BIG DATA DIFFERENTIATORS: A QoE PREDICTIVE MODEL FOR VIDEO SDN

Hui Zhang
Conviva

Abstract

Providing a positive viewing experience is critical for content owners relying on advertising and subscription-based revenue models to capitalize on the opportunity present in online video. However, with more viewers turning to online sources for consumption, understanding the quality of experience (QoE) and providing an optimal QoE becomes more critical. In this paper, we will discuss a possible online video predictive model for improving QoE using global intelligence extracted from analyzing billions of streams in real time utilizing a big data processing platform. While predictive modeling has been used in conjunction with big data to analyze historical and current trends for countless other disciplines, its application in digital media delivery has not yet been explored. Our research shows that on-line predictive modeling can provide tremendous value to those looking to monetize and enhance the viewer experience.

INTRODUCTION

Online video streaming is one of the most important applications on the Internet. Today, more than 57% of Internet traffic is video and the percentage is predicted to reach 69% in 2017 [1]. At the same time, users are demanding better and higher quality video (e.g. HD and Ultra-HD or 4K video) [2][3][4]. Ensuring video Quality of Experience (QoE) is becoming, and will continue to be, a challenge for both video content publishers and service providers.

Existing protocols that enable Internet video streaming assume two fixed end points (e.g., a video server and a client streaming the video) and varying resource availability between, or at, the two end points. The key mechanism to achieve better quality is to rapidly *adapt* or react to congestion and/or apply changes in resource availability along the path or at the end points. One example of this approach is the set of adaptive bitrate algorithms that have been implemented in many video players, often highly optimized for specific streaming protocols. While these adaptive solutions have served us well in the past, they become sub-optimal as video streaming services become more sophisticated and new opportunities emerge.

Towards a Video Software-Defined Network

Most scalable and reliable Internet video streaming services have many control knobs for adjusting video playback quality across different layers in the network stack that call for cross-layer optimizations. For example, many video services are implemented using multiple servers, typically distributed geographically, e.g., Content Delivery Networks (CDNs). For each video session, a video player can select one of many possible servers from which to start streaming. If the duration of the video session is long, it is possible to switch servers during the lifetime of the session. The control plane is further complicated by recent trends in content providers utilizing multiple CDNs [5] and/or CDN federations [6]. In addition to streaming server selection, for an adaptive bitrate video streaming protocol, the initial bitrate needs to be selected and the player needs to continuously adapt the bitrate during the video playback. We argue that by leveraging and extending the recently proposed Software-Defined Networking (SDN) approach [7][20], it may be feasible to select the best Internet route on a per-video-session basis. Furthermore, we will not be surprised if other knobs, such as video encoding profiles [8], client or server initial TCP window size [9], or even the transport protocol itself, can be opened up for control.

The Case for a Data-Driven Predictive Model

The new opportunities call for cross-layer optimizations where the decisions involving different control knobs need to be considered together, instead of separately. This causes the decision space to grow exponentially, making it extremely difficult for a reactive protocol to do the best possible job. Indeed, it will take a long time for such a protocol to converge to a good decision. Next, we briefly discuss the challenges faced by existing protocols.

Typically, these protocols use static initial configuration parameters that are often sub-optimal. For example, adaptive streaming protocols usually start with a statically configured bitrate. If this bitrate is too low, the protocol might not even be able to reach the optimal rate by the time the video has ended (e.g., for a 30s or 60s news clip). Additionally, such pre-configured bitrate may be sub-optimal during periods of congestion or compromised bandwidth.

Even after an initial decision is made, when these protocols react, they don't always make the optimal decisions, which may further impact user experience. For example, in case of congestion, an adaptive bitrate protocol may switch up to a bitrate that cannot be sustained, and, as a result, the user may experience re-buffering.

In this paper, we make three arguments. First, given the fundamental limitations of reactive approaches, we argue for an alternative predictive approach, which aims to accurately predict the outcome of making a particular choice, e.g., will a stream be able to sustain a particular bitrate? In theory, a perfect prediction would allow protocols to use "optimal" configuration parameters and make "optimal" decisions. For example, it would be possible to exactly pick the largest sustainable bitrate for a video stream at start time.

Second, in order to accurately predict the outcome of a given choice, one may be tempted to use an analytical approach to model the environment, the streamer, the network, or some combination thereof. However, we believe this is infeasible due to the huge complexity of the delivery ecosystem, and this is also unnecessary. We instead argue for a data-driven empirical approach to leverage the information available from other players, streams or connections, i.e., use the performance experienced by other "similar" sessions to predict the performance for a given session. For example, if sessions located at some organization can sustain 2Mbps on average when streaming from a CDN, then it's likely that a new session from the same organization will be also able to stream at 2Mbps from the same CDN.

A Cloud-Based Big Data Solution: V-SDN

To this end, we propose a global control plane architecture – or Video SDN (V-SDN) – that continuously collects data from various sources, e.g., the quality of current and historical video sessions, and uses this information to maximize quality of other sessions. There are several challenges to implement such a system. First, we need an Internet-scale solution. To provide some perspective, we collect data from over 4 billion streams per month, and at each point in time, we are collecting data from up to 2 million concurrent streams. When applying to all Internet videos, that number becomes at least one order of magnitude bigger. Second, we need to process this information and make decisions in real time, i.e., milliseconds or sub-millisecond response time. Third, to make the best decisions, we need on-line prediction models to accurately model the session performance as new data is continuously collected.

We present the design and early experience with deploying such a control plane architecture, called video Global Optimization (GO). Conviva has built a cloud-based data platform using a data hub architecture [10], where data ingested from different sources are stored in a distributed file system, and different tools built on top of the data provide batch processing, stream processing, search, graph computation, ad-hoc analytical querying, time-series analytics, and statistical analytics computation. The data platform is designed to be distributed, horizontally scalable, and highly available. With this data platform, applications such as GO do not need to worry about the distributed nature of the system and are able to explore the data more freely. GO uses the quality-related information (e.g., current bitrate, re-buffering rates, start time, etc.) which is continuously collected from each streaming video client. Next, GO processes the quality information in real time, and, based on this information, provides hints to clients about the best bitrate or server to start with or switch to. For ease of deployment in today's Internet ecosystem, we make two simplifying assumptions. First, for a given session, GO selects the server and Internet path at the CDN granularity, instead of server granularity. Second, GO currently makes decisions at the start of a video session, and not in the middle of the session. These simplifying assumptions reduce both the frequency and the number of decisions GO needs to make. Despite these restrictions, our experience with deploying GO to optimize video streaming across several video sites shows that it is a highly advantageous step towards a fully featured global control plane.

In particular, due to the flexibility of our data platform architecture, it would be easy to extend GO to handle additional data sources, to make new types of decisions (e.g., select an encoding format), or to implement different algorithms.

GO System Overview

As shown in Figure 1, GO consists of two main components: (i) a backend that collects and processes the information about the video quality across all clients, and (ii) a client library that collects quality information at each client and sends it back to the

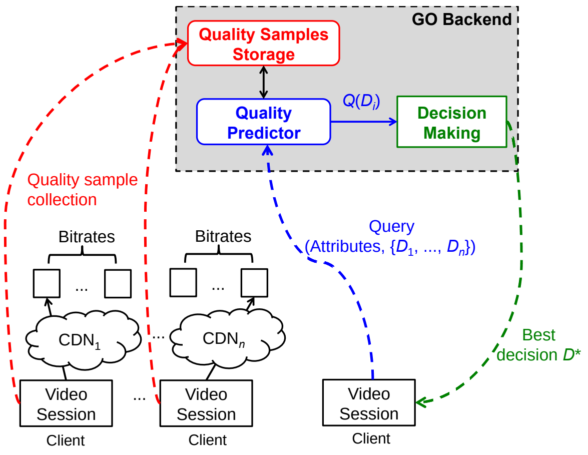


Figure 1: Go System overview

backend. In particular, the client library monitors the states of player and network condition, summarizes them in the form of quality samples, and sends these samples back to the GO backend.

The GO backend uses the streaming capability of the data platform to process the quality samples received from clients in real time, and predict the quality outcomes of future sessions for each possible decision. The GO system makes two decisions for each session: the initial or starting bitrate (most videos today are encoded at multiple resolutions), and the initial CDN to stream the content from (many content providers today are on or moving towards multi-CDN setup). The initial CDN and bitrate are chosen from a pre-determined set of options and they can be changed at any point during a session. Note that to play a video that is encoded at n different bitrates and stored on m CDNs, there are m × n options to chose from.

To pick one of these options, GO predicts the quality metric for each option, and then picks the one that corresponds to the highest quality. In particular, GO uses real-time, quality-related information from every client currently streaming video to predict the quality of another user, and uses this prediction to select the initial bitrate and CDN for a new user.

Quality Samples

As explained above, the client library collects various pieces of quality information, summarizes this information into quality samples, and sends these samples to the GO backend. More precisely, a quality sample contains a set of quality measurements and attributes. Based on the quality measurements, GO computes a set of quality metrics. In this paper, we focus on four industry-standard video quality metrics that have been shown to impact user's engagement [11][5]:

1. Buffering ratio: The percentage of time a session spends in buffering state, i.e., waiting for the player's buffer to replenish with enough data to continue the playback.

2. Join time: The time it takes to start playing the video from the time the user clicks the "play" button.

3. Average bitrate: Many of today's video players support adaptive bitrate switching to effectively react to changes in the bandwidth availability. The average bitrate of a session is simply the time-weighted average of the bitrates used in a given session.

4. Video start failures: Some sessions fail to start playing due to various reasons, including content unavailability, or CDN server overload.

In addition, a quality sample contains a

Name	*Description*	*Number of Unique Values*
ASN	The Autonomous System (AS) Number that the client IP belongs to	15K from multiple countries
Video asset	The video asset being streamed	80K
Content provider (site)	The video site providing the content	279
Initial CDN	The CDN that the video session starts with	19 including ISP, commercial, in-house
Initial bitrate	The bitrate that the video session starts with	15K video assets with multiple starting bitrates
Connection type	Type of last-mile connectivity	7 including WWAN, DSL, fiber-to-home

Table 1: Number of unique values for various session attributes

large number of client and video session attributes. Table 1 summarizes some information about these attributes. Results are derived from quality samples from over 800 million sessions or views (both successful and failed) over a one-month period.

CHALLENGES

Before we delve into solutions, we first discuss some of the challenges to building such a video quality prediction system. First, we describe the scalability challenges and then we discuss the algorithmic challenges in managing prediction errors.

Internet-Level Scalability

A global intelligence system needs to be scalable in order to make a large number of real-time decisions. This is especially true given the combinatorial nature of the session features, i.e., the number of feature combinations increases exponentially with number of base features. At the same time, the decisions need to be made in an extremely low (sub millisecond) constant time so that the decision making process doesn't adversely affect the video session.

Managing Prediction Error

Any prediction has error. The natural question is: where do errors come from and how do we manage them? We roughly follow the decomposition of prediction error developed in [12], specializing it to our setting. In quality prediction, there are four sources of prediction error:

1) Estimation error caused by limited data: All things being equal, more data will give a more accurate prediction because predictions are less impacted by random fluctuations in the available data. Given a large number of attributes and the combinatorial nature of the attributes (the number of partitions grow exponentially with more attributes), estimation error can be a serious problem in practice.

2) Bias due to missing or unused information: The bias occurs when one does not observe (or use) an attribute that is important for prediction. Bias is not alleviated by gathering data from more sessions, but by gathering more attributes from each session. There is a fundamental tradeoff between estimation error and bias that we need to address: more features reduce bias but also reduce the number of samples within each feature set and potentially increase estimation error if not designed carefully. On the other hand, the common approach to handle estimation error is to increase number of samples by aggregation, thus increasing bias.

3) Unavailability of recent data: In a practical system, there are delays in measuring, sending and processing quality samples. If conditions change rapidly, there may be no quality samples sufficiently close to the session under prediction. This is an extreme example of estimation error. In this case it may be necessary to model the evolution of the video ecosystem over time in order to extrapolate to the current time. Figure 4 shows per-minute quality variability. It shows that even with sufficient data, the mean value of quality samples belonging to sessions in the same partition could vary significantly. This clearly indicates that any practical algorithm has to be running in real time (on the order of one minute or less).

4) Noise: Outcomes may be affected by non-deterministic inputs that could not reasonably be observed or predicted by any system. For example, performance may be affected by exponential back-off at the data link layer, by the congestion generated by cross traffic at the network layer, or by a randomized algorithm somewhere in the networking stack. Though prediction error induced by such factors technically falls under the

category of bias, it is more useful to think of it as noise. Noise implies that some degree of prediction error is inevitable.

ALGORITHM

In this section, we present a practical algorithm that addresses the challenges presented in previous section.

The tradeoff between estimation error and bias naturally leads us to consider a class of algorithms that compute average quality outcomes for partitions of sessions under different feature sets, and then dynamically chooses the feature set that seems to work well for a given session. We propose the following basic structure for the algorithm. The algorithm chooses a set of features offline based on analysis. Then the algorithm starts collecting quality metrics for all the feature sets. Based on the quality metrics collected, each feature set will be given a weight so that the weights reflect statistical properties of the feature set and are thus updated when new quality samples are collected. Finally, when receiving a new session, we estimate the performance for each available decision using a weighted sum. Intuitively, the algorithm works as follows: for a given session, we look at the quality metrics of all the sessions that match this session exactly. If we have enough data and the metrics are statistically stable, the decision will be mostly based on that. Otherwise, we will have to remove a feature and look for quality metrics with similar sessions, etc.

The next question then becomes how to design a weighting scheme to balance between estimation error and bias to minimize overall prediction error.

George et al [13] considered this problem and proposed an algorithm called WIMSE (short for Weighted Inverse Mean Squared Error). As the name suggests, the weights w_i are chosen to be the inverse of an estimate of the mean squared error (MSE). Inverse-MSE weighting has the following desirable property: If the mean quality pi of each group is statistically independent and the mean squared error is estimated exactly, then the resulting prediction is an optimal estimator in the sense that it has the minimal overall prediction error among all samples drawn from the same distribution [13]. While in practice neither condition is met, George et al have found that WIMSE can nevertheless work well in most cases.

The basic algorithm still faces other practical challenges. However, due to space limitation in this paper, we concentrate on only the major challenges the algorithm faces.

DEPLOYMENT

We leverage recent advances in data systems to build our data platform, around Hadoop [14] and the Berkeley Data Analytics Stack (BDAS) ecosystems [16]. From a high-level design, we use HDFS as our primary storage layer and Spark as our primary computation engine. We also use other tools to provide both HA (high availability) and data processing abstraction such as batching (Spark [15]), streaming (Streaming Spark [15]), search (Solr [17]), analytical querying (Shark [18]), statistical analytics (R [19]), etc.

We implemented the GO system on top of our data platform using the streaming capability of the platform to train the model and produce decision tables every minute. The decision tables are then sent to a number of decision makers distributed across multiple locations. Finally, when a query comes into a decision maker, it runs the algorithm using the decision table and responds within milliseconds.

The GO system is currently deployed with multiple premium content publishers. However, there are several practical difficulties of evaluating the performance of GO in production deployment environments. First, most publishers do not want to perform A/B testing when they understand that the performance of some of the streams may not be optimal when they are grouped by the non-optimized version of the algorithm. Second, with all publishers, there are usually additional business considerations beyond the goal of optimizing the QoE of the video streams. For example, when using multiple CDNs, a publisher could get a lower price from a particular CDN if it would allocate more than a certain percentage of its total traffic to the CDN. This CDN allocation policy would put additional constraints on the GO optimization algorithm. Consider a scenario where a publisher has three CDNs X, Y, and Z, and has a minimum committed usage percentage on X and Y. This would mean that even if CDN Z was the best performing CDN based on the prediction algorithm, beyond certain percentage of traffic, no additional streams would be allocated to it.

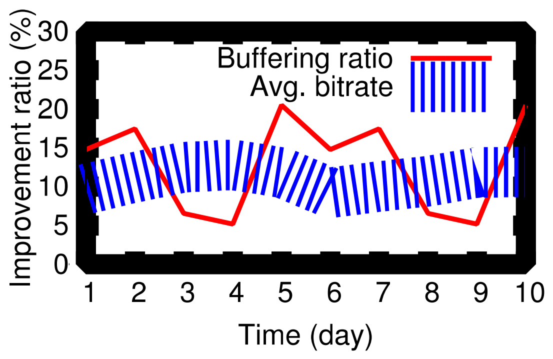


Figure 2: Performance improvement over time compared to baseline algorithm

Buffering Ratio and Average Bitrate: In Figure 2, we show the percentage of improvement of GO over the baseline (randomized decisions) with two performance metrics: buffering ratio and average bitrate. In particular, for buffering ratio, we show the percentage of reduction of buffering ratio for all sessions served by the GO algorithm in each day as compared to all sessions served by the baseline algorithm on the same day; for average bitrate, we show the percentage of increase of average bitrate over the entire session duration for all sessions served by the GO algorithm in each day as compared to all sessions served by the baseline on the same day. We present the comparison over a continuous time period of 10 days. There are several points worth noting. First, both metrics are improved simultaneously with GO as compared to the baseline. In contrast, with normal adaptive bitrate protocols without special optimization, the improvement of one metric usually results in the deterioration of the other. For example, the reduction of the buffering ratio usually comes together with the reduction of the average bitrate also. With GO, both metrics are improved simultaneously. Second, the performance improvement varies daily. The most likely explanation is that it is due to the CDN performance variation. To understand this

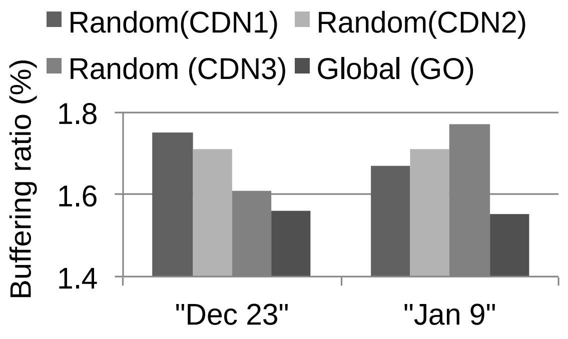
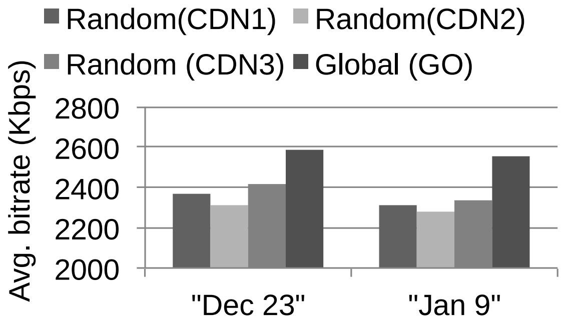


Figure 3: Buffering ratio for different CDNs at different points in time, illustrates CDN performance variability

Figure 4: Average bitrate for different CDNs at different points in time, demonstrates very much the same phenomenon as Figure 2

better, we compare the performance of all sessions under GO and the performance of all sessions on each of the three CDNs under the baseline algorithm. This is shown in Figures 3 and 4 respectively with buffering ratio and



average bitrate as performance metrics respectively. For each figure, we show the comparison in two separate days. Some additional points to note: first, the performances of sessions for different CDNs under the baseline do vary, with respect to both buffering ratio and average bitrate. Second, the sessions under GO perform better than the sessions for even the best of the three CDNs. This suggests that GO is not only looking for the best CDN on average, but also differentiates CDN performance in finer granular partition across time and space. In addition, if one compares the relative performance for each individual CDN, the ranking varies between the two days. In particular, with respect to the buffering ratio, CDN3 is the best on Dec 23, but CDN1 is the best on Jan 9; with respect to the average bitrate, CDN3 is the best on both days.

Interaction with the adaptive bitrate protocol: Since GO in this deployment only selects the bitrate and CDN at the beginning of each session and the HLS protocol controls the bitrate adaptation for the duration of the session, we would like to understand how the initial selection decisions by GO impacts the future adaptation decisions made by HLS. The number of bitrate switches per session made is a good indicator of how closely GO is able to select the ideal bitrate for a session. A good initial selection would result in a lower number of bitrate switches in the future. Figure 5 shows the comparison between GO, a static selection policy (traditional initial bitrate selection algorithm) and an algorithm that selects the starting bitrate at random. As shown in the figure, GO outperforms either case. Also note that static selection is as bad as picking a bitrate at random!

DISCUSSION

While our results look promising, the magnitude of the improvements may appear underwhelming. However, we believe that this is not due to the lack of potential of the approach, but it merely comes down to the

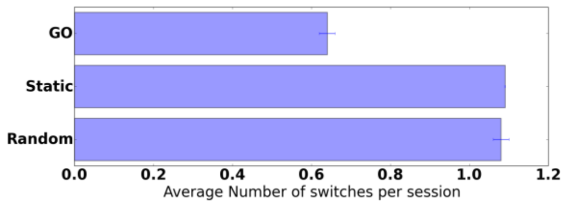


Figure 5: Average number of switches per session for GO initial bitrate selection, static initial selection (1200Kbps) and random selection

inherent limitations that are typical to any first instantiation of a radical approach. In the remainder of this section, we consider some of these limitations which, when removed, will result in much higher improvements.

Mid-stream selection: Currently, GO makes decisions at the sessions' start times. For a long session this may not be optimal, as, for example, the quality of the selected CDN may degrade during the session's lifetime. We are adding the ability to make the decisions during the mid-stream, as well. Note that in this case, the prediction is equally important, as switching to a new CDN is not guaranteed to increase the quality, especially if the quality degradation is due to the last mile or due to the client inability to render at a high bitrate.

Leveraging network and CDN information: GO makes decisions based on client side information only. While clients provide the most accurate information regarding the quality experienced by users and at the final viewing stage, this information may not always be optimal when making decisions. The decision process can be considerably improved if GO were to leverage information from other entities in the distribution ecosystem, including CDN servers, caches, switches and routers. Using such information, GO could significantly improve the prediction accuracy. For example, GO could learn much faster that a CDN server is overloaded by getting load information directly from that server than inferring this information from clients that experience quality issues when connected to that server.

Finer grain selection: Currently, GO selects the resource at the CDN granularity. This means that GO does not do much if the CDN redirects the client based on its location (e.g., IP address) to a set of congested servers. However, if the client were able to specify the servers to stream from, GO could avoid the overloaded servers and dramatically increase the quality. We believe that we will soon have the ability to perform such fine grain selection, as CDNs are incentivized to expose such information to clients. Indeed, a CDN operator will prefer that a quality-impacted client move to other servers in the same CDN rather than migrate to a different CDN. Furthermore, an ISP CDN that also runs its own software on set-top boxes or other user devices would be in the perfect position to run a GO-like algorithm that makes decisions at server granularity.

CONCLUSION

As the Internet infrastructure becomes more complex, the potential number of congestion and failure points will only increase. In this paper, we have shown that despite this increasing complexity, a predictive model for QoE, leveraging an Internet scale control plane architecture (GO), gives online video providers the ability to deliver an optimal viewing experience for their content. With new SDN technologies being developed and deployed to expand the capabilities of devices within the video ecosystem, we believe that a GO-like solution will be able to make even more granular optimizations and better control the delivery environment. Content publishers and service providers are in an excellent position to utilize a V-SDN architecture, such as GO, to ensure the integrity of their business as the industry moves to an Internet TV model.

REFERENCE

[1] Cisco Visual Networking Index: Forecast and Methodology, 2012–2017

[2] http://gigaom.com/2014/01/06/netflix-4k-ultra-hd-3d/

[3] http://mashable.com/2014/01/03/youtube-4k-ces/

[4] http://vimeo.com/tag:4k

[5] A Case for a Coordinated Internet Video Control Plane. Xi Liu, Florin Dobrian, Henry Milner, Junchen Jiang, Vyas Sekar, Ion Stoica, Hui Zhang, Proceedings of ACM Sigcomm, 2012

[6] Content Delivery Network (CDN) Federations, How SPs Can Win the Battle for Content-Hungry Consumers, Cisco Whitepaper

[7] The Road to SDN, Nick Feamster, Jennifer Rexford, Ellen Zegura, ACM Queue, December 30, 2013

[8] Dynamic Video Transcoding in Mobile Environments, Bo Shen, Wai-Tian Tan, and Frederic Huve, IEEE Multimedia, 2008

[9] Increasing TCP's Initial Window, IETF RFC 6928

[10] Big Data Requires a Big, New Architecture, Dan Woods, Forbes

[11] Understanding the Impact of Video Quality on User Engagement, Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, Jibin Zhan, Vyas Sekar, Ion Stoica, Hui Zhang, Proceedings of ACM Sigcomm, 2011

[12] A unified bias-variance decomposition, Pedro Domingos, Proceedings of ICML 2000

[13] Value Function Approximation using Multiple Aggregation for Multiattribute Resource Management, Abraham George, Warren B. Powell, Sanjeev R. Kulkarni, JMLR 2008

[14] http://hadoop.apache.org/

[15] http://spark.apache.org/

[16] https://amplab.cs.berkeley.edu/software/

[17] https://lucene.apache.org/solr/

[18] http://shark.cs.berkeley.edu/

[19] http://www.r-project.org/

[20] http://www.opendaylight.org/