To Integrate Heterogeneous devices to provide Astroinformatics Data

[1]Hardeep Singh Kang, Research Scholar, IKGPTU, Jallandhar, India,hardeep_kang41@rediffmail.com
[2]Dr. K S Mann, Prof. & HOD(Department of IT), GNDEC, Ludhiana, India, mannkulwinder@yahoo.com

Abstract - The domain of astronomy is now explored by armatures, hobbyist and professionals alike. One of the requirement is to view, analyses and compute astroinformatic parameters is to access data from device such as smart photos, computing tabs or books. The work implemented here and the outcomes of the performance of the mobile and android TV app shows that rendering of the large size astroinformatics data is not much a issues. The 472 second performance profile recorded shows that the loading time for all these five files is around 307.4 milliseconds and rendering time 1700 milliseconds. Which means that for each fits file the average loading time is about 60 milliseconds and 340 milliseconds is the rendering time. The scripting tasks took maximum time (average 12960 milliseconds) for all files to be processed on the client side. It should be noted that the series of function calls on the Astro Server Engine does not reveal much difference in the performance of the API while rendering on android TV or android device.

Keywords- Astronomy, Data Integration, Interoperability, API, FITS.

I. INTRODUCTION

Sincere, efforts are underway to build, raise and encourage interoperable[1][2][3] frameworks to connect astrophysics data repositories with smart phone devices. All this, due to the fact that mankind inherently has such level of curiosity that it's cannot stop itself from such endeavors. Discovering new bytes of useful information on how universe works seems to be incredible experience for most the people. Astronomy has potential to give thrill of scientific discovery even if one is no scientist[4]. As a collage or school student one can take deep dive into astrophysics experiments and explorations. This has been possible to availability of large public digital[5] archives of the astronomy scans of the sky. Due the availability of such datasets the person can expand his scope of curiosity to higher levels and even explore fields such as space weather[6][7][8], Geospace[9] sciences and astrophysics instrumentation[10] etc. The developments in machine learning and data mining[11][12]is helping tremendously in context of astronomy experimentations[13] and simulations.

The unintentional consequence of the motivation to connect with astrophysics comes from the progress and availability of computing hardware. Today, mobile devices[14] have enough capabilities to work in domain of Astrophysics and Astroinformatics [15][16][17][18] analysis. The algorithms can be parallelized[19] and can render images with great quality in terms of resolution in mobile devices also. Client based frameworks makes the connecting links better now. Standards such as json and xml also help in maintaining the speed of progress in this context. The next section gives a review of the initiatives in this context.

II. LITERATURE REVIEW

Normally, the purpose of the typical organizations working in the area of astronomy is to collect data in electronic form, improve the critical evaluation and combination of the astronomy data, disburse results to the international communities and conduct research using the data. Currently, many frameworks of java scripts[20][21] are making news in the field of astrophysics for providing flexibility and easy to build interoperable applications. This section gives insights from the contemporary literature in this context. The field of astronomy work is normally divided into identification and cataloging the astronomical objects and their measurements properties. The catalogues[22] may consist of the data in form of tables, observational logs, surveys and associated data based on queries. All these data forms are visualized using multiple techniques and catalogue cross matches are corroborated. Open data[23][24] initiatives is the norm in astronomy for promoting collaborative and open research. Lot of efforts had been made to make the astronomy data for standardization of formats, conventions and meta tags and description. Recent developments include hierarchical representation[25][26] of large image surveys, cubes, tables and catalogues. The APIs such as Alladin Lite are useful for building and extending astrophysics analysis. Astropy [27] projects is community effort for developing the packages that give power to construct the geometry of celestial objects, create multi-extension FITS files, accessing table data and many more facilities that are required to render, view and analysis the files. The Astropy library has computational facilities for doing cosmological computations, convolution, filtering and visualization of the

 DOI : 10.18231/2454-9150.2018.0102

astrophysics data. The Astropy community also allows integration of many other projects into it main project. Other scientific communities may join in and work to construct interoperable "affiliate" projects. Some of these projects are under planning and many are affiliate projects have released stable versions of their respective projects. AstoML[28], Astroquery[29], Astroscrappy, APLpy[30][31], Gammapy[32], Glueviz[33],Pyregion[34],Spectral cube[35], Synphot [36] are few projects worth mentioning as they are stable and being extensively in use by astrophysics community. There are many other astrophysics libraries that are precursors and overlapping in functionality along with Astropy are Astrophysics and Kapteyn[37]. The functionality of the Astrophysics library is optical astronomy and have tools for porting data /functionality to Astropy and the Kapteyn[37] library's work is focused on computing co-ordinates and transformation of the astrophysics data. With the passage time mobile apps have been ubiquitous[38] and there have been fair share of growth in domain astrophysics in this context. There are large numbers of mobile apps that are educational in nature and some have been released for the public in general from institutions. such as NASA. In fact, these organizations have released interoperable functions in form of APIs[39]. The 'space images' and NASA API [40]is a mobile app that uses images from the NASA repository. The application called moon atlas gives close up of moon surface and topology. The application Google Sky Map[41][42] is an android based app gives annotated views of the present and past night sky. The app Solar Walk 3D is an award winning app that allows users to explore solar system 3 D models. A similar app gives virtual sky walk and is called "Pocket Universe" works on iOS. The 'Galaxy Collider' improves the understanding of the physics by using simulations of the interacting galaxies. Start Chart and Cosmic Watch apps are also very useful.

The survey of these mobile apps show that these applications help in improving are awareness about our cosmic, solar and our sky systems and focus is normal educational in nature. However, this survey found that these applications are getting more responsive with the use of Javascript based frameworks such as "js9"[43][44]. Js9 is JavaScript especially build for astronomy domain and help us to render the large sizes astronomy data files. Using "in app browsers". These in-app browsers are available both in iOS and android and both the case instances of Web Views are made and url can be fed. They have crude navigation buttons and have local storage capabilities.

After conducting this review, following gaps and challenges need to be resolved.

- Responding to the change in scale: Big Data and data rendering on heterogeneous devices such as smartphone devices.
- Limited work has been reported in context of constructing applications that work on smart phone devices with astrophysics data.
- Support for Large astrophysics file such as Fits from mobile apps limited.

III. OBJECTIVE

Analysis of interoperable application programming interface (API) performance of astroinformatics data in terms of server abilities and availability, response timelines, latency at edge zone of the cloud and throughput. Above all, this application should be able to render astronomical images and large files formats such as fits using in-app browsers.

IV. METHODOLOGY FRAMEWORK

S.No	Dataset (Fits files)	Description
1	CAS –A Chandra [44][45]	It is a snapshot of the supernova remnant (structure as result of super nova explosion) in the constellation Cassiopeia and is one of the brightest extrasolar radio source in sky
2	3c273[44][45]	Snapshot /image data of quasar located in constellation Virgo. 3C 273 is visible in the month of May in both the hemispheres.
3	3c58[44][45]	It is pulsar and supernova remnant snapshot within the Milky way.
4	CTB 109 (Einstein)[44][45]	This is also galactic supernova remnant image dataset .
5	Ngc1316 AIP[44][45]	Fourth brightest objects at 1400 Mhzimage.

Table 1: Datasets used in the work

This section explains the steps taken to develop framework for making the astroinformatics data through heterogeneous devices such as smart phones. Secondly, the later part of this section gives performance analysis of the component used in the framework. The Datasets mainly consists of

 DOI : 10.18231/2454-9150.2018.0102

spectrum of the X-ray emission of the various astronomical objects are normally studied for deducing many aspects of our universe and the size of these image shots are large in size . The table [1] shows the list and description of the datasets used in this research work.

The objective of this work required a multiple tools, libraries and plugins. The mobile app development was conducted using ionic framework and Cordova plugins and JavaScript library known as Js9. These JavaScript frameworks also require 'node js' framework as prerequisite. Beside all these tools and library Github was used to maintain the code. The workflow figure [1] of this research based application can be understood in three steps. The creation of this application starts with the creation of the Cordova plugin that uses the libraries from the js9 JavaScript. The next step is integrating this plugin into the ionic framework and writes a client application.

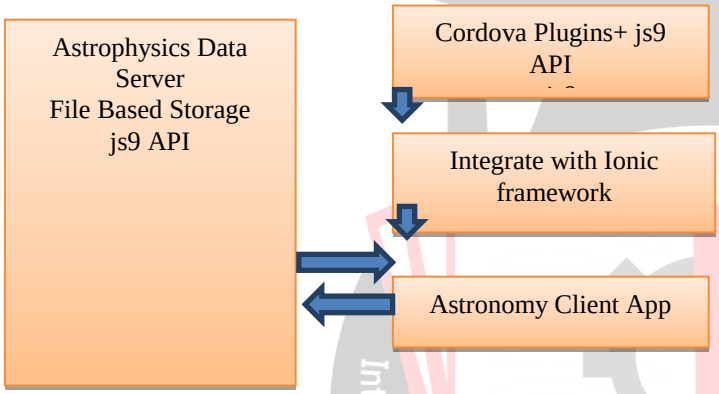

Figure 1: Workflow of the research work

V. FRAMEWORK AND ARCHITECTURE

There are many devices that can now hold data and applications that are based on the popular mobile based operating systems such as android, Microsoft windows, MeeGo, Bada Symbian, black berry and iOS. Most of these operating systems can be made working on the devices that include hand held, TV sets, in-car devices, scientific instruments and net books. But android has maximum installations and coverage on heterogeneous devices. This may be attributed to the fact that android is open source and since the time automation has hit the software industry, the development of frameworks, reusable libraries and API's is name of the game. The open source version of Ionic framework is one the most popular frameworks used for building applications. The architecture and design of the framework can be understood from the [2] diagram. The diagram shows the proposed work the astrophysics application as well.

Figure 2: Architecture of the Mobile Application

A. Building Astrophysics App

The figure [2] shows the process of building the mobile app using the architecture and modules of depicted in Figure [3]. The application constructed uses the client –server communication model for send and receiving data and process information.

Astro-Mobile Client:

(a) After the installation of the ionic framework using the node jsenvironment. The integration of the in-app browser needs to be done. This would enable the users to view the functionality of js9 as web page inside the android application.

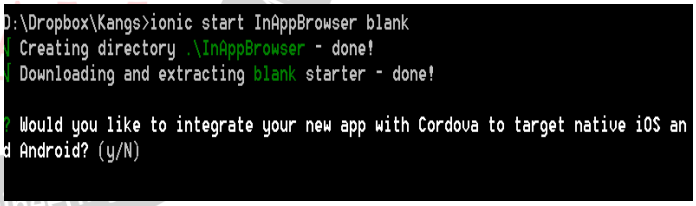

Figure 3: Installation of ionic framework and dependencies

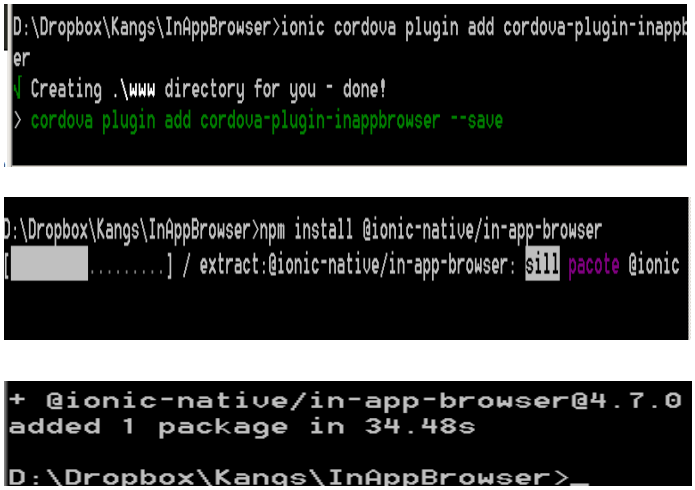

Figure 4: Incorporation of In-App browser plugin

 DOI : 10.18231/2454-9150.2018.0102

(b) The second step is to enable JavaScript so that js9 JavaScript library can be used and invoked.

Astro-Engine Server:

The Astro Engine server was configured using virtual machines on the digital ocean platform. After doing the trivial tasks for installing/configuring Linux (Ubuntu 14) and web server (apache2). The installation of the js9 was initiated.

```
D:\Dropbox\Kangs>git clone https://github.com/ericmandel/js9
```

Figure 5: Command to install and clone js9 in Astro Engine Server

```
C:\Bitnami\wampstack-5.5.29-1\apache2\htdocs\js9>ls
APICHANGES          js9.min.js          js9multi.html
LICENSE             js9.scpt            js9onchange.html
Makefile.in         js9Electron.js      js9panzoom.html
README.md           js9ElectronPreload.js  js9plugins.html
```

Figure 6: Installed JS9 framework and server application in Astro Engine Server

For maintaining the reproducibility of the research work source code is maintained at Github repository [https://github.com/hardeepkang/astroandroid].The code listing below shows 1) How NAS API is called to render astronomy images and code listing (2) gives the code of function used for calling the html page having JavaScript capabilities for handling large astronomy files.

```
export class HomePage {
  url: string;

  constructor(private inAppBrowser: InAppBrowser) { }

  openMobileAstro(url: string) {
    const options: InAppBrowserOptions = {
      zoom: 'no',
      enableViewportScale:'yes',
      toolbar:'yes',
      footer:'yes',
      location:'yes',
      presentationstyle:'pagesheet'
    }

    // Opening a URL and returning an InAppBrowserObject
    url = 'https://[redacted]/js9bitpix.html';
    const browser = this.inAppBrowser.create(url, '_self', options);
    browser.show();
    browser.close();
   // Inject scripts, css and more with browser.X
  }
}
```

Figure 7: Code Listing 1 of JS9 API Functional Calls

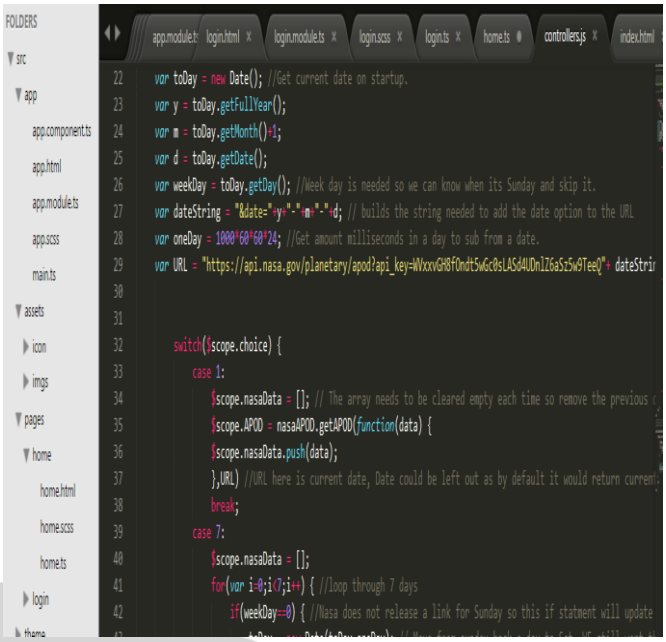

Figure 8: Code Listing 2 of NASA API Functional Call

Output : The Figure [8] show the screen shot of the mobile app view of the home page and the JS9 library capabilities. The mobile view has the zoom, scale, color change and file (fits) uploading facilities.

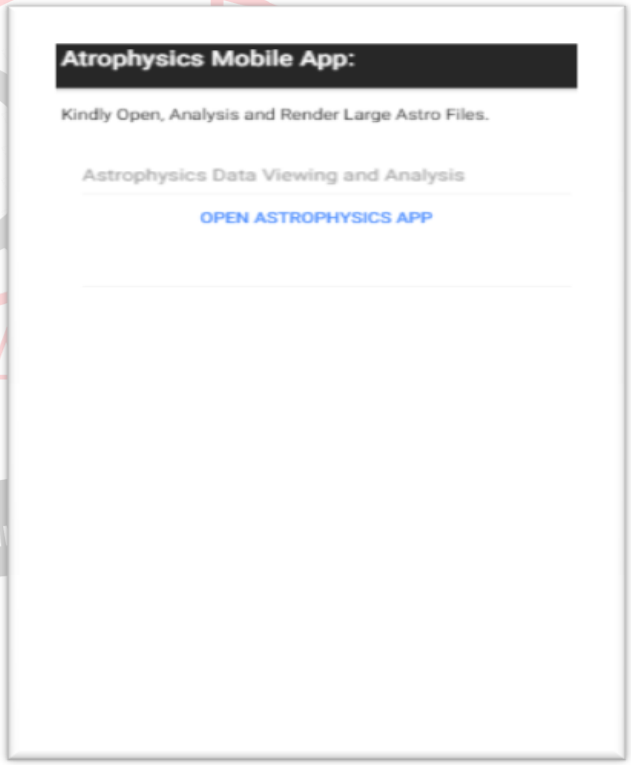

Figure 9: Load View of Mobile App main page

 DOI : 10.18231/2454-9150.2018.0102

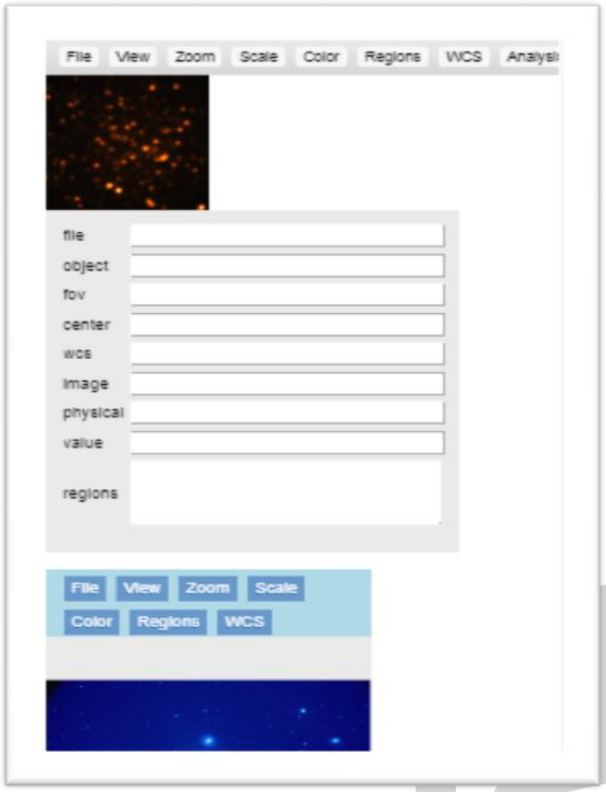

Figure 10: Mobile view of the Astro Client App

VI. RESULTS AND DISCUSSIONS

The performance of the mobile app needs to evaluate for it to become a success story. All though care has been taken to maximize the performance of the application. To keep the performance of the application high the unused variables and objects were not kept in the memory unnecessarily. The local memory has limitation in the quantity of data it can handle and it may not work well with browsers also. To overcome this challenges the astronomical files were stored at the Astro-Server and the client rendered the file data for analysis using a JavaScript client embedded in the mobile app framework. The hammering of the network with repeated requests was minimized with model view control pattern of the application. The use of CSS sprites was also minimized and application gives a plain look. This was done for enhancing the performance. The evaluation of the application was done based on the following parameters.

A. Evaluation Parameters

The performance of the mobile app basically depends upon three aspects i.e Device Performance, Server/API performance and Network performance. In context of our work we are considering the performance with respect to the Server/API communication in rendering the Astrophysics files.

S.NO	PERFORMANCE PARAMETER	DESCRIPTION
1	LOADING TIME	THE TOTAL TIME IN MILLISECONDS FOR PAGE TO LOAD IN THE DEVICE , WHEN A REQUEST IS SEND TO THE ASTROPHYSICS SERVER .
2	SCRIPT EVALUATION TIME	SINCE, THE CLIENT APPLICATION WORKS PRIMARILY BASED ON THE JAVASCRIPT , THIS PARAMETERS CHECKS THE RESPONSIVENESS OF THE JAVASCRIPT FUNCTION CALLS VIA API
3	RENDERING TIME	THIS IS TIME TAKEN BY THE PAGE TO FINALLY SHOW UP WITH ITS CONFIGURED RESOLUTION ON THE DEVICE THAT MAY BE ANDROID TV OR ANDROID MOBILE APP.
4	PAINTING TIME	TOTAL TIME TAKEN BY THE IN-APP BROWSER TO TRAVERSE THROUGH ALL THE VISUAL ELEMENTS IN ORDER TO DISPLAY ON THE SCREEN.

Table 2: Performance Parameters

A. Performance Measurement Tools

The Chrome DevTools and testing tools were used for measuring the performance of the in-app browser calls on the Astro engine server. The figure [8] shows the outcome of the performance profile recording.

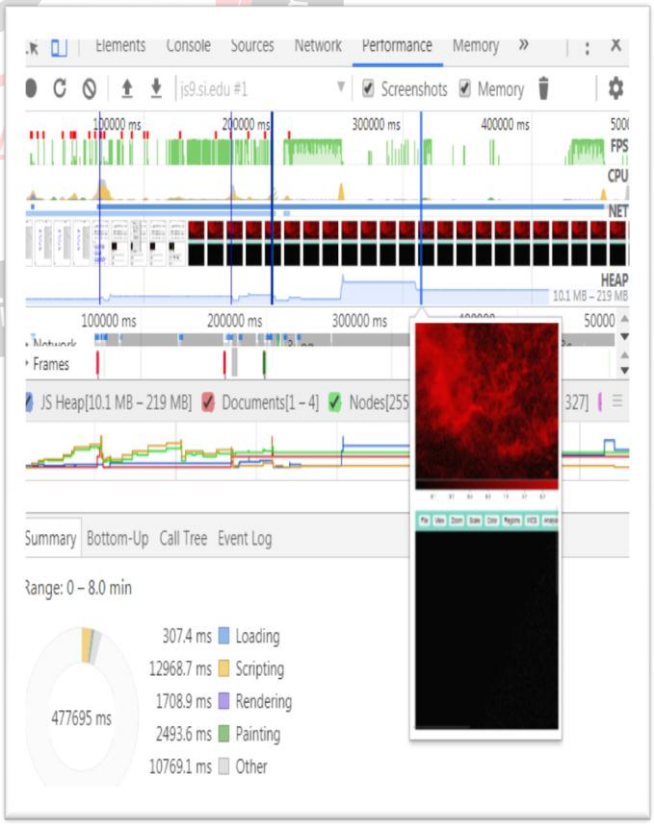

Figure 11: Timeline (472 ms) and Graphs to check performance.

 DOI : 10.18231/2454-9150.2018.0102

The interaction between the app and the server/API has been measured using the parameters [Table 2]. These parameters checks how efficiently the data is communicated to the device with acceptable response time. Each aspect of the process is measured, which include "how much time does the JavaScript takes to process and how much time does the page on mobile takes to load and finally get rendered. For this performance checking tool made a comprehensive profile of all the tasks that were initiated from the client machine. The full cycle of the CAS-A , CTB 10, m13 , 3c58 , Ngc1316 AIP from loading to rendering was done. The performance of the app is clear encouraging. The 472 second profiles shows that the loading time for all these five files is around 307.4 milliseconds and rendering time 1700 milliseconds . The scripting task takes the maximum time (average 12968 ms) for all files to be processed on the client side. It should be noted that the series of function calls on the Astro Server Engine does not reveal much difference in the performance of the API, while rendering on android tv or android device. For easy interpretation of the outcomes of the performance analysis the table [3] and Figure [11] should studied .

S.No	Tasks	Value(s) in milliseconds
1	Loading	307.4
2	**Scripting**	**12968.7**
3	Rendering	1708.9
4	Painting	2493.6
5	Other	10769.1
6		

Table 3: Analysis Task Running time

From these values, it is clear loading is taking minimum time . This is attributed to the mechanism of lazy –loading incorporated in the App.

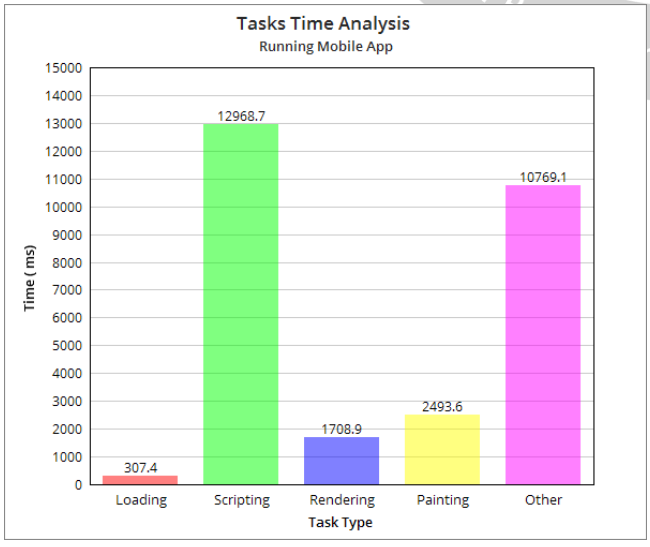

Figure 12: Analysis Task Running time

Running Mobile App to render astroinformatics data requires multiple tasks .The Figure [12] shows the performance of the application per task .

VII. CONCLUSION

From this work, it is amply clear that if JavaScript frameworks are used, integrating with the mobile based application is not hard. Facebook and Twitter also use the concept of in-app browsers for rendering their content. Both browsers provide basic navigation buttons and other capabilities. This concept is also useful in our context of research work. That is, it helps to brings large astronomy files rendering in mobile apps easy. But , all this was achievable only when a proper strategy was followed to minimize the initial loading of the objects in the main pages of the App. The initial screen was loaded and JS9 files and other JavaScript frameworks were loaded later on. Then , the caching mechanism was also used to keep track of the last retrieved data or state of the App page .JS optimization and minifying was also done to achieve good performance for rendering .For this purpose YUI optimizing tool (http://yui.github.io/yuicompressor/) was applied to the JS files . The parameter "Keep Alive" was also fine-tuned to support better response when image data changes continuously on the screen . Other than these steps, the optimization of User interface in terms of repaint/redraw operations was defined meticulously so as to avoid rendering of unnecessary CSS objects etc. In fact, the minimum CSS objects were used to compose the layout of the App pages as per the JS9 requirements.

By following all these steps the biggest obstacle considered to solve these issues were removed to achieve good responsiveness and quality of resolutions. Similar issues were faced when the video streaming got ignited. This work shows that it is not hard to integrate APIs with mobile /TV apps with concept of in-app browsers. This can be attributed to the statistical outcomes that were recorded during the testing session of the mobile application. The statistical results show that the maximum time consumed by the App is in loading and executing the Java script framework and they need to optimized and minimized for it to render in high performing timeline .

VIII. FUTURE DIRECTIONS

The work can be further extended to develop more applications in astrophysics domain and incorporate more plugins from js9 to support advance analysis of the astronomic datasets. Secondly, the current research work using js9 mobile app framework can be further enriched with specific language (python and C) bindings. This work can also be extended for building Android TV based apps, which can make the experience of viewing the astrophysical image dataset rich and it can reach another section of amateurs or curious people.

 DOI : 10.18231/2454-9150.2018.0102

REFERENCES

[1] B. Resch and M. Mittlboeck, "Live geography - Interoperable geo-sensor webs enabling portability in monitoring applications," in 2nd International Conference on Advanced Geographic Information Systems, Applications, and Services, GEOProcessing 2010, 2010, pp. 74–79.

[2] a. B. M. Russel, A. I. Khan, and D. Abramson, "Interoperable grid agent middleware for scientific instruments and sensors," Proc. 4th Int. Work. Middlew. grid Comput. - MCG '06, p. 21, 2006.

[3] H. Panetto, M. Zdravkovic, R. Jardim-Goncalves, D. Romero, J. Cecil, and I. Mezgár, "New perspectives for the future interoperable enterprise systems," Comput. Ind., vol. 79, pp. 47–63, 2016.

[4] P. J. Marshall, C. J. Lintott, and L. N. Fletcher, "Ideas for Citizen Science in Astronomy," Annu. Rev. Astron. Astrophys., vol. 53, no. 1, pp. 247–278, 2015.

[5] A. Govada and S. K. Sahay, "A Communication Efficient and Scalable Distributed Data Mining for the Astronomical Data," pp. 1–20, 2016.

[6] W. B. Cade, "The first space weather prediction," Sp. Weather, vol. 11, no. 6, pp. 330–332, 2013.

[7] J. C. Green, J. Likar, and Y. Shprits, "Impact of space weather on the satellite industry," Sp. Weather, vol. 15, no. 6, pp. 804–818, 2017.

[8] A. Bhardwaj, T. K. Pant, R. K. Choudhary, D. Nandy, and P. K. Manoharan, "Space Weather Research: Indian perspective," Space Weather, vol. 14, no. 12, pp. 1082–1094, 2016.

[9] R. M. McGranaghan, A. Bhatt, T. Matsuo, A. J. Mannucci, J. L. Semeter, and S. Datta-Barua, "Ushering in a New Frontier in Geospace Through Data Science," Journal of Geophysical Research: Space Physics, vol. 122, no. 12. pp. 12586–12590, 2017.

[10] J. W. Mitchell and T. Hams, "Astrophysics and space instrumentation," in Handbook of Particle Detection and Imaging, 2012, p. 56.

[11] Y. Zhang and Y. Zhao, "Astronomy in the Big Data Era," Data Sci. J., vol. 14, no. 0, p. 11, 2015.

[12] N. M. BALL and R. J. BRUNNER, "Data Mining and Machine Learning in Astronomy," Int. J. Mod. Phys. D, vol. 19, no. 7, pp. 1049–1106, 2010.

[13] R. Schilizzi, "Experimental Astronomy," Experimental Astronomy, vol. 17, no. 1–3, p. 1, 2004.

[14] B. Ryan, "Developing library websites optimized for mobile devices," Ref. Libr., vol. 52, no. 1, pp. 128–135, 2011.

[15] E. D. Feigelson, "The changing landscape of astrostatistics and astroinformatics," in Proceedings of the International Astronomical Union, 2016, vol. 12, no. S325, pp. 3–9.

[16] R. Vilalta, K. D. Gupta, and A. Mahabal, "Star classification under data variability: An emerging challenge in astroinformatics," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2015, vol. 9286, pp. 241–244.

[17] O. Laurino, R. D'Abrusco, G. Longo, and G. Riccio, "Astroinformatics of galaxies and quasars: A new general method for photometric redshifts estimation," Mon. Not. R. Astron. Soc., vol. 418, no. 4, pp. 2165–2195, 2011.

[18] P. Škoda, "Astroinformatics: Getting New Knowledge from the Astronomical Data Avalanche," Adv. Intell. Syst. Comput., vol. 210, p. 15, 2013.

[19] S. C. Muller, G. Alonso, and A. Csillaghy, "Scaling astroinformatics: Python + automatic parallelization," Computer (Long. Beach. Calif)., vol. 47, no. 9, pp. 41–47, 2014.

[20] A. Finkbeiner, "Astronomy: Planets in chaos," Nature, vol. 511, no. 7507, pp. 22–24, 2014.

[21] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to build high-performance network programs," IEEE Internet Comput., vol. 14, no. 6, pp. 80–83, 2010.

[22] R. Bose, R. G. Mann, and D. Prina-Ricotti, "AstroDAS: Sharing Assertions across Astronomy Catalogues through Distributed Annotation," Proven. Annot. data Int. Proven. Annot. Work. IPAW 2006, pp. 193–202, 2006.

[23] I. V. Pasquetto, A. E. Sands, P. T. Darch, and C. L. Borgman, "Open Data in Scientific Settings," in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16, 2016, pp. 1585–1596.

[24] H. M. Krumholz and J. Waldstreicher, "The Yale Open Data Access (YODA) Project — A Mechanism for Data Sharing," N. Engl. J. Med., vol. 375, no. 5, pp. 403–405, 2016.

[25] P. Greenfield, M. Droettboom, and E. Bray, "ASDF: A new data format for astronomy," Astron. Comput., vol. 12, pp. 240–251, 2015.

[26] T. Jenness, "Reimplementing the Hierarchical Data System using HDF5," Astron. Comput., vol. 12, pp. 221–228, 2015.

[27] Astropy Collaboration, "Astropy: A community Python package for astronomy," Astron. Astrophys., vol. 558, p. A33, 2013.

 DOI : 10.18231/2454-9150.2018.0102

[28] J. Vanderplas, A. J. Connolly, Z. Ivezic, and A. Gray, "Introduction to astroML: Machine learning for astrophysics," in Proceedings - 2012 Conference on Intelligent Data Understanding, CIDU 2012, 2012, pp. 47–54.

[29] B. Sipocz, "Astroquery: querying astronomical web forms and databases," in Python in Astronomy 2016, 2016.

[30] T. Robitaille and E. Bressert, "APLpy: Astronomical Plotting Library in Python," Astrophys. Source Code Libr., 2012.

[31] G. B. Berriman and J. C. Good, "The application of the montage image mosaic engine to the visualization of astronomical images," Publ. Astron. Soc. Pacific, vol. 129, no. 975, p. 58006, 2017.

[32] A. Donath et al., "Gammapy - A python package for γ-ray astronomy," in Proceedings of Science, 2015, vol. 30-NaN-2015.

[33] A. A. Goodman, "What is WorldWide Telescope, and Why Should Researchers Care?," in American Astronomical Society Meeting Abstracts, 2016, vol. 227.

[34] Astopy, "pyregion." 2018.

[35] Astopy, "spectral-cube." 2018.

[36] Astopy, "synphot." 2018.

[37] kapteyn, "kapteyn." 2018.

[38] A. Accomazzi, "Astronomy 3.0 Style," in Astronomical Data Analysis Software and Systems XIX - ASP Conference Series, 2010, pp. 1–9.

[39] A. A. Sawant and A. Bacchelli, "A dataset for API usage," in IEEE International Working Conference on Mining Software Repositories, 2015, vol. 2015–August, pp. 506–509.

[40] NASA, "https://api.nasa.gov/api.html." 2018.

[41] H. Ouilhet, "Google Sky Map : Using your Phone as an Interface," Search, pp. 419–421, 2010.

[42] T. S. Metcalfe, "Crowdfunding Astronomy Research With Google Sky," J. Astron. Earth Sci. Educ., vol. 2, no. 2, p. 109, 2015.

[43] X. Zhou et al., "Visualization technology and its application for massive astronomical data analyses," in Proceedings - 8th International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2015, 2016, pp. 105–108.

[44] E. Bertin, R. Pillay, and C. Marmo, "Web-based visualization of very large scientific astronomy imagery," Astron. Comput., vol. 10, pp. 43–53, 2015.

[45] chandra.harvard.edu/photo/openFITS/, Ed., "chandra.harvard.edu/photo/openFITS/." 2015.

 DOI : 10.18231/2454-9150.2018.0102