Mitigate Underspend in Display Advertising Marketplace to Boost Revenue: A Practice of Data Driven Engineering

Lei Wang
wanglei@yahoo-inc.com

Shoujun Liu
shoujun@yahoo-inc.com

Rui Lin
ruilin@yahoo-inc.com

Yaohong Deng
yaohong@yahoo-inc.com

Lei Zhang
leiz@yahoo-inc.com

Shinn-Der Lee
sdlee@yahoo-inc.com

ABSTRACT

In this paper we introduce the practice of data driven engineering to boost revenue by mitigating underspend in Right Media Exchange (RMX), the core Non Guaranteed Display Advertising serving platform of Yahoo! with 1 billion dollars annual revenue. Underspend occurs when advertiser's budget is not effectively spent. Utilizing historical serving log data, we draw the landscape of underspend and found there was in average 3.2 million dollars ($3.2M) budget left-on-table each day. Through simulation and modeling we identified several system deficiencies contributing to 10% of the $3.2M, while the left 90% were related with business practices and unrevealed system deficiencies.

Based on the data analysis, we carried out a series of algorithms enhancements which have brought 4.6% advertiser spend lift in production (annual run rate of $38M) and extra 2.5%∼4% in bucket testing. We then further improved the infrastructure by building up the Storm-based low latency platform to provide a foundation of intelligent optimization, which has brought 3% revenue lift in bucket test and extra benefits such as the way to overcome MoneyBall overspend. This data driven engineering methodology has made NGD Budget from a routine project to a revenue contributor, which is welcome by stakeholders including business.

Keywords

Revenue, Underspend, Data driven engineering, Agile

1. INTRODUCTION

1.1 Motivation

Right Media Exchange [9] is the first and one of the largest display advertising exchange platform, a business of more than one billion dollars annually. It provides an integrated marketplace for publishers and advertisers to monetize the digital inventories or to reach the audiences. Advertisers can specify their budget delivery rules [6], targeting profile and ROI goals. Based on this information, the system automatically bids on the eligible inventories in real time.

Overspend and underspend are the two critial problems that we need to solve for better customer satisfaction and more revenue. We had a poster in TechPulse 2012 [4] covering the overspend analysis and we have another submission this year for the followup work.

Intuitively, underspend is a much bigger pie in revenue. However, it is only a hypothesis before knowing the real amount of the potential. The possible causes for underspend are quite complicated, including not only system deficiencies such as the suboptimal budget allocation algorithms, but also business practices like low bidding price [12], improper targeting [13] and the drawback of certain pricing types etc., and even false positives like the advertisers set up unreasonably high budget but actually do not want to run it thoroughly. To get the money, we need to answer several questions: How much underspend is there? What are the primary causes, solutions and priorities? What is the end state?

1.2 Approach

We took a data driven engineering methodology to combat with underspend for revenue. We draw the landscape of daily underspend of line items, filtered outliers and found there was in average $3.2M left-on-table each day [3], a big pie while never be easy to get.

We started with what we know and what we can control to try to get a small portion first. We deep dived the system deficiencies that may lead to underspend based on our knowledgei n budget system. The modeling and simulation showed that 10% ($320K) of the underspend was caused by the suboptimal algorithms of both spatial and temporal budget allocation as well as some long time hidden bugs.

We took an agile approach to do analysis and deliver solutions iteratively and incrementally. Whenever we identified and quantized a deficiency, we would propose solution and implement it quickly to validate the revenue impact, which may be followed by deeper analysis and iteratively improvements. Following this methodology, we fixed the low hanging system issues first, then expanded to backend algorithms, and then customer visible algorithms.Since Q4 2012 we have made 4.6% revenue lift in production, extra 2.5%∼4% running in

bucket testing, as well as 3 more features in queue.

The incremental delivery gives stakeholders including business the confidence to let the team to drives its own roadmap. The practice gives team the chance to understand the system and the problems better to realize that latency is a root deficiency which is a major blocker for the efficiencies of optimizations to mitigate underspend. So besides the algorithms enhancements, the team built up low latency feedback loop utilizing Storm with 3% revenue lift observed in bucket testing and benefited MoneyBall in overspend issues. With the integration of ActiveMQ and Rainbow in future, the team is further driving it to be a real time & centralized budget system to serve as a foundation of intelligent system.

The other 90% of the $3.2M is mainly related with business practice and some unrevealed system deficiencies. We are doing deep analysis for this portion. Meanwhile, we are working with business team to learn their use cases to identify scalable solutions. For example, we are investigating a model with Yahoo! EMEA to automatically shift budget between line items within a same insertion order to boost spending and improve ROI. Manually testing has shown good results.

2. UNDERSPEND ANALYSIS

In this section we first draw the landscape of the overall left-on-table budget first to see whether it is pie worthy a try, then deep dive into the system efficiencies to see how much we can solve, and at last probe the unrevealed system issues and business practice to see the future potential.

2.1 Overall left-on-table budget

Our analysis approach is to compute the average daily budget and then deduct the daily spend to get the left-on-table budget for each line item. In RMX, advertisers can set up the lifetime budget as well as daily/monthly cap budget or unlimited budget etc. We use the metrics in table 1 to handle these different scenarios to avoid carrying over budget between days to get the effective daily budget. In the analysis, we used budget archive data and RMX serving log data (DH events) from April 2012 to June 2012 [24] to compute budget and spend respectively.

Sometimes the advertisers will specify an extremely high budget which they would not really be willing to run out thoroughly. To get rid of this noise, we did not count line items whose daily underspend is higher than $1,000 because they contribute to 90% of the observed underspend while just occupied 10% of the total spend. We further filtered out all line items of network 29507 and 46636 because they contributed to 88% of the CPA underspend [3].

2.2 Underspend of system deficiencies

Table 1: metrics of effective daily budget

	life limited	life unlimited
cap limited	$\min\{cap, \frac{life\_budget}{life\_time}\}$	cap
cap unlimited	$\frac{life\_budget}{life \mid time}$	NA

Based on the knowledge of our systems, we listed the major potential deficiencies and quantized their underspend impacts utilizing modeling and simulation. Detailed analysis methods are documented in the twiki pages referred in each deficiency below. As illustrated in Figure 1, the identified system reasons contribute to about 10% of the overall left-on-table budget, which fall into three categories: temporal budget allocation, spatial budget allocation, and legacy bugs.

2.2.1 *temporal budget allocation*

Many advertisers in RMX, though may be not specified in contracts, would like to spend their budget as even as possible. RMX provides daily cap and even pacing to help to spend the money smoothly between days and between hours. However, the supply traffic in RMX has significant divergence between days within a week and between hours within a day. Generally, the traffic in Wednesday is 20%∼40% higher than that in Saturday. The traffic in the afternoon is several times of the traffic in early morning. In this way, the rigid daily cap and even pacing may result in redundant budget in low traffic time while prevent from winning more opportunities in high traffic time and bring:

Underspend of inter-day budget allocation [21]: We selected line items who had overspend within a week, developed models to compute their spending potential with enough budget, and then cap with the redundant budget that can be carried over from previous days within this week. The weekly potential [21] turned out to be $500K, projecting to $75K in average daily.

Underspend of inter-hour budget allocation [5]: As illustrated in Figure 2, using the similar method we got daily underspend related with inter-hour budget allocation is $50K.

2.2.2 *spatial budget allocation*

RMX budget allocation takes a pre-allocation method. Each hour the budget system will allocate a certain amount of budget to each of the six serving colocations, and then the budget will be divided equally to be distributed to each ad server within the colo. Different colocations have different spending speed. Different ad servers with in a same colocation, though with elaborate traffic balancing, still vary significantly in the spend peed for specific line items. What's more, the colocations are isolated with each other, same for the ad servers. In this way, if some colocations or ad servers spend faster and run out of budget earlier, they can not

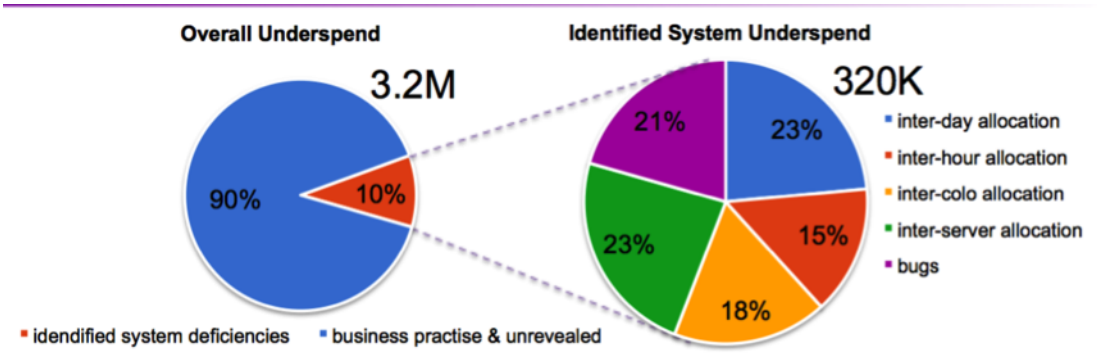

Figure 1: Breakdown of the overall and system underspend

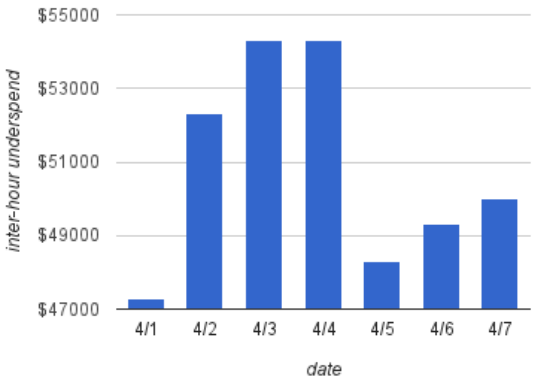

Figure 2: Inter-hour underspend

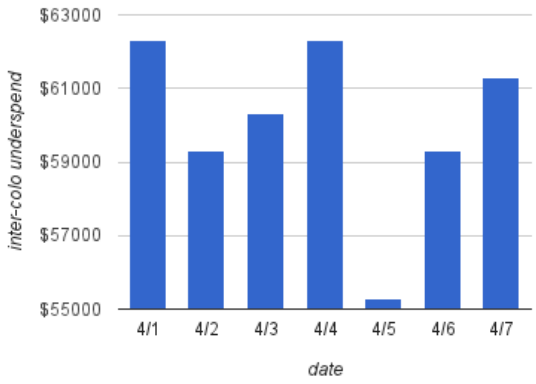

Figure 3: Inter-colo underspend

borrow budget from their peers even if these peers may have redundant budget, but will have to wait for the budget refresh in the next hour, which may bring:

Underspend of inter-colo budget allocation [8] [5]: Similar with the method of analyzing the inter-day allocation, we selected line items which overspent in some colocations in certain hour, then build up model to compute its spending potential with enough budget, and finally cap with the total budget of all colocations of this hour to compute the daily underspend which turn out to be $60K as illustrated in Figure 3.

Underspend of inter-server allocation [2]: As illustrated in Figure 4, using the similar method we got daily underspend related with inter-server budget allocation is $75K.

2.2.3 hidden legacy bugs

Some long history hidden bugs were identified when combating with the abnormal data. For example, when crossing day boundary, the system would allocate less budget than deserved for the first hour of the day; for the dCPM second pricing, the second price was used in charging correctly but the first price was used in online budget checking which may make the ad server stop delivering earlier [17]. These bugs brought $67K underspend daily.

2.3 Unrevealed system deficiencies

We found several other system deficiencies that may cause underspend, while we have not made the mechanisms clearly or quantized the impact accurately.

Over-prediction [19]: For CPC/CPA line items, we will predict the CTR/CVR to compute the eCPM to participate in the auctions and track online budget while we charge based on the real clicks/conversions. The over predicted CTR/CVR will inflate the eCPM spend and may stop online serving. On the other hand, the inflated eCPM means to bid high (eCPM) but pay low (real CPC price) which may ruin the opportunities of others.

Unequal eCPM distribution: As illustrated in Figure 6, the eCPM in the several minutes of the beginning of each hour is extremely high and then drops later. It may be related with the hourly budget refresh.

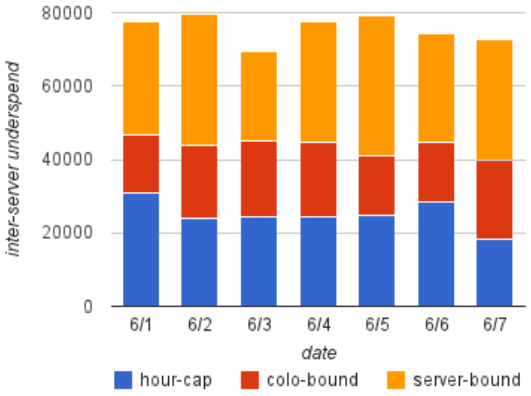

Figure 4: Inter-server underspend

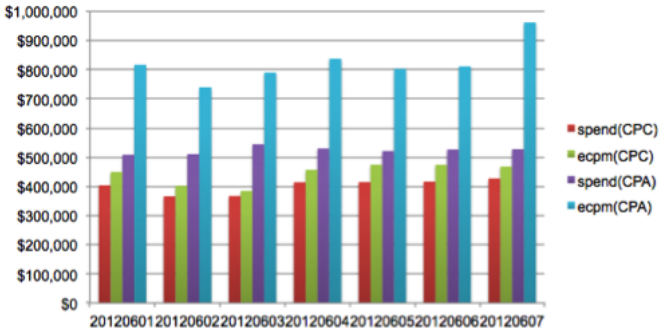

Figure 5: CTR/CVR over prediction

This distribution means in the hour beginning some low quality traffic may be sold in high price and vice versa for the end of the hours.

2.4 Business Practice

The deep dive analysis of this portion is still ongoing. As the first step, we partitioned the overall underspend according to pricing types and different tiers of line items. For details:

As illustrated in Table 2, CPA is the most inefficient type from spend perspective. CPC is the opposite. These observations match the experience of our account managers. The low bid price and low conversion rate contributed to 60% and 35% of the CPA underspend.

Figure 8 illustrated the average underspend rate of different decile of line items ranked by spend amount. It shows the top tiers of line items have pretty good

Table 2: metrics of effective daily budget

	CPA	dCPM	CPM	CPC
underspend percentage	55%	32%	16%	6%
underspend rate	90%	62%	55%	30%

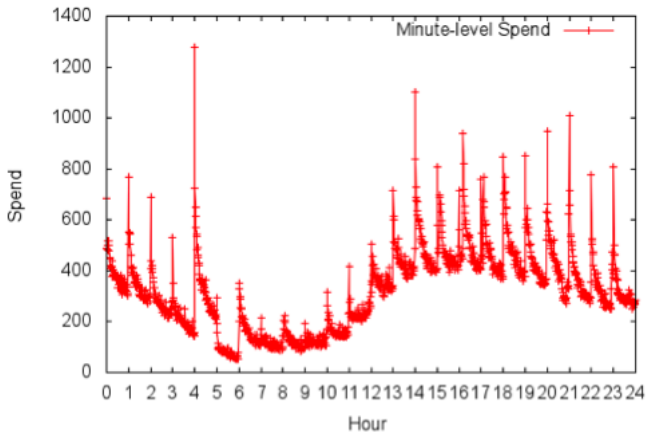

Figure 6: Spend/eCPM distribution

spend efficiencies. So we may take these line items as reference to improve the overall performance.

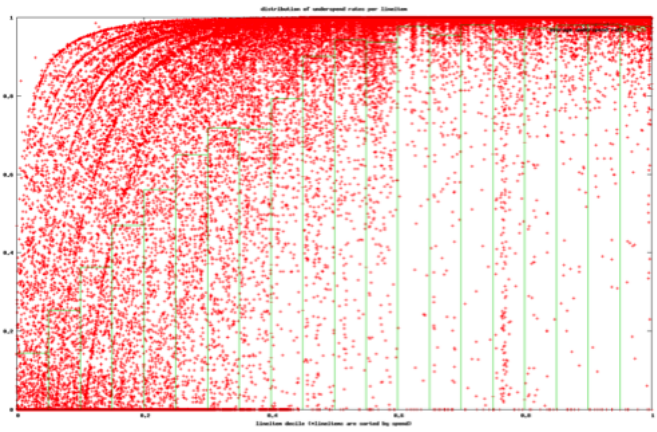

Figure 7: Underspend rate per tiers of line items

3. UNDERSPEND MITIGATION

Shortly after getting some preliminary results in the analysis, we have been taking actions to mitigate underspend incrementally and iteratively.

3.1 Revenue lift algorithms

We started with solving the low hanging fruits (the legacy bugs) since Q4 2012, and then carried out a series of algorithms improvements to mitigate the system deficiencies found in analysis.

To mitigate inter-day underspend, we developed 'dynamic daily cap algorithm' (Budget 2.7)[18] [1] to carry over the redundant daily cap budget of previous days to the following days. We might spend 20% more of the daily budget when more opportunities present themselves, but we will keep the average daily spend no more than the daily budget at any time. In bucekt testing, the rpm (the average revenue that publisher received

Table 3: revenue metrics of Budget 2.7

	rpm	adv_cpm	pub_rev	adv_spend
Y!	+0.6%	-0.4%	+4.1%	+5.9%
Y! EMEA	+1.8%	+1.4%	+1.8%	+1.6%
Y! Asia	+1.6%	+1.4%	+3.2%	+3.1%

Table 4: revenue metrics of Budget 2.1

	rpm	adv_cpm	pub_rev	adv_spend
Y!	-1.5%	-0.6%	+1.0%	+3.6%
Y! EMEA	+7.8%	+9.4%	+7.7%	+10.3%
Y! Asia	+3.2%	+3.3%	+3.8%	+3.6%

for each impression) lift is +2.6%, and the adv cpm (the average spend that advertisers paid for each impression) is +2.5%, projecting to $18M annual publisher revenue lift and $21M annual advertiser spend lift with the same performance in production. Detailed metrics [23] of each regions and seats please refer to Table 3.

To mitigate inter-hour underspend, we developed 'daily even pacing enhancement based on traffic pattern' (Budget 2.1) [15]. The previous daily even pacing algorithm distributes budget equally within the remaining hours of the day. The new algorithm allocates budget to different hours according to Right Media's global traffic pattern to reduce underspend. Meanwhile, it utilizes smoothing methods to make the budget allocation as even as possible to avoid overspend. The rpm lift is +1.7% (annual publisher revenue lift projection is $12M) , and the adv_cpm lift is +1.5%. Detailed metrics [16] are in Table 4.

To mitigate inter-colo underspend, we developed 'trafic trend based proactive colo shifting algoritm' (Budget 2.2) [22]. The previous colo-shifting algorithm[8] reactively depends on each colo's performance in the previous hour. The new algorithm proactively forecasts the traffic trend of each colo and then balances with the impact of last hour's performance. The rpm lift is +1.2%, and the adv_cpm lift is +1.8%. Detailed metrics are in Table 5.

As illustrated in Figure 8, besides the features that have been launched into production (+4.6% publisher revenue lift, projecting to $32M yearly) and that are in bucket testing (2.6%∼4.3% publisher revenue lift), there are several other featues in queue for further investigation, including the 'geo-targeting based budget allocation' [14], 'intra-hour pacing' and 'day-parting based budget allocation' etc. We expect to solve the deficiencies more thoroughly by delivering these features.

Table 5: revenue metrics of Budget 2.2

	rpm	adv_cpm	pub_rev	adv_spend
Y!	+0.6%	-0.4%	+4.1%	+5.9%
Y! EMEA	+1.8%	+1.4%	+1.8%	+1.6%
Y! Asia	+0.5%	+0.2%	+0.7%	+0.5%

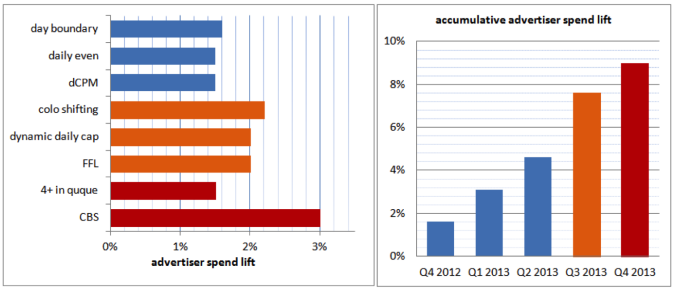

Figure 8: Revenue lift features

Along with the features developement, we keep investing in CI to improve. We automated the creation of CI pipeline to reduce the dev work from 3 hours to 10 minutes, and parallelized the functional test to reduce the runing time by 75%. These work improved the team's velocity and quality significantly. From Figure 8 we can see that in Q4 last year we delivered 1 feature. In Q1 this year we run 2, and in Q2 we run 4 in parallel and delivered all of them to bucket test with good performance.

3.2 advanced infrastructure

The iterative analysis and development gave us deeper insights to our system and the marketplace. We gradually realized that the potential of algorithm improvements upon current system has its ceiling. To get more out of the revenue pie, we need an intelligent system upon which we can apply advanced models for automatic optimization and support scalable business solutions to combat with the underspend better. To achieve this, we need an advanced infrastructure to reduce the latency in current Budget system. As illustrated in Figure 9, at present budget system uses Grid to accumulate the historical spend for each line item with about 50 minutes latency, much longer occasionally, which brings at least a 50 minutes blinding time of the spend information. In this way, the effectiveness of optimization models and algorithms are quite limited.

We build up Storm based fast feedback loop (FFL) [10] in reduce the latency to a few minutes or even subminute. Storm is a real time computation system consuming streaming inputs. As illustrated in Figure 10, Storm uses the minute files of raw DH events as inputs, filters the uninterested events, projects the fileds containing spend info, aggregates the spend and dumps into files to HDFS for downstream processing such as budget cache generation and low latency business metrics reports. With these efforts, we reduced the latency to 15 mintues in the first step. The 1% bucket testing (15 minutes latency & 1 hour budget cycle) shows quite

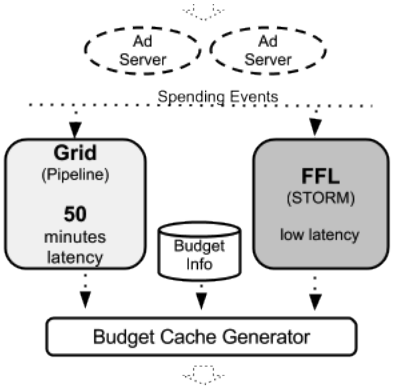

Figure 9: Budget system latency

postive metrics (about 3% adv cpm lift). MoneyBall is leaveraing on this infrastructure to reduce overspend as well [11]. We expect more after more use cases are developed upon FFL, such as to reduce the budget cycle to 15 minute as well.

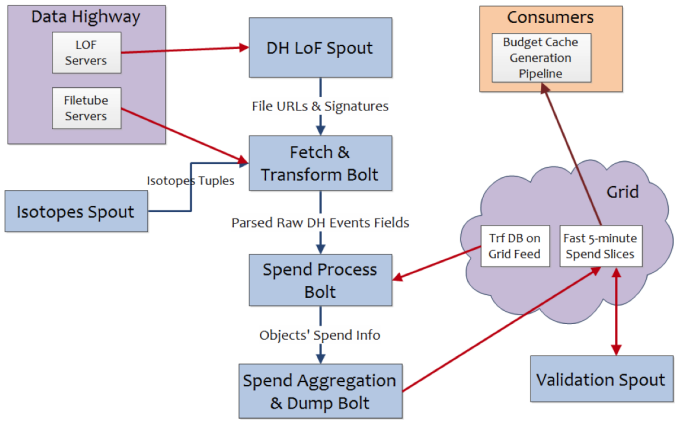

Figure 10: Storm based fast feedback loop [25]

15 minutes latency is just the beginning. We are moving on to integrate Storm with Rainbow (streaming DH events, to replace DH Classic and FileTube who has 4 minutes latency) and CMS (Y! ActiveMQ, to replace GDT which has 3 minutes latency) and stream every input to reduce the end-to-end latency to a few seconds. With doing this, we are moving forward to an 'end state' [7] of Budget infrastructure which consists two parts: 1) A slower feedback loop whose latency is measured in minutes. It takes the place of the existing budget cache generation feedback loop. Its main responsibility is to reset the system to its latest state which includes the activation of new objects, the reactivation of existing objects, and the removal of deactivated objects from the online system, and 2) A number of sub-minute latency feedback loops which implement the real time optimization logics, such as to pause overspending objects, shift budget from non-performing colocations/line items to performing ones, to rebalance the spending between ad servers, and to adjust intra-hour spend rates etc. In this way, it is actually a centralized budget system, a foundation of intelligent optimizaiton.

4. CONCLUSION

Data driven enginnering helps to make NGD budget from a routine project to a revenue contributor. When founded in Q4 2011, the initial scope of Budget team was to decouple the budget library from serving code bases so that it can be tested and maintained independently. We tried data driven method the first time in combating with overspend. The data analysis helped us to target the root causes and deliver effective solutions to reduce the site-up response time from 6 weeks to . The initial success gave us the confidence to strengthen the data driven engineering methodology in the challenge of underspend. Based on the analysis results improved revenue through algorithm and infrastructure enhancements and formed a clearer of the 'end state'. A team may need to learn to crawl first, then walk and run. We have benefited much from data driven engineering in the way.

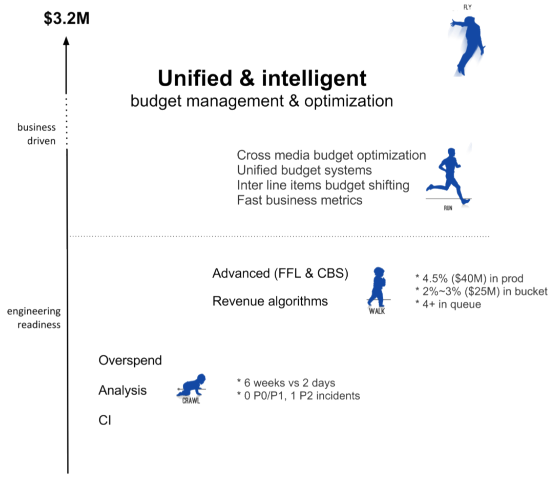

Figure 11: The past and the future

5. REFERENCES

[1] RMX KB Article. Daily even pacing enhancement. `https://kb.yieldmanager.com/article.php?id=1044`.

[2] Lei Wang Jianlin Zhang. Inter-server underspend. `http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendCausedByInterServersIsolation`.

[3] Lei Wang Jianlin Zhang. Overall rmx underspend. `http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendOverallUpperBound`.

[4] Lei Wang Jianlin Zhang, Lei Zhang. Overspend analysis: Recover lost money in display. TechPulse 2012.

[5] Lei Wang Jun Wangg. Inter-colo and inter-hour underspend. http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendCausedByBudgetAllocation.

[6] Shinn-Der Lee. Budget cap and pacing primer. http://twiki.corp.yahoo.com/view/NgdExchangeServing/BudgetCapAndPacingPrimer.

[7] Shinn-Der Lee. Budget system end state. https://docs.google.com/a/yahoo-inc.com/document/d/1QxmKiPaVynaZG6eMktPfLynnAS2R2v733wU9g4qwVa0/edit?usp=sharing.

[8] Shinn-Der Lee. Inter-colo budget distribution. http://twiki.corp.yahoo.com/view/NgdExchangeServing/InterColoBudgetDistribution.

[9] Shinn-Der Lee. Right media exchange. http://www.rightmedia.com/index.php.

[10] Mengjun Han Lei Zhang. Architecture design of budget ffl. https://docs.google.com/a/yahoo-inc.com/document/d/1gt8IWPGcy8tMNIoUyAEhN7DNdf9Imptf1ryGopegC24/edit.

[11] Shinn-Der Lee Lei Zhang, Shoujun. Design of moneyball low latency overspend feedback loop. https://docs.google.com/a/yahoo-inc.com/document/d/16us2MCfibTXHeq-SNPbNcycJ4IckgR0kyxR7D1pk4V4/edit#heading=h.epszskues0wf.

[12] Rui Lin. Underspend related with bidding price. http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendRelatedWithBidding.

[13] Rui Lin. Underspend related with targeting. http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendRelatedWithTargeting.

[14] Shoujun Liu. Geo-targeting based budget allocation. https://docs.google.com/a/yahoo-inc.com/document/d/1eW-6vv6YFRiX5hv2mXLuc_jaCwc14_ocflJRtwlzJZ4/edit?usp=sharing.

[15] Jing Hong Rui Lin, Yaohong Deng. Daily even pacing enhancement based on traffic pattern. http://twiki.corp.yahoo.com/view/NgdExchangeServing/PacingDayEvenDesign#A_2.3_Design_Phase_2.

[16] Lei Wang Rui Lin. 25% testing metrics of budget 2.1. https://docs.google.com/a/yahoo-inc.com/document/d/1CjxkfxO6xrixR122kZVUCRPy-aYYEClry6LZw3jqRDc/edit.

[17] Lei Wang Rui Lin. dcpm second price online deduction. http://twiki.corp.yahoo.com/view/NgdExchangeServing/BugFixOfECPMforDCPMInAdServer.

[18] Lei Wang Rui Lin, Yaohongt. Dynamic daily cap algorithm. https://docs.google.com/a/yahoo-inc.com/document/d/113sLwrTEHkRkd8GuMYEYQyRmU8kxabLPMmFNWSzc7eE/edit?usp=sharing.

[19] Lei Zhang Rui Lin. Underspend of over prediction. http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendCausedByOverPredict.

[20] Yaohong Deng Rui Lin. Moneyball overspend. https://docs.google.com/a/yahoo-inc.com/spreadsheet/ccc?key=0AueWDoO-C-DzdDhVUUp1TjFBdEJPN0t5UXhUZmI2M3c#gid=0.

[21] Rui Lin Shinn-Der Lee. Weekly capping. https://docs.google.com/a/yahoo-inc.com/document/d/113sLwrTEHkRkd8GuMYEYQyRmU8kxabLPMmFNWSzc7eE/edit?usp=sharing.

[22] Jianlin Zhang Shoujun Liu. Traffic trend based proactive colo shifting algorithm. https://docs.google.com/a/yahoo-inc.com/document/d/1wnZ5CcYJ9NVm0uEZgOHqIOpNp_Zn1u_4aYcMeWfSKzo/edit#heading=h.zhvh3ba8piov.

[23] RMX Team. Rmx business metrics definitionst. http://twiki.corp.yahoo.com/view/NgdExchangeServing/ExperimentMetricsDefinitions.

[24] Jun Wang. Data sources and pipeline for budget analysis. http://twiki.corp.yahoo.com/view/NgdExchangeServing/UnderSpendDataSources.

[25] Lei Zhang. Detailed design of budget ffl. https://docs.google.com/a/yahoo-inc.com/document/d/1pS-OQowZA3eqWAN9TjCR1L1KGxeri1Hwiy_yce2Ar2c/edit.