, not scaling it. When a cluster needs more capacity, new machines are added and MongoDB will figure out how the existing data should be spread to them. (Kristina Chodorow, 2010)

The MongoDB implements scaling using two alternatives – Replication and Sharding.

Replication is the way of storing same data on different node machines. Such replication allows the application or a machine to establish multiple parallel accesses to the database allowing faster and easier access to data. Replication is also known to make databases 'fault-tolerant'. If one of the sources of a replicated database fail, the other replicated copies of that database act as backups and allow application server to continue the operation.

The biggest challenge with replication however, is the data inconsistency. Multiple copies of same data clearly mean that they need to be updated from time to time or at the right time in order to provide the accurate results irrespective of the source.

Sharding is the ability of the database to split the data into pieces and store each piece on a different node machine, which is called a 'Shard'.

The result of this is a segregated collection of objects spread on different nodes. The database system must have a service to route the operations through each node, so data can be stored, retrieved and deleted on the right shard. (Maxmiliano F. BRAGA *et al*, 2011)

The MapReduce is the combination of two processes – Map and Reduce. This process cumulatively ensures the compression of large amount of data in order to provide aggregated results.

The Map function returns key-value pairs and for those keys that has multiple values, MongoDB applies reduce function to produce the aggregated data. Both of this function are JavaScript functions. MapReduce can return the results of a map-reduce operation as a document, or may write the results to collections. The input and the output collections may be sharded. (*www.docs.mongodb.org*)

6. Dart with MongoDB

A Dart program alone, is not capable of establishing a connection to a MongoDB database. However, there are external libraries like Mongo-Dart, which allow a Dart program to easily connect and interact with a Mongo database.

Each Mongo database, when hosted on server, has its own URL. This URL can be passed to MongoDart's'Db( )' class' constructor which in turn connects to that database, and returns an object. Thus obtained object is then used to select one of the database's collection. Further using the same object of Db, one can call a '.open( )' method, which opens the collection in background, asynchronously.

Once the collection is opened, the dart program is free to read or write data into this collection.

```
{
    Dbdb = new Db("mongo://127.0.0.1/mongo-
    example/sample");
    varsample_coll = db.collection("first");
    db.open().then(() {
        returnsample_coll.find();
    });
}
```

Fig.3 Example of Mongo-Dart's connection to database

Future work

Considering the various advantages of Dart and MongoDB over conventional web technologies available, it is certain that many new and powerful web applications would be built based on them. The next step in this realm is of developing a large scale information system by harnessing the powers of Dart's properly structured code and swift operations withMongoDB'sflexible and scalable databases.

Conclusion

From this, it is easy to conclude that Dart may as well be backbone of upcoming web applications. With its close resemblance to the well-known languages and sophisticated code structure Dart is very much capable of replacing JavaScript in the near future.

MongoDB paired with Dart makes a solid platform for the web applications to build upon. Using technologies like AngularJS for the front end it is very much possible to build a completely working cloud-based, platform-independent application from the scratch.

Acknowledgement

We would like to thank our project guide professor ShikhaMoondra for helping and inspiring us throughout this project. We are also grateful to our head of the department, ProfessorMahendraPatil for the support. Lastly we would not be able to conduct this research without the infrastructure and facilities provided by Atharva College of Engineering.

References

Paul Rubens (2013), Can Google Dart Solve JavaScript's Speed and Scale Problems? *http: // www. cio.com/ article/2382855.*

www.dartlang.org.

Kathy Walrath and Seth Ladd (2012), Dart: Up and Running, *O'Reilly Media, Inc.*

Chris Buckett (2013), Dart in Action, *Manning Publications Co.*

Buzz Moschetti (2014), MongoDB vs SQL: Day 1-2, *www.mongodb.com/blog.*

www.docs.mongodb.org.

Kristina Chodorow (2010), MongoDB: The Definitive Guide, *O'Reilly Media, Inc.*

Maxmiliano F. BRAGA, F´abio N.D. LUCENA (2011), Using NoSQL Database to Persist Complex Data Objects. *Brazilian Society for the Progress of Science (www.sbpcnet.org.br)*