Dynamic Web Development

Pratik Ghule[1], Vrushabh Doshi[2], Kishor Binwade[3], Siraj Bagwan[4]
Department of Computer Engineering
[1,2,3,4] Student, Dr. D. Y. Patil School of Engineering, Pune Savitribai Phule Pune University, Pune.

Abstract- *The absence of benchmarks for websites with dynamic content has been a significant impediment to analysis during this space. we have a tendency to describe 3 benchmarks for evaluating the performance of websites with dynamic content. Then benchmarks model 3 common kinds of dynamic content websites with wide varied application characteristics: an internet bookshop, Associate in Nursing auction web site, and a bulletin hoard. For the net bookshop, we have a tendency to use the TPCW specification. For the auction web site and therefore the bulletin hoard, we offer OUT own specification, sculptural once ebay.com and slahdot.org, severally. for every benchmark we have a tendency to describe the look of the information and therefore the interactions provided by the net server.*

We have enforced these 3 benchmarks with a spread of ways for building dynamic-content applications, as well as java, Java servlets . all told cases, we have a tendency to use ordinarily used ASCII text file code. we have a tendency to additionally give a consumer ape that permits a dynamic content internet server to he driven with varied workloads. Our implementations square measure on the market freely from our computing device for different researchers to use

Keywords- ID, GUI. graphical Web pages

I. INTRODUCTION

There square measure variety of committal to writing languages out there so as to create websites. thus there's have to be compelled to understand anyone committal to writing language to create an internet page. This innovative code application permits users to create websites while not knowing any committal to writing language. it's specifically designed for the interior use for firms. This code helps to build/design graphical websites. This code is embedded in an exceedingly web site for skilled use, and can be out there for patrons to tailor websites as per their want. this method helps the user to create web content with effective graphical interface. User doesn't ought to work specifically for graphical user interface. System can show varied webpage templet user will choose any templet consistent with his preference. this method can save time of the user and can facilitate the user to target main practicality needed in their webpage. this method can offer user with skilled webpage templates they will choose the templates supported the need and practicality. Here during this system user will build whole web site by simply choosing the content and pictures needed in their webpage. User will style every and each web content in their websites consistent with their preference by choosing the content and pictures. User will even specify position of the content to be placed. once user logins to the system, system can show varied templates to the user. User must choose the templet. System can show varied pages of that templet. By clicking on explicit page he should specify the content and pictures that must be placed within the webpage. User will place the content and might read the templet at the same time. Once the user clicks onto the submit button system can generate the web site. System can send code and .rar file to the user's email ID. User will build custom websites simply exploitation this application. User doesn't ought to understand any committal to writing languages to create websites. With the assistance of this method user will work on main practicality needed in their web content.

II. METHODS

We have implemented these three benchmarks with a variety of methods for building dynamic-content applications, including , Java servlets . In all cases, we use commonly used open-source software. We also provide a client emulator that allows a dynamic content Web server to be driven with various workloads. Our implementations are available freely from our Web site for other researchers to use.

1) Translation engine - translating your templates along with prescribed properties and values into actual html page,

2} Rule engine - applying validations, evaluating expressions, executing actions etc.. specified against html components.

3) UX engine - applying specified cosmetics on HTML components

4) JSF, SPRING - data handling, action executor framework

5) XML - data storage and conversion.
S = { U , K,}
Input:
LT, P
1] Set of users
U = {U1, U2, U3, U4 . . . . . Un}

2] LT = log file for each user contains user profile attributes on our site .

3] Set the attribute which is defined in web site that key word related tweets to extracted .

A ={a1,a2,a3,a4,. . . . . an}
4] set of web templates web pages
W = {w1 , w2, w3, w4, ... wn }
Process
UA = user Attributes .
UA=(UA 1, UA 2, UA 3…… UA n)
UR= User requirements .
UR =(UR 1, UR 2, UR 3….. UR n)
K=Web Translation engine
K=(K1,K2,K3……Kn)
UX= UX engine
UX =(UX 1, UX 2, UX 3….. UX n)

Output:
P= {PD, PT, Pv, PL}
Pp = {PR}

III. LITERATURE SERVE

[1] An subject area analysis of Java TPC-W, the use of the Java programing language for implementing server-side application logic is increasing in quality, nevertheless there's little legendary concerning the subject area needs of this rising industrial work. we have a tendency to gift a close characterization of the dealings process Council's TPC-W net benchmark, enforced in Java. The TPC-W benchmark is meant to exercise the online server and dealings process system of a typical e-commerce computer. we've enforced TPC-W as a set of Java servlets, ANd gift an subject area study particularization the memory system and branch predictor behavior of the work. we have a tendency to additionally measure the effectiveness of a coarse-grained multithreaded processor at increasing system output victimization TPC-W and different industrial workloads. we have a tendency to live system output enhancements from 8 May 1945 to forty first for a 2 context processor, and twelve-tone music to hour for a four context uniprocessor over a single-threaded uniprocessor, despite weakened branch prediction accuracy and cache hit rates.

[2] Exploring Processor style choices for Java-Based Middleware Java-based middleware may be a speedily growing work for high-end server processors, significantly Chip Multiprocessors (CMP). to assist architects style future microprocessors to run this vital new work, we offer a close characterization of 2 standard Java server benchmarks, ECperf and SPECjbb2000. we have a tendency to initial estimate the number of instruction-level correspondence in these workloads by simulating a really wide issue processor with good caches and ideal branch predictors. we have a tendency to then determine performance bottlenecks for these workloads on a a lot of realistic processor by by selection idealizing individual processor structures. Finally, we have a tendency to mix our findings on accessible ILP in Java middleware with results from previous papers that characterize the availability of TLP to analyze the optimum balance between ILP and TLP in CMPs. we discover that, like different industrial workloads, Java middleware has solely a little quantity of instruction-level correspondence, even once run on terribly aggressive processors. once run on processors resembling presently accessible processors, the performance of Java middleware is restricted by frequent traps, address translation and stalls within the memory system. we discover that SPECjbb2000 differs from EC perf in 2 significant ways: (1) the performance of EC perf is affected {much a lot of|far more |rather more |way more} by cache and TLB misses throughout instruction fetch and (2) SPECjbb2000 has more memory-level parallelism.

[3] Web Caching and Zipf-like Distributions: proof and Implications This paper addresses 2 unresolved problems concerning net caching. the primary issue is whether or not net requests from a hard and fast user community square measure distributed in keeping with Zipf's law [Zip29]. many early studies have supported this claim [Gla94, CBC95, ABCdO96] whereas different recent studies have urged otherwise [NHo + ninety eight, ACC +98]. The second issue relates to variety of recent studies on the characteristics of net proxy traces, that have shown that the hit-ratios and temporal vicinity of the traces exhibit sure straight line properties that square measure uniform across the various sets of the traces [CI97, RV98, DMF97, GB97a, KLM97]. particularly, the second issue is whether or not these properties square measure inherent to net accesses or whether or not they square measure merely AN unit of the traces. a solution to those unresolved problems can facilitate each net cache resource coming up with and cache hierarchy style. we have a tendency to show that the answers to the 2 queries square measure connected. we have a tendency to initial investigate the page request distribution seen by net proxy caches victimization traces from a spread of sources. we discover that the distribution doesn't follow Zipf's law exactly, however instead follows a Zipf-like distribution with the exponent variable from trace to trace. moreover, we discover that there's (i) a weak

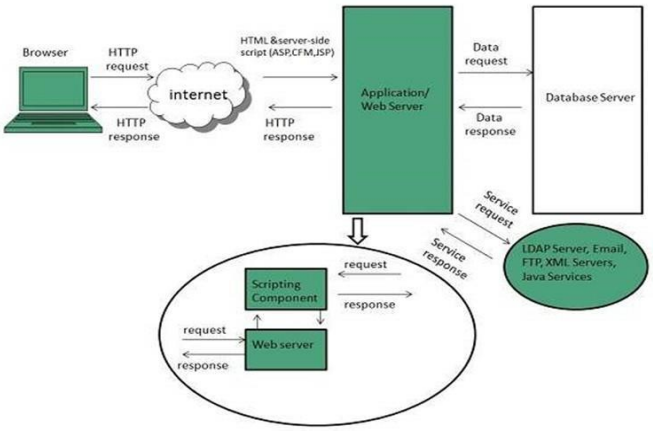
corr elation between the access frequency of an online page and its size and (ii) a weak correlation between access frequency and its rate of amendment.

[4] Web Caching and Zipf-like Distributions: proof and Implications. This paper addresses 2 unresolved problems concerning net caching. the primary issue is whether or not net requests from a hard and fast user community square measure distributed in keeping with Zipf's law many early studies have supported this claim whereas different recent studies have urged otherwise.The second issue relates to variety of recent studies on the characteristics of net proxy traces, that have shown that the hit-ratios and temporal vicinity of the traces exhibit sure straight line properties that square measure uniform across the various sets of the traces particularly, the question is whether or not these properties square measure inherent to net accesses or whether or not they square measure merely AN unit of the traces. a solution to those unresolved problems can facilitate each net cache resource coming up with and cache hierarchy style.

We show that the answers to the 2 queries square measure connected,. we have a tendency to initial investigate the page request distribution seen by net proxy caches victimization traces from a spread of sources. we discover that the distribution doesn't follow Zipf's law exactly, however instead follows a Zipf-like distribution with the exponent variable from trace to trace. moreover, we discover that there's solely (i) a weak correlation between the access frequency of an online page and its size and (ii) a weak correlation between access frequency and its rate of amendment. we have a tendency to then take into account an easy model wherever the online accesses square measure freelance and therefore the reference likelihood of the documents follows a Zipf-like distribution.

IV. PROPOSED SYSTEM

As per today's fast moving world towards internet and in need of faster webpages access, our motivation is to develop a fine and better framework which provide a ease for generating dynamic webpages for any enterprise level, middle level organization, low level firms with just \ Dynamic Web Generation



click of their finger. 21

1) Translation engine - translating your templates along with prescribed properties and values into actual html page

2) Rule engine applying validations, evaluating expressions, executing actions etc.. specified against html components.

3) UX engine - applying specified cosmetics on HTML components

4) JSF, SPRING - data handling, action executor framework

5) XML - data storage and conversion.

V. RESULT AND DISSECTION:

We have implemented these three benchmarks with a variety of methods for building dynamic-content applications, including , Java servlets . In all cases, we use commonly used open-source software. We also provide a client emulator that allows a dynamic content Web server to be driven with various workloads. Our implementations are available freely from our Web site for other researchers to use.

We are making the source code of our implementations freely available on our Web site. We hope other researchers will use them, making performance results of dynamic content Web sites more reproducible and easier to compare. The world of today is characterized by tremendous opportunities for personal and professional development, firstly due to the progress in transportation technology allowing people to relocate rapidly from one geography to another and secondly due to easy access to information and knowledge through the internet. These world conditions of global mobility together with available web resources and tools can strongly contribute to the development of the unique talents of individuals, which is a critical factor for one's future career and life success.

VI. CONCLUSION

We have implemented the three dynamic content benchmarks and a workload generator tool that allows us to vary the workload driving the dynamic content server. We have used our implementations to carry out a bottleneck characterization of the benchmarks. Different benchmarks show different bottlenecks: the database CPU for the online bookstore, and the Web server CPU for the auction site and the bulletin board. Complex queries cause the database CPU to be the bottleneck for the online bookstore. In contrast, the queries for the other applications are simpler. We are making the source code of our implementations freely available on our Web site. We hope other researchers will use them, making performance results of dynamic content Web sites more reproducible and easier to compare.

REFERENCES

[1] Bill Marshall Spiderwriting.co.uk. (2017). Static v Dynamic Website Design – Spider Writing Web Design. [online] Available at: http://www.spiderwriting.co.uk/static-dynamic.php [Accessed 10 Nov. 2017].

[2] Chris K. Codeconquest.com. (2017). Static vs. Dynamic Websites. [online] Available at: http://www.codeconquest.com/website/static-vs-dynamic-websites/ [Accessed 10 Nov. 2017].

[3] Mayers, D. (2017). What is the difference between a back end and a front-end web programming language? - Kevin Chisholm - Blog. [online] Kevin Chisholm - Blog. Available at:https://blog.kevinchisholm.com/web-development/difference-between-back-front-end-language/ [Accessed 10 Nov. 2017].

[4] Carey Wodehouse Home, H., Development, W. and Experience, F. (2017). What is Client-Side Scripting? Choosing the Scripting Languages for your Web Application. [online] Hiring — Upwork. Available at: https://www.upwork.com/hiring/development/how-scripting-languages-work/ [Accessed 10 Nov. 2017].

[5] Asha Mandava and Solomon Antony Murray , A review and analysis of technologies for developing web applications state University Murray, Kentucky.

[6] Carlos Castillo ,A framework for the design and AND IMPLEMENTATION OF WEB SITES University of Chile / Newtenberg Digital Publishing Ltd. 2120 Blanco Encalada /10 Estado 3rd floor, 2002.

[7] Wutthichai Chansuwath, A Model-Driven Development of Web Applications Using AngularJS Framework, 29 June 2016.

[8] OMG, Documents Associated With XML Metadata Interchange (XMI), Version 2.4.2[Online].Availa http://www.omg.org/spec/XMI/2.4.2/. Last Accessed: 30 Mar 2016.

[9] N. Jain, P. Mangal, and D. Mehta, "AngularJS: A modern MVC framework in JavaScript, J. Global Research in Computer Science, vol. 5, no. 12, 2015, pp. 17-23.