International Journal of Web Portals
Volume 9 • Issue 1 • January-June 2017

Personalized Chunk Framework for High Performance Personalized Web

K S Shailesh, IGNOU, New Delhi, India
Suresh Pachigolla Venkata, IGNOU, New Delhi, India

ABSTRACT

Dividing the web site page content or web portal page into logical chunks is one of the prominent methods for better management of web site content and for improving web site's performance. While this works well for public web page scenarios, personalized pages have challenges with dynamic data, data caching, privacy and security concerns which pose challenges in creating and caching content chunks. Web portals has huge dependence on personalized data. In this paper the authors have introduced a novel concept called "personalized content chunk" and "personalized content spot" that can be used for segregating and efficiently managing the personalized web scenarios. The authors' experiments show that performance can be improved by 30% due to the personalized content chunk framework.

KEYWORDS

Content Chunk, Personalization, Personalized Content Chunk, Personalized Web Acceleration, Web Content, Web Performance, Web Performance Optimization

INTRODUCTION

Most of the modern-day websites are personalized wherein the web site content and functionality will be rendered based on user preferences and other context parameters such as access device, language and such. Users expect Internet-facing applications to be responsive, interactive with optimal performance (Galletta et al., 2004). The key success criteria for an Internet-facing web site would be its usability and performance (Schmiedl et al., 2009). Web architecture should also satisfy the SLAs and quality attributes such as extensibility, performance, scalability and flexibility (Shivakumar, 2014).

In Internet-facing Information portals and public web site scenarios, the web site content is mostly stored in content management system (CMS) and we can apply the content chunking or content fragmenting concept. A web page content can be logically divided into modular fragments (Griffin et al., 2005). Content chunks are independent content pieces and represent a semantic entity (Challenger et al., 2005).

On a typical web page, the marquee image, right-hand content boxes can be created as a content chunk. Once the page content is segregated into logical content fragments, each of the individual content chunks can be managed. Content chunks can be independently authored, updated, published and cached. Chunks can also be tagged with metadata for easier identification. Chunks can also be cached individually with unique cache keys and this can improve the page performance.

In case of personalized web scenario, the page content would depend on many implicitly and explicitly specified parameters such as user preference, user profile attributes, user device, locale,

DOI: 10.4018/IJWP.2017010104

security roles, permissions, location, time and such. Web sites would push targeted content, relevant ads and effective recommendations using personalized web. Personalized web mostly contains dynamic values specific for a user group or for an individual user. Due to its dynamic nature, managing the content on a personalized web page is different than that of a static web page. We cannot statically author a personalized content chunk as the content author would not know of the dynamic values beforehand. Caching personalized content has its own set of challenges: caching a variant of content for each user would quickly become unmanageable. Global caching of personalized content would also violate privacy and security of individual users.

In this paper, we introduce a novel concept called "Personalized chunk" and "Personalized content spot" which brings the best of both worlds: flexibility and reusability of content chunk and security/ privacy adherence of personalization scenarios.

Based on our experiments we were able to achieve about 30% performance improvement through usage of personalized chunk and personalized content spot.

PAPER ORGANIZATION

In the remaining portions of the introduction section we will look at state of the art methods along with related work and the significance of the work. We will discuss the complete details of the personalized content chunk framework, process steps, caching design 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, threats to validity and future scope of improvements in "discussion" section.

Literature Review and Related work

Content chunk is a predominant theme in this paper and we will be extending the concept of the chunk to personalized scenarios. Content chunk was proposed for reducing network traffic (Zhang et al., 2015) and for reducing download time in mobile device (Jang et al., 2014). Content chunk would improve the performance in CDN scenarios (Zhu et al., 2016). The concept of chunking can also be used to remove duplicates (Griffin et al. 2005) and can be used for device specific rendering and appropriate positioning (Davis et al. 2006). Keyword based chunk detection is another technique proposed by Brodie et al. 2004. Chunking concept is used to differentiate static chunks with dynamic chunks (Khaing & Thein 2005) and are used for prefetching (Maghoul et al. 2008).

For identifying and creating content chunks, Ramaswamy et al., 2004 proposes caching and reusability as the key criteria and Christos et al., 2004 and Ioannis et al., 2004 propose user driven personalization and page's internal structure for creating fragments and Chan & Woo (1999) proposes content similarity in a site for identifying content chunks. Challenger et al. (2005) and Challenger et al. (2000) propose an update mechanism for dynamic content update of content chunk and fragment based publishing.

Dynamic content caching (Challenger et al., 1999) and active cache (Cao et al., 1998) can be used for caching dynamic content chunks.

Content chunks are used for performance improvement (Datta et al. (2002) and for improving quality of service in dynamic content (Mohapatra & Chen (2001). Chunks are also used to improve learning on mobile devices

As far as content caching goes, web graph technique as discussed in Mohapatra & Chen (2001) can be used for chunk caching (Bruck et al. 2012 and Nagler et al., 2007).

In case of web portals, portlets are often used to partition the page into logical sections and portlets are used for rendering the personalized information based on the context (Shivakumar, S. K. (2015).

Gaps with State of the Art Techniques

Table 1 presents the gaps with the existing state of the art in personalized web scenarios and how the "Personalized chunk" concept addresses these gaps:

To elaborate more about the personalized chunk, let us look at the key differences between a public content chunk and the personalized chunk.

Table 2 provides high level differences between a public content chunk and a personalized chunk.

SCOPE AND SIGNIFICANCE OF THE WORK

The personalized content chunk and personalized content spot presented in this paper is a significant contribution to the personalized web scenario. Based on the literature review, we did not find any state of the art framework or methodology that would improve the performance of personalized web. Given below are the key contributions of this paper:

Table 1. Gaps with state of art chunking techniques

Category	Gaps with existing state of the art	How the gap is addressed with "Personalized chunk" proposed in this paper
Identifying content fragment	Chunks are identified on their reuse potential (Ramaswamy et al., 2004) or interrelationship between pages (Challenger et al. 2005) The techniques cannot be used for personalized web as personal data cannot be reused Heavy overhead in data transfer rate Does not handle dynamic data	Personalized chunk efficiently manages personalized data and dynamic data. The refresh can be done with light weight fine-grained content. Personalized content smartly refreshes only the changed content.
Application scalability	The techniques proposed by (Ramaswamy et al., 2004) and (Challenger et al., 2005) pose scalability challenges in handling personalized data during peak traffic.	Personalized chunks are designed to be light-weight to handle surge in traffic.
Content Type	State of the art techniques do not handle mix of static and dynamic content. State of the art techniques do not efficiently handle the update of dynamic content.	Personalized content and personalized content spot handles mix of static and dynamic content efficiently. The personalized content chunk and personalized content spot is designed to handle dynamic content.
Caching	State of the art techniques only stores content in global cache (Challenger et al. 2000). But they would not handle the caching of personalized content.	Personalized content and personalized content spot handles caching for personalized content.
Context parameters	State of the art techniques do not consider context parameters such as user preferences, user device for the content chunk.	Personalized content chunk considers context parameters to design and render a content chunk. Context parameters are used to populate the dynamic data in the personalized content spot.
Performance	State of the art techniques efficiently cache the public content chunk but would face performance challenges during peak load for personalized content for processing dynamic content (Shivakumar, 2014).	Personalized content chunk is tested with various load scenarios and performs well during high load.

Table 2. Public content chunk vs personalized content chunk

	Public Content Chunk	Personalized Content Chunk
Main feature	Modular and reusable static content chunk that can be delivered to all users.	Personalized chunk is mainly used for dynamic and personalized web scenarios.
Identity parameters	Content chunks are mainly identified by tags, metadata and page URLs.	Personalized chunks are identified by context parameters such as user role, preferences, user device.
Content type	Static content	Mix of static and dynamic content
Reusability	Can be mainly reused in public web scenarios.	Can be reused in personalized web scenarios.

- A framework based on personalized content chunk and personalized content spot to efficiently handle the dynamic nature and personalized data in personalized web scenarios.
- Cache design and handling for personalized web scenarios.
- Improvement in scalability and performance of personalized web scenarios.

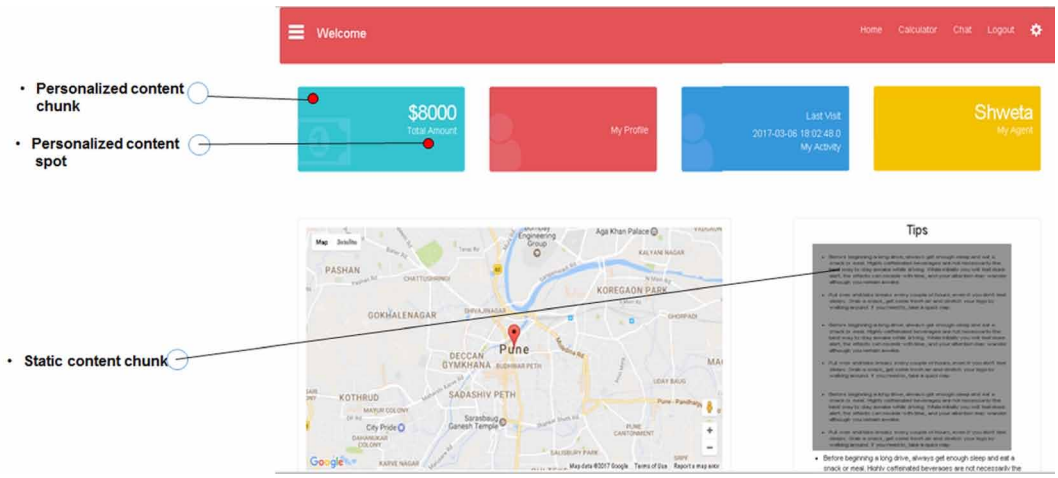
METHOD

The details of personalized content chunk framework is elaborated here. We will look at the methods for identifying and rendering the personalized chunk in this section. Figure 1 provides a snapshot of the web application implementing the personalized content chunk framework.

The dashboard page depicted in Figure 1 is from a personalized dashboard page. The dashboard represents the set of key content pieces for an insurance web application. User would see his/her personalized premium details, profile details, activities and agent details. Naturally the values for these content pieces would vary across users. In this dashboard, we would not be able to create static content chunks for premium, my profile, last activity and my agent as the data would be dynamic for each user. On the dashboard page, the "Tips" can be created as a static content chunk as the information would not vary across various users and this qualifies as a public content chunk.

The premium value depicted in "Total Amount" section is a classic example of personalized content chunk and depicts all the key tenets of a personalized chunk:

Figure 1. Personalized chunks of a personalized dashboard page



- **Dynamic/Private information**: "Total Amount" has user-specific private data which cannot be shared across users.
- **Modularity**: The chunk represents a logical piece of information which can be used as a semantic unit.
- **Reusability**: The chunk has static content (text "Total Amount") that can be reused across potential across various users (without the private data).

Using the same tenets, we can also categorize "My Profile", "Last visit" and "My Agent" chunks as personalized chunks. "Tips" would become regular public content chunk that can be reused across various users.

Analysis of a Personalized Content Chunk

As mentioned earlier, each personalized content chunk is a mix of static and dynamic content and has a dynamic value specific for a given user. In the case of "Total Amount" personalized chunk "*$8000*" is the dynamic value specific for the logged in user. We term the dynamic value as "*personalized content spot*".

The static portion of the chunk is content "Total Amount" which can be used across various users. We can also cache the static portion of the personalized content chunk.

We will designate the dynamic portion of the personalized content chunk with personalized content spot. In this case "$8000" would be marked as personalized content spot.

The static portion of the personalized content will be authored while authoring the content and the dynamic value marked by personalized content spot will be injected at the page run time based on the context parameters such as user profile attributes, security role, user device and such.

During caching only the static portion of the personalized content chunk will be cached.

The HTML code snippet for the personalized content chunk and personalized content spot are shown below:

Box 1

```
<div class="body">
                <div class="personalized-content-chunk">
                        Total Amount
                        <div class="personalized-content-spot" >
                    {{total_amount_value}}
                        </div>
                </div>
</div>
```

Personalized Content Chunk Framework

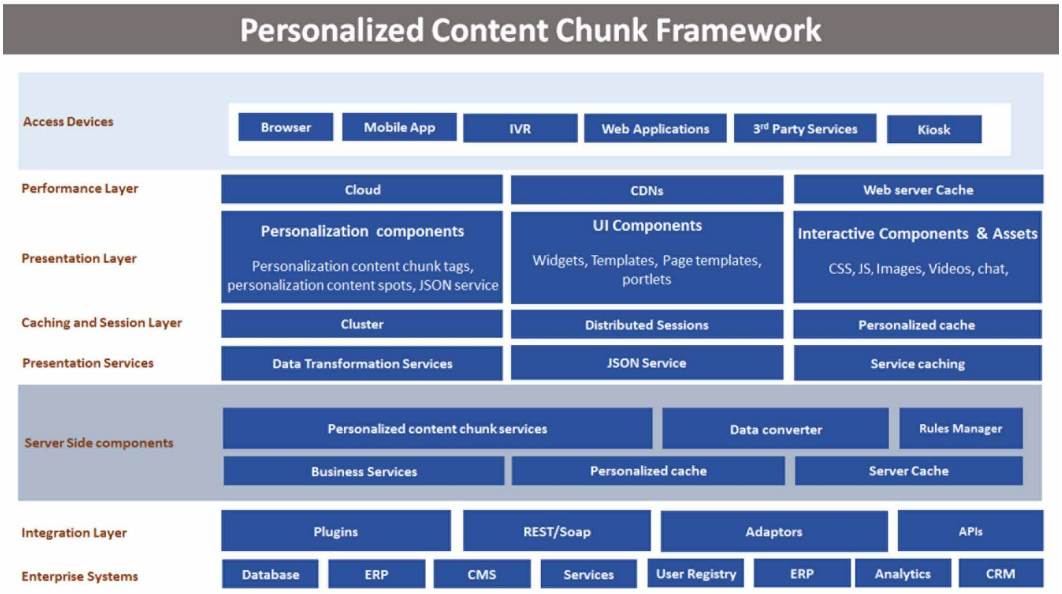
In this research paper we are proposing a novel personalized content chunk framework to address the performance of personalized web scenario.

The logical architecture of the personalized content chunk framework is depicted in Figure 2.

The framework mainly consists of presentation layer components and server side components. Other layers provide the necessary infrastructure elements to support the core components. Performance layer consists of CDN, web server cache and cloud to forward cache the static elements such as images, videos, JS/CSS files for optimal performance.

Presentation services consist of JSON services to invoke the business components to get dynamic value for personalized content spot. Integration layer consists of APIs, REST/SOAP services, plugins and adaptors to integrate and fetch data from enterprise interfaces such as database, ERP, CRM needed for personalized web page.

Figure 2. Personalized content chunk Framework



As the core components of the framework are present in the presentation layer and server side layer, we are going to elaborate those next.

Presentation Layer Components

The presentation layer components mainly consist of components that can be categorized into three categories: personalization components, UI components and Interactive components and assets.

Personalization components consist of personalization HTML tags, personalization content spots and JSON service. Personalization content HTML tags mainly consist of "div" HTML tag with a pre-determined class name "personalized-content-chunk". The personalization content chunk contains the static portion of the personalized chunk. For instance, in the "Total Amount" personalized chunk example, the static text "Total Amount" is added as part of "personalized-content-chunk" div class. The dynamic value of the personalized content chunk will be embedded within "personalized-content-spot" div class. For instance, in "Total Amount" personalized chunk the dynamic value of $8000 will be denoted by {{total_amount_value}} within "personalized-content-spot" div class which gets populated during run time based on context parameters. JSON service is mainly responsible for parsing the pre-determined div classes to get the dynamic value at the run time. In the "Total Amount" example, the JSON service would read the {{total_amount_value}} from the "personalized-content-spot" div class and invokes the backend service by passing the context parameters such as user role and user profile attributes. JSON service would then populate the actual value $8000 and replace the {{total_amount_value}} at run time. JSON service uses asynchronous service to invoke the services which form minimum overhead on the page performance.

UI components consist of page layouts, widgets and portlets that are normally used in an Internet-facing web applications such as web portals. Few advanced web portals also consist of drag-and-drop UI components and widget library. The widgets and other UI components would mainly fetch the web content from CMS in case of information portals.

Interactive components and assets are part of most of the web pages. They constitute static assets such as images, JavaScripts, stylesheets, videos, JSON/XML files and other binary files that are usually cached in web server cache or at CDN layer.

Server Side Components

The server side components mainly consist of personalized content chunk services, data converter and personalized cache. *Business services* would handle the core business logic needed for the application and *rule manager* would provide configurable rules for the application.

Personalized content chunk services would expose light-weight REST based services to the clients to provide the dynamic values. The chunk services would use the provided personalized content spot place holders (such as {{total_amount_value}}) and the applicable context parameters (such as user id and user profile attributes) and would internally invoke the business services to return the dynamic value ($8000 for total amount). In case of Total Amount personalized content chunk, the personalized content chunk services would invoke the policy business service to get the exact policy amount for the given user.

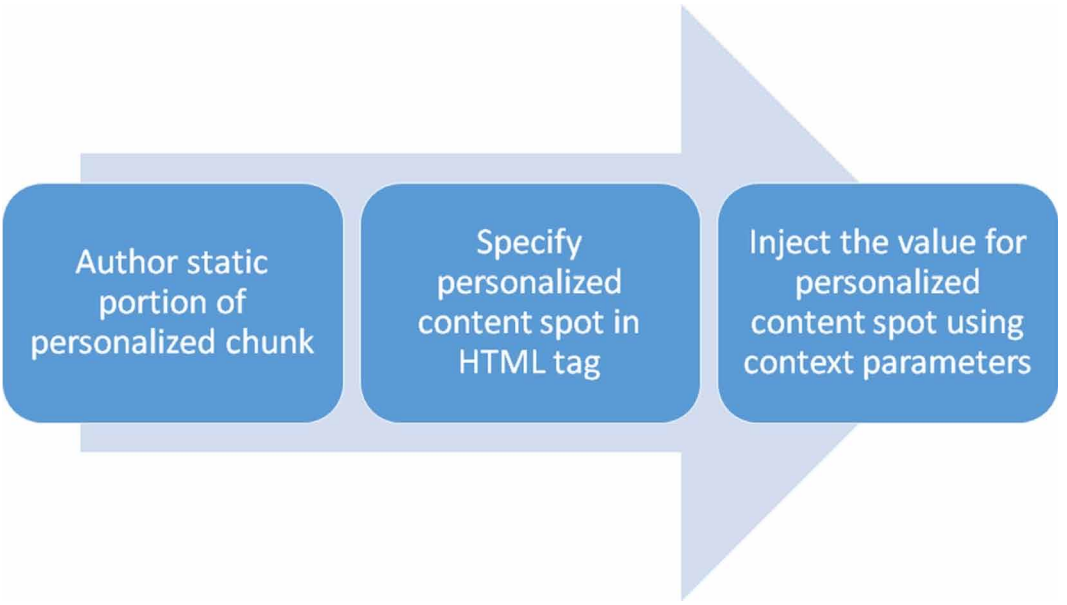
Data converter provides data transformation services needed by the business components. The response of frequently used personalized services would be cached in "Personalized cache" with user profile attributes (such as user id, user role) as cache key. This will be a private cache accessible only for a given user session and will not be shared with other user sessions due to security and privacy concerns.

Personalized Content Chunk Processing Steps

In this section we will look at the steps involved in processing a personalized content chunk and populating the dynamic value in personalized content spot at run time.

Figure 3 shows the high-level steps involved in personalized content chunk framework. Given below are the process steps that would occur in the process:

Figure 3. Steps of personalized content chunk framework



- **Step – 1 Authoring personalized content chunk**: Content authors would mainly author the static section of the personalized chunk. For instance, "Total Amount" will be authored a static content chunk for the Total Amount personalized chunk. After authoring the content chunk would be tagged with metadata and would be published in appropriate formats (HTML, JSON, XML).
- **Step – 2 Specifying dynamic value**: The dynamic portion of the personalized chunk would add the placeholder under "personalized-content-spot" div class. This will be later replaced by the actual value at the run time. For instance, {{total_amount_value}} would be added as a placeholder for the "Total Amount" personalized content chunk.
- **Step -3 Dynamic value injection at the run time**: JSON service would invoke the personalized content chunk service to replace the placeholder with the actual value at the run time using the context parameters.

The personalized content chunk rendition algorithm is given in Algorithm 1:

Algorithm 1

```
void render_personalizedchunk(userid) {
 //scan the entire web page for personalized-content-chunk tags
and compiles the // list of all personalized chunks on the page
 List personalizedchunk_tag_list = getPZNChunksOnPage(pageid);
  //Process all the personalized chunks on the page
 for each personalizedchunk_tag in personalizedchunk_tag_list do
  //Extract the personalized content spot from the personalized-
content-spot tag
   PZNchunk_name = get_pznchunk_name(personalizedchunk_tag);

   // Invoke the personalized chunk service with all needed
parameter values.
   PZN_chunk = get_PZNChunk (PZNchunk_service, PZNchunk_name)
   //Inject the dynamic value of the personalized content spot
chunk into the //corresponding HTML tag
   Update_HTML(PZN_chunk);
end do;

}
```

CACHING DESIGN

The personalized chunk cache is a two-level cache. First level stores the mapping of personalized chunk's uniqueid to a hash value computed using the generic context params (such as user id, device id and page id). The hash value is computed using context parameters. The cache entries for "total_amount" is given in Table 3:

The <object> in the second level cache would be the actual personalized chunk content identified by uniqueid 649345.

Table 3. Two-level personalized content chunk cache

Cache Key	Cache Value
649345	I76e428e5n346cew78bfdf5a005477521
I76e428e5n346cew78bfdf5a005477521	<object>

RESULT

We implemented the personalized content chunk framework in a prototype application. Details are elaborated in this section.

EXPERIMENT DESIGN

For measuring the performance improvement, we created another web application named as *Dashboard web app* which was used as a baseline application to compare the performance improvement of the personalized web app. Each of the dashboard web app and personalized content chunk app had same number of pages with similar content volume. This is necessary to compare the performance of the applications under similar scenario.

We used Apache JMeter to load both the applications and captured the average page response times. The performance metrics at the various user loads is given in following Table 4.

We also monitored the server resource values such as CPU, memory and bandwidth consumed during the performance testing process and the CPU consumption was within 40% at the peak load for both applications. Memory consumed for dashboard web app was 6 GB during peak load whereas the memory consumed by the personalized app was 4 GB during peak load.

DISCUSSION

We will discuss the explanation and significance of the results and the effectiveness of personalized content chunk in this section.

EXPLANATION OF MAIN FINDINGS

As shown in Table 4, the average page load time after using personalized content chunk framework is 1.707 seconds as compared to the dashboard web app's 2.44 seconds which brings about 30%

Table 4. Performance metrics for personalized content chunk app and dashboard app

User Load	E_A, Average page response time of dashboard web app (seconds)	E_B Average page response time of personalized web app (seconds)	$\delta = E_B - E_A$
100	2.75	3.15	0.4
200	2.77	1.45	-1.32
300	1.78	1.52	-0.26
400	1.97	1.9	-0.07
500	2.63	1.76	-0.87
600	2.53	1.38	-1.15
700	2.69	1.27	-1.42
800	2.38	1.5	-0.88
900	2.55	1.34	-1.21
1000	*2.35*	1.8	-0.55
Mean:	**2.44**	**1.707**	**-0.733**
Standard deviations:	**0.331997323**	**0.54847364**	**0.59988**

improvement in average page response time. After calculating the standard deviation and degree of freedom of 9 we get the t-statistic as 3.863 and we could prove our hypothesis (that the personalized web app has improved performance over dashboard web app) with 99.81% certainty that the performance times are improved with personalized content chunk framework.

We can clearly see 30% improvement in performance of the personalized web app which uses personalized content chunk framework. The key reason for the improvement in the performance of the personalized web is due to the asynchronous refresh of the dynamic value in the personalized content spot. The dashboard web app had to refresh the entire content chunk that impacts the page performance whereas the personalized web app need to refresh only the minor portion of the chunk in the personalized content spot area.

We could also notice a 33% improvement in the memory needed by the personalized web app. This memory optimization is due to the 2-level cache design of the personalized content chunk framework. Due to the low granularity of the personalized content chunk, the memory needed to cache the personalized content chunk is reduced by 33% as compared to the regular content chunk.

SIGNIFICANCE AND IMPLICATIONS OF THE RESULTS

The key finding from the personalized content chunk framework is the high improvement of web performance for personalized web pages in page response times. This would make the personalized content chunk framework a promising option for personalized web scenarios. As the web is increasingly adopting personalization scenarios personalization content chunk can be adopted in those scenarios.

THREATS TO VALIDITY

We conducted the experiments for 40 pages with maximum of 1000 concurrent users load. The performance improvement and resource behavior was observed in this setup. The performance numbers might change with increase in content volume and with increased concurrent user load. The experiments need to be conducted for those parameters.

Frequency of updates to personalized content is another factor that might impact the performance numbers. If the personalized content is frequently updated, the frequency of calls would impact the overall performance.

FUTURE SCOPE OF IMPROVEMENTS

In the current framework, the content authors need to be trained or be made aware of the structure of the placeholder in the personalized content spot. This is one key areas of improvement for the personalized content chunk framework.

Another enhancement to the personalized content chunk framework is to automatically convert the existing personalized web page into personalized content chunks and personalized content spots.

CONCLUSION

In this research paper, we have proposed a novel personalized content chunk framework that would efficiently manage, cache the dynamic information of the personalized web through personalized content spots. We discussed various components such as HTML tags, JSON service to create the personalized pages. We elaborated the framework components to achieve this. We built a prototype application to quantify the performance improvements and as per our experiments we were able to notice 30% improvement in response time using personalized content chunk framework.

REFERENCES

Brodie, D., Gupta, A., & Shi, W. (2004). Keyword-based fragment detection for dynamic web content delivery. *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters WWW Alt. '04*. doi:10.1145/1013367.1013444

Bruck, P.A., Motiwalla, L., & Foerster, F. (2012). Mobile learning with micro-content: a framework and evaluation. *Proceedings of the 25th Bled eConference eDependability: Reliable and Trustworthy eStructures, eProcesses, eOperations and eServices for the Future* (pp. 17-20).

Candan, K. S., Agrawal, D., Li, W.-S., Po, O., & Hsiung, W.-P. (2002) View Invalidation for Dynamic Content Caching in Multi-tiered Architectures. *Proceedings of VLDB '02.*

Cao, P., Zhang, J., & Beach, K. (1998). Active Cache: Caching Dynamic Contents (Objects) on the Web. *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98)*, the Lake District, England doi:10.1007/978-1-4471-1283-9_23

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. doi:10.1145/1064340.1064343

Challenger, J., Iyengar, A., & Dantzig, P. (1999). A Scalable System for Consistently Caching Dynamic Web Data. *Proceedings of IEEE INFOCOM '99*.

Challenger, J., Iyengar, A., Witting, K., Ferstat, C., & Reed, P. (2000). Publishing System for Efficiently Creating Dynamic Web Content. *Proceedings of IEEE INFOCOM '00*.

Challenger, J., Iyengar, A., Witting, K., Ferstat, C., & Reed, P. (2000). A publishing system for efficiently creating dynamic Web content. *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. doi:10.1109/INFCOM.2000.832259

Challenger, J., Iyengar, A., Witting, K., Ferstat, C., & Reed, P. (2000). A Publishing System for Efficiently Creating Dynamic web Content. *Proceedings of IEEE INFOCOM*.

Chan, M. C., & Woo, T. W. C. Cache-Based Compaction (1999): A New Technique for Optimizing Web Transfer. *Proceedings of INFOCOM '99.*

Christos, B., Vaggelis, K., & Ioannis, M. (2004). Web page fragmentation for personalized portal construction. *Proceedings of the International Conference on Information Technology: Coding and Computing ITCC '04*. doi:10.1109/ITCC.2004.1286475

Datta, K. Dutta, H. Thomas, D. VanderMeer, Suresha, and K. Ramamritham. (2002) Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation. *Proceedings of SIGMOD'02.*

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

Douglis, F., Haro, A., & Rabinovich, M. 1997, December. HPP: HTML Macro-Preprocessing to Support Dynamic Document Caching. *Proceedings of the USENIX Symposium on Internet Technologies and Systems* (pp. 83-94).

Galletta, D. F., Henry, R., McCoy, S., & Polak, P. (2004). Web site delays: How tolerant are users? *Journal of the Association for Information Systems*, *5*(1), 1.

Griffin, W., Jones, B., & Lee, S. K. (2005), U.S. Patent No. 11/192,791. Washington, DC: *U.S. Patent and Trademark Office*.

Ioannis, M., Vaggelis, K., & Christos, B. (2004). Web Page Fragmentation and Content Manipulation for Constructing Personalized Portals. In Advanced Web Technologies and Applications, LNCS (pp. 744-754). doi:10.1007/978-3-540-24655-8_81

Jang, I., Suh, D., & Pack, S. (2014). Minimizing content download time in mobile collaborative community. *IEEE International Conference on Communications (ICC).* doi:10.1109/ICC.2014.6883697

Khaing, A., & Thein, N. L. (2005). Efficiently Creating Dynamic Web Content: A Fragment Based Approach. *Proceedings of the 6th Asia-Pacific Symposium on Information and Telecommunication Technologies*. doi:10.1109/APSITT.2005.203648

Maghoul, F. and Yiu, P. and Davis, M. and Athsani, A. and Yi, J. (2008). U.S. Patent No. 12/240,323. Washington, DC: U.S. Patent and Trademark Office.

Mohapatra, P., & Chen, H. (2001). A Framework for Managing QoS and Improving Performance of Dynamic Web Content. *Proceedings of GLOBECOM '01.* doi:10.1109/GLOCOM.2001.966219

Naaman, M., Garcia-Molina, H., & Paepcke, A. (2003). Evaluation of ESI and Class-Based Delta Encoding. *Proceedings of WCW* '03.

Nagler, W., Ebner, M., & Sherbakov, N. (2007, September 26-28). Flexible teaching with structured micro-content-How to structure content for sustainable multiple usage with recombinable character. *Proceedings of the Conference ICL2007.* Kassel University Press.

Ramaswamy, L., Iyengar, A., Liu, L., & Douglis, F. (2004). Automatic detection of fragments in dynamically generated web pages. *Proceedings of the 13th Conference on World Wide Web WWW '04*. doi:10.1145/988672.988732

Schmiedl, G., Seidl, M., & Temper, K. (2009, September). Mobile phone web browsing: a study on usage and usability of the mobile web. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (p. 70). ACM.

Shivakumar, S. K. (2014). *Architecting High Performing, Scalable and Available Enterprise Web Applications*. Morgan Kaufmann.

Shivakumar, S. K. (2015). *A complete guide to portals and user experience platforms*. CRC Press. doi:10.1201/b19182

Souders, S. (2009). *Even Faster Web Sites: Performance Best Practices for Web Developers*. O'Reilly Media.

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.*

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

Shailesh K S has more than 15 years of industry experience, His interest areas are performance engineering and web portals.

Suresh Pachigolla Venkata is currently Director of School of Computer and Information Sciences at Indira Gandhi National Open University. His research areas include Software Engineering and Mobile Technologies.