Capacity Planning and Headroom Analysis for Taming Database Replication Latency

- Experiences with LinkedIn Internet Traffic

Zhenyun Zhuang, Haricharan Ramachandra, Cuong Tran,
Subbu Subramaniam, Chavdar Botev, Chaoyue Xiong, Badri Sridharan
LinkedIn Corporation
2029 Stierlin Court, Mountain View, CA 94002, USA
{zzhuang, hramachandra, ctran,
ssubramaniam, cbotev, cxiong, bsridharan}@linkedin.com

ABSTRACT

Internet companies like LinkedIn handle a large amount of incoming web traffic. Events generated in response to user input or actions are stored in a source database. These database events feature the typical characteristics of Big Data: high volume, high velocity and high variability. Database events are replicated to isolate source database and form a consistent view across data centers. Ensuring a low replication latency of database events is critical to business values. Given the inherent characteristics of Big Data, minimizing the replication latency is a challenging task.

In this work we study the problem of taming the database replication latency by effective capacity planning. Based on our observations into LinkedIn's production traffic and various playing parts, we develop a practical and effective model to answer a set of business-critical questions related to capacity planning. These questions include: future traffic rate forecasting, replication latency prediction, replication capacity determination, replication headroom determination and SLA determination.

Keywords

Capacity Planning; Espresso; Databus; Database replication

1. INTRODUCTION

Internet companies like LinkedIn handle a large amount of incoming traffic. Events generated in response to user input or actions are stored in a source NoSQL database. Though these events can be directly consumed by simply connecting to the source database where the events are first inserted, today's major Internet companies feature more complicated data flows, and so require database replications to isolate the source database and events consumers (i.e., applications that read the events).

ICPE'15, Jan. 31–Feb. 4, 2015, Austin, Texas, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3248-4/15/01 ...$15.00.
http://dx.doi.org/10.1145/2668930.2688054.

Database replication is needed mainly for the following two reasons. Firstly, the source database may need to be protected from heavy or spiky load. Having a database replication component can fan out database requests and isolate the source database from consumption. Figure 1 illustrates the typical data flow of event generation, replication, and consumption. When users interact with a web page, the corresponding user updates (events) are sent to databases. The events are replicated by a replicator and made available to downstream consumers. Secondly, when Internet traffic is spread across multiple databases or multiple data centers, a converged and consistent data view is required, which can only be obtained by replicating live database events across data centers.

The database replication process has to be fast and incur low latency; this is important both for the benefits of the particular business and for enhanced user experience. Any event replicated by the events replicator has an associated replication latency due to transmission and processing. We define replication latency as the difference in time between when the event is inserted into the source database and when the event is ready to be consumed by downstream consumers. Minimizing the replication latency is always preferred from the business value perspective. While a delayed user update can be a bit annoying (for example, LinkedIn's user profiles fail to show the users' newly updated expertise), other delayed updates can incur additional business cost or reduced business income. For example, with web sites that display customers' paid ads (i.e., advertisements), the number of ads impressions (i.e., the number of times an advertisement is seen) across multiple data centers has to be tightly correlated to the pre-agreed budget. A significantly delayed ads event update will cause additional cost and reduced income for the company that displays the ads.

Keeping a low database replication latency of events is a challenging task for LinkedIn's traffic. The events feature the typical characteristics of Big Data: high volume, high velocity and high variability. These characteristics present tremendous challenge on ensuring low replication latency

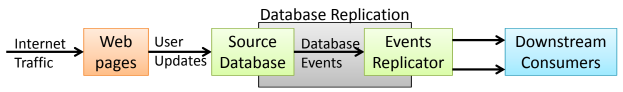


Figure 1: Data flow of database replication

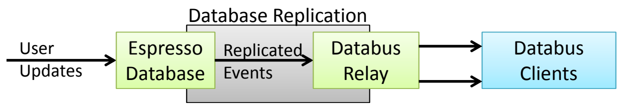


Figure 2: Data flow of LinkedIn Databus replication and consumption

while without significantly over-provisioning the deployment. Though a naive solution to minimizing the replication latency is to provision the capacity based on *maximum* traffic rates, the high-variability feature will incur high business cost associated with the over-provisioning. On the other hand, allowing certain levels of replication latency can significantly reduce business cost. Striking the balance between replication latency and business cost turns out to be a challenging task that requires appropriate capacity planning.

Database replication latency is closely tied to capacity planning. Capacity planning will help understand the current business and operational environment, assess and plan for future application needs based on traffic forecasts. Capacity planning can particularly help reduce the business operation cost. For example, given incoming traffic patterns, we can plan the replication capacity (hardware/software provisioning) so that replication latencies do not exceed business requirements. Though a naive solution is to aggressively provision resources to meet business SLAs (Service Level Agreements) in the worse cases such that the replication capacity always exceeds the foreseen peak traffic rates (for example, maximum replication latencies less than 60 seconds), it would incur unnecessary business cost. On the other hand, based on appropriate capacity planning models, it is possible to significantly reduce business cost without violating business SLAs.

To reduce database replication latency and save business cost, appropriate capacity planning is required. For the problem we attempt to address in this work, we need to understand the relationship among incoming traffic volume, replication capacity, replication latency, and SLAs in order to ensure desired replication latency. In addition, by carefully considering both incoming traffic rate and replication capacity, we can also use capacity planning to foresee future replication latency values given a particular replication processing capacity. Moreover, most Internet companies' traffic also show an ever-growing traffic rate, and we need to improve replication processing capacity to accommodate the traffic increase.

Specifically, the following set of questions need to be answered in the context of capacity planning to tame database replication latency:

- *Future traffic rate forecasting*: Given the historical data of incoming traffic rates, what are the expected traffic rate in the future? This question will also help answering latter questions;

- *Replication latency prediction*: Given the incoming traffic rate and replication processing capacity, what are the expected replication latencies? The values are important to determine the SLA of maximum replication latency;

- *Replication capacity determination*: Given the increased incoming rate and largest allowed replication latencies (SLA), how much replication capacity do we need to achieve? This will help define replication capacity requirements;

- *Replication headroom determination*: Given the replication capacity and SLA, how much more incoming traffic can we handle? With the current replication capacity, how long (i.e., how many days) will it take before SLA violation? This helps plan for future requirements.

- *SLA determination*: Given the incoming traffic rate and replication processing capacity of today or future, how to determine an appropriate SLA? Apparently, we don't want to over-commit or underestimate the SLA.

Based on our observations into production traffic and various playing parts, in this work we develop a practical and effective model to forecast incoming traffic rates and deduct corresponding replication latency. The model can be used to answer the set of business-critical questions related to capacity planning defined above. In this work, we share how we perform capacity planning by answering the five questions related to capacity planning. These questions offer different aspects of capacity planning and can be applied to different scenarios. We use the models on one of LinkedIn's database replication products to describe the designs and demonstrate usage, but the models can easily be applied to other similar usage scenarios. Moreover, our observations on internet traffic patterns can also help shed light on solving similar capacity planning problems in other areas.

For the remainder of the writing, we first present the various observations regarding our production incoming traffic to motivate our design in Section 2. We then define the problems being addressed in this writing in Section 3. We then present the high level designs in Section 4 and the detailed designs of forecasting in Section 5. Section 6 presents the evaluation results of the proposed designs. We discuss some further relevant questions in Section 7 and share our learned lessons in Section 8. We also present certain related works in Section 9. Finally Section 10 concludes the work.

2. OBSERVATIONS OF LINKEDIN TRAFFIC

In this section, we first give some background of LinkedIn's Databus, then present some observations of our production traffic and deployments. These observations will help our design in later sections.

2.1 LinkedIn Databus

LinkedIn's Databus [7] is a data replication protocol responsible for moving database events across data centers. It can also be used to fan out database usage to reduce the load on source databases. It is an integral part of LinkedIn's data processing pipeline. Replication is performed by the Databus Relay component, which processes incoming database records and makes them ready to be consumed by downstream Databus clients. A simple illustration of the data flow is shown in Figure 2. The raw events are inserted into the source database, LinkedIn's home-grown Espresso [15]. These events are replicated and queued for Databus Relay to consume. Databus Relay fetches and processes the queued events so that the events can be later pulled by clients.

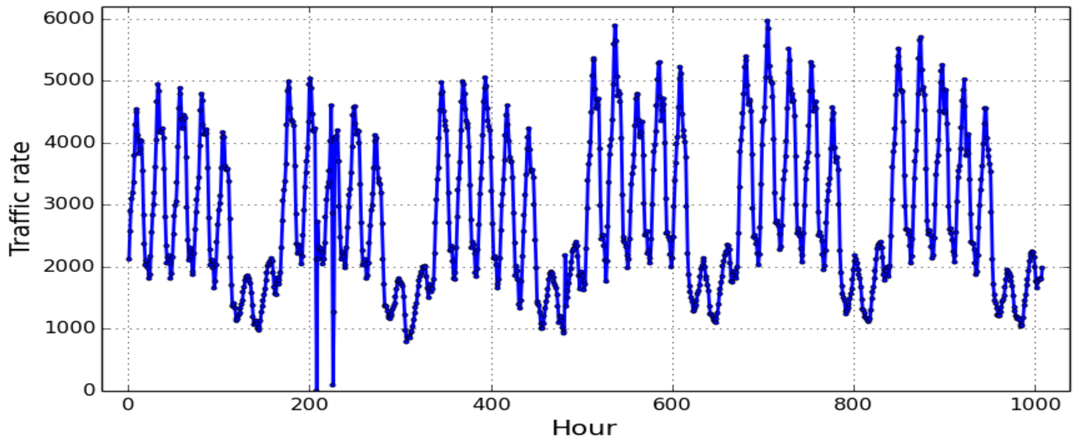
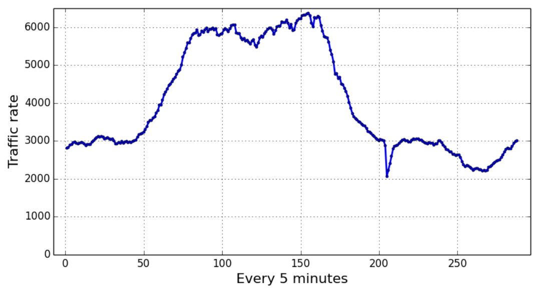
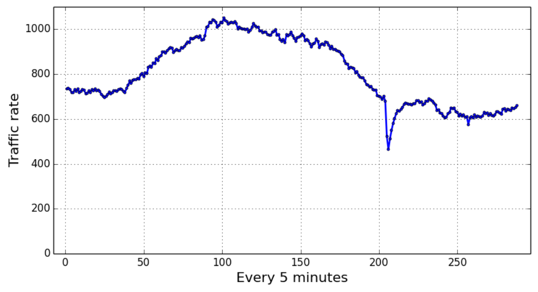


Figure 3: Six weeks (42 days) of incoming traffic (totally 1008 data points; note that two close-to-zero data points are invalid)



(a) Monday traffic (May 5th, 2014)



(b) Saturday traffic (May 17th, 2014)

Figure 4: Individual days of traffic (Each data point represents 5 minutes, totally 288 data points)

A critical performance factor is the Databus Relay Lag (i.e., Databus replication latency), which is calculated as the difference in time between when a record is inserted into source Espresso and when the record is ready to be consumed by the Databus clients. The extent of relay lag largely depends on two factors: the rate of incoming raw database events (i.e., the "producing rate" of the events) and the Databus Relay processing capacity (i.e., the "consuming rate" of the events). So if the producing rate is always lower than the consuming rate, there will be no substantial relay lags, other than the event transmission delay which is usually very small. But when the producer overwhelms the consumer, the incoming events will be queued and a non-trivial relay lag will result due to an events queue backlog. To minimize relay lags, appropriate capacity planning is needed after careful consideration of both producing and consuming rates.

2.2 Periodic pattern of incoming traffic rate

We first studied the incoming traffic rate of multiple Espresso instances across many months; our first impression was the strong periodic pattern - the traffic shape is repeated for each day of the week. Figure 3 below shows 42 consecutive days of data for a single Espresso node. The incoming traffic consists of two types of database events: insert and update. Databus Relay processes both types of traffic, so we aggregated the total incoming rate for each minute.

We observed weekly repeating patterns in incoming traffic rates. Every week, the five workdays have much higher traffic rates than the two weekends. Such a weekly pattern repeats with similar traffic shapes. Within a single day, irrespective of being workday or weekend, we found that the traffic rate peaks during regular business hours, while drops to the lowest at night.

2.3 Incoming traffic of individual days

We then studied the periodic pattern of incoming traffic for each day. For each workday and weekend, we noticed that the traffic shape is a well formed curve. In Figure 4, we show the workday of May 5_{th}, 2014 (Monday) and the weekend day of May 17_{th}, 2014 (Saturday).

We observed that the traffic shapes of these two days were quite similar, except for the absolute values of each data point. Specifically, for each day, the peak periods were about 8 hours (that is, 6AM to 2PM in the West Coast, or 9AM to 5PM in the East Coast). Not surprisingly, the workday peak value (6367 records/sec) was much higher than that of weekends (1061 records/sec).

2.4 Relay processing capacity

We also examined the relay processing capacity. The relay processing rate was maximized only when there was a buildup in queued events. To force Databus Relay to work at

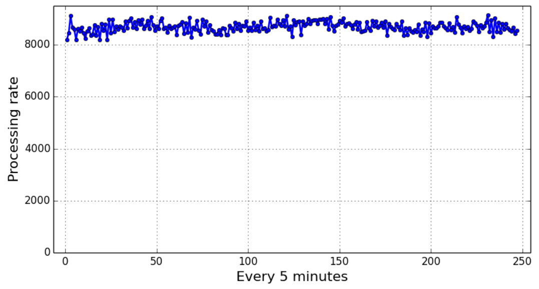


Figure 5: Relay processing rate (Relay capacity)

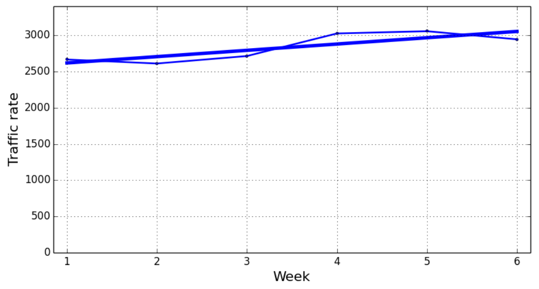


Figure 6: Traffic growth in 6 weeks

full capacity, we rewound the starting System Change Number (SCN) of the relay component to artificially introduce sufficient relay lag. For a particular Databus Relay instance, the replay processing rates over 20 hours of a recent day are plotted in Figure 5.

We observed that the relay processing rate is relatively flat over time. The reason for relative stable relay processing rate is because the processing rate is dominated by the replication protocol, which is relatively constant for each event.

2.5 Growing traffic

We studied the historical data of incoming traffic and found that: (1) overall, the traffic volume grows over time; (2) the traffic for individual days is affected by many factors including weekends, holidays, and production releases. We analyzed the incoming traffic rates for a duration of 6 weeks and plotted the monthly traffic rates in Figure 6. Linear fitting of the curve demonstrates about 20% increase in 6 weeks.

Note that this analysis and graph are based on a particular instance of the Espresso node. We also studied other instances and found that though the exact growth rates differ for different periods and instances, the overall traffic growth rate is quite significant. Even though it is possible to build capacity planning models for each instance for higher accuracy, we want to build a generic model to cover all instances.

2.6 Summary

Our study of LinkedIn's production traffic showed that the database events producing rate (i.e., the incoming traffic rate) follows strong patterns in the form of repeated weekly curves, while the consuming rate (i.e., the replication processing capacity) is relatively constant. Because of varying incoming traffic rates, we used to see that replication latency can accumulate during peak hours when production rate is higher than the relay capacity, and the accumulated replication latency decreases during non-peak hours when the incoming traffic rate is lower than the relay capacity.

The incoming traffic rate keeps growing, thanks to the user and activity growth, so it is important to take into account the traffic growth when doing capacity planning. We need to forecast into future traffic to understand the future replication latency, estimate the capacity headroom, define the replication capacity requirements, and determine SLAs.

3. PROBLEM DEFINITION

In this section, we formally define the problems we attempt to address in this work. We first give the definitions of SLAs (service level agreements); then present the four types problems related to capacity planning.

3.1 Forms of SLA (service level agreements)

SLA is a part of a service contract where a service is formally defined. Thought normally the service contracts are between companies, it can extend to between different divisions inside a company or even between different software components. For instance, a web service component may define a SLA such as maximum call latency being 100 ms, so that the callers (e.g., a web page rendering component) can rely on the SLA to make design choices and further define their SLAs.

The goal of taming the database replication latency is to fulfill SLA, which can come in different forms. A straightforward SLA metric can be expressed as the "largest" replication latency experienced, which is the form we use to describe our designs. For instance, *largest replication latency should not exceed 60 seconds.* However, other forms of SLA metrics are also possible, depending on specific requirements. To give a few examples: (1) certain percentiles (e.g., p99) of all replication latencies in a day, or (2) the duration of replication latencies exceeding certain values. Despite the differences in SLA forms, we believe the underlying mechanisms required to answer the five types of questions in Section 1 are very similar. So in this work, we will use the largest replication latency as the SLA metric to present our solution.

3.2 Problems being addressed

For simplicity in presentation, we fix the time-granularity of the traffic rates to *per-hour* average, but other granularity can be similarly defined. The following variables are needed to define the problems: (1) Relay capacity: $R_{i,j}$ in day d_i and hour h_j, where $0 \leq j \leq 23$; (2) Incoming traffic rate: $T_{i,j}$ in day d_i and hour h_j; (3) Replication latency: $L_{i,j}$ in day d_i and hour h_j; (4) SLA of replication latency: L_{sla}, which is the largest replication latency.

With the above variables, we can formally define the set of the problems we will address in this work:

- *Forecast future traffic rate* Given the historical traffic rate of $T_{i,j}$ of time period P (e.g., past 30 days), what are the future traffic rate $T_{r,k}$, where $r > i_{max}$ in day r and hour k?

- *Determine the replication latency* Given the traffic rate of $T_{i,j}$ and relay capacity $R_{i,j}$, what are the replication latency of $L_{i,j}$?

- *Determine required replication capacity* Given SLA requirement L_{sla} and traffic rate of $T_{i,j}$, what are required relay capacity of $R_{i,j}$?

- *Determine replication headroom* There are two questions: (1) Given the SLA of L_{sla} and relay capacity of $R_{i,j}$, what are the highest traffic rate $T_{i,j}$ it can handle without exceeding SLA? (2) what are the expected date d_k of the previous answer?

- *Determine SLA* There are multiple forms of SLAs that can be determined. In this work we consider two of them: (1) what is the largest replication latency L_{max} in a week; (2) for 99% of time in a week the replication latency should not exceed L_{p99}, what is the L_{p99} value?

4. DESIGN

This section will start with the overview of our design, followed by two models to perform forecasting of future traffic rates. After that, the other four types of questions described in Section 1 will be respectively answered.

4.1 Design overview

Before we jump into the details of the design, it helps to gain a conceptual grasp of the problem we try to address. The extent of relay lag largely depends on two factors: the rate of incoming raw database events and the Databus Relay processing capacity. The Databus Relay is conceptually a queuing system and has the features of a single server, a First-In-First-Out (FIFO) queue, and infinite buffer size. Unlike typical queuing theory problems, in this work, we are more focused on the "maximum" awaiting time of all events at a particular time.

In order to answer all the capacity planning questions, we need to obtain the future traffic rates. We will employ two models for traffic forecasting: Time series model and Regression analysis. Given historical traffic rates of certain period, the time series model will be able to forecast the future rates based on the discovered trend pattern and seasonal pattern. The historical data we can obtain is mostly per-hour based due to current limitations on the storing mechanisms, so for each past day it consists of 24 data points. We also observed that the traffic rates exhibit strong seasonal pattern with period of a week (i.e., 7 days, or 168 hours).

For this type of time series data, typically ARIMA (autoregressive integrated moving average) model [3] is chosen for modeling and forecasting. Due to its design rationales and computation overhead, ARIMA is not suitable for long period seasonality (e.g., 168). In addition, the capacity planning needs to obtain per-hour forecasted data rather than per-day data. Because of this, we are not allowed to aggregate per-hour data into per-day data to reduce seasonality period to 7 (since a week has 7 days). To accommodate the long period seasonality as observed in traffic data and the per-hour forecasting requirements, we propose a *two-step* forecasting model to obtain future per-hour traffic rates. Briefly, it firstly obtain the aggregated traffic volume of each day/week, it then "distribute" (or "convert") the aggregate to each hour inside a day/week. The "conversion" of traffic volume from day/week to hours relies on *seasonal indexes*, which roughly represent the traffic portion of each hour inside a week. Specifically, the model consists of two steps: (1) Step-I of forecasting the average incoming rates *per day* of future days using ARIMA model and (2) Step-II of forecasting the average rates *per hour* of each future days using seasonal indexes.

Regression analysis is targeted for estimating the relationships among variables. Regression analysis can also be used to forecast and predict. Specifically, for our purpose of forecasting future traffic rate, the independent variable is the time, while the dependent variable is the traffic rate at each time. Such analysis is also referred to as "time series regression". Since we observed strong seasonal pattern with the period of a week, we propose to forecast the average traffic rates of future "weeks" based on historical weekly data. Once we obtained the weekly average, we can then convert the weekly average to per-hour data of within a week using the same step-2 of time series model presented above. Another important reason why we choose *per-week* aggregation is that, regression analysis is based on the assumption that the adjacent data points are not dependent. Compared to finer scale of data points, weekly aggregated data are less dependent on each other. We will discuss more on this in Section 7.

For answering the other three types of capacity-planning related questions, we develop respective mechanisms which utilize numerical calculations and binary-searching. For ease of description, we assume the time granularity is per-hour. Once the traffic rates are obtained, then a numerical calculation method will be used to deduct the relay lags of any time point. Based on the numerically calculated results, the relay capacity required to sustain a particular traffic rate can also be obtained using binary search. The maximum traffic rates as well as the corresponding future dates can also be determined. Finally, examining the deducted replication latencies we can also determine appropriate SLAs.

4.2 Forecasting future traffic rate with ARIMA model

The incoming traffic rates of each time unit are a sequence of observations at successive points in time, hence they can be treated as a time series. We can employ time series method to discover a pattern and extrapolate the pattern into the future. The discovered pattern of the data can help us understand how the values behave in the past; and the extrapolated pattern can guide us to forecast future values.

ARIMA time series forecasting model will be used to predict the incoming rates of any future days. Though it is obvious to see the weekly pattern for our particular time series data, identifying seasonal periods for other time series might not be so trivial. Examining the ACF (Autocorrelation Function) [3], we see a strong signal at 7, indicating a period of 1 week (i.e., 7 days). Based on our investigations into the properties of the traffic data, we decided to use ARIMA(7,1,0) to fit the historical data and forecast the traffic rates of future days, the details will be presented in Section 5.

Answering various capacity planning questions requires the per-hour granularity of traffic rates. To convert the forecasted per-day rate to per-hour rate, we perform time series decomposition. A typical time series consists of three components: trend component, cycle (or seasonal) component and irregular component [3]: $X_t = T_t + S_t + R_t$, where: X_t is the value of the series at time t; T_t is the trend component, S_t is the seasonal component, and R_t is the random effect or irregular component. T_t exists when the time series data

```
$L_{i,j} = predict(T_{i,j}, R_{i,j})$: // Latency in seconds.
1  // assuming no latency buildup in $d_{i-1}$, so $L_{i,0} = 0$.
2  for each hour of $h_j$, where $1 \leq j \leq 24$:
3      if $T_{i,j} > R_{i,j}$: // increase latency buildup
4          $L_{i,j} = L_{i,j-1} + \frac{3600(T_{i,j} - R_{i,j})}{R_{i,j}}$
5      else: //decrease latency buildup if any
6          $L_{i,j} = L_{i,j-1} - \frac{3600(R_{i,j} - T_{i,j})}{R_{i,j}}$
7          if $L_{i,j} < 0$:
8              $L_{i,j} = 0$
```

Figure 7: Algorithm of predicting replication latency

gradually shift across time; S_t represents repeating patterns (e.g., daily or weekly); while R_t is time-independent and is characterised by two properties: (1) the process that generates the data has a constant mean; and (2) the variability of the values is constant over time.

Time series decomposition can eliminate the seasonal component based on seasonal index, which measures how much the average for a particular period tends to be above (or below) the expected value. Seasonal index is an average that can be used to compare an actual observation relative to what it would be if there were no seasonal variation. In this work, we utilize seasonal index to convert per-day data to per-hour data. Calculating seasonal index requires the setting of seasonal period, which we choose 168 hours (i.e., 1week, or 7 days * 24 hours). Once the seasonal indexes are calculated, each day of data can be "seasonalized" to obtain per-hour data based on the particular day in a week. We will present details in Section 5.

4.3 Forecasting future traffic rate with regression analysis

When regression analysis (i.e., time series regression) is used to forecast future traffic rates, the weekly average values are used instead. Specifically, for the consecutive weekly average traffic rate of W_t, where t is the week id, a trend line (i.e., the linear fitting of W_t) can be obtained in the form of $Y_t = aW_t + b$. The a is the slope of the change, or the growth rate. With the trend line, future weekly average traffic rate W_t can be forecasted.

Future W_t values obtained with time series regression are actually the "deseasonalized" forecast values. To convert to specific hourly rate, W_t needs to be "seasonalized", similar to the process of using ARIMA model. The details of the conversion will be presented in Section 5.

4.4 Predicting replication latency

For a particular day of d_i, assuming the per-hour traffic rate of $T_{i,j}$ and the relay capacity $R_{i,j}$ are known (totally 24 data points). Apparently if the processing capacity is larger than or equal to the incoming traffic rate, or $T_{i,j} \leq R_{i,j}$, no replication latency buildup will result; otherwise, there will be replication latency buildup. We define "peak period" as the time period where the incoming rate exceeds the relay capacity, and"non-peak period" otherwise. For the latency buildup during peak traffic time, it can be gradually consumed during non-peak traffic time when the traffic rate is relatively low. We assume eventually all the latency buildup

```
$R_{i,j} = capacity(T_{i,j}, L_{sla})$:
$variables$:
1  $R_{i,j(min)}$: a $R_{i,j}$ such that $max(L_{i,j}) \geq L_{sla}$
2  $R_{i,j(max)}$: a $R_{i,j}$ such that $max(L_{i,j}) \leq L_{sla}$

1  $R_{i,j(lf)} = R_{i,j(min)}$
2  $R_{i,j(rt)} = R_{i,j(max)}$
3  while $R_{i,j(lf)} < R_{i,j(rt)} - 1$:
4      $R_{i,j(mid)} = \frac{R_{i,j(lf)} + R_{i,j(rt)}}{2}$
5      $L_{i,j} = predict(T_{i,j}, R_{i,j(mid)})$
6      if $max(L_{i,j}) \leq L_{sla}$:
7          $R_{i,j(rt)} = R_{i,j(mid)}$
8      else:
9          $R_{i,j(lf)} = R_{i,j(mid)}$
10 return $R_{i,j(rt)}$
```

Figure 8: Algorithm of determining replication capacity

will be consumed by the end of non-peak traffic time of any day. This is a reasonable assumption as otherwise the replication latency will continually grow across days, which is unusable scenario for any business logic or SLA definition.

The mechanism to predict replication latency for any hour of a day is based on numerical calculations. Assuming there is no latency buildup from previous day of d_{i-1} (i.e., $L_{i,0} = 0$, for each successive hour h_j where $j > 0$, the traffic rate is compared to relay capacity at h_j. If the incoming rate is higher, it will incur additional replication latency at that time. Otherwise, previous replication latency, if any, will be decreased, as the relay has additional capacity to consume previously built-up latency. The amount of latency change is based on the difference between the two rates, that is, $L_\delta = \frac{T_{i,j} - R_{i,j}}{R_{i,j}}$ hours. This process continues for each hour that follows, and the entire data set of relay lag is constructed. The algorithm is shown in Figure 7.

4.5 Determining replication capacity

Given SLA requirement L_{sla} (i.e., the maximum allowed replication latency) and traffic rate of $T_{i,j}$, we need to obtain the minimum required relay capacity of $R_{i,j}$. Previously we have shown how to predict the replication latency $L_{i,j}$ given $T_{i,j}$ and $R_{i,j}$. For simplicity we denote the process as a function of $predict()$, so we have $L_{i,j} = predict(T_{i,j}, R_{i,j})$. For each day d_i, we can also easily obtain the maximum replication latency of $L_{i,max}$ of $L_{i,j}$, denoted by $L_{i,max} = max(L_{i,j})$.

To find out the relay capacity $R_{i,j}$ that having $L_{i,max} \leq L_{sla}$, we can do a binary searching on the minimum value of $R_{i,j}$. In order to do binary searching, we need to have two base values of $L_{i,j}$ that are below and above L_{sla}, respectively. These two values can be easily found out. With these two base values (denoted by $R_{i,j(min)}$ and $R_{i,j(max)}$, we can perform a binary searching in the value space between them. The time complexity is $log(R_{i,j(max)} - R_{i,j(min)})$. The algorithm is shown in Figure 8.

4.6 Determining replication headroom

Determining replication headroom consists of two questions. Firstly, given the SLA of L_{sla} and relay capacity of

```
$S_{i,j} = headroom(R_{i,j}, L_{sla})$: // The scaling factor of day $d_i$
variables:
1  $S_{i,j(min)}$: a $S_{i,j}$ such that $max(L_{i,j}) \leq L_{sla}$
2  $S_{i,j(max)}$: a $S_{i,j}$ such that $max(L_{i,j}) \geq L_{sla}$

1  $S_{i,j(lf)} = S_{i,j(min)}$
2  $S_{i,j(rt)} = S_{i,j(max)}$
3  while $S_{i,j(lf)} < S_{i,j(rt)} - 0.01$:
4      $S_{i,j(mid)} = \frac{S_{i,j(lf)} + S_{i,j(rt)}}{2}$
5      $L_{i,j} = predict(T_{i,j} * S_{i,j(mid)}, R_{i,j(mid)})$
6      if $max(L_{i,j}) \leq L_{sla}$:
7          $S_{i,j(lf)} = S_{i,j(mid)}$
8      else:
9          $S_{i,j(rt)} = S_{i,j(mid)}$
10 return $S_{i,j(lf)}$
```

Figure 9: Algorithm of determining headroom (scaling factor)

$R_{i,j}$, what are the highest traffic rate $T_{i,j}$ it can handle without exceeding SLA? (2) what are the expected date of the previous answer?

To determine the highest $T_{i,j}$ without compromising L_{sla}, we can again use binary searching. For this purpose, we assume the same shape of traffic rates inside a day, and scale the rate values by multiplying a scaling factor (e.g., 2.0). Specifically, First we find two base scaling factors (denoted by $S_{i,j(min)}$ and $S_{i,j(max)}$, where $S_{i,j(min)}$ will fulfill L_{sla}, while $S_{i,j(max)}$ will exceed L_{sla}. Then binary searching is performed on the $T_{i,j}$ such that $predict(T_{i,j}, R_{i,j}) \leq L_{sla}$. The algorithm is shown in Figure 9.

Once the highest $S_{i,j}$ (denoted by $T_{i,j(h)}$)is obtained, determining the corresponding date will be done by considering the traffic growth rate r, which denotes yearly growth rate. Specifically, with the forecasting model, the traffic rate of day d_k can be obtained and denoted by $\frac{365 S_{i,j}}{r} + d_i$.

4.7 Determining SLA

We have shown above that the replication latency values for each time unit can be obtained if we know the incoming traffic rate and the replication processing capacity. Once the replication latencies are known, determining the appropriate SLA for replication latency is quite straightforward. For the first form of SLA determination (i.e., L_{max}), we can simply iterate through the entire week, find the largest replication latency value, and assign it to L_{max}. For the second form of SLA determination (i.e., find p99.9 value of L_{p99}, we can apply a adapted binary searching algorithm to find the value of L_{p99}, such that 99% of time the replication latency is below L_{p99}. For simplicity, we will not elaborate on the detailed algorithm.

Table 1: The estimates, standard errors, t value and corresponding significance

Lag/Diff.	Estimate	Std. Err.	t	Significance
AR Lag 1	-0.198	0.104	-1.903	0.063
AR Lag 2	-0.312	0.105	-2.983	0.004
AR Lag 3	-0.245	0.107	-2.290	0.026
AR Lag 4	-0.259	0.106	-2.435	0.018
AR Lag 5	-0.284	0.109	-2.611	0.012
AR Lag 6	-0.214	0.106	-2.030	0.047
AR Lag 7	0.655	0.108	6.083	0.000
Difference	1			

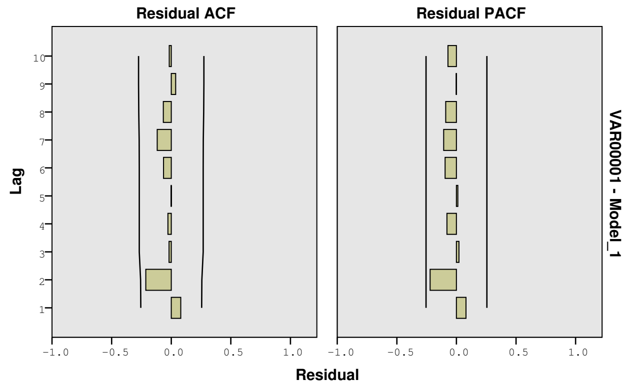


Figure 10: ACF and PACF of ARIMA(7,1,0)

5. FORECASTING TRAFFIC RATE

In this section, we address the forecasting question by using both time series model and regression analysis model. Both models requires 2 steps: (1) forecasting rate of future days (or weeks) and (2) converting per-day (or per-week) rate to per-hour rate.

5.1 Forecasting using time series model

For 6-weeks of time (i.e., 42 days), the average traffic rate of each hour is plotted in Figure 3. From the figure, we can easily see that the time series is not a *stationary time series*; instead, it exhibits strong *seasonal pattern*. It can be seen to have a weekly seasonal pattern. There is also a trend pattern in the time series which we will discover later. The trend pattern is not easy to spot by human eyes due to two reasons: (1) the trend is shadowed by other patterns (e.g., seasonal patterns); and (2) the trend is not obvious since the shift is slow.

5.1.1 Step-I: Forecasting rate of future days

ARIMA(p,d,q) model needs to know the exact values of (p,d,q). Based on various investigations into the properties of the data, we choose ARIMA(7,1,0). In Figure 10 we show the ACF and PACF of the model, we can see that the model of ARIMA(7,1,0) adequately fits the data.

We obtained the specific parameters (AR lags and difference) of the ARIMA(7,1,0). With confidence interval width of 95%, we show the estimates, standard errors, t value and corresponding significance in Table 1. The stationary R^2 is 0.888.

5.1.2 Step-II: Converting per-day rate to per-hour rate

We will *decompose* the time series to obtain seasonal indexes of each hour in a week (i.e., totally 168 indexes in a week). Calculating the seasonal indexes can be done as follows:

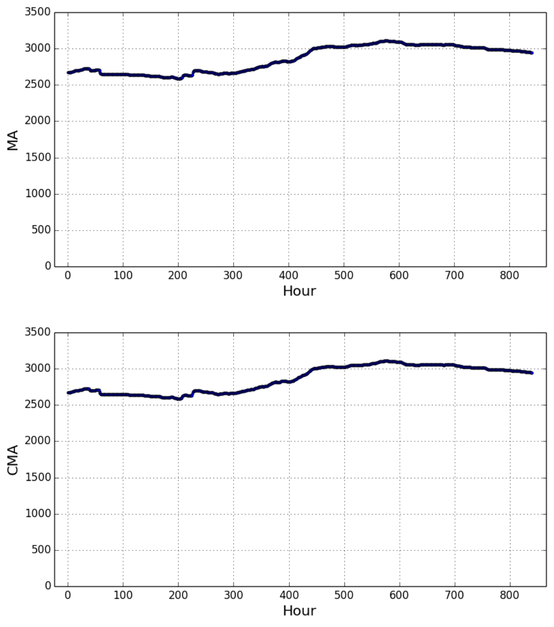


Figure 11: (a) Weekly moving average (MA_t) and (b) centered moving average (CMA_t)

1. *Calculating weekly moving average (MA)* $MA_t = \frac{\sum_{i=-63}^{64} X_{t+i}}{168}$. MA_t gives the average weekly traffic rate.

2. *Calculating centered moving average (CMA)* $CMA_t = \frac{MA_t + MA_{t+1}}{2}$. Since the period is an even number (i.e., 168), the moving average values we computed do not correspond directly to the original hours of the time series. This can be resolved by computing the average of the two neighboring averages. The centered moving averages CMA_t represent the trend in the time series and any random variation that was not removed by using moving averages.

3. *Calculating seasonal irregular values* $SII_t = \frac{X_t}{CMA_t}$. Dividing the time series value by the centered moving average, we can get the seasonal irregular values SII_t for each hours.

4. *Calculating seasonal indexes* $SI_t = \overline{SII_t}$. for each of the 168 hours, its seasonal index can be averaged over all corresponding seasonal irregular values. Eventually, we get 168 seasonal indexes SI_t, which are the "scaling" factor of the particular hours.

5. *Adjusting seasonal indexes* $SIndex_t = \frac{SI_t * 168}{\sum_{i=0}^{167} SI_{t+i}}$. Since the average seasonal index should equal 1.00, we need to adjust the seasonal indexes by multiplying 168 and dividing the sum of all unadjusted seasonal indexes.

Once the seasonal indexes are obtained, per-day traffic rate can be easily converted to per-hour rate by "seasonaling" the forecasted data. Specifically, to convert the traffic rate of a forecasted day, let's assume it is the i_{th} day in a week, where $0 \leq i \leq 6$ of a week, say d_i. We use $r_{i,j}$ to denote the forecasted rate of hour j in d_i. For each hour j, we first obtain its in-week hour offsets $h_j = 24i + j$. Then for hour h_j, the per-hour traffic rate is calculated as $d_{i,j} * SIndex_{h_j}$, where $SIndex_{h_j}$ is the corresponding seasonal index for hour h_j.

5.2 Forecasting using regression analysis

The process of forecasting using regression analysis shares the same Step-II with the time series model described above. So we will focus on the Step-I. In Step-I of this model, the per-week traffic rates in future weeks are obtained, which is different from the time series model. The reason for this is the weekly seasonality.

An alternative way of forecasting using regression analysis is to separate the forecasting of each weekday and treat them as independent time series. Specifically, for each of the weekdays (e.g., Monday and Sunday), a separate trending line is obtained. Based on the respective trending lines, the traffic rates of each of the seven weekdays will be forecasted. Though theoretically this treatment should also work, we feel it is a overkill of our problem. In addition, maintaining and processing of seven time series models is a non-trivial tasks.

Once future weekly traffic rates are forecasted, they can be converted to hourly traffic rates using the same seasonal indexes we described before. Specifically, assuming the weekly traffic rate is w, for each hour j ($0 \leq j < 168$) in a forecasted week, the forecasted traffic rate $r_j = w * SIndex_{h_j}$.

In this section, we use the a particular Databus instance to demonstrate the effectiveness of the proposed models when answering the four types of questions in Section 1.

5.3 Some headroom is necessary

Any prediction model with Internet traffic will inevitably have errors for various reasons. The major cause of prediction errors is the high dynamics of Internet traffic. When the prediction underestimates the traffic volume, the actual replication latency will be lower than expected. Note that such underestimation will not violate the performance SLA which is defined based on predicted values. The only cost is over-provisioned computing resources.

However, when the predicted traffic volume is lower than actual value, the performance SLA could be violated. To address this issue, it is absolutely necessary to give certain headroom when determining various metrics such as SLA. The exact amount of headroom given depends on a set of factors including: (1) historical performance of forecasting;

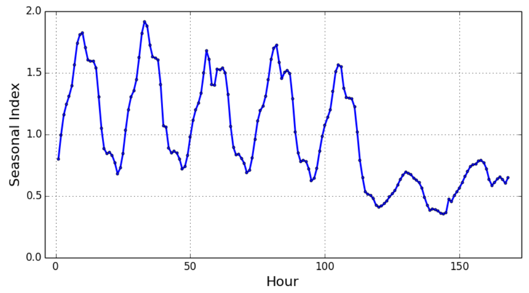


Figure 12: Adjusted seasonal index ($SIndex_t$) (Each hour of the week has a data point, totally 168 data points)

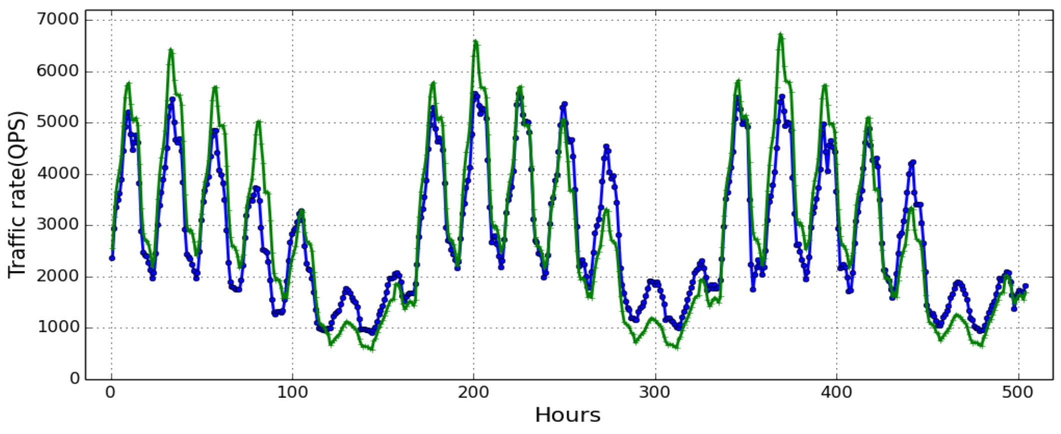


Figure 13: Observed (in blue) and forecasted (in green) hourly traffic using *ARIMA time series* model for 3 weeks

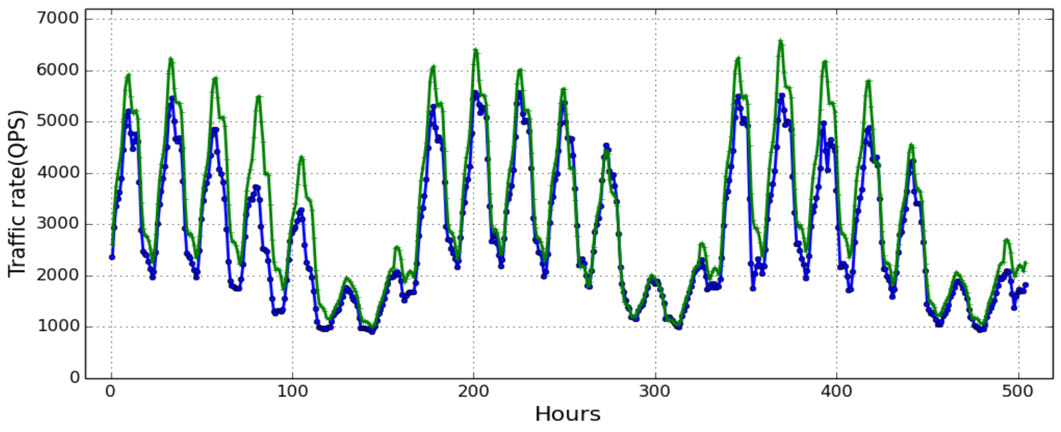


Figure 14: Observed (in blue) and forecasted (in green) hourly traffic using *regression analysis* model for 3 weeks

(2) the consequences (e.g., business cost) of violating SLA; (3) the cost of over-provisioning computing resources. We will not elaborate on this in this writing.

6. EVALUATION

6.1 Future traffic rate forecasting

According to the common practice of forecasting using ARIMA model, the forecasted data length should not exceed half of the historical data length [3]. So with 42 days of historical data, typically up to 21 days into the future can be forecasted using ARIMA model.

We show individual steps of calculating seasonal indexes according to Section 5.1.2. The MA (moving average) and CMA (centered moving average) are shown in Figure 11, respectively. We also show the adjusted seasonal index in Figure 12.

Using both the ARIMA time series model and regression analysis model, we obtained the forecasted hourly traffic rates for the later 21 days. In Figure 13 we show both the observed (raw) values and the forecasted values of ARIMA time series model. Figure 14 shows the results of regression analysis model. We also evaluated the forecasting accuracy by calculating the root mean square deviation (RMSD). ARIMA model has a RMSD value of 547, while regression analysis gives 578. For comparison, we choose a baseline MA (Moving Average) model which predicts an hour j's traffic rate based on historical data of exactly 1-week ago, that is, $T_{i,j} = T_{i-7,j}$. The corresponding RMSD value is 781, considerably larger than both of our proposed methods. Also note that though for this particular time series data, ARIMA model gives better accuracy, the difference is only about 6%. We also evaluated with other time series, the performance results vary, and both models give quite similar accuracy.

6.2 Replication latency prediction

To understand how well the numerical analysis model predicts replication latencies, we compared the numerically calculated results to observed values in one of our most resource-limited instance. We choose one day of replication latencies for a single Databus Relay instance as the base value. We fed the incoming traffic rates and relay capacity for that Relay instance into the numerical analysis model and then calculated the latencies.

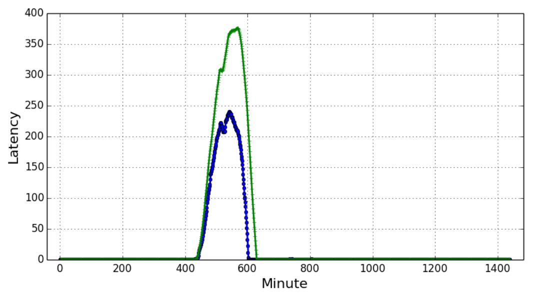


Figure 15: Comparing calculated relay lag to real lag (The shapes of the two curves are similar, the peak values differ by 1.6X.)

The two sets of results are shown in Figure 15. From the figure, we see that these results have almost identical shapes across the time line. The absolute peak values are also very close (i.e., 376 vs. 240, less than 1.6X). Given the data complexities and variations in the production environment (for instance, network bandwidth impact, network latency, cpu and other resource contentions, etc.), we feel the accuracy of the numerical analysis model is satisfactory.

6.3 Replication capacity determination

To determine the replication capacity, we chose a single day and would like to determine the replication capacity needed to ensure a SLA that is defined as the maximum replication latency being 60 seconds.

The average traffic rate of the day is about 2386 event/s. We firstly found that the capacity of 2500 event/s results a maximum latency of 8871 seconds, which violates the SLA. The other base value of capacity being 5000 event/s results in less than 1-second latency. Using the two base values, we perform binary searching. We plotted the min/max/mid values used in the algorithm during the binary searching in Figure 16(a). It only takes 12 steps to finally determine the exact replication capacity needed - 3474 event/s, which results in 57 seconds of replication latency.

6.4 Replication headroom determination

We chose the replication capacity being 5000 event/s and attempted to determine the maximum scaling factor that it can handle without violating the same SLA as defined before (i.e., maximum of 60 seconds). The raw traffic rates (i.e., scaling factor being 1.0) does not violate SLA, while a scaling factor of 2.0 results in more than 8000 second latency. Starting with these two scaling factors, binary searching takes 9 steps to determine the maximum scaling factor is 1.44. Assuming a yearly growth rate of 30%, it will take about 13 months to reach that traffic rate.

6.5 SLA determination

To determine SLAs, let's assume the time period is the entire 3 future weeks and the replication processing capacity is 6000 event/sec, we find the maximum replication latency is about 1135.2 seconds. Based on this, the maximum latency value in an appropriate SLA can be 1136 seconds, that is $L_{max} = 1136$. Correspondingly, we perform binary searching on the percentile values and find the $p99$ value of the SLA $L_{p99} = 850$.

7. DISCUSSION

We now discuss a few interesting issues/aspects in the problems and solutions.

7.1 Strong seasonal pattern of web traffic

LinkedIn's production web traffic shows strong patterns of seasonality, and we believe such seasonality also exists in the traffics other Internet companies. Both strong daily and weekly seasonal patterns are observed. Due to limited data length, in this work we did not study the yearly seasonal pattern. Though such seasonality adds difficulty to capacity planning, the latter actually can potentially take advantage of the seasonality to reduce operation cost and improve the performance. For instance, if the seasonal pattern predicts that the traffic rate will lower down in the next few hours, we might want to shrink the computing resources (e.g., network bandwidth, number of instances) to save operation cost. Similarly, if the traffic rates are expected to increase soon, we can allocate more computing resources to improve the performance. Such dynamic allocation of resources fit well into a cloud computing deployment. We will leave this to future work.

7.2 Forecasting using time series vs. regression analysis

We have presented two methods to forecast future traffic rates: time series model and regression analysis. Both methods then predict hourly traffic rates based on the same seasonal indexes. Though the accuracies of these two methods are quite similar, the design rational for these two methods are *fundamentally* different. Regression analysis assumes little dependency between two neighboring data points (i.e., the traffic rates of two succeeding time), while time series model better fits otherwise. The fact that the outcomes from the two methods are similar suggest the following two characteristics about the data dependency. First, the dependency between neighboring hourly-aggregated data points in our particular production traffic is minor, hence time series model does not gain advantage over regression analysis. Intuitively, since our data points are hourly-aggregated, it is not surprising to see little dependency between two hours. Second, our regression analysis model works on weekly-aggregated data points, which have even less dependency between two neighboring data points. Thus, both ARIMA and regression analysis models suffice in our particular scenario.

7.3 Applying to other areas

Though this work is specific to database replication capacity planning, the observations we made regarding traffic patterns and the models we proposed can help solve similar problems in other areas. For instance, online log processing applications are desired to keep up with the incoming traffic rate. Typical designs/deployments of log processing are fixed, not adaptive to log generation rates. As a result, the processing may incur delays when incoming log generation rates are high. High log generation rates are caused by high traffic rates during peak hours. Similarly, the accumulated delays will decrease during non-peak time. It also has similar capacity planning questions that need to be answered. The characteristics are very similar to those of the database replication process. All the mechanisms we proposed in this work can be applied there.

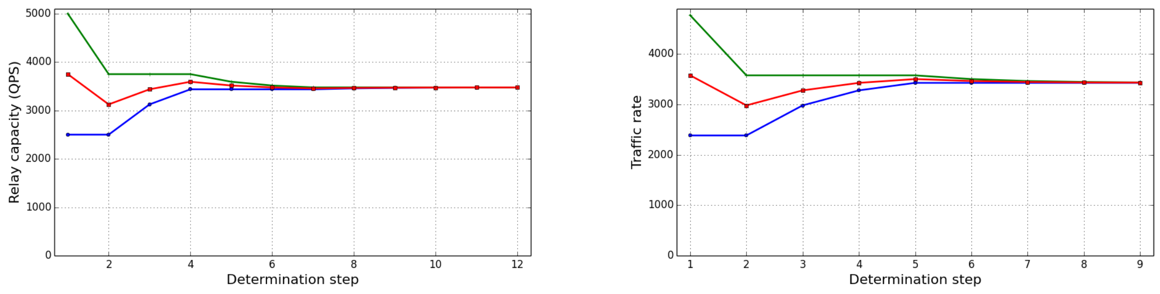


Figure 16: The binary searching steps to determine replication capacity (a) and replication headroom (b)

8. LESSONS LEARNED

During the course of this work, we learned a set of lessons that we would like to share.

8.1 Approximation model works well when the entire data set is not available

When only limited values of incoming traffic are known, while the full curves (i.e., the complete data points corresponding to the incoming traffic rates) are not available, it is difficult to determine the replication latency by performing numerical analysis as presented in Section 4. Partly for this reason, we developed an approximation model to quickly determine the maximum replication latency. This model requires knowing the average traffic rates. Based on average rate, the peak traffic rates can be approximated by scaling the average traffic rate. Based on our experience, for such web-type traffic rates, the rule of thumb is to scale average traffic rate by $1.5X$ to obtain the peak traffic rates.

Once we know the incoming peak traffic rate, we assume the peak hours will last for about 8 hours (which roughly correlates with the length of typical business hours). We denote these 8 hours as "busy period". To estimate the maximum replication latency, we also need to know the non-busy incoming traffic rate. For simplicity, we approximate the non-busy rate by halving the average traffic rate. To gain an idea of these values compared to the entire day's traffic, in Figure 17 we plot the raw traffic rates of the day of April 15th, 2014 for a single instance. Each data point represents a 5-minute average of traffic rate. The thick blue line shows the daily average of the traffic rate, red line is the busy traffic rate, and green line is the non-busy rate.

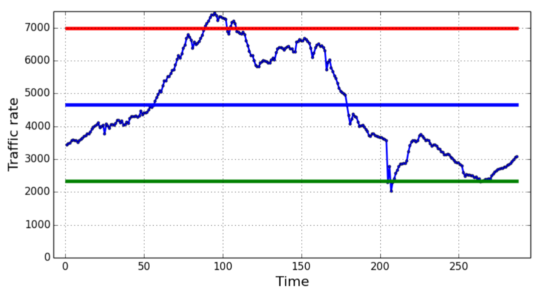


Figure 17: Approximation model (Each data points represents 5 minutes)

Let R_{cap} be the relay capacity (e.g., 6K event/s), T_{avg} be the average incoming traffic rate, T_{busy} be the busy traffic rate, T_{idle} be the non-busy traffic rate, and P be the duration of peak period; we can deduct the maximum replication latency (denoted by La) by $La = \frac{(T_{peak} - R_{cap})P}{R_{cap} - T_{idle}}$. Note that $R_{cap} - T_{idle}$ gives the extra consumption rate that drains the previous latency buildup. Let $T_{busy} = 1.5T_{avg}$ and $T_{idle} = 0.5T_{avg}$. Also assume the time granularity is per second. The equation calculating La in seconds becomes $La = \frac{28800(1.5T_{avg} - R_{cap})}{R_{cap} - 0.5T_{avg}}$.

With the above equation, we can similarly address other capacity planing questions. The corresponding solutions only need a simple linear transformation of variables. Specifically, to determine the needed relay capacity, we use $R_{cap} = \frac{(43200+0.5*La)R_avg}{28800+La}$. To determine the maximum average incoming rate we can handle, we have $T_{avg} = \frac{(28800+La)T_{cap}}{43200+0.5La}$. We evaluated the accuracy of this approximation model by comparing this model to the numerical calculation model, we found that the maximum replication lag obtained by approximation model is about $1.7X$ of that of the numerical calculations. Though it is not quite accurate and only gives the *single* maximum lag, it is a quick way to roughly estimate the worst replication latency. In addition, it does not need the full data set as required by the numerical calculations.

8.2 Internet traffic is highly dynamic, so give enough headroom in capacity planning

Based on our observations into the production traffic, the traffic rates across time can be very dynamic, sometimes bursty. The variability is present in multiple granularities including day, hour and minute. We observed highest variability inside a day. For instance, for the first day in our presented 6-week data, out of 24 data points (hourly aggregated), the mean is 3017, while the standard deviation is 1195. The primary reason for the high variability inside a day is the human activities with regard to business hours.

Even with daily aggregated data of 60 days (totally 60 data points, each for a day), the variability is still high with mean of 2787 and standard deviation of 864. We believe the primary reason is difference between the weekdays and weekends inside a week. Compared to the variability of hourly and daily traffic, the variability inside an hour (i.e., per-minute aggregation) is much smaller. For instance, for a particular workday, we find the maximum variability has the mean of 2484 and standard deviation of 374.

Given the high variability of web traffic and the traffic growth trend, it is important to give enough headroom in capacity planning. A careless capacity planning strategy would simply use the average values (e.g., average traffic rate), however, the average value can be significantly misleading due to the dynamic property. For instance, for our 60-day of hourly aggregated traffic, we found that the highest traffic rate can be $2.5X$ of the mean traffic rate.

9. RELATED WORK

This section will present some relevant works in literature.

9.1 Database replication latency

Database replication latencies have been studied in several works. An approach of replicating the commit operations rather than the transactional log is proposed in [12] to reduce replication latency. Database replication latencies in particular applications are also studied [16]. Some other works [2, 1] focus on MySql replication latency and evaluate the latencies of different types of transactions.

9.2 Time Series Model

Time series data are commonly decomposed into trend, season and noise. Noise is often modeled by stationary ARMA (autoregressive moving average) [4, 10] process. Box-Jenkins proposes ARIMA model (differencing) [3], which is a generalization of ARMA model. This model is generally referred to as an ARIMA(p,d,q) model where parameters p, d, and q are non-negative integers that refer to the order of the autoregressive, integrated, and moving average parts of the model respectively. Other models include Holt-Winters (HW) [5] and State Space Model (SSM) [8].

9.3 Forecasting network and Internet traffic

Work in [6] presents both neural networks and two time series (ARIMA and Holt-Winters) to forecast the amount of TCP/IP traffic. [13] found that IP backbone traffic exhibits visible long term trends, strong periodicities and variability at multiple time scales; Work [14] proposes time series based model to forecast when to add more network bandwidth. [11] uses time series model to extract the trends and seasonal patterns from page view data. Other works [9] study the fine-grained time series characteristics of network traffic at an Internet edge network to facilitate anomaly detection.

10. CONCLUSION

In this work we study the problem of taming the database replication latency for LinkedIn Internet traffic. Based on our observations into production traffic and various playing parts, we develop practical and effective models to answer a set of other business-critical questions related to capacity planning. These questions include: future traffic rate forecasting, replication latency prediction, replication capacity determination, replication headroom determination and SLA determination.

11. REFERENCES

[1] *How fast is MySQL replication?* http://www.xaprb.com/blog/2007/10/23/how-fast-is-mysql-replication/.

[2] *Investigating MySQL Replication Latency in Percona XtraDB Cluster.* http://www.mysqlperformanceblog.com/2013/03/03/investigating-replication-latency-in-percona-xtradb-cluster/.

[3] G. E. P. Box and G. M. Jenkins. *Time Series Analysis: Forecasting and Control.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 3rd edition, 1994.

[4] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting.* Springer, 2nd edition, Mar. 2002.

[5] C. Chatfield. *The Analysis of Time Series: An Introduction, Sixth Edition (Chapman & Hall/CRC Texts in Statistical Science).* Chapman and Hall/CRC, 6 edition, July 2003.

[6] P. Cortez, M. Rio, M. Rocha, and P. Sousa. *Expert Systems*, 29(2):143–155, 2012.

[7] S. Das, C. Botev, and et al. All aboard the databus!: Linkedin's scalable consistent change data capture platform. SoCC '12, New York, NY, USA, 2012.

[8] J. Durbin and S. J. Koopman. *Time series analysis by state space methods.* Oxford Univ. Press, 2001.

[9] C. James and H. A. Murthy. Decoupling non-stationary and stationary components in long range network time series in the context of anomaly detection. In *Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012)*, LCN '12, Washington, DC, USA, 2012.

[10] T. H. Lai. Time series analysis univariate and multivariate methods: William w.s. wei, (addison-wesley, reading, ma, 1990). *International Journal of Forecasting*, 7(3):389–390, 1991.

[11] J. Li and A. Moore. Forecasting web page views: Methods and observations. *JMLR (Journal of Machine Learning Research)*, 9(Oct):2217–2250, 2008.

[12] H. Mahmoud, F. Nawab, A. Pucher, D. Agrawal, and A. El Abbadi. *Proc. VLDB Endow.*, 6(9):661–672, July 2013.

[13] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot. Long-term forecasting of internet backbone traffic. *Trans. Neur. Netw.*, 16(5):1110–1124, Sept. 2005.

[14] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot. Long-term forecasting of internet backbone traffic. *Trans. Neur. Netw.*, 16(5), Sept. 2005.

[15] L. Qiao, K. Surlaker, S. Das, and et.al. On brewing fresh espresso: Linkedin's distributed data serving platform. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, New York, NY, USA, 2013.

[16] P. E. Tun, T. T. Soe, C. Myint, N. N. Ei, M. M. Oo, L. L. Yee, and A. Thida. In *Proceedings of the 3rd International Conference on Communications and Information Technology*, CIT'09, Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS).