Incremental Configuration Update Model and Application in Sponsored Search Advertising

Wei Yuan[1(✉)], Pan Deng[1], Biying Yan[1], Jian Wei Zhang[2], Qingsong Hua[3], and Jing Tan[4]

[1] Institute of Software Chinese Academy of Sciences, Beijing, China
{814498287,30081046,352957054}@qq.com
[2] Beihang University, Beijing, China
1177926@qq.com
[3] Power System Research Centre, Qingdao University, Qingdao, China
8988596@qq.com
[4] Shenyang Artillery Academy, Shenyang, China
tanj_tanjing@163.com

Abstract. Sponsored search advertising is a significant product contributing impressive revenue to an internet search company. It serves for thousands of partners including owned properties and syndication partners. To boost revenue, configuration should frequently be changed to satisfy every partner's personalized requirements, partners on-boarding or off-boarding, features enabling or disabling, etc. To better address these demands and make them take effect rapidly in system, a model for incremental configuration update and its application in sponsored search are put forward in this paper to do configuration update automatically. The results in application in product system turn out to be encouraging demonstrating its efficiency, helping company reduce costs .

Keywords: Sponsored search advertising · Incremental Configuration Update Model · Data reload

1 Introduction

As one of the largest and fastest growing advertising channels [1], sponsored search is the delivery of relevance-targeted text (image and video ads are also included in large general search engines such as Google.com, Bing.com and Yahoo.com) advertisement as part of the search experience [2]. In Q1 of 2015 [3], 54.6 billion explicit core desktop searches which is free for consumers were conducted in the top 5 American search engines, indicating a robust 26 % growth for the US paid search market [4]. Moreover, advertising expenditures on sponsored search is forecast to grow to more than $110B in 2015. By contrast, other media format (like television, newspaper) advertising spending in the US will encounter a significant decline in growth. Obviously, sponsored search advertising is becoming a key or preferred market promotional tool for more and more enterprises and organizations.

J. Wan et al. (Eds.): Industrial IoT 2016, LNICST 173, pp. 70–78, 2016.
DOI: 10.1007/978-3-319-44350-8_8

In practice, all major sponsored search players use keyword auctions to allocate display space of advertising material alongside other functionality landing pages. Generally, the keyword auction is based on a pay-per-click model in which advertisers are charged only if their advertisement is clicked by a user [5, 6]. For a specific product or service, advertisers select suitable keywords under some professional or keyword recommendation tools likely to be searched by users in the search page. Meanwhile, advertisers need to provide a bid for each of those keywords indicating the amount of money they would pay for a click. When a query is searched, the engine matches the keywords of all advertisers again the keyword in this query and decides the most eligible ad in an auction which is achieved by a mechanism, namely a generalized second price auction [5].

Along with the crazy growth of sponsored search adverting users, the online serving system is also expanding rapidly to meet amount of personalized demands. Generally, this system usually contains a bunch of specific configuration to manage and serve for hundreds of thousands of customers. Configuration Management (CM) [7] (e.g., configuration version control, changes adoption) becomes more complicated due to continuous customer requirement changes. Configuration need to be updated frequently and most of requirements only impact part of the configuration as usual. This process should also not impact the online system without restarting the engine and come into effect in real time.

In this paper, an incremental configuration update model is abstracted to improve the velocity and efficiency for these frequency changes from practice in Sponsored Search. This model is also applied to automatic configuration update system offering 24×7 h reliable service.

The paper is organized in the following fashion. Our scheme proposed for incremental configuration update is detailed in Sect. 2. Section 3 describes the application in Sponsored Search. Some results and analysis are presented and discussed in Sect. 4. Finally, the paper concludes in Sect. 5.

2 Incremental Configuration Update Model

2.1 Elements of This Model

To achieve the practice requirements of pushing the only to-update parts for all kinds of configuration to production as automatically and rapidly as possible with less engineering resources, this paper abstracts the Incremental Configuration Update Model.

Figure 1 shows this model briefly. There are four mainly steps: (1) incremental configuration extractor; (2) quality verification; (3) cross-colo delivery; (4) delta configuration reload. The whole procedure can be described as follows: the requestor provides to-update configuration changes via specific portal which also can cooperate to do some verification; Then, the incremental configuration are derived according to the deriving strategy in step (1) and distribute the results to step (2) for quality verification; After all automatic verification pass and signed by the original requestor, step (3) deploys the incremental configuration to production environment of cross-colo; Finally, step (4) reloads the delta configuration to take effect without any server changes.

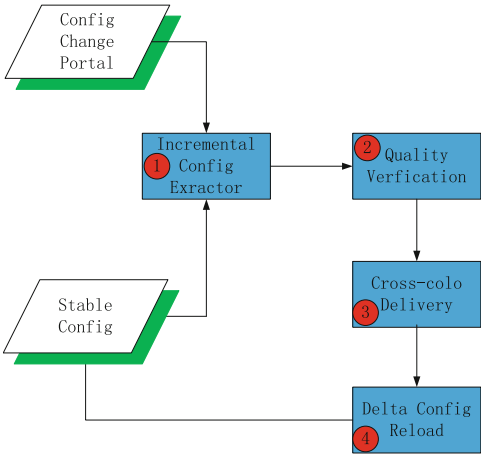

Fig. 1. Incremental configuration update model

2.2 Derive Delta Configuration

There are usually two dimensions for configuration data. One dimension is we called partner, which is a unique value to identify the various advertisers or publishers, and another dimension is we called feature, which is used to describe one special behavior for one or some partners. So in consequence, one partner may include one or more features, and one feature can belong to one or more partners.

In Table 1, "/search/amazon/xml/" represents the partner Amazon and it enables feature 'Sitelink' [10, 11] and partner Pinterest (/search/pinterest/xml/) enables feature 'Sitelink' and 'ImageAds' [12].

Table 1. Partner and feature

Index	Partner	Feature
1	/search/amazon/xml/	Sitelink
2	/search/pinterest/xml/	Sitelink
3	/search/pinterest/xml/	ImageAds

The incremental configuration is usually to express one or more features enabled/disabled for some partners, so the key part of updating incremental configuration is to divide the configuration and compare with the last stable configuration to extract the incremental parts. We can divide the configuration by dimension. Thus, it can be elicited to three methods.

(1) Dividing configuration by partner. In this situation, configuration data are stored in a whole file, and can be divided into many small files, and each of those files is used to indicate one partner configuration. Every partner configuration file contains the whole feature sets.

(2) Dividing configuration by feature. In contrast with the first method, the whole configuration file is divided into many small files according to features. Every feature configuration file contains all the partners who enable this feature.
(3) Dividing configuration by both partner and feature. In this method, each small configuration file only contains the setting for one feature on one partner. Obviously, this method leads to so many file pieces.

But, how to choose the proper dividing method? The goal should be to choose the one which creates minimum (most necessary) incremental configuration for each update. Based on this goal, we should consider following criteria.

a. Compare the size of partner set and feature set. Bigger size leads to more file pieces and impact more for each update.
b. Consider on the pattern for each update. If we immerse ourselves in developing on one new feature, the second method is more proper for each update. If the main activity for each day is to introduce new partner or enable/disable feature for some partners, the first method is a better choice.

2.3 Incremental Configuration Quality Verification

For an automatically updating system, continuous integration and self-verification are critical. The test automation and propagation automation tool can enormously make the verification easier and simpler. Continuous Integration (CI) [8, 9] is a continuous integration test tool and globally used in Internet company. CI job is comprised of abundant basic test cases and can be triggered by every code or configuration change. GDT is automatically propagation tool internally and commonly used for data deployment by service engineer (SE) team for production environment propagation [13, 14]. Once the changes are committed, CI job is triggered automatically. The job consists of following components.

(1) Basic test cases: functional test
(2) Integration test: aim to simulate the incremental configuration launch and verify the incremental configuration and launch. Establish the current production environment in CI and simulate incremental configuration propagation with tool like GDT. And the test queries are created automatically based on the incremental configuration and sent to test environment for integration.

With continuous improvements, the whole model could provide reliable verification for both incremental configuration and launch.

2.4 Cross-Colo Delivery

After the quality of the incremental configuration are verified, the next key component is to deliver the incremental configuration to different data centers. The reason we need cross-colo delivery is that the large-scale system runs in multiple data centers to support BCP or global service.

The key point for this step is to carry the data to the destinations quickly, consistently and under monitoring.

2.5 Configuration Data Reload Automatically

When the incremental configuration arrive at destination machines, the system needs to have the capability to reload it into machine memory and make it come into effect. The reloading should process smoothly, completely, and under monitoring.

(1) Smoothly means the system should not discontinue the service and then switch to serving with the new configuration for new requests.
(2) Completely means the system must reload the whole incremental configuration at once, and if there are some failures happened, the system must cancel the reloaded configuration and rollback to last stable status.
(3) Under monitoring means the whole process should be monitored and alerts should be sent out for both reloading success or not.

3 Application in Sponsored Search

In Sponsored Search engine, there are various configuration data, like partner configuration, system configuration, server configuration, etc. Generally, as partner business and server requirements, the configuration data need to change frequently and update to production as real-time and reliably as possible.

For partner configuration, we have been applying the incremental configuration update model in Sponsored Search. According to production online feedback, it can overwhelming shorten the partner configuration update cycle from one month to daily, even aiming to near real-time in the future; it has reduced engineering resource enormously with only Partners and Account Managers involved in this model and other work are finished by model automatically; The most important point is that it is more reliable because it can verify itself and get verification from requestor.

3.1 Partner Delta Configuration Deriving

In this step, method “Dividing configuration by partner” is used, because the partner size is much bigger than the feature size, and there are more requests to update the partner’s existed configuration than developing new feature.

Figure 2 illustrates how incremental partner configuration is derived automatically:

(1) Account Manager will collect requirements from all partners and advertisers, and finish the requirements on config change portal;
(2) Original partner config CI flattens partners configuration changes and hands it over to upcoming CI;
(3) Extractor CI divides the whole partner configuration file into single files according to partners, and compare them with production stable configuration to extract

incremental configuration, then pack it for next step Incremental Configuration Quality Verification.

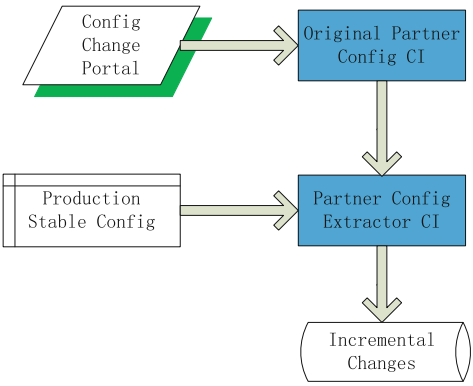

Fig. 2. Partner incremental configuration deriving

3.2 Partner Incremental Configuration Quality Verification

When Verification CI receives incremental configuration, it pushes them into verification runway which is imaged from production environment and send query for this partner configuration changes to verify. After machine verification, it sends result and query sample to Account Manager for launch staging via mail. This step can guarantee changes quality which meets the partners' requirements and system standard. Then, it delivers to next step Cross-Colo Delivery (Fig. 3).

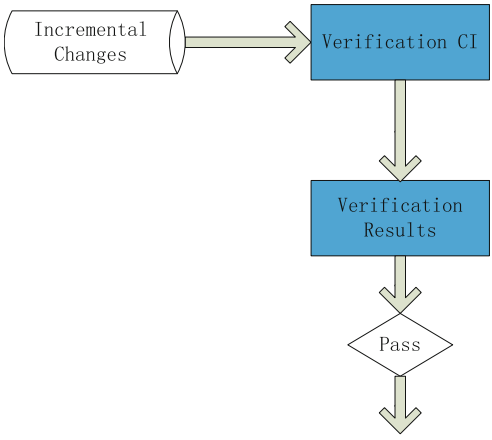

Fig. 3. Partner incremental configuration quality verification

3.3 Partner Incremental Configuration Cross-Colo Delivery

There are amount of tools to support file delivery in cross-colo in different companies. However, it is very crucial to choose proper tools and syncretize them into pipeline.

When the CI gets the successful signal of incremental configuration verification from previous step, it delivers the incremental configuration to different colos. For every launch trigger, it creates a new folder including the incremental configuration and then relink the production partner configuration to soft link to this new folder for following operations (Fig. 4).

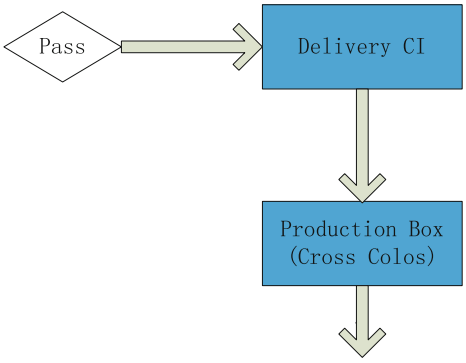

Fig. 4. Partner incremental configuration cross-colo delivery

3.4 Partner Incremental Configuration Reload Automatically

In order to make the reloading smoothly, completely and under monitoring, following process are designed.

A timing thread is created to check whether the system needs to reload the partner configuration periodically. As the partner configuration has divided into single files according to partners, it just needs to find changed partner configuration and reload.

(1) The timing thread tries to find the partner configuration as per soft link time
(2) Read the file and load incremental configuration into memory
(3) Replace existed partner configuration using the delta configuration through memory buffer-switch technology without impacting normal serving

For every reloading, it sends out events for monitor.

4 Results and Analysis

Having described in details our proposed model, we now concentrate on its evaluation. For this purpose, some evaluating indicators have been collected before and after using this model in production launch.

Table 2 shows the application results, indicating the partner configuration to push to production reduce from 24 MB to 80 KB; manpower resource reduces sharply from

8 man/day to 0.5 man/day. The launch cycle shortens to daily and there is no error after using this model for launch.

Table 2. Application results

Item	Before using model	After using model
File size	24 MB	80 KB
Resource	8 man/day	0.5 man/day
Launch cycle	One month	Daily
Error	1	0

Notes: Error represents the number that the changes do not meet requirements during verification.

5 Conclusion

This Incremental Configuration Update Model is extracted based on practice in partner configuration management of Sponsored Search. The application has shown that this model can accelerate the update speed, improve the quality and reduce the involved resource. Of course, this model can be also useful for other large serving systems and industries.

Although it gets some impressive improvement after applying this model in project, there are still some improvements to do in future.

(1) Each step of the model can be improved to reduce latency to reach to the goal that the configuration update can be near real-time from happening to online
(2) Enhance the notification and archiving. Currently, the model only sends a mail after each successful/fail launch and no other records. The function of archiving each change to twiki automatically is a good improvement.

References

1. Yao, S., Mela, C.F.: A dynamic model of sponsored search advertising. Mark. Sci. **30**(3), 447–468 (2011)
2. Fain, D.C., Pederson, J.O.: Sponsored search: a brief history. Bull. Am. Soc. Inform. Sci. Technol. **32**(2), 12–13 (2006)
3. Accessed http://tinyurl.com/nb6cazh, http://tinyurl.com/ph59teu, http://tinyurl.com/nkd3xoe
4. Accessed http://searchengineland.com/report-q1-us-paid-search-growth-strongest-in-3-years-217850
5. Edelman, B., Ostrovsky, M., Schwarz, M.: Internet advertising and the generalized second price auction: selling billions of dollars worth of keywords. Am. Econ. Rev. **97**(1), 242–259 (2007)
6. Graepel, T., Candela, J.Q., Borchert, T., Herbrich, R.: Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In: 27th International Conference on Machine Learning, ICML, pp. 13–20 (2010)
7. Dart, S.: Concepts in configuration management systems. In: Proceedings of the 3rd International Workshop on Software Configuration Management, pp. 1–18 (1991)

8. Accessed https://jenkins-ci.org/
9. Accessed https://en.wikipedia.org/wiki/Continuous_integration
10. Varian, H., Chan, W., Jindal, D., Ranganath, R., Patel, A.: Facilitating the serving of ads having different treatments and/or characteristics, such as text ads and image ads, US 20050251444 A1
11. Cheng, H., van Zwol, R., Azimi, J., Manavoglu, E., Zhang, R., Zhou, Y., Navalpakkam, V.: Multimedia features for click prediction of new ads in display advertising. KDD, 777–785 (2012)
12. Rubens, S., Thomas, D.: Advertising search system and method, US 20060287919 A1
13. Yuan, W., Deng, P., Taleb, T., Wan, J., Bi, C.: An unlicensed taxi identification model based on big data analysis. IEEE Trans. Intell. Transp. Syst. **99**, 1–11 (2015)
14. Wan, J., Lai, C., Mao, S., Villar, E., Mukhopadhyay, S.: Innovative circuit and system design methodologies for green cyber-physical systems. Microprocess. Microsyst. **39**(8), 1231–1233 (2015)