Near Real-Time Experimentation for Quick Decision Making

Arjun Reddy Lingala

arjunreddy.lingala@gmail.com

Abstract

With modern applications growing at rapid pace, it also grows the user base and it is very important to maintain and improve the user experience of the application. A/B Testing or experimentation is used for launching or making any change to the application. As companies are not sure which change users like and to decide which change users like more without launching the actual change, experimentation is used a lot these days in all major companies. Experimentation usually runs over multiple days or weeks in some cases depending on the population or exposures it has reached. Applications sometimes want to launch a new feature or change on a critical surface of the application for some amount of time. Critical surfaces quickly exposes the change to users and running experiments on critical surfaces for multiple days or weeks will impact the experience of the user and may change entire perspective of the application for the user. Running experiments for multiple days or weeks on surfaces which have high exposure is risky for the application. In this paper, we discuss near real-time experimentation address this risk by running experiments and computing results of the experiment frequently to check if the experiment results are statsig and stop the experiment and launch the variant that has highest positive results from the experiment

Keywords: Experimentation, A/B Testing, Real-Time, Statsig, Decision Making, Iot Systems, Data Pipelines, Adaptive Systems, Kafka, Batch Processing, Spark, Flink

I. Introduction

In modern world, ability to make quick data-driven de- cisions is critical for any industry such as finance, social networks, e-commerce et. al. With today's systems generating large amount of data in real-time, traditional experimentation techniques that run for multiple days or weeks to get to statsig results struggle to keep pace with demand and in some cases where experimentation is ran on surfaces of the application which has most number of users might impact the user experience of the app. For example, if Instagram wants to run an experiment for a campaign that has multiple variants on home surface of the app. Running multiple variants for multiple days with traditional experimentation methods will result in bad experience for users as the home screen appearance changes frequently for a user. Experimentation results of these heavy user facing surfaces can be statistically significant within few hours from the start of the experiment. Near real-time experimentation facilitates quick decision making by creating a loop where data is gathered, analyzed and acted upon very quickly. The duration can be set based on how quickly exposures of various experiments fill in. This ability helps in crucial scenarios like autonomous decision making. Traditional experimentation techniques analyze data in bulk after each day after collection which lead to delayed insights, rendering them ineffective in dynamic environments. Near real-time experimentation reduce latency in data process- ing and enable immediate action capability.

Near real-time experimentation also presents unique tech- nical challenges. High data throughput, low latency require- ments, and the need for scalable infrastructure impose demand- ing requirements on data processing framworks. In this paper, we deep dive into building resilient system architectures, and data handling mechanisms to compute experiment exposure results without compromising performance or reliability and explain about implementation of these systems ensuring both accuracy and speed. This paper explores the foundational components of near real-time experimentation, infrastructure required and challenges that involve in implementation and key methodologies that enable near real-time experimentation for making quick decision making

II. Background

Traditional experimentation models are generally evaluated by relying on historical data which is often collected and analyzed using batch processing systems which run every day. With modern systems growing very rapidly, the impact on users and application by providing incorrect experience happens very quickly. Not all use-cases require quick decision making, but in areas like finance, marketing and key surfaces on social networks where the exposures of the test variance of the experiment grow very quickly and impact user experience. Various iterative models have been developed for improving the responsiveness of decision making, but these approaches are limited to time-consuming analysis.

With recent advancements in real-time data processing, many organizations have started leveraging building analytics in near real-time to get insights, which includes implementing near real-time batch processing that runs batch processing in every few minutes/hours chunks instead of running entire day's data using HDFS and Spark and implementing real- time systems using Kafka and Flink. Near real time exper- imentation enables decision-makers to test hypotheses and observe outcomes very quickly and make decision which has high potential to enhance decision speed and accuracy from experimentation in social networking to optimizing logistics.

III. Framework and Methodology

Near real-time experimentation involves two major key components to design that can deliver actionable insights rapidly and efficiently – Data Infrastructure, Experiment De- sign and Evaluation. Data Infrastructure is the key component which makes sure that data required for experiment is captured at scale and storage is scaled with increase in data volume. Near real-time experimentation requires the experiments to be setup differently compared to traditional experiment setup as they deal with much faster data in hours or minutes batches.

A. *Data Infrastructure*

1) Data Collection: The foundation of near real-time ex- perimentation is robust data infrastructure which is capable of handling high velocity and high volume data. Data collection is very key component of the data infrastructure. This layer is the entry point of data to the system which captures data from multiple sources that are required for evaluating experiment which involves high throughput systems. It is designed to handle high velocity and high volume data which may change rapidly based on the number of exposures of the experiment increases by region or time of the day. A robust data collection layer reduces the overhead by a lot to upstream (source systems that generate data) and downstream (systems that process generated data) systems. Pub-sub systems like Kafka is best suited for data collection layer in distributed environments. Systems like Kafka separates the overhead of connecting data generator (source) and data processor (trans- formation layer) making integrating systems easier and scaling the data collection layer easier. New sources like transactional systems, sensors, external APIs and data feeds that want to send data for experiments can implement Kafka producer and start sending data to Kafka topics. Any experiment that requires data of a specific use case can subscribe to a specific topic by implementing Kafka consumer. Note that, a single source data can be used for multiple experiments, where as infrastructure where experiment evaluation system directly consumes data from source each source system needs to send data to multiple systems which adds additional network overhead on the system. In this case, it is up to consumer on how much data to consume reducing overhead on data generator.

2) *Data Processing:* Data processing layer in data infras- tructure for near real-time experimentation is essential as it processes, transforms and enriches data as it flows through the system. This layer transforms raw data which might not have complete meaning in most cases into data that explains what the event is about. In some cases, this layer involves aggregating the data into small groups to provide better insights. From near real-time experimentation point-of-view this step, can be implemented as real-time on micro-batches depending on the scale of the experiment. In first step, data is enriched with dimensions that explains what the event is about, where is it happening, who is performing et.al. In case of real-time, we use processing frameworks like Apache Flink that applies transformation logic defined on enriched data and passes data to downstream on storage. In real- time transformations can be performed by defining windows. Windowing is a technique used to analyze data over defined time periods which is important for capturing time-sensitive insights. In case of micro-batching of data of minutes to hours, Apache Spark can be used to transform the data. Apache Spark in case of micro-batching schedules batch processing job for every X minutes (based on batch duration defined) and applies the transformation logic which could involve filtering to complex joins and complex aggregations. At the end of processing layer, data is saved to storage for further processing or consumption for experiment evaluation.

1) *Data Storage:* Data storage layer in data infrastructure for near real-time experimentation is responsible for storing and managing both raw and processed data. Data storage solution should ensure that data saved is resilient to any network breakages and server failures. Modern distributed systems and data lakes like HDFS, Amaon S3, Azure Data lake storage are capable of storing large volumes of both raw and processed data in scalable repository. These systems also have historical data backups that serve as long-term archive preserving all historical experiments related data for retro- spective analysis, model training and compliance purposes. Systems like Amazon Glacier can be used for saving archival data which is not accessed frequently for lesser cost compared to other storage options.

B. *Experiment Design and Evaluation*

1) *Holdout Group:* In any experiment or A/B testing, a holdout group is a subset of users who are excluded from participating in an experiment to serve as a control group for the experiment. This helps in comparing the test group that receive treatment of the experiment against baseline behavior. Experiments can be more accurately evaluated by observing both holdout and test groups and measure the impact of key metrics between these groups. Random assignment is one of the method of designing holdout groups in which users are assigned randomly between holdout and test groups to ensure that groups are comparable and any differences between groups is because of experiment. Typically 5% - 10% of the overall user base is used for holdout group which may vary based on the experiment use case and intended user behavior. For applications, that are in early stages or pre-MVP it might require to have 50% of the user base as holdout group. For near real-time experimentation, a separate dataset (Hive table in case of distributed system) needs to be maintained and is populated before starting the experiment and right after control and test groups are finalized. This dataset is very simple with columns that explains about which users are which group for a given experiment. Columns for this dataset includes *experiment name, user id, experiment group*. This setup helps in evaluating experiments like feature rollouts, marketing campaigns, UI changes et.al. This dataset will be joined with user activity data comparing holdout group and tests variant groups to evaluate the experiment.

Experiment Evaluation: Experiment evaluation involves assessing the results and impacts of an experiment to quickly understand the impact of experiment and make a decision. Every organization should implement vital metrics and monitor those against each experiment. Vital metrics are the key metrics to the company which includes metrics like user engagement, revenue etc. Any drop or negative change on these vital metrics impact overall app. All experiments should consider these vital metrics as critical ones and stop the experi- ment immediately if these metrics are dropping. For evaluating the experiment in near real-time a separate job (Spark job for micro-batches, Flink for real-time) will be implemented that joins the holdout dataset with enriched data that contains user activity related to experiment. This final dataset explains the result of the experiment by comparing holdout group with test groups. Key metrics related to experiment can be positive, but it is also very important to keep track of other metrics which might be negative and have impact on the application. Early stopping criteria rules should be defined for both successful and unsuccessful experiment scenarios to save from unnecessary resource usage and varied experience to users. Once the experiment results are statsig, the experiment can be stopped and if the results are positive, test variant that has more expected behavior will be launched or no change is launched and the developers plan for better change in the application.

In some organizations, experiment dashboards are devel- oped which lets users select the experiment details and ex- periment results can be viewed. These dashboards make it easy for stakeholders to monitor results and make data-driven decision even with no significant knowledge of how systems work. Stakeholders can make segmented analysis by segmen- tation experiment results across various dimensions including *device_type, region, user_tier, age and others*. Near real-time experimentation reduces the risk of negative outcomes quickly and address them by stopping the experiment.

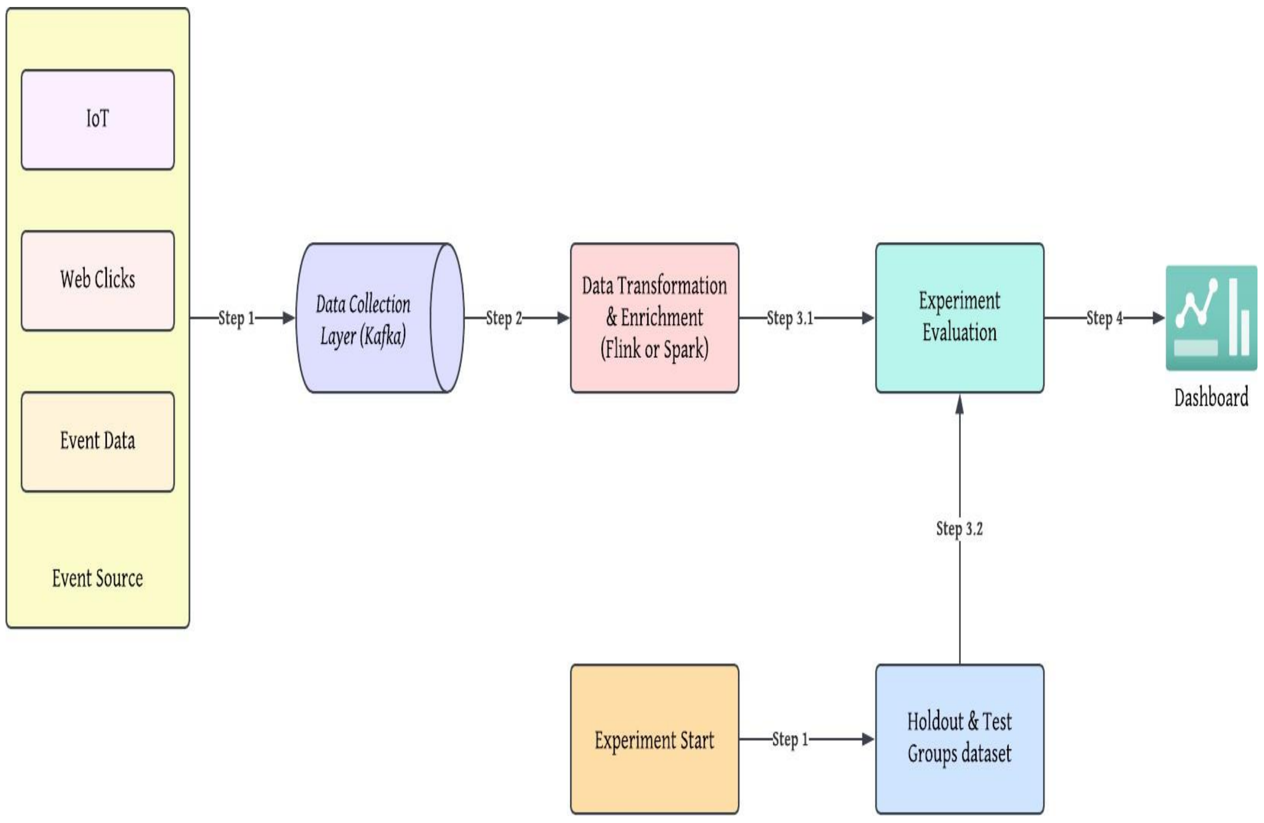

Fig. 1. Near real-time experimentation flow

IV. Implementation Challenges

Near real-time experimentation is very useful in environ- ments where rapid decision making is needed and it helps in saving resources by making decision early and stopping the experiment. It also comes with few implementation challenges like – data infrastructure complexity, data quality, managing user experience, compliance and privacy.

1) Data Infrastructure Complexity: With near real-time experimentation, data should be collected in real-time from different types of sources which some times lead to data overload. High frequency, efficient data ingestion systems like Kafka which can handle a high volume of events per second is required. Kafka lets data generators and data processors disconnect by having a separate storage solution that scales with volume of data. Overhead of storing the data is moved to Kafka layer from data generator, data generators still buffer the data every few seconds and send the data to Kafka in small batches to minimize network overhead cost. Data processors can consume the data at its own piece not worrying about the data is being lost when the processing takes time. Setting up systems like Kafka is additional overhead but it helps in scaling by data volume and one solution for all near real-time experiments for the organization. On the other hand, orga- nizations also needs to setup stream processing systems like Flink or micro-batch processing systems like Spark depending on the evaluation criteria for the experiment. Though, these infrastructure is additional setup, it works and scales as the number of experiments grow with the organization.

2) Data Quality: Real-time data pipelines are prone to errors, and in high frequency data environments, missing or duplicated events is very common which might impact the experiment evaluation leading to unreliable decision making. It is very common for data source to batch the event data and send it to minimize network overhead and in some cases, though the data is sent from the source network delays might cause the delay and the events arrive at much later time. To address these, users can implement a data validation layer to detect and correct inconsistencies in real-time, use idempotent operations and dedup techniques to handle repeated events and implement automated tests for data change to maintain compatibility.

3) Managing User Experience: Holdout groups have dif- ferent experiences for different set of users and ensuring users have a consistent experience will be challenging. As experiments scale there is an increased risk of overlapping ex- periments contaminating the test and holdout groups impacting experiment results. Especially in near real-time experimen- tation often requires analysis by segmentation that includes device_type, region etc. In real-time scenarios, these segments can change very frequently and that can introduce complexities in experiment evaluation.

4) Compliance and Privacy: Any organization handling large volumes of data from across the world must comply with regulations like GDPR, HIPAA et.al. Complying to privacy regulations, organizations should implement framework to anonymize user data to avoid the risk of privacy breaches. As near real-time experimentation only requires data for less duration compared to traditional experimentation, data retention can be minimized and can let users consent about opting in for experiments.

V. Conclusion

In this paper we discussed about how near real-time exper- imentation can help organizations make quick decisions and how it is invaluable for organizations where rapid iteration and quick responsiveness are critical; We also discussed how near real-time experimentation represents a significant shift towards quick decision-making; We then deep dived into how orga- nizations can built near real-time experimentation infrastruc- ture by collecting data, processing data, logging experiment exposures, evaluating experiment in real-time using modern distributed systems tooling like *HDFS, Amazon S3, Kafka, Flink, Spark et.al.*.; We then discussed how to create holdout groups and create a dataset for experiment exposures and use it for experiment evaluation to make quick decisions; We then discussed about infrastructure and experiment evaluation challenges, including the need for a scalable infrastructure with real-time analytics capabilities to ensure accuracy and minimize false positives. These challenges can be addressed by a well-coordinated pipelines and consideration of privacy and compliance requirements with investments in automated monitoring and alerting systems.

In conclusion, near-real-time experimentation represents a major shift towards quick decision-making. It offers a com- petitive advantage by enabling quick, evidence-based actions that are aligned with real-world user feedback. As technologies for real-time data processing continue to evolve, near-real-time experimentation will likely become an essential practice for organizations seeking to remain adaptive and data-driven in an increasingly dynamic landscape.

References

1. Kreps, J., Narkhede, N., Rao, J. (2011). ”Kafka: A Distributed Messag- ing System for Log Processing.” In Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece, 2011, pp. 1-7.
2. Alexandrov, A., Bergmann, R., Ewen, S., Freytag, J.-C., Hueske, F.,
3. Heise, A., Kao, O., Leich, M., Leser, U., Markl, V., Naumann, F., Rheinla¨nder, A., Scha¨ler, M., Schelter, S., Ho¨ger, M., Tzoumas, K., Warneke, D. (2014). ”The Stratosphere Platform for Big Data Analytics.” The VLDB Journal, vol. 23, no. 6, pp. 939-964, Dec. 2014. doi: 10.1007/s00778-014-0357-y.
4. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., Stoica, I. (2012). ”Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Com- puting.” In Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI), San Jose, CA, 2012, pp. 1-14.
5. Shvachko, K., Kuang, H., Radia, S., Chansler, R. (2010). ”The Hadoop Distributed File System.” In 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, NV, USA, 2010, pp. 1-10. doi: 10.1109/MSST.2010.5496972.
6. Amazon Web Services, Inc. ”Amazon Simple Storage Service (Amazon S3) – Object Storage Built to Store and Retrieve Any Amount of Data from Anywhere.” Available: https://aws.amazon.com/s3/.
7. Amazon Web Services, Inc. ”Amazon S3 Glacier – Storage Classes for Long-Term Data Archiving,” Available: https://aws.amazon.com/glacier/.
8. Cohen, J. ”A Power Primer.” Psychological Bulletin, vol. 112, no. 1, pp. 155-159, 1992. doi: 10.1037/0033-2909.112.1.155.
9. Goodman, S. N. ”p Values, Hypothesis Tests, and Likelihood: Implica- tions for Epidemiology of a Neglected Historical Debate.” American Journal of Epidemiology, vol. 137, no. 5, pp. 485-496, 1993. doi: 10.1093/oxfordjournals.aje.a116695.
10. Kohavi, R., Longbotham, R. ”Online Controlled Experiments and A/B Testing.” The Handbook of Marketing Research: Uses, Misuses, and Future Advances, pp. 1-33, 2017. doi: 10.1002/9781119212177.ch2.
11. Chen, J., Zhao, Y. ”A/B Testing: A Practical Approach to Optimize Product Performance.” In 2018 IEEE International Conference on Man- agement of Engineering and Technology (PICMET), 2018, pp. 1-5. doi: 10.23919/PICMET.2018.8481555.
12. P. L. V. de Haan, M. D. H. M. van der Kloet, M. L. H. van Eijndhoven, J.
13. W. de Jager. ”A/B Testing with Relational Databases: Framework and Case Study.” In Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, France, 2015,

14. pp. 1-8. doi: 10.1109/DSAA.2015.7344830.

A. C. G. Ferreira, A. M. N. A. Pereira, R. L. S. Oliveira, R. S. C. P. de Lima. ”A Comparative Study of Batch Processing Architectures for Big Data Applications.” In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 1488-1496.

15. Wang, Y., Zhao, Y. ”Batch Processing in Big Data: A Survey.” IEEE Access, vol. 6, pp. 72450-72463, 2018. doi: 10.1109/AC- CESS.2018.2884161.