MCWDF - Micro-chunk based Web Delivery Framework

Shailesh Shivakumar, P.V Suresh
Indira Gandhi National Open University
New Delhi, India
shailesh.shivakumar@gmail.com

ABSTRACT: *Segregating the web page content into logical chunks is one of the popular techniques for modular organization of web page. While chunk based approach works well for public web scenarios, in case of mobile-first personalization cases, chunking strategy would not be as effective due to dynamic nature of the content and due to the nature of content granularity.*

In this paper, we propose a novel framework Micro chunk based Web Delivery Framework which uses micro chunk. The framework aims to address the challenges posed by regular chunk in a personalized scenario. We will look at the techniques for creating micro chunk and we will discuss the advantages of micro chunk when compared to a regular chunk for a personalized mobile web scenario. We have created a prototype application implementing the Micro chunk based Web Delivery Framework and benchmarked it against We have benchmarked the performance statistics against regular personalized web application to quantify the performance improvements achieved by micro chunk design.

Keywords: Micro Chunk, Web Performance, Caching Strategy, Performance Engineering, Personalized Web Acceleration, Web Performance Optimization

Received: 23 July 2017, Revised 2 September 2017, Accepted 17 September 2017

1. Introduction

Modern web architecture needs to address multiple concerns and cater to various stakeholders. Internet-facing web sites should be responsive, interactive and should achieve optimal performance. The success of online strategy depends on its usability and performance. The web application should satisfy quality attributes such as performance, scalability, extensibility and flexibility.

Web pages that display enterprise information would mainly get its web content from content management systems (CMS). We could broadly categorize such web pages as public web which renders web pages for public users and private web which renders the web content for personalized/logged-in scenarios. There are number of techniques such as layered architecture, caching, service-oriented-architecture, asynchronous resource loading, content compression that can be used to achieve better scalability and performance.

Web content chunking (also known as HTML fragmenting) is one of the popular techniques. In this technique, we logically segregate the page content into multiple chunks (or fragments). The chunks are modular logical entities which are independent

and have reuse potential. A content chunk is a logic piece of cohesive HTML content on a web page which can exist independently and describes a semantic concept. Also known as html fragment.

For instance in a products page, the product brief description can constitute as a chunk and product specifications can form another chunk. A chunk consists of rich content and multi-media content. A chunk for the web content will be created from its origin system which is normally a content management system (CMS). The content chunk created will be tagged with metadata so that it can be appropriately retrieved on the correct web page. Chunk-based web page architecture addresses concerns such as scalability, performance and reusability.

While content chunking works as an effective strategy for public web scenarios, personalized web poses its own set of challenges. In a personalized web scenario a chunk contains the content applicable for a given context (such as user attributes, device, and language, location, and security roles and such). This means that it is not possible to statically author these content chunks in CMS and cache it globally. This further impacts the scalability and performance attributes. More and more web sites are adopting hyper personalization strategy wherein all content is heavily personalized for the user and the context. In such scenarios there will be seldom static content chunks which can be applied to all users.

One of the ways to effectively achieve good performance in this scenario is to identify the static chunks for a personalized page and globally cache it. This partially addresses the problem as the dynamic and personalized chunks still cannot be cached and authored thereby impacting the overall page performance.

In this paper we introduce a novel concept called "micro chunk" and Micro-chunk based Web Delivery Framework (MCWDF) which addresses the challenges in authoring, management and performance for personalized web scenario. A micro chunk is highly granular dynamic content chunk which is created and managed at the run-time with minimal overhead on page performance. Micro chunk is created mainly based on context parameters such as user attributes, device name, security roles and other filters. As micro chunk caters to dynamic content scenarios, it can be effectively used in personalized web pages with minimal overhead in content authoring.

We will look at MCWDF and how it solves the dynamic content challenge in personalization scenarios.

Based on our experiments we were able to notice 90% improvement in page response time by usage of micro chunk when compared to regular content chunk for personalized mobile web scenarios.

Paper Organization: In this section we will look at the main concepts of micro chunk along with literature review and the significance of the work. We will discuss the complete details of micro-chunk based web delivery framework, caching design and the authoring and publishing aspects of micro chunk in the Method section. In "Results" section we will look at the benchmarked results of performance numbers at various user load and content metrics. . Finally we will discuss the significance of results, explanation of the main findings, and impact on content authoring and future scope of improvements in "discussion" section.

1.1 Case for Micro Chunk

Most of the modern web applications are personalized. Almost every content chunk on a personalized page has a personalized and contextual value. This brings in two set of differences from the regular content chunk/HTML fragment based approaches: The chunk as a whole is not dynamic; it is partially dynamic and partially static. For instance consider the content chunk:

"Your policy is due for renewal on *01-Jan-2017* and your new annual premium would be *$250*"

The italicized content pieces in the above content fragment are dynamic and the rest is static. As a result of this mixed nature of the chunk it is not possible to cache it nor can we afford to make a page refresh for updating this content. The dynamic nature of the content also poses challenges for authoring these types of chunks. We see a good number of this tiny content chunks in modern mobile-first web sites.

1.2 Core concepts of Micro-chunk based Web Delivery Framework (MCWDF)

The key concepts using in MCWD is detailed below

Content Chunk: A content chunk is a logic piece of cohesive HTML content on a web page which can exist independently and describes a semantic concept. Also known as html fragment.

Micro Chunk: A micro chunk is a modular and dynamic component of a page. Micro chunk is constructed at the run time at the micro-chunk server based on context parameters and uses light-weight services to retrieve the dynamic values.

Context Parameters: These parameters represent the variables which can fully describe the run time environment. Session attributes, user attributes, device, security roles, current page are some of the examples of context parameters.

Micro Chunk Server: A micro server is responsible for constructing the micro chunk based on context parameters. The micro-chunk server would manage the micro-chunk through its entire lifecycle including authoring, updating, caching and purging the micro chunk. The micro chunk server also manages all the service invocations needed to manage the micro chunk and acts as a façade for the web page.

Micro Chunk Service: The light-weight services which can be used to obtain the micro chunks and they are hosted on micro chunk server. Micro chunk services would take the context parameters to create the micro chunk.

1.3 Literature Review, Related Work and Gaps with State of the Art Techniques

The core concepts discussed in the paper are related to "*chunking content in dynamic scenarios*" and "*improving performance and management of dynamic content and personalization scenarios*". We will look at state of the art in these two focus areas.

Firstly the generic concept of chunking is used for web page construction for many purposes:

The concept of chunk is used to reduce network traffic [1], and to reduce download time in mobile devices [2]. Chunked data is also widely used in CDN for performance optimization [3]. The technique discussed in patent [4] uses chunk concept for identifying and removing duplicates and another technique [5] uses chunking method for device specific rendering and appropriate positioning. The technique [6] uses chunking concept to differentiate static chunk from dynamic chunk and pre-fetch appropriately. ESI (Edge side include) [9] provides language for defining web page components and assembling it at edge caches using chunking concept. [25] Compares ESI and deltaencoding using analytics and simulation studies.

Content fragments are also used to handle dynamic content. The paper [8] proposes algorithm to automatically detect the fragments based on its caching and reusability potential. Papers [15] and [16] proposes user driven personalization which uses fragments based on page's internal structure. Papers [17] and [18] provide an efficient update of dynamic content in a web page and dynamic content publishing using fragments. Paper [24] efficiently encode structurally similar web pages of a site.

As far as dynamic content performance is concerned, papers [10], [11] discusses dynamic content caching and paper and paper [14] uses active cache for caching dynamic web content. Paper [23] discusses invalidation algorithms of dynamic chunks and exploits the difference between cache invalidation and view maintenance to delivery fresh content. Paper [12] proposes fragment based performance improvement. Paper [13] uses web graphs using fragment for managing quality of service in dynamic content.

Once the chunks are categorized as static and dynamic, caching policies can be accordingly applied. The web graph technique in [13] handles dynamic content by adding various attributes such as QoS, security and these attributes are used to refresh the dynamic content.

Another technique [21] uses micro content for efficient learning on mobile devices and the paper [22] discusses micro content for e-learning. In these two techniques, content is broken down into smaller, easily-digestible learning content.

As papers [8] and [17] are closely related to the subject matter of this paper, we will elaborate the differences between the topics discussed in those papers and our proposal. Paper [8] the core idea is the automatic fragment detection based on its reuse potential and its lifetime and personalization characteristics and caches it and paper [17] presents technique to establish relationship between pages and its composite fragments represented by object dependence graph. It also presents algorithm to detect and update changed fragments and publish them. The content fragments discussed in these papers are at coarse grained

and the updates are done at the chunk level. Based on the amount of content present within the fragments, the web page incurs data transfer overhead. In our micro-chunk based approach, we propose a light-weight fine-grained content where content updates are highly manageable with least data transfer overhead. Papers [8] and [17] would also encounter scalability challenges when it needs to handle huge amount of private and personalized content scenarios as it leads to high frequent content updates whereas our micro-chunk based approach is tested for scalability in high traffic personalized scenarios. The content fragments of [8] and [17] are also not designed to handle a mix of static and dynamic values within a content fragment. In the case of a micro chunk, we could efficiently handle the dynamic values embedded within a chunk.

To summarize, state of the art techniques use the content fragments to handle dynamic content using following methods:

1. Identifying the reusable chunks and segregating static chunks from dynamic chunks.

2. Use dynamic fragments to automatically push the updated content to page through dynamic fragments

3. Design the cache differently for static and dynamic chunks.

The main gaps with state of the art techniques with respect to personalized mobile-first scenario are as follows:

1. In a personalized mobile-first web scenarios, most of the content chunks would contain a mix of static and dynamic data and it would not be easy to neatly separate static content from dynamic content

2. Personalized data is more dynamic than a regular dynamic data. A personalized data is specific for an individual user. We could explain the difference between the regular dynamic data and personalized data through a dashboard example. A public dashboard of an employee portal would render the regular dynamic data by retrieving the web content from CMS; whenever the content in CMS changes the dashboard would get the updated dynamic data from the CMS. A personalized dashboard rendered for a logged-in employee would specifically display the content for the employee such as employee's leave balance, profile details and such. This personalized dashboard is a classic example of personalized data. State of the art techniques are designed to manage and cache regular dynamic data but falls short of handling the personalized data. None of the caching techniques mentioned earlier can efficiently handle the personalized data due to security and privacy concerns.

3. State of the art techniques also do not automatically consider context parameters such as user's device, time, location and such to render the personalized content.

4. Due to the challenges with applicability of content chunk with personalized data, the performance and scalability would be impacted due to the increased run-time overhead in retrieving and processing the dynamic content. The impact would be more if there is high volume of dynamic data and high concurrent user traffic, the impact would be more.

Our novel concept of micro chunk is designed to handle all types of personalized and context parameters for optimal page rendering.

In the next section we will briefly discuss the drawbacks of content chunks in the context of personalized page scenarios.

1.4 Scope and Significance of the Work

The scope of this research work covers the following:

- Performance optimization of a mobile-first web page in personalized scenarios using micro-chunk concept. This improves upon the existing content chunk based technique to handle dynamic content scenario

- Improving the scalability for the personalized and dynamic web content scenarios.

The research paper proposes a novel micro-chunk based web delivery framework which can be used in personalized web and dynamic web scenarios.

2. Method

In this section we will look at the detailed design of micro chunk based web delivery framework (MCWDF) along with caching and authoring scenarios for micro chunk.

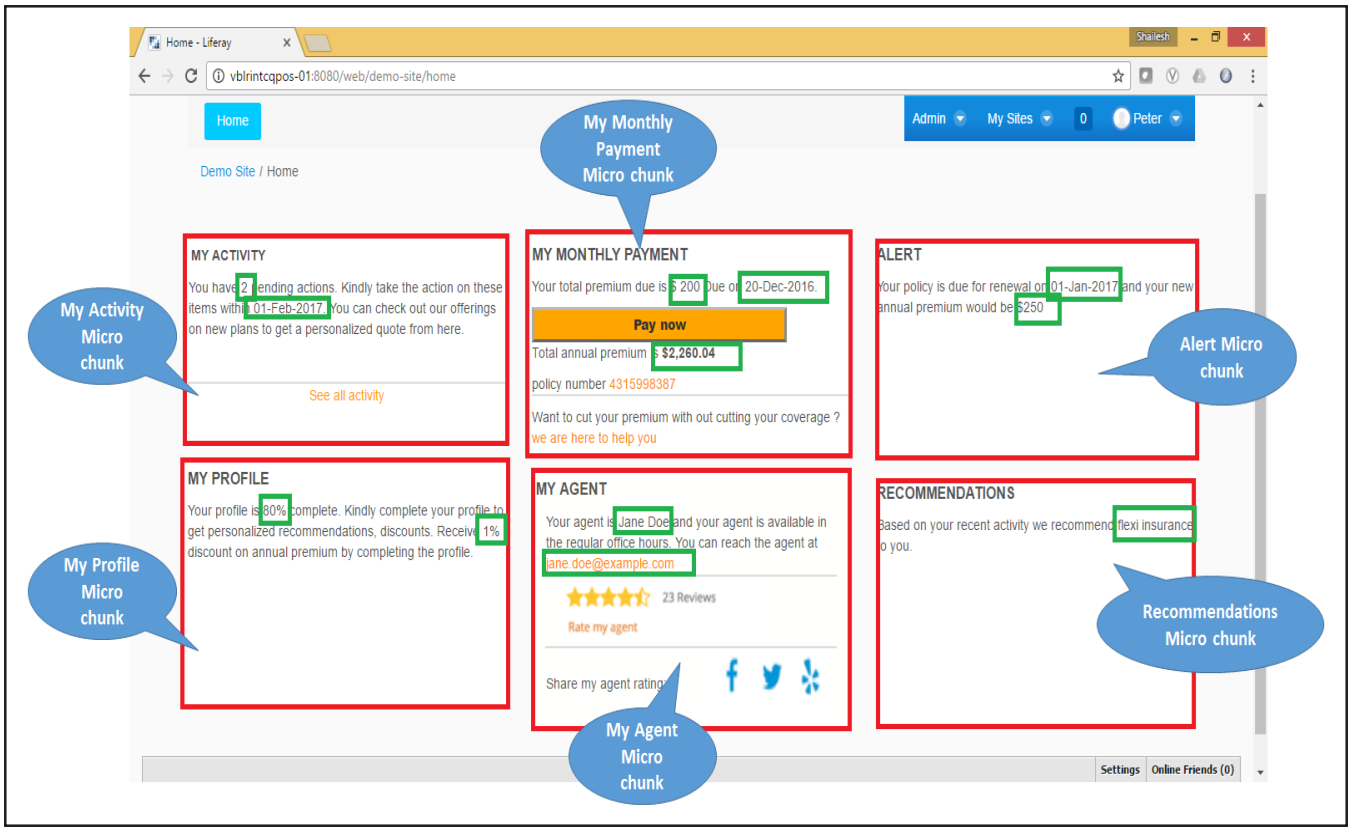

Figure 1. Micro chunks of mobile enabled personalized dashboard page

2.1 Anatomy of a Micro Chunk

Let us look at the concept of micro chunk using an insurance dashboard page we have developed for demo application. Figure 1 shows the micro chunks on a personalized insurance dashboard which is developed for mobile web.

As we can see the personalized dashboard has a heavy mix of static content and dynamic values. It would not be possible to cache the content as-is due to privacy and security concerns and dynamic nature of the data. We can also notice that there is not much content chunk which are fully static. Almost all content has personalized information to some extent.

A micro chunk can be created to handle a variety of dynamic personalized information which depends on user information and context information.

The key tenets of a micro chunk are: Independence (nil or minimal dependency on other content), Dynamic nature (should contain dynamic and personalized information), Reusability (the content within a micro chunk should be reusable), shorter content length (content sentences ranging from 1-5) and Modularity (logical grouping of content).

In the dashboard example, we have identified 6 micro chunks and the dynamic values within each micro chunk is highlighted within a box. Each micro chunk identified satisfies 5 key tenets listed above. For instance the “Alert micro chunk” contains the logical grouping of policy alert information and is both modular and independent. The micro chunk contains the dynamic information 01-Jan-2017 and $250 which is personalized for logged-in user Peter. The alert message verbiage and format can be reused across all users with varying dynamic values applicable for each user. The micro-chunk can be effectively used for enabling the mobile web and implement mobile first strategy. The rendition of the responsive dashboard page on a mobile device is shown in figure 2.

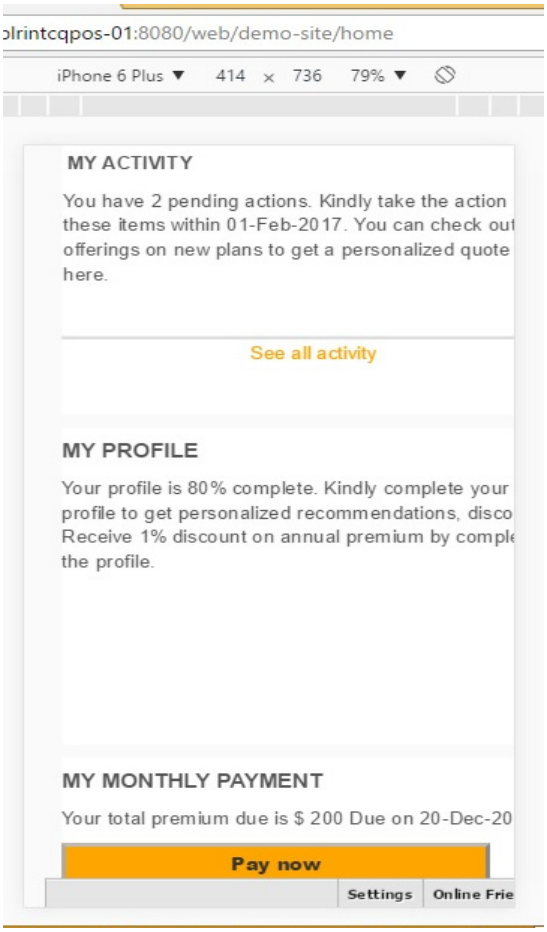

Figure 2. Mobile simulation of the dashboard page

In the next section we will discuss in detail about the MCWDF and the role of micro chunk in rendering the personalized page.

2.2 Micro Chunk based Web Delivery Framework Architecture

We are proposing MCWDF that uses micro chunk to efficiently manage the content on mobile-first personalized pages.

MCWDF is responsible for creating the micro chunk, its associated JSON and manages its lifecycle along with the caching of micro chunk.

The logical architecture of MCWDF is depicted in figure 3.

The MCWDF consists of components that are part of web page and components which are part of micro chunk server. There are basically three framework components that are a part of web page.

Micro Chunk JSON defines a micro chunk. The micro chunk JSON will be loaded as part of initial page load. Each micro chunk is identified by a uniqueid and contains the details of a microchunk such as content type (text or image or video), serviceurl (that provides the service end point to the micro chunk service based on context parameters). The context parameters are essentially personalization tokens which provide the system-specific and application-specific context values such as user roles, device, account id and such. The micro chunk preprocessor would send these values to the micro chunk service. Let us look at two micro-chunks to understand its design and utility in figure 4.

JSON representation of micro-chunks are given below:

Each micro-chunk has a name is identified by a unique identifier. The “contenttype” indicates the type of the content rendered by the micro chunk and “serviceurl” represents the service end point which provides the needed dynamic information for this micro chunk. We can also pass all context parameters such as userid, device, page and such.

```
{
   "microchunks" : [
    {
        "name": "user_promotion_message",
        "uniqueid": "12345",
        "contenttype": "text",
        "serviceurl": "http://localhost:9091/getpromomsg",
        "contextparams": {
            "generic": {
                "page": "{page_id}"
            },
            "private": {
                "userid": "{user_name}",
                "account": "{account_id}"
            }
        }
    },
    {
        "name": "user_account_info",
        "uniqueid": "672345",
        "contenttype": "image",
        "serviceurl": "http://localhost:9091/getacctinfo",
        "contextparams": {
            "generic": {
                "device": "{device_id}",
                "lang": "{lang_id}"

            },
            "private": {
                "userid": "{user_name}",
                "account": "{account_id}"

            }
        }
    }
]}
```

Figure 4. Micro chunk JSON

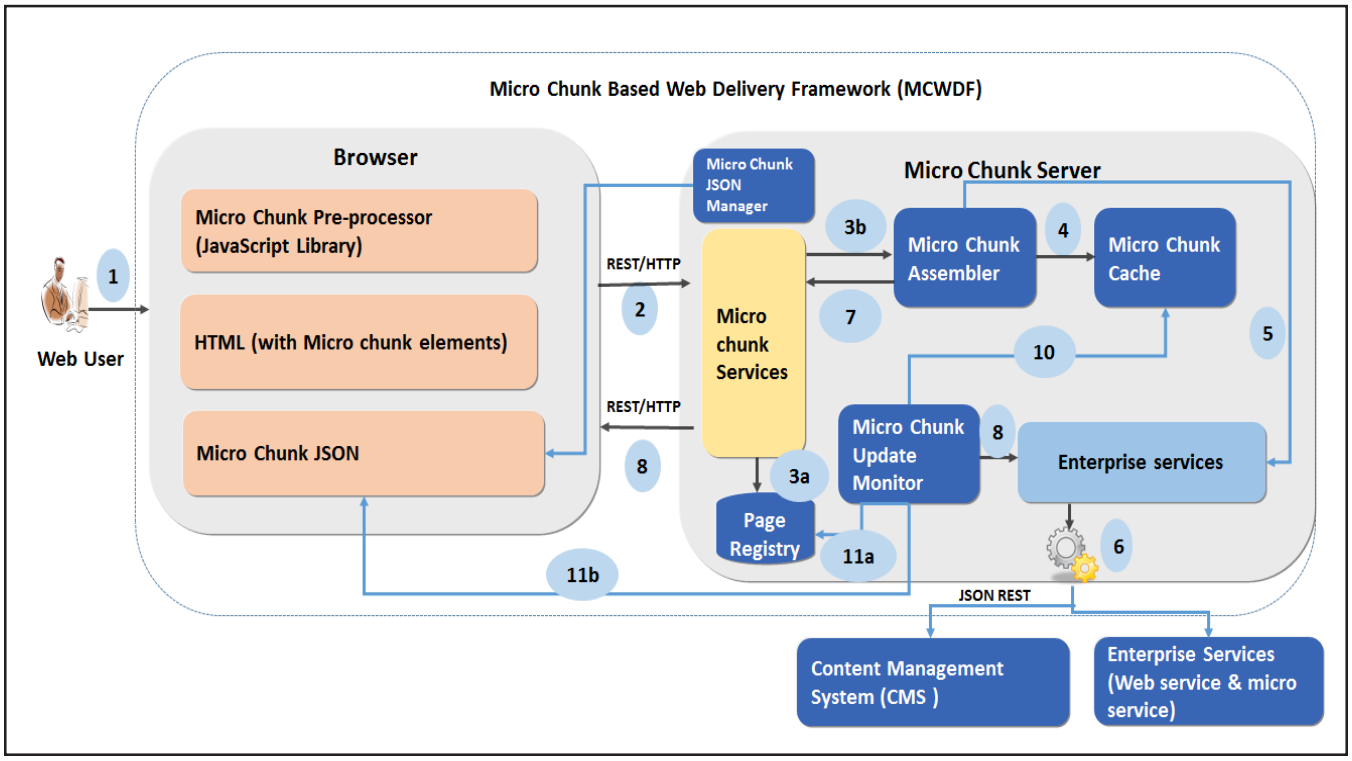

Figure 3. Micro chunk based web delivery Framework

The contextparams could be “generic” which contain non-user specific details such as page name, device id or language. “private” context parameter provides user specific personalization tokens such as user id, account id and such.

Micro Chunk Preprocessor is a custom JavaScript library that uses the micro chunk JSON to invoke the micro chunk service and injects the dynamic micro chunk using the dynamic micro chunk content obtained from the micro chunk service. The micro chunk would be inserted at the HTML element identified by the micro chunk name as depicted in figure 5. The pre-processor checks the micro chunk JSON for updates and if there are updates, it will refresh the micro chunk content.

HTML Tags is basically a HTML div element with a pre-defined style class “mcwdf-microchunk”. The div embeds the name of the micro chunk. The micro chunk preprocessor is responsible for populating the div with the dynamic micro chunk value obtained from the micro chunk service. The update happens asynchronously and whenever there is an update to the micro chunk.

```
<div class="borderless-container" style="">
    <div class="body">
        <div class="journal-content-article">
            <div class="mcwdf-microchunk" >
                {{user_promotion_message}}
            </div>
        </div>
    </div>
</div>
```

Figure 5. HTML for rendering micro chunk

The key components of the micro chunk server are as follows:

Micro Chunk Services which act as the service end point for the micro chunk requests. Once the request for micro chunk is received along with context parameters, the requesting page and its will be added to the page registry and the request is transferred to the micro chunk assembler.

Micro Chunk Assembler is responsible for identifying all the upstream services needed to create the requested dynamic micro chunk. Initially the assembler checks for the existence of micro chunk in the micro chunk cache using the micro chunk's uniqueid as the cache key. If the corresponding micro chunk is not cached, the micro chunk assembler invokes corresponding services from enterprise services layer and creates the dynamic micro chunk. A single micro chunk may need invocation of multiple services. For creation of "user_account_info" micro chunk the assembler would invoke 3 services: firstly a content service to get the base static content of the micro chunk which gets the content: "Your policy is due for renewal on *#due_date* and your new annual premium would be *#due_amount*".

Then the assembler invokes two micro services to get the dynamic values for #due_date and #due_amount for the said customer.

Only the base static version of the micro chunk is cached against the micro chunk's uniqueid and the context parameters.

Enterprise Services would interact with enterprise interfaces such as content management system (CMS) and other upstream enterprise services. Enterprise services layer could also invoke micro services needed for the micro chunk

Page Registry contains the mapping of the micro chunk ids and the page URLs on which those micro chunks are used.

Micro Chunk Update Monitor registers for the updates from CMS and enterprise services. When the data changes in the system of record, it would purge the micro chunk cache for the corresponding micro chunk and would update the micro chunk JSON on all pages where the micro chunk is used. The monitor uses the mapping stored in page registry to identify the page JSONs.

Micro Chunk JSON Manager is mainly responsible for creating the initial version of the micro-chunk JSON that is used to by the micro chunk preprocessor of the page. The micro chunk JSON manager provides a web interface for the administrator to configure various parameters of a micro chunk such as uniqueid, serviceurl, contextparams and others.

The MCWDF operates in two modes: pull mode and push mode. The pull mode happens during first-time page load and during this mode, the micro chunk preprocessor gets the dynamic micro chunk from the micro chunk service. Subsequently the micro chunk lifecycle is managed in push mode. If there is any change to the micro chunk data in the source system, the micro chunk update monitor pushes the updated content to all pages which contain the micro chunk.

At a broad level, micro-chunk server essentially plays the role of a façade to invoke needed services and manage the micro chunk. An intermediate content services layer could serve this purpose; however micro chunk server is a sophisticated platform that can manage the micro chunk's lifecycle more efficiently.

2.3 Caching Design

The micro chunk cache is a two-level cache. First level stores the mapping of micro chunk's uniqueid to a hash value computed using the generic context params (such as device id and lang id). The hash value is computed using all the "generic" context parameters for a micro chunk. The cache entries for "user_account_info" is given in table 1:

Cache Key	Cache Value
672345	m6e428e5n346cew78bfdf5a 005477521
m6e428e5n346cew78 bfdf5a005477521	 <object>

Table 1. Two-level microchunk cache

The <object> in the second level cache would be the actual micro chunk content identified by uniqueid 672345: "Your policy is due for renewal on *#due_date* and your new annual premium would be *#due_amount*".

2.4 Micro chunk Authoring and Publishing

A separate authoring template would be created for managing micro chunks. As micro-chunks are modular and reusable across many pages, authors would be able to manage micro chunks centrally. As part of micro chunk creation process, author needs to specify the micro chunk name with other metadata tags such as language, device, role and other applicable personalization parameters. There could be multiple variants of the same micro chunk varying based on language, device and user role. Authors can add the static text of the micro chunk and indicate the dynamic variables. Here is the sample authoring for "user_account_info" micro chunk:

"Your policy is due for renewal on *#due_date* and your new annual premium would be *#due_amount*"

Dynamic values #due_date and #due_amount are resolved at run time by the micro chunk assembler based on the context parameters.

The micro chunk is published as a JSON which can be invoked through REST web service by providing the uniqueid and context parameters.

3. Results

We implemented the MCWDF in a prototype application, Microchunk Web App, with 40 pages with 3 hierarchy levels whose hierarchy level is shown in figure 6. The dashboard page depicted in Figure 1 is the home page of the Microchunk web app. We created micro services to get the dynamic data needed for the microchunk. The microchunk server in the Microchunk Web App leveraged the micro services and each micro chunk used an average of 1.4 micro service to create a fully dynamic micro chunk content. Each page has an average 4.5 micro chunk in the micro chunk web app.

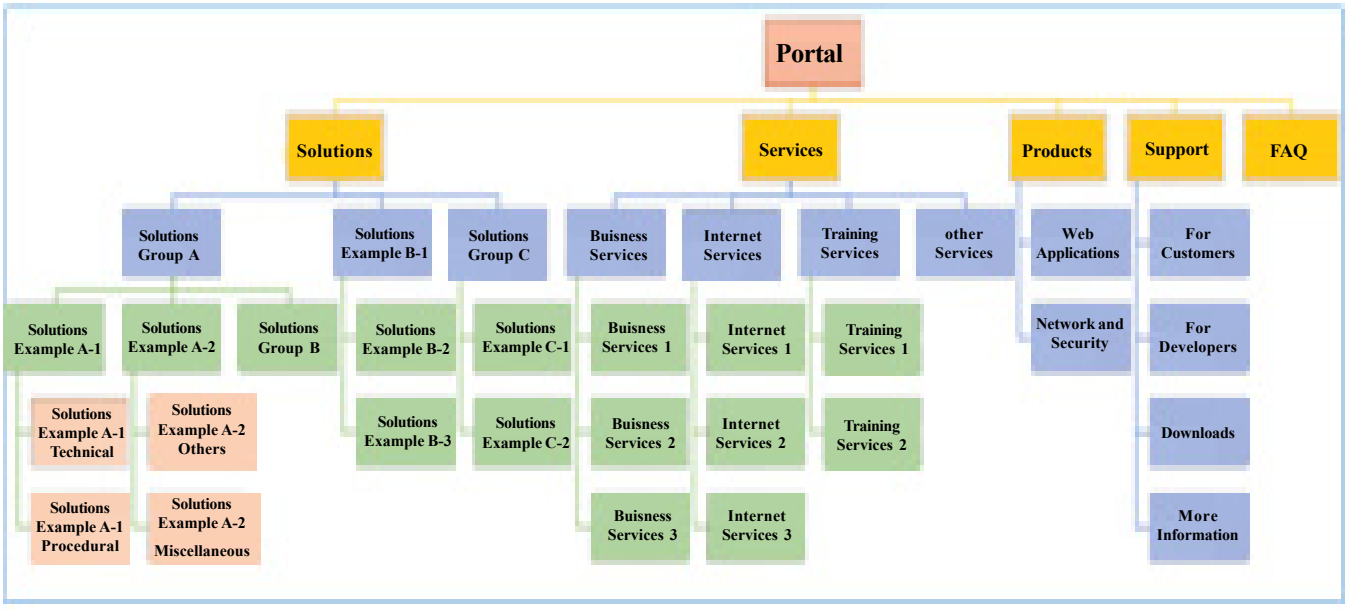

Figure 6. The 3-level hierarchy of Microchunk web app

For measuring the performance improvement, we created a mirror replica of the microchunk web app as a regular web application, named as Dashboard web app. In this case we used database services and content services directly invoked from the page to create the dynamic data on the web page. The content required for dashboard web app was authored as regular content chunk with dynamic values being added at the run time by the presentation engine.

3.1 Performance Comparison of Microchunk Web App vs Dashboard Web App

We used Apache JMeter to load both the applications and captured the average response times. The performance metrics at the load of 100 concurrent users is given in following table 2

Application	Average First Byte time (S)	Average page load time (S)	Average Total number of requests per page	Average asset load time (image, CSS,JS) (S)
Microchunk web app	0.125	1.088	4	1.039
Dashboard web app	1.142	1.999	20	1.849

Table 2. Performance metrics for microchunk app and dashboard app

We took the average times for pages to arrive at the above numbers.

We measured the average page response time (PRT) at various loads as depicted in the figure 7. We consistently noticed an average of 90% improvement in page response time at various loads.

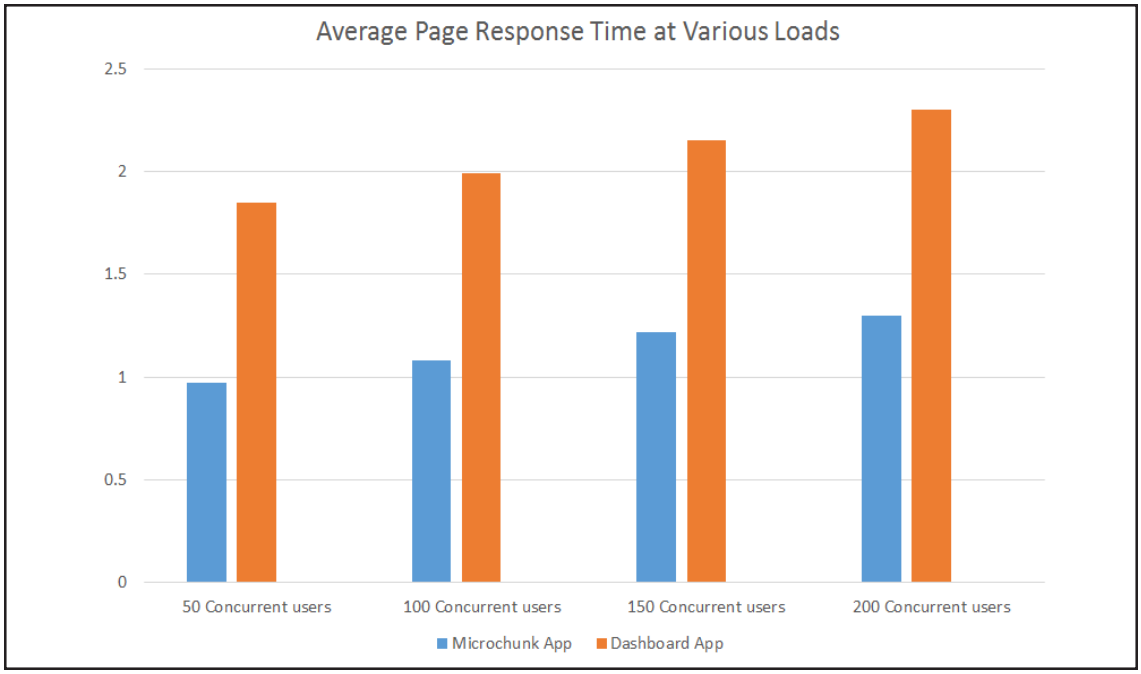

Figure 7. Average response times

We also measured the content authoring time needed for the micro chunks of microchunk web app and content chunk of dashboard web app. Here are the main metrics:

- Total number of content templates needed:
    - Micro chunk templates in microchunk app: 15
    - Content chunk templates in dashboard app: 10
- Average Content authoring time

- Average time for authoring a Micro chunk in microchunk app: 15 mins
- Average time for authoring a content chunk in dashboard app: 10 mins

- Average content publishing time
  - Micro chunks in microchunk app: 5 mins
  - Content chunks in dashboard app: 5 mins

There is 50% increase in the number of templates needed for microchunk app as compared to regular content chunk templates needed for dashboard web app. The increase in the template number for micro chunk app can be mainly attributed to dynamic values and its placement which creates variations among templates. The authoring time for micro chunk is also higher when compared to dashboard app for the same reason.

4. Discussion

We will discuss the significance of the results and the effectiveness of micro chunk in this section.

4.1 Key Motivations for Micro Chunk

The concept of content chunk and fragment needed a fresh look in the context of hyper-personalized modern applications which are developed with mobile-first strategy. The traditional definitions of content chunks or fragments [1], [3] would not be fully relevant for a personalized modern application. Firstly the amount of content chunk has reduced to cater to mobile and Omni-channel requirements. Secondly and more importantly the content fragment would no longer fit into a pure definition of a "static chunk" or "dynamic chunk" [6]; it is mix of static content along with dynamic values. So we need a briefer content fragment which could cater to dynamic and personalized scenarios and the concept of micro-chunk is designed for this purpose. While the traditional content chunks were pre-dominantly driven by CMS or HTML content, the content in modern web applications get the information from various sources. So micro chunk server would use micro chunk assembler and invoke all necessary services to create the micro chunk.

The cache management of micro chunk would also be different from that of regular content chunks [10], [11], [13]. Hence micro chunk cache uses two-level caching mechanism with Micro chunk update monitor doing an on-demand cache invalidation when the source content changes.

4.2 Explanation of Main Findings

As we can see from table 2 and figure 7, the micro chunk web app has a distinct performance advantage in mainly 3 areas: average page load time, average time to first byte (TTFB) and average resource requests per page. As the micro chunks are created, managed and cached by the micro chunk server, the average page response time has improved by more than 90% and the average asset load time has improved by 80%. We could also notice that average number of resource requests of microchunk app has reduced to 25% of the dashboard web app. This is mainly due to the design of micro chunk server which acts as the façade to manage and handle all the underlying micro services. The asset load time reduction is due to asynchronous asset loading process.

We could also notice that on content management side, the number of micro chunk templates and content chunk authoring time is lagging behind that of content chunk. While the micro chunk template development is a one-time process which can be reused for future micro chunks, the increase in the micro chunk authoring effort can be attributed to the complexity involved in the authoring process. The author needs to select the right micro chunk template and use the correct dynamic values.

4.3 Significance and Implications of the Results

The key finding from MCWDF is the drastic improvement of web performance for personalized mobile-first site in page load time and asset load time metrics. This is a significant improvement over traditional ways of dynamic information aggregation. This makes the micro chunk a promising design alternative for personalized mobile web applications. As more and more web applications are built with mobile-first design, the concept of micro chunk becomes more relevant.

Another aspect of MCWDF is about the handling of personalized and dynamic values. The MCWDF also handles the scenario which has partially mixed dynamic values within static content chunk. Micro chunk server provides an efficient management

mechanism for creating, updating and caching micro chunks.

4.3.1 Impact on content authoring and Publishing

On the content authoring front, additional micro chunk templates need to be created and authors need to be trained to use those templates. For the microchunk web app we created 15 new micro chunk templates to create all distinct micro chunk each with an average of 2 dynamic values.

Development of additional micro chunk templates and author training was the overhead involved in this effort. Authors were also confused about the definition and usage of dynamic values at times.

For content publishing we leverage the in-built JSON services of the CMS to expose the micro chunk content as a web service. There was no impact on content publishing side.

4. 4 Comparison with Alternate Techniques

The main techniques which are closer to MCWDF are regular content chunk and custom services aggregation. Let us look at the key differences between Micro chunk and these two techniques.

Following table provides high level differences between a normal content chunk and a micro chunk.

We will look at a detailed example in coming sections.

	Content Chunk	Micro chunk
Key Tenets	• Cohesiveness • Semantic independence • Static nature	• Modular • Dynamic and context based content
Content Granularity	Relatively coarse grained	Fine-grained granularity
Data Content	Normally static which can be used across all contexts	A mix of static and dynamic and driven by context parameters
Reuse potential	Can be fully reused as-is in static scenarios and public pagesCannot be reused in private web pages which need personalized data	Can be reused across public and private scenarios

Table 3. Examples for Micro and Content Chunks

The functions of a micro chunk server can loosely be compared to that of a custom services layer, micro chunk server does more activities than a services layer such as assembling, caching and managing micro chunk. The micro chunk server exposes micro services to the presentation components and aggregates services needed for developing a micro chunk.

4.5 Threats to Validty

We conducted the experiments for 40 pages with maximum of 200 concurrent users load. The performance improvement was observed in this setup. We would need to observe the performance improvement of MCWDF with higher number of pages and with greater user load to check the scalability and performance of the micro chunk cache.

The performance improvement is also largely dependent on the frequency of content updates in the source systems. If there is a frequent content update activity, it would trigger cache invalidation leading to cache misses; this in turn would negatively impact the page performance. This also would impact the performance metrics of the MCWDF

4.6 Future Scope of Improvements

Currently the micro chunk JSON is mainly developed manually in micro chunk JSON manger interface. The process of initial micro chunk JSON creation process needs to be automated. More research needs to be done in that front.

The authoring process of micro chunk needs to be improved. Currently the authors mainly depend on the training material and this could lead to confusion. A more structured and easy-to-use process needs to be designed. Efforts are needed to reduce the content authoring time and to design an efficient way to manage micro chunk templates.

5. Conclusion

We have proposed a novel framework MCWDF which can be used for performance optimization of personalized mobile web pages. The MCWDF uses the concept of micro chunk to logically partition the page into modular, dynamic and reusable micro chunks. Micro chunk server would be responsible for creating, updating and caching the micro chunks for the page.

We developed a demo site which implemented the MCWDF and benchmarked the performance against a regular web application of similar nature. As per our experiments we have noticed > 90% improvement in page performance.

References

[1] Zhang, X., Wang, N., Vassilakis, V. G., Howarth, M. P. (2015). A distributed in-network caching scheme for P2P-like content chunk delivery. *Computer Networks.*

[2] Jang, I., Suh, D., Pack, S. (2014). Minimizing content download time in mobile collaborative community. *In:* 2014 IEEE International Conference on Communications (ICC).

[3] Zhu, M., Li, D., Wang, F., Li, A., Ramakrishnan, K. K., Liu, Y., Liu, X. (2016). CCDN: Content-Centric Data Center Networks. *In:* IEEE/ACM Transactions on Networking IEEE/ACM Trans. Networking, 1-14.

[4] Griffin, W., Jones, B., Lee, S.K., *U.S. Patent No. 11/192,791*. Washington, DC: U.S. Patent and Trademark Office.

[5] Davis, P.E., Dean, S.E., Meliksetian, D.S., Milton, J., Weitzman, L., Zhou, N. U.S. Patent No. 7,076,728. (2006). Washington, DC: U.S. Patent and Trademark Office.

[6] Maghoul, F., Yiu, P., Davis, M., Athsani, A. and Yi, J., U.S. Patent No. 12/240,323. (n.d.). Washington, DC: U.S. Patent and Trademark Office.

[7] Douglis, F., Haro, A., Rabinovich, M. (1997), December. HPP: HTML Macro-Preprocessing to Support Dynamic Document Caching. *In:* USENIX Symposium on Internet Technologies and Systems (p. 83-94).

[8] Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F. (2004). Automatic detection of fragments in dynamically generated web pages. *In:* Proceedings of the 13th Conference on World Wide Web - WWW '04.

[9] Edge Side Includes - Standard Specification. http://www.esi.org.

[10] Challenger, J., Iyengar, A., Dantzig. P. (1999). A Scalable System for Consistently Caching Dynamic Web Data. *In:* Proceedings of IEEE INFOCOM, March.

[11] Challenger, J., Iyengar, A., Witting, K., Ferstat, C., Reed, P. (2000). Publishing System for Efficiently Creating Dynamic Web Content. *In*: Proceedings of IEEE INFOCOM 2000, May.

[12] Datta, K., Dutta, H., Thomas, D., VanderMeer., Suresha., Ramamritham, K. (2002). Proxy-Based Accelaration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation. *In*: Proceedings of SIGMOD-2002, June.

[13] Mohapatra and H. Chen. A Framework for Managing QoS and Improving Performance of Dynamic Web Content. *In*: Proceedings of GLOBECOM-2001, November 2001.

[14] Cao, P., Zhang, J., Beach, K. (1998). Active Cache: Caching Dynamic Contents (Objects) on the Web, *In*: Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), the Lake District, England, September.

[15] Christos, B., Vaggelis, K., Ioannis, M. (2004). Web page fragmentation for personalized portal construction, *In:* International

Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.

[16] Ioannis, M., Vaggelis, K., Christos, B. (2004). Web Page Fragmentation and Content Manipulation for Constructing Personalized Portals. *In:* Advanced Web Technologies and Applications Lecture Notes in Computer Science, 744-754.

[17] Challenger, J., Dantzig, P., Iyengar, A., Witting, K. (2005). A fragment-based approach for efficiently creating dynamic web content. *ACM Trans. Inter. Tech. TOIT ACM Transactions on Internet Technology,* 5 (2) 359-389.

[18] Challenger, J., Iyengar, A., Witting, K., Ferstat, C., Reed, P. (n.d.). A publishing system for efficiently creating dynamic Web content. *In:* Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064*)*.

[19] Khaing, A., Thein, N. L. (2005). Efficiently Creating Dynamic Web Content: A Fragment Based Approach, *In: 6th Asia-Pacific Symposium on Information and Telecommunication Technologies*.

[20] Brodie, D., Gupta, A., & Shi, W. (2004). Keyword-based fragment detection for dynamic web content delivery, *In*: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters - WWW Alt. '04.

[21] Bruck., Peter, A., Luvai Motiwalla., Florian Foerster. (2012). Mobile learning with micro-content: a framework and evaluation, *In:* Bled eConference, 25th Bled eConference eDependability: Reliable and Trustworthy eStructures, eProcesses, eOperations and eServices for the Future. 17-20.

[22] Nagler., Walther., Martin Ebner., Nick Sherbakov. (2007). Flexible teaching with structured micro-content-How to structure content for sustainable multiple usage with recombinable character. *In:* Conference ICL2007, September 26-28, 2007. Kassel University Press.

[23] Candan, K. S., Agrawal, D., Li, W.-S., Po, O., Hsiung, W.-P. (2002). View Invalidation for Dynamic Content Caching in Multitiered Architectures. *In*: Proceedings of VLDB. September.

[24] Chan, M. C., Woo, T. W. C. (1999). Cache-Based Compaction: A New Technique for Optimizing Web Transfer. *In:* Proceedings of INFOCOM-1999.

[25] Naaman, M., Garcia-Molina, H., Paepcke, A. (2003). Evaluation of ESI and Class-Based Delta Encoding. *In*: Proceedings of WCW - 2003.