Scholar Publishing Group
Machine Learning Theory and Practice
https://doi.org/10.38007/ML.2026.060104
ISSN 2790-0983 Vol. 6, Issue 1: 30-37

Exploration of Multi-Channel Conversion Path Optimization Methods Based on A/B Testing

Xiangping Yu

Marketing, Point Stone Technology Limited Companny, Sunnyvale, 94085, CA, US

Keywords: A/B testing system, Dynamic policy distribution, Traffic management model, Sequential testing, High concurrency performance optimization

Abstract: The rapid development of the Internet industry drives the demand for rapid iteration of products. As a core tool for data-driven decision-making, A/B testing faces the challenges of high concurrency performance bottlenecks, traffic cost control and reliability of statistical methods. This study adopts the idea of dynamic strategy distribution to construct a modular architecture, decoupling the information production cache consumption module. It combines the traffic layered reuse model and the double-sided sliding window algorithm to achieve dynamic traffic management, and integrates sequential testing and fixed level testing to construct an anti data intrusion traffic management model. The system achieves a thousand level QPS processing capability and millisecond level response, supports experimental flow expansion, bucket deletion, and traffic push operations, shortens the cycle and reduces sample consumption through experimental self iteration, and ensures statistical reliability. Research has found that dynamic policy distribution significantly improves system lightweighting and scalability, while data privacy models achieve traffic and time cost savings while ensuring data reliability. The integration of technology stacks forms a fully automated closed-loop from experimental configuration to result analysis. This study provides a systematic solution for optimizing multi-channel conversion paths, promoting the efficient application of A/B testing in complex business scenarios.

1. Introduction

The background of the exploration of multi-channel transformation path optimization method based on AB test [1] stems from the fact that the improvement of Internet technology and infrastructure promotes the rapid development of the industry, the proliferation of product types and quantities (covering traditional services, consumer grade products, emerging technology services and other fields), the multiple pressures of user demand upgrading, industry competition intensification, technology update iteration, and security and privacy requirements, which require rapid iterative upgrading of products. How to launch effective iterations into the market without visible strategic effects has become a core challenge. A/B test has become a key tool for data driven decision-making by virtue of its advantages of objectivity (random grouping, double-blind characteristics to eliminate subjective interference), economy (reduce development costs and trial and error risks), and controllability (standardized processes and scientific statistical methods). Previous literature research has shown that A/B testing faces significant challenges in system architecture and data analysis: in high concurrency scenarios, the experimental grouping, data

collection, and allocation processes need to handle massive requests, and traditional architectures are difficult to meet real-time and accuracy requirements, resulting in system performance bottlenecks; In terms of traffic cost control, limited user resources, extended testing cycles, and unreasonable traffic allocation can easily lead to insufficient sample size or waste, affecting the credibility and efficiency of experiments; Traditional statistical methods pose a risk of data intrusion in real-time monitoring, and frequent testing may increase the rate of hypothesis errors, requiring a balance between sample size and accuracy. The motivation of this article focuses on solving the above challenges and achieving efficient iteration through multi-channel conversion path optimization. The specific goals are divided into two aspects: first, to improve the high concurrency performance of A/B testing systems, design a configuration driven traffic reuse model, adopt dynamic policy distribution ideas to achieve asynchronous decoupling and traffic layered reuse, and enhance system lightweighting and scalability; The second is to achieve cost control of traffic, combining sequential testing and fixed level testing methods to construct a traffic management model that prevents data intrusion, reduces sample size while ensuring data reliability, and supports experimental self iteration and seamless connection. The main contributions include proposing the idea of dynamic policy distribution, solving high concurrency problems through asynchronous distribution of policy configuration information through experiments, and improving system response speed and stability by combining traffic layered reuse[2]; Design anti data intrusion statistical methods, integrate sequential testing to control hypothesis error rates, achieve experimental flow and time cost savings, ensure data analysis reliability, and provide a systematic solution for optimizing multi-channel conversion paths.

2. Correlation theory

2.1 High concurrency A/B testing model based on dynamic policy distribution concept

Under the background of the rapid growth of Internet business and the surge in demand for A/B testing services, enterprises have gradually shifted from third-party services to independent research and development of A/B testing systems to support multiple business systems sharing the same platform. Business growth and diversification require systems to have robustness, reusability, and scalability, manifested in the ability to conduct multiple experiments simultaneously, respond quickly to a large number of user policy requests, and distribute experimental strategies (i.e., withstand high concurrency pressure). In traditional architecture, experimental users directly access the experimental management module. When the scale of experiments and the number of users increase dramatically, this communication method will cause the module storing experimental strategies to face enormous concurrency pressure. To this end, this chapter proposes the idea of dynamic policy distribution, which achieves asynchronous decoupling of the system through message middleware and caching modules, optimizes the architecture to improve high concurrency performance. Related technologies include message middleware (such as Apache Kafka[3], RabbitMQ[4], etc., which use publish/subscribe or queue models to achieve loosely coupled communication between components, improve system reliability, scalability, and flexibility, support message filtering, persistence, transactions, and other features) and hash algorithms (such as MD5, SHA series, CRC, MurmurHash, etc., which map arbitrary length data to fixed length digests to achieve data sharding, load balancing, and user sharding, ensuring uniform distribution, uniqueness, and fast querying of data, and supporting horizontal scalability).

2.2 Dynamic strategy distribution concept

This mechanism is based on asynchronous decoupling [5] and achieves modular collaboration of

the system through message middleware and caching modules, solving performance bottlenecks in high concurrency scenarios. In the specific implementation, the core functions such as strategy editing, user sharding, and data collection are clearly divided into modular designs, and communication between modules is only achieved through interfaces with small data volumes and unified formats, reducing coupling. Its innovative "dynamic" distribution mechanism is manifested as: the system actively pushes lightweight configuration information to various modules and business systems through message middleware during initialization, restart, or policy changes, ensuring real-time synchronization of policies; During the period of no policy changes, there is no redundant communication within the system. When a user requests, the business side parses the configuration to generate a complete policy response, avoiding the high bandwidth consumption of traditional architectures that require communication for every request. The strategy configuration information is stored in JSON format, including experiment ID, bucket information, and parameter list. The traffic bucket and hierarchical reuse are implemented through hash algorithm, supporting the same user to participate in different attribute experiments across multiple experiment layers. Through performance testing, it has been verified that under concurrent requests from tens of thousands of users, the average response time of the system is 410ms, 99% of requests can be completed within 1.7 seconds, with an error rate of 0%. The throughput reaches 2335.4 times/second, and the response time is stable within 4500 concurrent requests. After exceeding this limit, there may be slight fluctuations due to performance limitations of message middleware. Overall, it meets the high concurrency requirements of multi business system shared scenarios.

3. Research method

3.1 A/B testing cost control model for preventing data intrusion

In the context of rapid product iteration, A/B testing needs to balance data reliability and cost control. Traditional frequency theory methods have limitations due to fixed sample sizes, long periods, and data peeking issues (such as false positive accumulation and multiple comparison errors caused by "peeking" data), making it difficult to meet the needs of high-efficiency experiments. This chapter proposes an experimental cost control traffic management model based on anti data intrusion statistical methods. By combining sequential testing and fixed level testing, dynamic sample size adjustment is achieved - significance is continuously tested in the experiment to avoid the risk of misjudgment caused by frequent "peeping" and reduce unnecessary sample consumption; Combining traffic layering and reuse technology, supporting multi experimental layer traffic sharing to improve utilization. This model is suitable for long-term strategy evaluation scenarios (such as dual bucket experiments), balancing experimental reliability and time/traffic costs under sufficient user traffic conditions, solving statistical errors caused by data intrusion, and improving A/B testing efficiency and result accuracy.

3.2 Cost benefit analysis of sequential testing and fixed level testing

Sequential testing (SPRT) [6]determines the stopping time through dynamic monitoring of experimental results, without the need for pre-set sample sizes. The core formula includes the control group indicator

$$A = \frac{D-\Delta}{2} \tag{1}$$

(as shown in formula 1), the experimental group indicator

$$B = \frac{D+\Delta}{2} \tag{2}$$

(as shown in formula 2), and the stopping probability

$$P_{n,\Delta} = (\frac{p}{p+q})^{\Delta}(\frac{q}{p+q})^{D-\Delta} \quad (3)$$

(as shown in formula 3). In one-sided testing, the probability of stopping when the null hypothesis is true needs to be controlled below the significance level α, and the critical value $\Delta*$ is approximately $z_{a/2}\sqrt{D}$; The critical value for bilateral testing is simplified to $z_{a/4}\sqrt{D}$. Sample size comparison shows that sequential testing saves more samples in most scenarios: when the baseline conversion rate is 1%, the significance level is 0.01, the experimental power is 0.8, and the minimum improvement rate is 10%, sequential testing requires a maximum of 4644 "implemented" samples, and fixed level testing requires 4026 samples, saving 13.3%; Table 1 shows that when the actual effect is twice the minimum improvement rate, sequential testing saves 51.9% in the 10% improvement rate scenario. (As shown in Table 1)

Table 1 Baseline Conversion Rate and Parameter δ Dynamics

Baseline Conversion Rate	Parameter δ	Total Users	Active Users	Converted Users	Baseline Users	Conversion Rate Change %	Total Transaction Amount	Transaction Conversion Rate
0.01	0.2	808	56	632	569	10	336	46.9
0.02	0.2	808	56	626	569	9	336	46.3
0.05	0.2	808	56	606	569	6	336	44.5
0.1	0.2	808	56	573	569	0.6	336	41.3
0.2	0.2	808	56	506	569	-12.4	336	33.6
0.3	0.2	808	56	440	569	-29.4	336	23.6
0.4	0.2	808	56	373	569	-52.4	336	10.1
0.5	0.2	808	56	307	569	-85.4	336	-9.4

Table 2 indicates that the advantage of the time-series test is significant when the baseline conversion rate is below 10%, but the fixed level test is better when it exceeds 50%. (As shown in Table 2)

Table 2 Comparison of Sequential and Fixed Test Sample Sizes under 20% MDR

Baseline conversion rate	Boundary point	Sample size
0.01	0.35	211
0.02	0.338	224
0.05	0.297	279
0.1	0.215	495
0.15	0.14	108

Provide a decision boundary point, such as when the baseline conversion rate is 0.01, the minimum improvement rate for saving samples in sequential testing should be ≤ 0.35.Sequential testing is sensitive to effective strategies (such as saving over 50% of samples in high delta scenarios), but its ability to exclude invalid strategies is weaker than fixed level testing. Its advantage lies in the dynamic sample adjustment that reduces time and traffic costs, making it suitable for long-term strategy evaluation scenarios and balancing experimental reliability and efficiency.

3.3 Design of Dynamic Traffic Management and High Concurrent A/B Testing System

The dynamic traffic management model achieves intelligent traffic allocation by combining fixed level testing and sequential testing. It adopts a double-sided sliding window mechanism to evenly and randomly distribute users within the experimental layer, ensuring both traffic sharing and internal mutual exclusion, effectively avoiding user preference interference with experimental results. The system constructs a cost control model based on anti data intrusion statistical methods. By comparing the sample size of sequential testing and fixed level testing, the optimal testing scheme selection boundary point is determined for different baseline conversion rates and minimum detectable improvement rates, achieving a balance between experimental costs and data reliability. The high concurrency A/B testing system adopts a B/S architecture design, integrating experimental management, data analysis, user permission management, and system logging functions. The core functions include dynamic configuration of black and white lists, definition and management of experimental indicators, manual/automatic scheduling of traffic, random grouping of users, real-time data collection and statistical analysis, dynamic adjustment of experimental strategies, and generation of visualized results reports. The system needs to meet the performance requirements in high concurrency scenarios, ensuring millisecond level response and thousand level QPS processing capabilities. At the same time, security and maintainability are guaranteed through data encryption, automatic fault recovery, and operation log tracking, ultimately forming a scalable and easy-to-use comprehensive experimental platform to support rapid iterative decision-making in multiple business scenarios.

4. Results and discussion

4.1 Design and Implementation of A/B Testing System in High Concurrent Scenarios

This design focuses on improving the high concurrency performance and controlling experimental costs of A/B testing systems. A high concurrency model is constructed using dynamic policy distribution ideas, and traffic management optimization is achieved by combining anti data intrusion statistical methods. The system adopts a B/S architecture for independent deployment, supporting multiple business systems, multiple experiments, and multi-user collaboration. Its core functions include experiment management (including black and white list settings, manual/automatic traffic scheduling, user random grouping), data analysis (real-time data collection, statistical method operation, experimental effect visualization), user permission management, and system log tracking. In terms of non functional requirements, the system needs to meet millisecond level response time and thousand level QPS processing capability to ensure data security and high availability, while supporting modular expansion and operation log tracking, ultimately forming a reusable experimental iteration loop to assist in business decision optimization and cost control.

4.2 Model experiment

This system adopts the B/S mode to build an independently deployed A/B testing platform, and achieves traffic reuse and dynamic management in high concurrency scenarios through a layered design of data management layer, policy layer, interface layer, and report layer. The data management layer stores experimental, metric, and policy data based on MySQL, combined with configuration information for Redis cache high-frequency access [7] to improve response speed; The strategy layer integrates a layered reuse model and diversion algorithm, supporting traffic sharing across experimental layers and mutually exclusive allocation within layers. The MurmurHash algorithm is used to achieve uniform random mapping of user IDs to experimental buckets,

ensuring conflict free traffic allocation in high concurrency scenarios. The calculation method is shown in Figure 1.

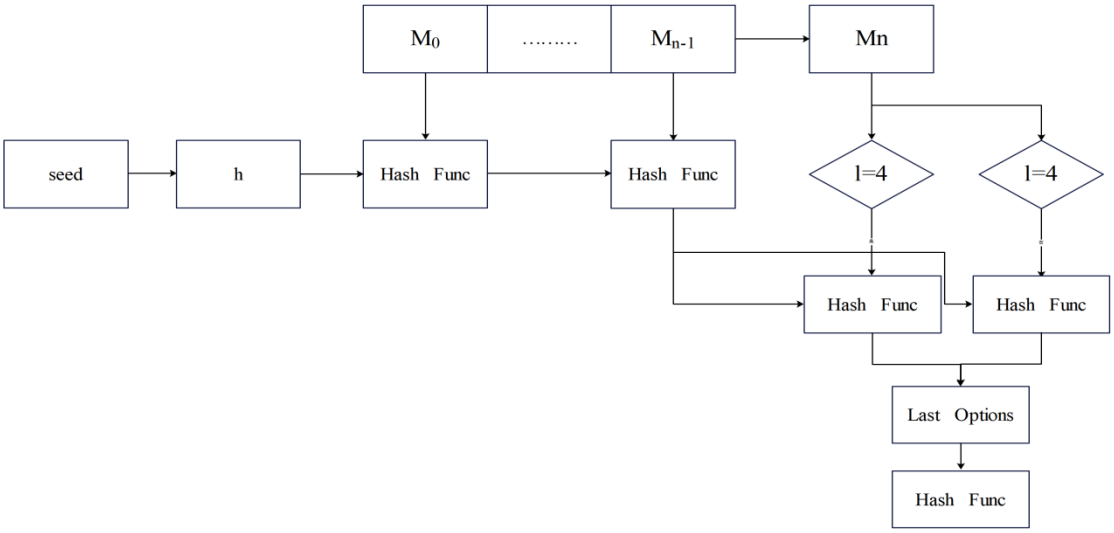

Figure 1. Iterative data processing flowchart based on hash function

In terms of technology selection, Java Spring framework is used to develop web applications, and RocketMQ is used as a message middleware to ensure high throughput and low latency asynchronous communication, meeting the requirements of thousand level QPS processing. The interface layer is responsible for converting experimental configurations and strategy content into internal operational instructions, and interfacing with the report layer to achieve visual statistics and analysis of experimental data, ultimately forming a comprehensive A/B testing platform that integrates experimental management, dynamic traffic scheduling, real-time data collection, and strategy iteration, supporting rapid decision-making and cost optimization in multiple business scenarios.

4.3 Effect analysis

The system integrates four core functions: experiment management, strategy analysis, indicator management, and report management. It achieves dynamic traffic allocation and user bucket mapping through the double-sided sliding window algorithm [8] and MurmurHash diversion algorithm. The experimental management module supports traffic information, bucket parameter configuration, and policy generation, synchronizing configuration information to Redis cache and RocketMQ message queue; The strategy analysis module monitors and analyzes strategy data, the indicator management module defines core indicators such as retention rate and page views, and the report management module generates visual experimental conclusions through statistical analysis, forming an iterative loop. When a user requests, the system filters through black and white lists, regions/genders/ages, determines buckets through hashing and parses experimental parameters, and finally completes the strategy display. After logging the experimental data, it is sent to the analysis system to generate decision-making basis for dynamically adjusting traffic. The system interface supports the creation of experimental layers, configuration of multi-dimensional traffic filtering rules, and bucket traffic allocation operations. The technology stack adopts the Java Spring framework, combined with MySQL data storage, Redis cache, and RocketMQ asynchronous communication, to ensure the thousand level QPS processing capability and millisecond response in high concurrency scenarios, realizing the full process automation management from experimental

configuration to result analysis.

5. Conclusion

This study focuses on the optimization of A/B testing systems, implementing modular architecture design through dynamic policy distribution ideas, decoupling information production [9], caching, and consumption modules, relying on policy caching to support dynamic traffic management (such as flow expansion, bucket deletion, and traffic push), and serving multiple business scenarios as an independent system to effectively cope with high concurrency pressure and complex business requirements. At the level of statistical methods, a traffic management model that prevents data intrusion is constructed by combining sequential testing and fixed level testing. The experiment self iterates by traversing a uniform random user list through double-sided sliding windows, shortening the experimental period and reducing sample consumption while ensuring statistical reliability. It supports real-time monitoring and early termination of significant conclusions during the experimental process. At the implementation level, the system integrates experimental management, strategy analysis, indicator management, and report analysis functions, using Java Spring framework, MySQL data storage, Redis cache, and RocketMQ asynchronous communication technology stack to ensure thousand level QPS processing capability and millisecond level response in high concurrency scenarios, forming a fully automated closed loop from experimental configuration to result analysis. Future research directions include exploring the collaborative application of traffic layering reuse and multi arm gambling machines [10]to adapt to short-term multi strategy scenarios; Optimize the sample efficiency of sequential testing in high minimum detectable improvement rate experiments; Study effective utilization strategies for unrealized sample sizes in high baseline conversion rate scenarios.

References

[1] *Kurz A F, Kampik T, Pufahl L, et al. Business process improvement with AB testing and reinforcement learning: grounded theory-based industry perspectives. Software & Systems Modeling, 2025, 24(1).*

[2] *Li Y, Su Y. A Network Traffic Prediction Model Based on Layered Training Graph Convolutional Network. IEEE Access, 2025, 13:24398-24410.*

[3] *Goel V, Kaur A. Software Maintainability Datasets collection across Android and Apache Kafka Versions. Procedia Computer Science, 2025, 258:3944-3957.*

[4] *Devaraj V, Bagyam J E A, Poongodi T. Optimising communication and performance in IoT with RabbitMQ: a bulk arrival single server retrial queueing model with multi-phase service. International Journal of Mathematical Modelling and Numerical Optimisation, 2025, 15(1):6-26.*

[5] *Hong, Y. (2025). Architecture Design and Performance Optimization of a Large-scale Online Simulation Platform for Business Decision-making. Advances in Computer and Communication, 6(4).*

[6] *Hong, Y. (2026). Research on Warehouse Capacity Optimization Methods Based on Predictive Modeling. Engineering Advances, 6(1).*

[7] *Jin Li. Performance Analysis of Efficient Microservice Architecture in the Financial Industry. Machine Learning Theory and Practice (2026), Vol. 6, Issue 1: 1-9.*

[8] *Yixian Jiang. Performance Optimization and Improvement of Advertising Machine Learning Platform Based on Distributed Systems. International Journal of Big Data Intelligent Technology (2026), Vol. 7, Issue 1: 9-17*

[9] Zhengle Wei. Research on Innovative Design of Financial Derivatives and Market Risk Management Strategies. International Journal of Social Sciences and Economic Management (2026), Vol. 7, Issue 1: 19-27

[10] Linwei Wu. Data Visualization and Decision Support Analysis Based on Tableau. Socio-Economic Statistics Research (2026), Vol. 7, Issue 1: 10-18

[11] Wang, C. (2026). Research on the Control of Uncertainty Risks in Investment Decision-making by Financial Modeling.