Optimization of Waiting Time in Complex RIA by Combined Pseudo-Hierarchical-Lazy-Loading Algorithm

Yun Chen, Ai Zheng, Shi Jing, and Qi Huang
School of Energy Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
Email: justice1200@126.com; {812807433, 835995993}@qq.com; huangqi@uestc.edu.cn

Abstract —In complex RIA (Rich Internet Application) environment, the browser may have to load and process very large amount of data, which may result in long waiting time and bad waiting experience. This may even cause to lose customers for commercial sites. To avoid such problems, it is necessary to replace the natural logic algorithms with some optimization algorithms. In this paper, a combined algorithm, which combines pseudo-hierarchical algorithm and lazy loading algorithm, is proposed. The theoretical analysis is performed. Then a test is designed and the performance of the combined algorithm is compared with that of natural logic algorithm. The results shows that the combined algorithm can give users a better waiting experience by rebalancing loading time and operation response time, matching the analysis very well.

Index Terms—RIA, pseudo-hierarchy, lazy loading, waiting experience

I. Introduction

Rich Internet Application (RIA) is one product of the modern Internet which is characterized by having a rich data model and rich interface elements and it can give users a highly interactive user experience [1]. As RIA technology becomes more mature, there have been many huge-scale RIAs. And with more extensive applications, RIA is involving many disciplines and fields [2]. A large 3D visualization online management system and 3D online web games are the most typical complex RIA and are able to demonstrate the complexity and the difficulty of RIA. When an instance of RIA becomes more and more complex, especially accompanied by loading large graphic resources and processing a lot of data, the problems of loading time and operation response time will come out and have a negative influence on waiting experience directly of users. For commercial companies, this may means incalculable customer loss and economic loss [3].

Fortunately, with the development of RIA optimization theory and technology, various optimization algorithms have been proposed [4] and involved in RIA system architecture, RIA server, RIA browser-side code, RIA internal communication and so on [5]. But with the emergence of 3D visualized RIA which has ultra large and complex data structures, RIA optimization still requires more effort. In this paper, the algorithm proposed based on combining pseudo-hierarchical algorithm and lazy loading algorithm is a rebalancing optimization of RIA for browser-side code. The lazy loading algorithm is a classical algorithm proposed decades ago [6], its core idea is to quickly load some basic data to complete starting application, which can accelerate the application starts, thus reducing the waiting time to start, and until now it's still widely used [7]. Pseudo-hierarchy algorithm is a solution by decoupling and managing the logic data structure of applications, which is widely used in data management for applications with complex data structures [8]. The algorithm rewrite the dependency of data nodes, so that each logical node can no longer be tightly coupled to their parents, children or siblings, thus the nodes can be individually loaded and managed thus forming a pseudo-hierarchy structure.

In short, the algorithm proposed in this paper is an attempt to optimize the algorithm of browser-side code, and the final goal is to optimize the waiting experience of users for complex RIA.

II. Analysis of Existing Algorithms

For complex RIA, each data node require ~~a~~ certain load time that cannot be ignored, and because RIA is running in browsers, the time for processing node cannot be ignored neither in general [9]. Under the precondition of the above, suppose there is a parent-child hierarchical RIA, and the logic layer of the RIA is K, each layer has δ_i nodes, then assume that the expected time $t(i, j)$ for loading and processing each node can be represented by the follow formula:

$$t(i, j) = t_{load} + t_{process}(i, j) . \quad (1)$$

In the above equation, *i* and *j* represent the *j*-th node in *i*-level, the total initial load time is T_s . In order to characterize the efficiency of response speed in RIA, assuming that the nodes of last level have siblings, then the expected time for accessing the nodes in the same level is T_u [10]. Then to some extent, the startup time of RIA can be defined as T_s , and the operation response time of RIA can represent by T_u . Therefore the key of optimizing user waiting experience converted to optimization of T_s and T_u .

Manuscript received July 6, 2015; revised November 17, 2015.
Corresponding author email: justice1200@126.com
doi:10.12720/jcm.10.11.859-863

A. Analysis of Natural Logic Algorithm

Under the above premise，assume that the system uses a doubly linked list to store the relationship between nodes, the RIA need to load all nodes to finish initialization, then T_s of the system is：

$$T_s = \sum_{i=1}^{K}\sum_{j=1}^{\delta_i} t(i,j) + T_{sys} \tag{2}$$

where T_{sys} is the time for system basic start which is indispensable and it is determined by the realization method of RIA and machine environment for RIA [11].

As all nodes in the system have been loaded, the time for accessing the related nodes in the same level is negligible, as follow:

$$T_u \approx 0 \tag{3}$$

B. Analysis of Lazy Loading Algorithm

Under the same premise, using lazy loading algorithm as initial loading algorithm, when all nodes in the first level are loaded and processed, the results can be display to users. So T_s can be expressed as

$$T_s = \sum_{j=1}^{\delta_1} t(1,j) + T_{sys} \tag{4}$$

At this time, the nodes in last level are neither be loaded or processed. Because the structure of the RIA is parent-child hierarchy, for the nodes in last level, if a node wants to visit another node in the same level, it will be completed until the whole family of the target node is loaded, so the operation time can be expressed as

$$T_u = f(\delta)\sum_{i=1}^{K}\sum_{j=i}^{\delta_i} t(i,j) \tag{5}$$

where $f(\delta)$ is a random variable that indicates the percentage of related nodes need be loaded to finish operation, where $\delta = (\delta_1, \delta_2, \cdots \delta_k)$. So it's predicted that if system uses conventional lazy loading algorithm separately, the initial load time will be decreased greatly [12], but because the hierarchy of system uses the natural logic, it will take much time for nodes in last level to visit the nodes in the same level. Moreover the system is more complex, the value of $f(\delta)$ is closer to 1 and the value of T_u will be larger.

C. Analysis of Pseudo-Hierarchical Algorithm

Pseudo-hierarchical algorithm has been proposed for many years, and was used in network optimization widely [13].

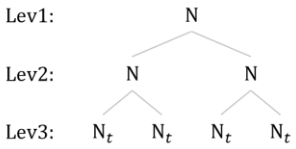


Fig. 1. natural logic structure of RIA nodes

For natural hierarchical logic RIA, the data structure of a simple RIA is a traditional binary tree which can be represented in Fig. 1.

The characteristic of the structure is that child nodes cannot exist without parent nodes. So in the case of lazy loading algorithm, a complete branch in a tree has been loaded and processed completely. But if one leaf node attempts to access the related nodes in the same level, operation response time will follow the rules mentioned in B , because another tree have to be loaded.

To solve this problem, pseudo-hierarchical algorithm packages the structure of RIA once more. Each node in one level will be set as a new combination of variables to generating another data structure, then the structure of the RIA is shown in Fig. 2.

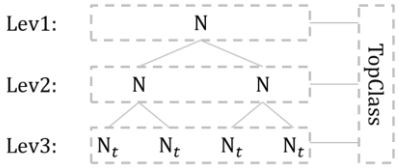


Fig. 2. structure of pseudo-hierarchical algorithm

In this case the relationship between nodes is no longer hierarchy only, thereby every node can exist independently without considering the relationship of parents or siblings, so that each node can ignore the parent-child relationship and exist in isolation, and this is generalized pseudo-hierarchical algorithms. The RIA can get rid of the strict logic relationship [14], [15], and the structure of the RIA transverse from the broad into a generalized longitudinal.

Therefore, in the context of the above, if the RIA uses the pseudo-hierarchical algorithm individually, the system's initial loading will be almost unchanged [16]. So T_s satisfies the equation (2), and expected time for accessing nodes in the last level the RIA T_u will basically remain unchanged too, as satisfying the equation (3).

Thus it can be concluded that if taking an implantation of a separate pseudo-hierarchical optimize algorithm only, the RIA will not be substantially optimized [17].

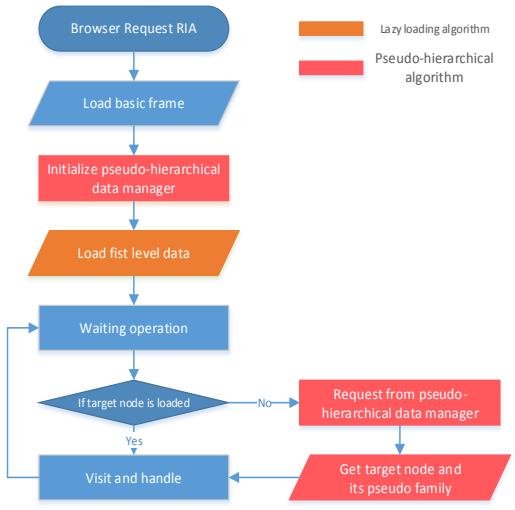


Fig. 3 Flow chart of combined algorithm

III. Analysis on Combined Algorithm

Under the same background, it will be analyzed what's different if using pseudo-hierarchical algorithm combined with lazy loading algorithm to replace natural logic algorithm. At first the flow chart of the combined algorithm is shown in Fig. 3.

Unlike common RIA, for a RIA implanted combined algorithm, it will initialize the pseudo-hierarchical data manager and load first level data of the RIA for startup. When a operation comes, the RIA will request data from pseudo-hierarchical data manager installed of analyzing the whole data structure of the RIA as using lazy loading algorithm only, and the pseudo-hierarchical data manager will response for quickly returning the target node without analyzing the whole family of the target node.

So it can be assumed that when RIA initializes, the processing time $t_{process}(i, j)$ for each node in a minor will increase a little because the initialization of pseudo-hierarchical data manager comparing to equation (1), but contrasted with the load time $t_{load}(i, j)$, its influence is negligible, because node resources are generally larger in complex RIA, so the node loading time often occupies a major time. The results of the following experiment will also support this view. Therefore:

$$t'(i, j) \approx t(i, j) \tag{6}$$

At this time, the initial load time of the system T_s meets with the initial load time by lazy loading algorithm in the II-B, i.e., T_s satisfies the (4).

Meanwhile, T_u of RIA is affected by pseudo hierarchy and no unnecessary loading phenomenon occurs, so that the expected time for loading and processing the nodes in the last level end of the layer satisfies the formula:

$$T_u = \sum_{j=1}^{\delta_k} t(K, j) / \delta_k . \tag{7}$$

Since the magnitude of one single $t(i, j)$ is nearly millisecond, it is general that T_u is also in this range, so users will not wait long as like II-B.

If we only use natural logic algorithms, the initial load time is so long that users cannot accept; Using lazy loading algorithm can solve the problem that the initial load is slowly, but in logic tightly coupled RIA it prone to emerge unnecessary loading issues, so that user may wait too long to operate; Using pseudo-hierarchical algorithm only is of no practical significance, and it will increase the size of RIA. By combining pseudo-hierarchical algorithm and lazy loading algorithm, we can realize generalized longitudinal structure of RIA which can reduce the initial loading time relative to the natural logic algorithm compared with the results of the pure lazy loading algorithm. It also can speed up the response time for user operation. In summary, combining of pseudo-hierarchical algorithm and lazy loading algorithm is a re-balancing optimization for T_s and T_u.

IV. Algorithm Validation and Results Analysis

A. Algorithm test Environment

Algorithm was tested and verified in a typical RIA realized with Flex and J2EE which is established based on Adobe Flex technology. A series of hierarchies and nodes are loaded, and the number of levels and nodes are manageable. The node includes 3DS data and many other types of data. The volume of the node is huge, it has 4 levels, and thereby we can simulate the real RIA system approximately. In the realization of RIA, the structure of this system is show in Fig. 4.

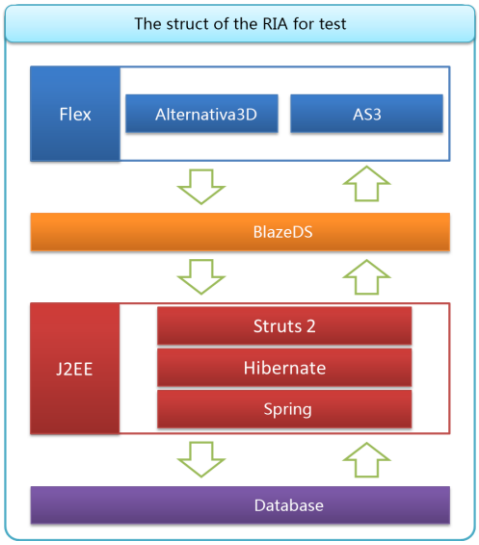


Fig. 4. Technology architecture of test RIA

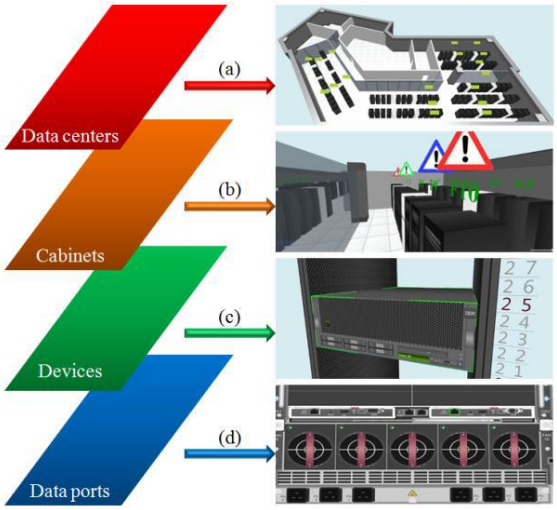


Fig. 5. Data levels of the test RIA

The test RIA is a typical 3D visualized data management system for datacenters. The main function of the RIA is to simulate a real datacenter in a 3D scene and manage the data visually. The data of the RIA are individuated into 4 levels as Fig. 5. So the startup time T_s of the RIA can be described by finishing the show of the datacenters. The operation response time T_u can be simulated by the time of a data port getting the information of the peer port which stands by the max

operation waiting time, because for lazy loading algorithm only, the operation must load the whole peer cabinet, devices and ports to finish the operation.

During the experiment, the database server and J2EE servers were placed on two standard servers. CPU of the server is 12-core at 2.4 GHz, Memory of the server is 8GB. Server and client are in the same 100-M network. Experiments were carried out. We deployed natural algorithm code the lazy loading algorithm, pseudo-hierarchical algorithm code and combine algorithm code on the server respectively. We add a timer in the code to record the start time and operation response time of different nodes in the system.

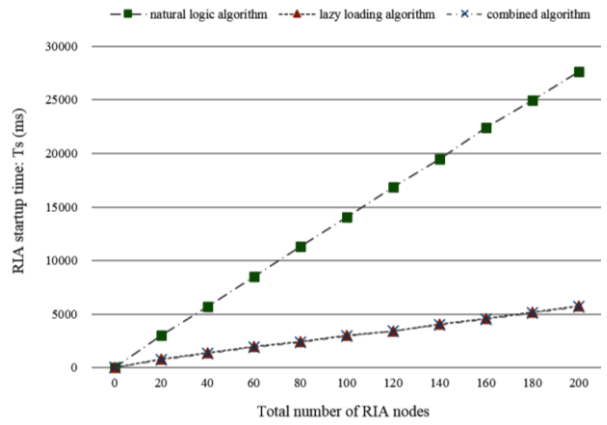


Fig. 6. Result of tests on startup time of RIA

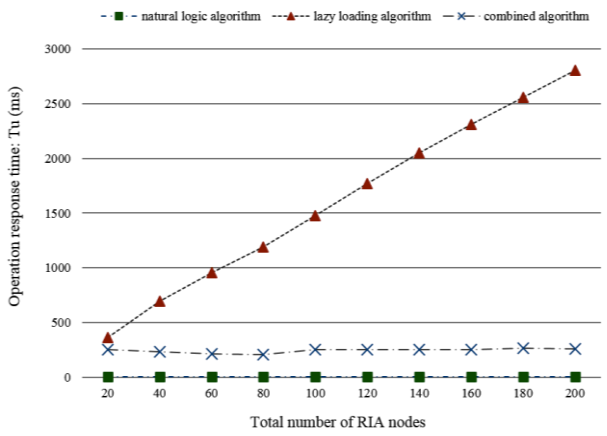


Fig. 7. Result of tests on operation response time of RIA

B. Analysis of Results

Timer in RIA server records the time of different algorithm. The load time is from the average of 20 times start experiment, every experiment was conducted in response to unrelated operation more than 30 times, taking the average as load time. The result of the experiment is shown in Fig. 6 and Fig. 7.

As can be seen from Fig. 6, using natural logic algorithms, the startup time of RIA will proportional increase as the number of nodes at a higher speed. The relationship is in keeping with equation (2). Lazy loading algorithm and combined algorithm will grow at a relatively lower rate of growth comparing to the speed of natural logic algorithms. The ratio is related with the level of logical node, satisfying the equation (4). Average operation time of three above-mentioned algorithms is shown in Fig. 7. It can be seen that as lazy loading algorithm logic tightly coupled constraints, the growth of operating time is rapid, the relationship between the number of nodes and operation is linear approximately, proving equation (5) is reasonable. However, the natural algorithm, the response time of operation is near zero, in line with forecast by equation (3). The response time of combine algorithm is stable and nearby constant, that is consistent with the results of equation (6).

In summary, the experimental results show that combination algorithm, as expected, improves the initial loading speed by the expense of part of the response speed. It is proved that combination algorithm is optimization for a typical complex RIA. Experimental analysis of the final data is consistent with the mathematical model, which proves the effectiveness of combination algorithm.

V. Conclusion

Pseudo-hierarchical algorithm combined with lazy loading algorithm can effectively rebalance the initial load time and operation time through internet analysis and experiments. The initial load time is greatly reduced while response operation time within an acceptable range. In a word, the overall waiting time of RIA is improved greatly. This algorithm is an optimization for RIA browser-side code, and is suitable for typical complex RIA such as 3D visualization management applications and 3D online web games. The findings may offer some new references for future research on optimization of RIA.

References

[1] B. Marco and P. Fraternali, "Large-scale model-driven engineering of web user interaction: The webml and webratio experience," *Science of Computer Programming*, vol. 89, pp. 71-87, 2014.

[2] Q. Pascale, *et al*., "Using a rich internet application to teach histology," in *Proc. 12th European Congress of Digital Pathology*, 2014.

[3] N. Rostislav, V. Vožnílek, and M. Balun, "Rich internet application for crisis management support–case study on Floods in Olomouc City," in *Proc. Fifth International Conference on Innovations in Bio-Inspired Com puting and Applications*, 2014.

[4] Ali, M. Montaz, "Some modified stochastic global optimization algorithms with applications," Loughborough University of Technology, 1994.

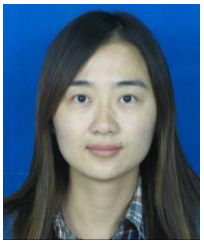
[5] B. Masiar, A. Gallidabino, and C. Pautasso, "Liquid stream processing across web browsers and web servers," in *Engineering the Web in the Big Data Era*, Springer International Publishing, 2015, pp. 24-33.

[6] L. Sheng and G. Bracha, "Dynamic class loading in the Java virtual machine," *ACM SIGPLAN Notices*, vol. 33, no. 10, pp. 36-44, 1998.

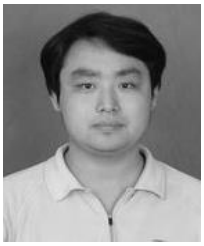
[7] L. Y. Yuan, *et al*., "A demonstration of rubato DB: A highly scalable NewSQL database system for OLTP and big data applications," in *Proc. ACM SIGMOD International Conference on Management of Data*, 2015.

[8] J. T. Zhou, *et al*., "hierarchical model of network planning capability for the construction method," *Computer Integrated Manufacturing Systems*, vol. 20, no. 8, pp. 1819-1826, 2014.
[9] M, H. Hu and P. Bian, "Human-Machine interface: Design principles of interactive waiting in Web and App," in *Proc. 2nd International Conference on Soft Computing in Information Communication Technology*, 2014.
[10] N. F. Fui-Hoon, "A study on tolerable waiting time: How long are Web users willing to wait?" *Behaviour & Information Technology*, vol. 23, no. 3, pp. 153-163, 2004.
[11] P. Pascal and D. Fortin, "An optimal algorithm to recognize robinsonian dissimilarities," *Journal of Classification*, vol. 31, no. 3, pp. 351-385, 2014.
[12] C. Suryakant, *et al*, "Model-based rich internet applications crawling: menu and probability models," *Journal of Web Engineering*, vol. 13, no. 3-4, pp. 243-262, 2014.
[13] F. Marcello, N. Bertoldi, and M. Cettolo, "IRSTLM: An open source toolkit for handling large scale language models," *Interspeech*, 2008.
[14] F. Fouquet, *et al*., *An Eclipse Modelling Framework Alternative to Meet the Models@ Runtime Requirements*, Springer Berlin Heidelberg, 2012.
[15] B. Mark, *et al.*, "Data Analysis WorkbeNch (DAWN)," *Journal of Synchrotron Radiation*, vol. 22, no. 3, 2015.
[16] L. Nabil, B. Dhupia, and S. Rubab, "Review of cross-platforms for mobile learning application development," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 1, 2015.
[17] J. R. Alder and S. W. Hostetler, "Web based visualization of large climate data sets," *Environmental Modelling & Software*, vol. 68, pp. 175-180, 2015.

Yun Chen was born in Chengdu, Sichuan Province, in 1991. He received the B.S. degree in College of automation engineering from UESTC in 2013 and now working on a M.S. degree in Energy science and engineering college from UESTC. The main research direction is smart grid.



Ai Zheng was born in Tangshan, Hebei province, in1992. She received the B.S. degree in School of automation engineering from UESTC in 2014, and now works on a M.S. degree in Energy Science and Engineering College from UESTC. The main research direction is smart grid.



Shi Jing was born in Nanjing, Jiangsu province. He received the Ph.D. degree in Energy science and engineering college from UESTC in 2013. The main research direction is the smart grid information technology, automation.

Qi Huang was born in Guizhou Province in the People's Republic of China. He received his BS degree in Electrical Engineering from Fuzhou University, Fuzhou, Fujian, P.R. China, in 1996, MS degree from Tsinghua University, Beijing, China, in 1999, and Ph.D. degree from Arizona State University, Tempe, AZ, US, in 2003. He is currently a professor at University of Electronic Science and Technology of China (UESTC) and the Deputy Dean of School of Energy Science and Engineering, UESTC, and the Director of Sichuan State Provincial Lab of Power System Wide-area Measurement and Control. His current research and academic interests include power system high performance computing, power system instrumentation, power system monitoring and control, and integration of distributed generation into the existing power system infrastructure.