AUSTRALIAN JOURNAL OF BASIC AND APPLIED SCIENCES

ISSN:1991-8178 EISSN: 2309-8414
Journal home page: www.ajbasweb.com

Node JS: Building an High Performance Event Manager in Android Platform

Kavitha Subramani, V. Hemapriya, A. Anto Libertina and V.R. Yazhini

Department of Computer Science Panimalar Engineering College Chennai, India

Address For Correspondence:
Kavitha Subramani, Department of Computer Science Panimalar Engineering College Chennai, India.
E-mail: kavitha_bhskr@yahoo.com

ARTICLE INFO

Article history:
Received 10 December 2015
Accepted 28 January 2016
Available online 10 February 2016

Keywords:
Google cloud messaging (GCM), Web Socket's, real time, scalable, high performance.

ABSTRACT

As mobile technologies evolve, their applications have changed various aspects of human life. Here, we attempt to examine the impact on services for scholars and research and educational organization .Our research is thus designed to utilize wireless mobile communication technologies, Google cloud messaging (GCM) and WebSocket's in an effort to build a high performance event manager in android platform. This research also changes the traditional way in which events take place and to provide real time status to scholars from the organization. Based on the investigation conducted, this paper describes a scalable platform for real time status details between organization and scholars involved in an event. This research also looks into impact of management of scholar detail in organization's database.

INTRODUCTION

Mobile technologies have changed various aspects of human life. In recent time's smart phones have become prominent and a latest survey indicates that it is increasing dramatically. As mobile devices have become increasingly more prevalent, they have made a severe impact on human's day today life. Mobile applications are overtaking PC's. With the rise in mobile technologies and applications it is becoming essential to build a high perform-able and scalable application.

Here, Attempt is made to examine the potential impact of various technologies such as wireless mobile communication, GCM and WebSocket's (Pimentel, V. and B.G. Nickerson, 2012) to build an high performance and scalable event manager for scholar's and research and educational organizations .This research will change the traditional way in which events are advertised and it also provides an real time status update about the events to the registered participant's.

It creates a scalable event manager i.e., the system will be able to process 1000 request's per second without any slow down's or server problem's. The organizer of the event will also be provided with a dashboard with an automatic mailing feature added to it. For this purpose RFID tags used for automated identification of objects and people in moile devices (Juels, A., 2006). High performance is achieved with the help of node js whose performance is very high when compared with others. The main aim of the paper is to provide real time status update to scholars and interaction opportunities between the organizer and the participant's.

METHODS AND MATERIAL

A. Existing work:

In the existing system it is mandatory that there must be a domain name and a good knowledge of web languages to the organizer or he needs to spend some huge amount in order to hire a programmer to advertise

To Cite This Article: Kavitha Subramani, V. Hemapriya, A. Anto Libertina and V.R. Yazhini., Node JS: Building an High Performance Event Manager in Android Platform. ***Aust. J. Basic & Appl. Sci.,*** *10(1): 174-177, 2016*

their event. The existing model include in creating a website and advertising them. Apart from that the communication between the organizer and all the participant of the event will be affected due to agent based monitoring (Ma, K., 2012). There won't be any real time status updating.

B. Proposed work:

Our proposed system of implementation is different from the existing ways.

- The proposed system establishes a supportive advertisement for organization communities. Here, we focus on the simplest and effective way for publishing events
- First, it is not necessary that the organization must have a domain name and programming languages for creating websites regarding the events.
- Second, the burden of hosting the websites and cost involved in advertisement of those websites can be avoided.
- The communication problem between the institution's and scholar's will be avoided with the help of socket io.

C. Google Cloud Messaging (GCM):

Google Cloud Messaging (GCM) for Android is a service that allows to send data from the server to the users' Android-powered device, and also to receive messages from devices on the same connection. The GCM service handles all aspects of queuing of messages and delivery to the target Android application running on the target device, and it is completely free. GCM replaces beta version of C2DM (Android Cloud to Device Messaging). C2DM is deprecated, and new C2DM sign-ups are no longer allowed.

Google-provided GCM Connection Servers take messages from a 3rd-party app server and send these messages to a GCM-enabled client app (the "client app"). Currently Google provides connection servers for HTTP and XMPP.

The 3rd-Party App Server is a component that implements to work with the chosen GCM connection server(s). App servers send messages to a GCM connection server; the connection server enqueues and stores the message, and then sends it to the client app. The Client App is a GCM-enabled client app. To receive GCM messages, this app must register with GCM and get a registration ID. If the client is using the XMPP (CCS) connection server, the client app can send "upstream" messages back to the 3rd-party app server.

D. Algorithm for real time update:

The registration id of the user which is stored in their local database at the time of registration and the length of the event array are the important parameter for the algorithm

Pseudo code:

Step 1: Assign a unique registration id R for the user's at the time of registration.
Step 2: The unique id R is stored in their local database.
Step 3: At the time of registration, the elements of the event array A will be filled with the R.
Step 4: Calculate the length n of the array A.
Step 5: Select the type of status to be updated by the organizer.
Step 6: For n, each R in A send socket io web server send the value of R to GCM server.

RESULTS AND DISCUSSION

A. Scalability and Performance:

On an average Node.js wins hands down. Even though there are few spikes that could be attributed to various disk related anomalies. . The Performance Comparison (Fig. 1) shows that two lines start to intersect towards the end of the test run, while that might start to give the impression that overtime the performance for .NET and node.js converge the reality is .NET starts to suffer even more over time.

Table I: Performance of .Net.

Concurrent Requests	Average Response time (ms)	Requests/second
10	23	422
50	119	416
100	243	408
150	363	411

The response time deteriorates as the number of concurrent requests increases. The response time was 23 ms on average at 10 concurrent requests, and 243 ms on average at 100 concurrent requests. The interesting part is that the average response time has an almost linear correlation to the number of concurrent requests, so that a tenfold increase in concurrent requests leads to a tenfold increase in response time per request. This makes the

number of requests that can be handled per second is pretty constant, regardless of whether we have 10 concurrent requests or 150 concurrent requests. At all observed concurrency level the number of requests served per second was roughly 420.

Table II: Performance of node JS.

Concurrent Requests	Average Response time (ms)	Requests/second
10	19	509
50	109	453
100	196	507
150	294	506

As before the average response time has a linear correlation to the number of concurrent requests, keeping the requests that can be served per second pretty constant. Node.js is roughly 20% faster, e.g. 509 requests/second (TABLE II) vs. 422 requests/second (TABLE I) at ten concurrent requests.

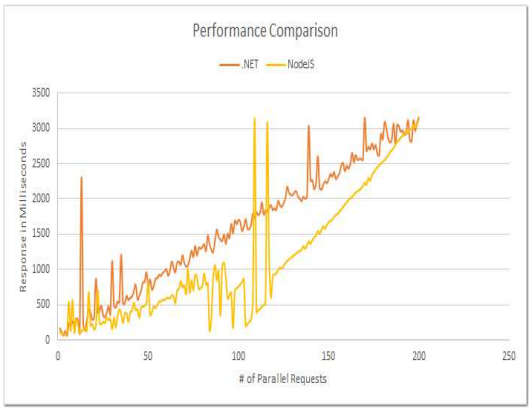

Fig. 1: Performance comparison of .NET and node js.

B. Implementation:

The System Architecture (Fig.2) shows that the Organizer has to enter into web portal in order to create events whose backend is mongo dB (NoSQL) (Li, T., 2012; Cattell, R., 2010). Whenever the organizer tries to login to the portal a jQuery will be executed which will check for the organizer Id and the password with the Collection (similar to table in SQL) called organiser detail Mongo DB.If the entered id and password details are correct he will be allowed to create events.

Once the event has been entered they will be stored in the database (Madden, S.R., 2005) and displayed in the mobile phone, with the help of socket.io. The emit function in socket.io is used to transfer the event details in the database into the mobile phone.

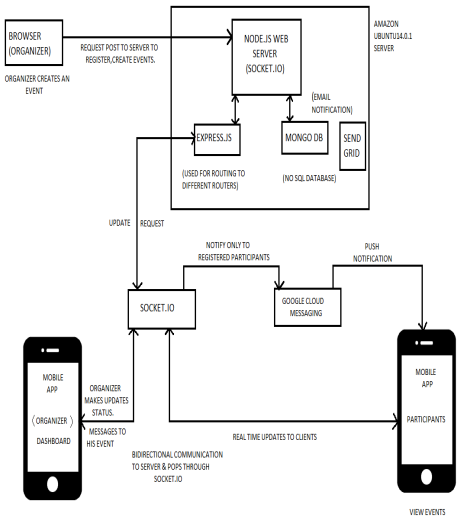

Fig. 2: System Architecture.

Syntax:

Socket. Emit ('Message', JSON. Stringify (data), Call back):

Participants for conference or symposium can register for the event with the help of the app. He can register for N number of events at a single click since users details will also save in his mobile local database.

The details of the registered participants will be stored in the main database called participant detail [8]. Organizer can log in to the mobile app with the help of this id and password which he used for creating events. He will be able to view the list of registered participants, whose mail id, name, and other details will be displayed in the dashboard by fetching it from the main database. Organizer will be able to send a confirmation mail to the registered participants with the help of a send Grid API. The user will be updated in the real time by the organizer regarding the status of the events. The Participant will get a push notification for the updates done by the organizer. Updates like event winners, broadcast messages and event status will be given to participants.

The user will be assigned a unique id at the time of registration. This id will be stored in the user's local database. These details will be stored in an event array, the elements of this array will be the unique id of various user. When a real time update needs to be given, for the length of the array the unique id will be sent to the Google Cloud Messaging (GCM) with the help of socket io.

Conclusion:

The Node.js is 20% faster than the JavaEE solution for the problem at hand. An interpreted language as fast as or faster a compiled language on a VM in which years of optimization have gone into [9]. Not bad at all. It is important to take this with a grain of salt: this type of application is perfectly suited for Node.js. This findings can be extended to other applications, because of the interpreted nature of JavaScript and the lack of established patterns for programming in the large Node.js application are best kept small.

Both Node.js and Java EE scale beyond what a normal server needs. 400-500 requests per second is quite a lot. Google, the largest website in the world, has about 5 billion requests per day. If it is divided by 24 hours, 60 minutes, and 60 seconds it comes out to 57870 requests/ second. That is the number of requests across all Google domains worldwide, so if a website running with 400 requests per second on one machine your website is already pretty big. 1 million requests per day on average means 11.5 requests per second.

REFERENCES

Pimentel, V. and B.G. Nickerson, 2012. "Communicating and displaying real-time data pimentel webSocket,"IEEE Internet Computing, 16(4): 45–53.

Li, T., Y. Liu, Y. Tian, S. Shen, W. Mao, 2012. "A storage solution for massive IoT data based on NoSQL," in Proceedings of IEEE International Conference on Green Computing and Communications, 50–57.

Cattell, R., 2010. "Scalable SQL and NoSQL data stores," SIGMOD Record, 39(4): 12–27.

Juels, A., 2006. "RFID security and privacy: a research survey," IEEE Journal on Selected Areas in Communications, 24(2): 381–394.

Ma, K., R. Sun, A. Abraham, 2012. "Toward a lightweight framework for monitoring public clouds," inProceedings of the 4th International Conference on Computational Aspects of Social Networks, 361–365.

Syme, D., A. Granicz, A. Cisternino, 2012. "Building mobile web applications," in Expert F# 3.0, 391–426 Springer.

Madden, S.R., M.J. Franklin, J.M. Hellerstein, W. Hong, 2005. "TinyDB: an acquisitional query processing system for sensor networks," ACM Transactions on Database Systems, 30(1): 122–173.

Leavitt, N., 2010. "Will NoSQL databases live up to their promise?" Computer, 43(2): 12–14.

Downie, N., 2013. "Chart.js," http://www.chartjs.org/.

"Introducing Socket.IO," 2013, http://socket.io/.

KaLei Inst. of Big Data Technol., Peking Univ., Shenzhen,China Yining Ma ; Zhi Tan"Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js"