Performance Optimization for Mobile Web

Gunjan, Bharti Sharma, Deepak Pathania, Ankita Dhillon

Abstract*: Abstract Most of the existing web pages have poor visibility, high bounce rates, and low conversion rates due to performance issues on mobile devices. The aim of this research work is to develop a framework which optimizes the performance of mobile web. The primary technology used in the framework is Google's AMP (Accelerated Mobile Pages). The proposed framework converts a non-AMP web page into an AMP compliant web page, with additional feature of internationalization, and caches it on cloud for future reference. The proposed framework is tested on various content rich sites and performance of converted AMP pages is com- pared with original pages based on various parameters likes first meaningful paint, first interactive, etc.

Keywords: Mobile Web, Performance Optimization, Accelerated Mobile Pages, AMP convertor.

I. INTRODUCTION

Accelerated Mobile Pages (AMP) [8] is a Google-backed project designed as an open standard to optimize the performance of content based web pages. The project allows developers to create elegant and performance optimized web pages by following a set of guidelines issued by the standard. AMP pages are also given preference in search rankings and are displayed in a separate carousel on top of normal search results, thus helping in both discovery and retention.

Publishers have to modify their websites as per the compliance rules specified in AMP which can be a blocker in the mass adoption of this open standard. This research highlights the benefits of AMP as an open standard and provides a method of converting existing web pages into AMP compliant ones with an easy to use interface, thus reducing the friction in adoption of the standard.

II. LITERATURE SURVEY

KLOTSKI [5] gives an overview of how the mobile web is getting more and more complex and bounce rates are higher and there is a growing need for a more specific strategy catering to the same. To maximize web page speed, many optimization techniques have been used in literature mostly dealing with reducing content size, caching, etc. [6]

Minification: Code styling rules ensure that the code looks well styled to enhance developer experience but serving the same good looking code to end user can lead to performance issues. Minification results in performance optimization through the removal of unnecessary whitespaces and other descriptors that do not affect the code flow directly. It is generally followed by the project implementation to deliver a better experience to the end user.

Image Transcoding: Images have always been performance killers, especially in the mobile format but with the introduction of image formats like WebP, smaller images can be created which leads to faster web pages. **HTML5** < *picture* > **element**: Conditions can be specified using the new < *picture* > tag to direct the browser to download only the image that matches the criteria, rather than downloading the entire image set and then applying resizing rules on top of it.

Gzip compression: gzip is a well-known compression technique that allows us to not only reduces the size of web pages but also the style sheets and scripts associated with it, prior to being served on the browser.

Data URLs: Data URLs removed the dependence on external paths for linking images, which is an additional network call. Images or other small documents can be encoded in base64 format and used directly in- side HTML documents to access an external resource. [3]

Caching: Caching entire pages is not a feasible option in a dynamic environment because of the level of uncertainty in what an end user might request but certain common resources can be cached to save additional network calls which leads to a smoother experience for returning users.

On demand data loading: On demand data loading allows data to be loaded in response to a certain trigger rather than loading all the data from the beginning which may lead to longer response times. In case of pagination, data related to a page currently not visible to the user should be fetched only if the user decides to visit that page. Similarly data in modals should be fetched in response to the opening of that modal. This saves extra overhead in the initial network call which leads to better first paint time.

Revised Manuscript Received on November 06, 2019.

Gunjan, CSE department, Name Maharaja Surajmal Institute of Technology, New Delhi, India. Email: gunjanbeniwal@msit.in

Bharti Sharma, CSE department, Name Maharaja Surajmal Institute of Technology, New Delhi, India. Email: bhartisharma@msit.in

Deepak Pathania, CSE department, Name Maharaja Surajmal Institute of Technology, New Delhi,India. Email: deepakpathania789@gmail.com

Ankita Dhillon, CSE department, Name Maharaja Surajmal Institute of Technology, New Delhi,India. Email: ankita.miamor@gmail.com

Pre-connect and Pre-fetch: Resolving domain names before a user actually follows a link can lead to smaller response times. Proxy can predict to a certain extent what a user might request next based on the behavioral aspect and association of resources which allows it to make smart decisions about what to pre-fetch based on the user behavior. Connection establishes delay can also be performance killers which can be avoided using pre-connect to eliminate round-trip latency. [4]

CDN: Content Delivery Network replicates the content delivery server at multiple locations for static assets. When an asset is requested, the source of request is determined based on which it is served by the closest server replicant. [7]

Apart from above discussed techniques, optimization can also be done in the software design phase itself which includes minimizing HTTP requests, replacing JavaScript by CSS as much as possible, loading JavaScript wisely, converting table layout to CSS layout, avoiding inline styles, split components across domains, minimizing the number of iframes, avoiding redirects, using responsive designs, minimize DOM access, etc.

III. ACCELERATED MOBILE PAGES

AMP basically creates the page layout without downloading all the resources based on certain size computations it does. It specifies that any asset that consumes some space on the page, say image, video, advertisement, etc. must have its height and width property set as attributes. This allows AMP to pre-compute the space they are going to occupy on the page.

AMP thus gives preference to the content on the page and loads it first by creating the placeholders of the specified dimensions for these assets. These placeholders can be later populated by the actual content and in the meantime, the user can consume the content they came for.

This benefits the publishers as well as users since if the user stays for longer by getting the content early, the chances of monetization are also higher for the publisher compared to when the user immediately leaves when the content takes too long to load. AMP also ensures that their content is easily crawled, indexed, displayed and easily viewed.

AMP takes some drastic measures to ensure performance of a webpage is not compromised. It forbids any external resources like linked style sheets & scripts, and limits the styling options to an inline style block, not exceeding 50 Kb. This can be a blocker for certain publishers who rely heavily on styled pages and heavy animations; however to get good performance these should be avoided in general.

AMP also pre renders certain content based on the likelihood of the content being requested as a background process, while employing an asynchronous approach to ensure that there are no blocking resources that hinder the rendering of the entire page. Google also provides a caching service to AMP compliant web pages so that a cached copy can be served to users instantaneously while also ensuring it is updated properly.

All these factors together help achieve AMP its remark- able performance and are in general good web practices.

A. AMP COMPONENTS

AMP has three basic components:

AMP HTML

AMP HTML is both a subset and a superset of HTML. It uses most of the HTML tags with certain modifications to ensure performance is given preference in the structure layout as well. Some components like image, audio are prefixed with 'amp-' to specify custom components that perform better than their normal counterparts without any computational modifications. AMP pages are linked to their counterparts with a < *linkrel* = "*amphtml*" > tag so that you can have both non-AMP and AMP versions of the page.

AMP JS

The AMP JS library is a black box that is not directly exposed to the developers but provides all of the performance benefits without any additional efforts on the developer's end. It implements asynchronous rendering to ensure no resource is blocking in nature and pre-calculates the page layout based on the size attributes specified in the AMP tags.

AMP CACHE

The AMP cache is an added incentive for developers to switch to AMP which delivers validated AMP documents through Google's cache server almost instantaneously. It enforces a strict same origin policy for serving all assets for maximum efficiency. It also auto validates the pages it delivers ensuring that the page being served doesn't have any external dependencies and works properly. A version of this validator is also available to every page locally to get validation errors in the browser console.

B. AMP RULES

The AMP rules are listed in Fig. 1.

C. AMP RESTRICTIONS

- Custom scripting, internal or external is not allowed.
- User input tags like input or text-area are not al- lowed.
- Style-sheets linked to the document using link tag are not allowed

- Start with the doctype <doctype html>.
- Contain a top-level <html> tag (<html amp> is accepted as well).
- Contain <head> and <body> tags (They are optional in HTML).
- Contain a link rel="canonical" href="$some_url"> tag inside their head that points to the regular HTML version of the AMP HTML document or to itself if no such HTML version exists.
  Contain a <meta charset="utf-8"> tag as the first child of their head tag.
- Contain a <meta name="viewport" content="width=device-width,minimum-scale=1"> tag inside their head tag. It is also recommended to include initial-scale=1.
- Contain a <script async src="https://cdn.ampproject.org/v0.js"></script> tag inside their head tag.
- Contain the AMP boilerplate code (head> style[amp-boilerplate] and noscript> style[amp-boilerplate]) in their head tag.

Fig. 1. AMP Rules

- Inline styles are not allowed.
- Single style tag with style rules not exceeding 50kb.
- HTML tag restrictions are shown in Table 1.

Table-I: AMP HTML Restrictions

HTML TAG	PAGE IN AMP HTML
script	Prohibited. Exception is the tag using type application/ld+json, mandatory script in for AMP runtime and script tags for extended components.
noscript	Allowed. The content will display when JavaScript is disabled by user.
form	Allowed. Must be replaced by amp-form tag.
input elements	Allowed. Exceptions are input types: image, button, password, file.
style	Allowed in head tag for custom styling. This style tag must containing attribute amp-custom.
link	Some values that can influence the browser behavior are not allowed, such as: preconnect, prerender, ...
meta	Allowed. Except http-equiv attribute.
a	Allowed if not prefixed with javascript:.

IV. PROPOSED WORK

The major motivation in this work is to develop a tool to convert a non-AMP compliant web page to an AMP compliant web page. This is done by dynamically applying the rules of AMP on an existing website to generate the AMP compliant HTML of it which can be later automated to create AMP pages on the fly for any website.

The methodology is not restricted to any technology stack used to build the website since it is based on the DOM selectors that are finally generated, thus giving the flexibility of avoiding any complex build processes to generate the required page. The back end can be accompanied by either a chrome extension or a visual iframe based dashboard that allows users to select the elements they would like to be present on the generated page, thus allowing us to work with dynamically loaded content as well.

The project has four main components:

- A core AMP converter that takes URL of a webpage along with some configuration options and returns AMP compliant HTML for it.
- A REST API that exposes the core module through an endpoint.
- A cluster to store and retrieve all converted data in the cloud.
- A visual interface that allows users to interact with the REST API.

A. *AMP converter*

The AMP converter (Fig. 2) is a standalone node module that provides a convert function which takes a URL and options object as arguments to return the AMP compliant HTML for that URL as per the specified options.The user selects the elements he wants to be avail- able in the AMP-compliant version. The project has four main components:

- A core AMP converter that takes URL of a webpage along with some configuration options and returns AMP compliant HTML for it.
- A REST API that exposes the core module through an endpoint.
- A cluster to store and retrieve all converted data in the cloud.
- A visual interface that allows users to interact with the REST API.

B. *AMP converter*

The AMP converter (Fig. 2) is a standalone node module that provides a convert function which takes a URL and options object as arguments to return the AMP compliant HTML for that URL as per the specified options.

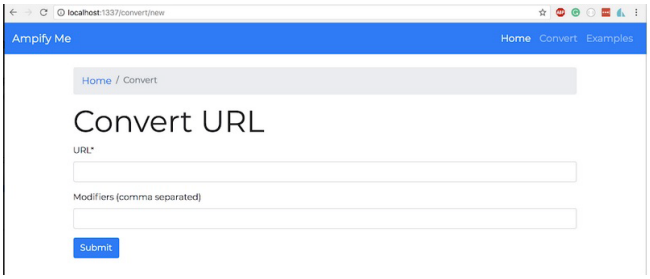

Fig. 2. AMP converter

1. The user selects the elements he wants to be avail- able in the AMP-compliant version.
2. The CSS selectors of the selected elements are collected to prepare an options object.
3. The options object along with the URL of the page to be converted is sent to the server where the selectors are extracted from options to get the content of the web page from the specified URL by targeted scraping.
4. A template is generated

with basic AMP-compliance features which are populated by the scraped content after sanitizing through steps 5-9.
5. The script tags are removed from the scrapped version to ensure no network calls block the content.
6. Meta tags present in the original site are added to the modified content to maintain integrity.
7. Custom elements that are not supported in the AMP documentation are looked for and removed if present; polyfills may be added in the future.
8. Assets are fetched separately and made compliant to the AMP rules. All external stylesheets are discarded and only the configurable boilerplate style stays in effect, stylesheets can be fetched, minified and inlined if they do not violate the 50 kb limit imposed by AMP for style.
9. The scrapped content populates the initially created template to add the necessary content.
10. Tags are renamed according to the compliance policies, for e.g. ”img” tag changes to ”amp-img” .
11. Content is translated to regional languages and stored on the server to serve that in case a search query comes in that particular language on a mobile device.
12. A md5 hash is created for each converted file which acts as a unique identifier for specifying the AMP URL on each page.
13. Modified compliant HTML is then returned which is written to a file if it doesn't already exist.

C. REST API

Sails is used to implement the converter, which comes with built in support for API generation using Blueprints. Sails command line tools are used to set up the basic structure.

1. Sails is installed globally using npm.
2. A new sails app is created using the globally installed sails cli.
3. Necessary configurations are provided to expose the REST endpoints.
4. New API is created using the cli tool which in turn created a controller and a model.
5. Necessary attributes are specified in the generated model along with custom model methods.
6. In the controller, handlers corresponding to different endpoints are specified, each handler by default handles the common HTTP verbs like create, up- date and destroy which can be overridden.
7. Handlers for convert and example endpoints are added.
8. When the convert endpoint is hit with the required data, it calls the AMP converter with the received data which returns the AMP compliant HTML.

D. *Data Cluster*

For data storage and retrieval, Hasura is used to create clusters online, which were then populated through the exposed REST API. This allows access of the converted pages from anywhere and a unique URL is generated to access them for providing canonical URLs of the web page.

1. Install the hasura cli to create a base hasura project locally and login to the account.
2. Set up a cluster of nodes (VMs) on the cloud that is required for hosting and deployment.
3. The File API is used to store the file that is created and written locally from the AMP compliant HTML that the convert function receives from the converter.
4. For reducing redundancy, an in memory check is provided to ensure that the file is written only if it doesn't already exist in that state, by matching the generated hash.
5. Data API is used to retrieve a list of ids of all the converted files which can then be used to access the file at any given moment.
6. The list of ids fetched from the data API is sent to the view layer where all the contents are displayed.

E. *Interface*

Embedded javascript templates are used for the view layer. The data from the controllers is passed to the view layer and similarly user input is passed from the view layer to the corresponding controller.

1. Sticking to the MVC architecture provided by sails, views are used to provide an interface to the user to interact with.
2. The landing page simply specifies the two CTAs which include converting a new page and seeing examples of the already converted ones.
3. The converted page leads to a form that accepts the user input in form of the URL to be converted and the options list.
4. The example interface fetches and shows the ids of the already converted ones which can be used to see the converted pages.
5. The interface follows the blueprint architecture such that all functions in a controller act as unique endpoints.

F. *Integration*

Here is how all the components talk to each other in a normal flow.

1. The user comes to the interface and enters the URL to be converted.
2. The form data is sent to the REST API controller.
3. After sanitization, the controller passes on the data to the converter.
4. The converter returns AMP compliant HTML which is written to a file locally after hashing.
5. The file is stored in the online cluster if it doesn't exists already.
6. When the user visits the example page, the same flow happens but with a different endpoint and thus a different controller which simply fetched the ids of all converted pages.

G. *Processing*

The returned AMP compliant HTML can serve as the 'amphtml' page whereas the original page serves as the canonical version. Both of these are required so that the search engine knows that if an AMP version of a page exists and the user agent trying to access it is a mobile device then the 'amphtml' version of the page should be loaded rather than the canonical one.

A regex matching automation system can be employed to generate the canonical version for all pages having the same DOM structure, making it ideal for content rich websites which generally use the same template for publishing articles providing a consistent DOM structure.

The User Agent can be extracted for the internationalisation functionalities and thus pages can also be served in regional languages if a query is received in that particular language, this allows websites to handle traffic that was not originally possible for them to capture.

H. *Validation*

AMP pages come with an in built validator that allows discovery of non-compliant areas. There are few chrome plug-in available as well that take the HTML and validate it against the rules to discover policy violations.

V. INTERNATIONALISATION

To incorporate internationalization in the framework, Googles translation API is used to convert the AMP compliant HTML text to multiple regional languages based on user search and a regex based model that allows achieving this for all URLs following a certain pattern. This permits translation of the AMP compliant page to the specified regional languages, which can be pushed at a different URL and served when a query for that page is generated in that particular language.

VI. RESULTS

An existing Huffington post article was converted to an AMP compliant one (Fig. 3 and Fig. 4). This article had a lot of images and external assets making it an ideal test case for evaluating the framework. Each asset was dealt with in such a way that it is non-blocking, passes all the validation tests while maintaining the readability, and also does not take away the original look of the page which is important for branding purposes. The page was trimmed by selecting only the content that were needed to be present in the AMP version by choosing the appropriate selectors on the page, and then a mobile page template was populated with that content after applying all the compliance rules. Since the selectors remain consistent on every article of the website, a single config file allows replication of the behavior for all the articles. The translation functionality was also tested and it worked as per the expectations. The process of internationalization is shown in Fig. 5 and Fig. 6.

The converted AMP pages were compared with their non-AMP versions based on several benchmarks like first meaningful paint, first interactive, consistently interactive, perceptual speed index, estimated input latency, performance score, and best practice score. The results are shown in Table 2.

Fig. 3. Huffington Post Article(NON-AMP) [2]

Fig. 4. AMP Compliant Huffington Post Article

CAR NEWS

New 2017 Hyundai Verna Bags Over 4,000 Bookings! Prices Start at Rs 7,99,900

August 24, 2017 | null

Yatharth Chauhan

Hyundai India launched the ***new look Verna 2017*** on August-22. In less than 48 hours since launch, the car has garnered over 4,000 bookings, thereby hinting at the great response that the car has got from prospective C2-segment sedan buyers. Akin to previous all-new models, the new 2017 Hyundai Verna is new inside and out. It may be noted that the Verna is known as the new Accent internationally. This means that the car will not just a new exterior design but will also have a new interior. Prices of the new car start at Rs 7,99,900 (ex-showroom) and are introductory for first 20,000 KMs. Read on for all about the new 2017 Hyundai Verna launch date, price, specifications and

Fig. 5. AMP Page of Existing Website[1]

VII. CONCLUSION AND FUTURE SCOPE

Mobilegeddon (change in Google's mobile search algorithm) was a major change in terms of how publishers looked at performance. It specifies that the performance would now be a factor in deciding the search engine ranking making AMP even more significant and impactful. This work could act like a proof of concept for future projects which can act like an interface for non-tech people to keep themselves updated with the latest in technology through a layer of abstraction that can be implemented in a generic manner for variety of cases. The conversion module can be easily integrated in an existing developer workflow so that a corresponding version can be generated on the fly for each developed web page.

कार समाचार

नई 2017 हुंडई वर्ना बैग 4,000 से अधिक बुकिंग! कीमतें 7,99, 9 00 रुपये से शुरू होती हैं

24 अगस्त, 2017 |

Yatharth Chauhan

हुंडई इंडिया ने ***नया वर्ना 2017 का*** शुभारंभ किया अगस्त 22 को लॉन्च के 48 घंटों से कम समय में, कार ने 4,000 से अधिक बुकिंग शुरू की है, जिससे इस महान प्रतिक्रिया पर संकेत मिलता है कि कार को संभावित सी 2 सेगमेंट सेडान क्रेताओं से मिला है। पिछले सभी नए मॉडल के लिए समान, नई 2017 हुंडई वेरना अंदर और बाहर नया है। यह ध्यान दिया जा सकता है कि वर्ना को अंतरराष्ट्रीय स्तर पर नए एक्सेंट के रूप में जाना जाता है। इसका मतलब यह है कि कार सिर्फ एक नया बाहरी डिजाइन नहीं बल्कि एक नया इंटीरियर भी होगा। नई कार की कीमतें 7,99, 9 00 रुपये (एक्स-शोरूम) से शुरू होती हैं और पहले 20,000 के.एम. के लिए परिचयात्मक हैं। 2017 हुंडई वेरना लॉन्च की तारीख, कीमत, विनिर्देशों और माइलेज के बारे में सभी के लिए पढ़ें।

FIG. 6. TRANSLATE AMP COMPLIANT PAGE

REFERENCES

1. carblogindia. https://www.carblogindia.com/new-2017-
2. hyundai-verna/. Accessed: 2018-03-30
3. Huffington post. https://www.huffingtonpost.in/akanksha-
4. sharma/photoblog-my-journey-to-the-end-of-the- world a 22137387. Accessed: 2018-03-30
5. Agababov, V., Buettner, M., Chudnovsky, V., Cogan, M.,
6. Greenstein, B., McDaniel, S., Piatek, M., Scott, C., Welsh, M., Yin, B.: Flywheel: Google's data compression proxy for the mobile web. In: NSDI, vol. 15, pp. 367–380 (2015)
7. Agarwal, A., Gupta, G., Mishra, H., Kumar, M.:
8. Packyard-data compression proxy for mobile web (2017)
9. Butkiewicz, M., Wang, D., Wu, Z., Madhyastha, H.V., Sekar, V.: Klotski: Reprioritizing web content to improve user experience on mobile devices. In: NSDI, vol. 1, pp. 2–3 (2015)
10. Elgazzar, K., Martin, P., Hassanein, H.: Mobile web ser-
11. vices: state of the art and challenges. Int. J. Adv. Comput. Sci. Appl. **5**(3), 173–188 (2014)
12. Firtman, M.: High Performance Mobile Web: Best Prac-
13. tices for Optimizing Mobile Web Apps. " O'Reilly Media, Inc." (2016)
14. MINÁRIK, D.: Accelerated mobile pages. Ph.D. thesis,
15. Masarykova univerzita, Fakulta informatiky (2017)

AUTHORS PROFILE

Gunjan is a assistant professor in Department of Computer Engineering of Maharaja Surajmal Engineering College, Delhi, India. She has completed her B.Tech from M.S.I.TGGSIPU, New Delhi and M.Tech from NorthCap University, Haryana. She has over 3.5 years of teaching experience. Her area of interest are IOT, Soft Computing, Machine learning, AI, Data Mining.

Bharti Sharma is a assistant professor in Department of Computer Engineering of Maharaja Surajmal Engineering College, Delhi, India. She has done his BTech. And M.Tech from Punjabi University Patiala, Punjab. She has over 9 years of teaching experience. Her areas of interest are Recommender System, Soft Computing, Swarm Intelligence, Data Mining.

Deepak Pathanai Deepak Pathania is a Senior Software Engineer at INDwealth, a fin-tech startup based out of Gurugram. Deepak did his graduation from MSIT with a specialization in Computer Science. Deepak has a keen eye for building elegant interfaces and a proven record of writing scalable code that has been used by millions of users.

Ankita Dhillon is a M.tech student at SKITM. She did her graduation from MSIT, Delhi with a specialization in Computer Science. Her areas of interest include designing user-friendly interfaces and mapping user journeys to reduce friction in usage.