SCTE·ISBE CABLE-TEC EXPO 2017
DENVER, CO
OCTOBER 17-20

2017 Fall Technical Forum
SCTE·ISBE • NCTA • CABLELABS

SMART RECORDINGS

DYNAMIC SEARCH AND RECORD OF LIVE TV

A Technical Paper prepared for SCTE
By

Chris Lintz
Sr. Principal Architect
Comcast VIPER
1515 Wynkoop St.
Denver, CO 80218

Table of Contents

List of Figures

Introduction

X1 – Comcast's advanced video platform and the X1's Cloud DVR (cDVR) are used by millions of customers across the country. They expect uninterrupted service, seamless access to thousands of programs, and world-class product features. Comcast VIPER designs and develops IP video solutions supporting X1, cDVR, and mobile technologies.

Users searching for linear programming on their set-top box or mobile device typically enter static metadata such as the program title or series they are interested in. This synchronous search will immediately return results highlighting channels and scheduling for any programming results. Channel surfers may simply find programming by continuously paging the on-screen guide or by rotating through their favorite set of channels.

A user's interest in content cannot always be defined by searchable static programming metadata. The dialogue within a program can often be a much richer description of the content, but today's services offer no way to search the dialogue in near real-time and inform a user that their interest is appearing on a live program.

Traditional DVR search and record functionality is also based on static metadata such as show title, program series and genre. Users can press the record button and instantly start recording a program or alternatively, schedule future recordings. While these are key functionalities for DVR use, automated recordings triggered from continuous search across linear dialogue offers a broader content discovery.

Searching and recording based on linear dialogue opens up more value and provides a leap forward in linear and DVR services. Imagine coming home and finding multiple video clips on your DVR containing in-depth interviews of your favorite sports stars appearing on various programs. Channel surfing becomes more interactive when you receive a guide notification informing you that a topic of interest is appearing on a channel you rarely watch.

Shorter, more relevant videos of interest offer a convenience for both at-home and mobile users always on the go. Investors can stay on top of every important conversation about their stocks. Fantasy football fans can capture news and interviews with their roster of players. Researchers and entertainment fans will no longer miss content that brings value to their lives. The possibilities are endless when given the ability to continuously search live TV dialogue 24/7 across all available channels.

This future smart search and record product is running in Comcast's lab and planning is underway for a customer trial launch, which will be in full compliance with all applicable laws and contractual obligations. The innovation was invented and pushed forward by engineers with the passion to bring more value and content discovery to X1 and mobile device customers. This document focuses on the architecture and technologies supporting the product functionality.

Logical Architecture Overview

Searching linear program dialogue and recording content of interest is achieved through multiple Comcast systems (Figure 1). Video Analysis is a system that provides real-time video, audio, and Closed Captioning analysis. Cloud DVR (cDVR) is a system running in regional Comcast data centers, which supports scheduling, recording, and playback of video for various devices. Linear Search consumes output from a Video Analysis pipeline and user queries, performing search as dialogue streams in a program.

Successful search matches result in notifications sent to the Notification System. Entitlements are checked before recording content – purposely not before linear search. This provides a broad based search across all available linear streams and allows customers to be notified that content of interest is appearing on a stream – even if they are not yet entitled to the stream. Customers entitled to the linear stream result in scheduling requests to the cDVR system. The Smart Search Service provides application interfaces for publish and retrieval of search results, video clip metadata, and notifications.

Future customer notification methods will be supported, including SMS, email, and push notifications to mobile devices. This gives customers the ability to tune instantly to the program and at a position in front of the content of interest.

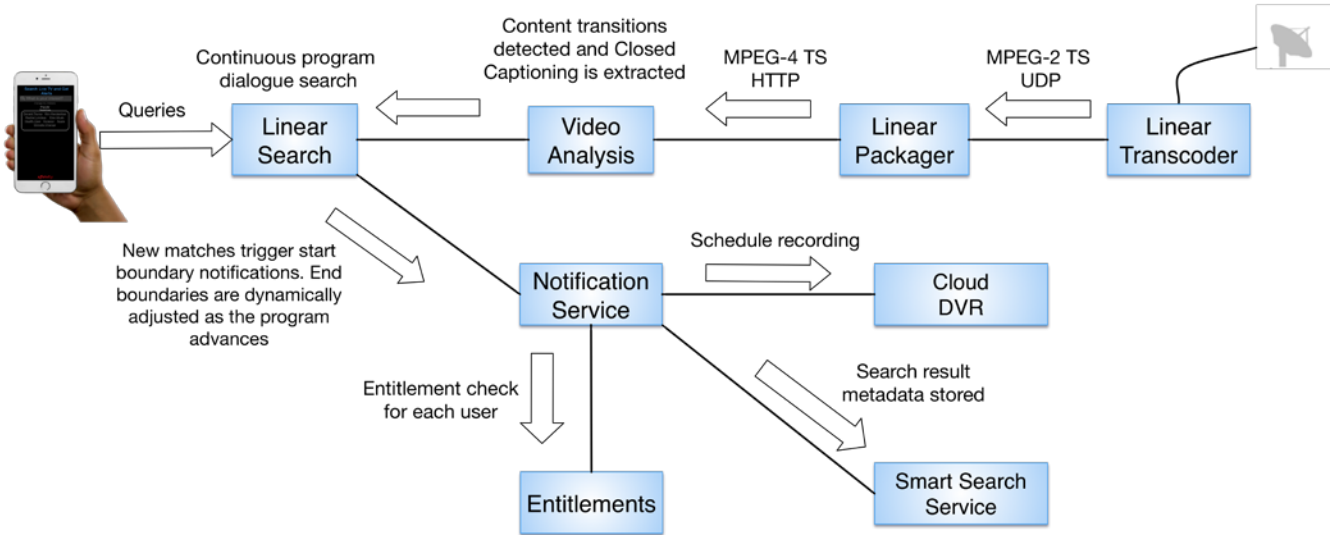

Figure 1 – Linear Search and Record Workflow Detailing System Components

Closed Captioning and Video Analysis

Video segments carry all the necessary metadata needed for searching linear dialogue. Linear textual dialogue and identified content transitions are what downstream linear search components utilize when executing user queries and processing results. Dialogue in the form of Closed Captioning (CEA-608/708) is carried in the picture user data on the transport stream. I-frames contain image data, which is used when analyzing the video to determine any content transitions.

Transitions within the content of a program provide opportunities to define boundaries around content when a match occurs from linear dialogue search. A shot change (Figure 2) is a slightly different camera perspective within the same scene of content. Scene changes occur when an entirely different camera perspective occurs within the same program.

Figure 2 - Visual Example of a Shot Change

SCTE-35 signaling in the manifest can be relied on for local ad spots, identifying a scene change. When this signal is not available, shot and scene detection algorithms help identify content changes in the video. This involves decoding image packets for color and edge information and applying mathematical formulas to detect movement from one frame to the next.

While these algorithms can be computationally expensive and require storing previous computations from frame analysis, they are achievable in real-time and provide a high detection accuracy. Detecting shot changes is less computationally expensive and requires less storage of previous frame analysis. These algorithms are the first step in a video analysis pipeline (Figure 3).

A process called a Stream Reader monitors linear manifests from a Linear Packager. Stream Readers scale horizontally consuming in aggregate over 10,000 local and national streams. The Linear Packager is an internal video packager supporting at rest encryption and an intermediate format derived from DASH. Each time the monitored manifest is updated, video segments are pulled and video frames are analyzed for shot and scene changes.

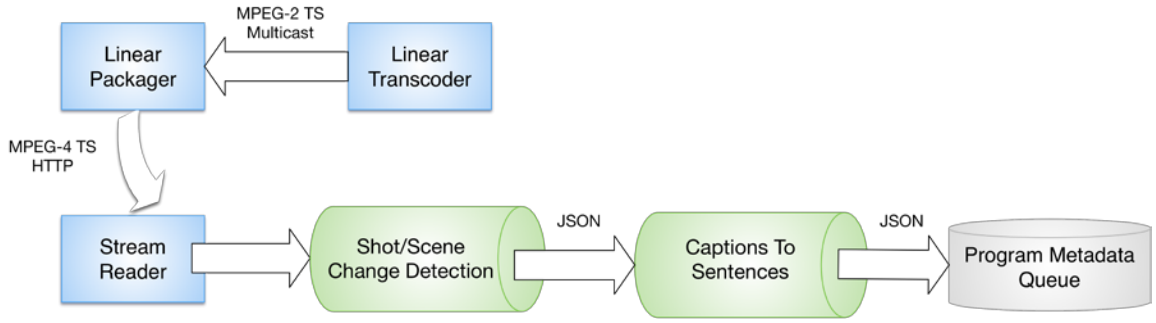

Figure 3 - Video Analysis Pipeline

Each segment carries an encoder boundary point (EBP) containing a sequential timestamp relative to the transcoder. These EBP timestamps are extracted along with the textual Closed Captioning data. Sentence formation is constructed if there is a partial phrase. A series of phrases, which ultimately form a sentence, may be spread over multiple segments. Multiple segments may result in more than one shot or scene change. All shot and scene change times are reflected as an array of EBP times in the program metadata document (Figure 4).

```
{
    "program": "Mad Money",
    "streamName": "CNBC HD",
    "streamId": "8951620516683951163",
    "ebpTime": 1496438204,
    "text": " >> Tesla passes Ford in market value now up over $350 per share
            as it continues an unreal run ahead of the Model 3”,
    "shotChange": [1496438204, 1496438208],
    "sceneChange": [1496438212]
}
```

Figure 4 - Program Metadata Example

Once a sentence is formed it is also included in the resulting program metadata document, which is then pushed onto the queue making it immediately available for search.

Linear Search and Record

Typical search systems store static documents, inverted indexes are built, and queries are executed against the indices. When a large amount of queries are searched over the same document, optimizations are made when inverting this concept. As streaming documents arrive they are tokenized and searched against query indices. Candidate query matches are returned, requiring a document search in order to resolve search hits and relevancy. This inverted search concept is commonly known as stream search, or reverse search, and can greatly reduce the number of queries executed.

Searching linear dialogue at Comcast scale can equal hundreds of thousands of queries running across hundreds of linear streams in large regions. Linear packagers output two-second segments resulting in dialogue changes received at that cadence per linear stream. In a typical Comcast region with hundreds of linear streams, the Video Analysis Pipeline produces streaming text documents at a rate of 200-300 per

second. The volume of queries combined with the influx of text documents makes streaming search a desirable technique for near real-time dialogue search.

Querying dialogue between the program start and live point of the program also provides value to users. A query added during a live program will search the past dialogue allowing for back-in-time notifications and recordings if the video segments are still available in the Linear Packager.

1. User Queries

Queries added to the system exist as live searches until removed by the user. Query Parsers (Figure 5) receive the submitted user queries and are responsible for filtering and expanding queries. Supported user query types are simple terms and phrases with limited conjunction and disjunction clauses.

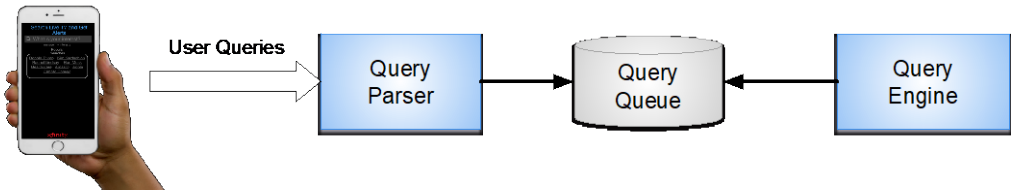

Figure 5 –Query Parser and User Queries

Within a program scene, a close distance between two phrases in Closed Captioning text represents conversationally related phrases. Matches resulting from proximity queries on textual dialogue identify conversational relevance. Proximity queries are used internally when expanding some multi-phrase queries.

Editorialized synonyms help expand popular queries into broader meanings. As one example, the two queries “Donald Trump” and “President Trump” would result in the same query “President Trump OR Donald Trump”. Queries are then normalized into an internal query representation and submitted to the queue.

2. Query Partitions

The Query Queue is partitioned so that each partition holds a subset of user queries. Using simple hashing on expanded queries provides a common routing technique resulting in the application writing identical queries to the same partition: *Partition ID = Hash(Query) % Total Partitions*

This approach ensures that the same Query Engine handles identical queries. This allows us to create one-to-many relationships of queries to users so that only a single query is executed for multiple users. The reduction in the amount of queries can be drastic for popular queries. It also provides optimizations resulting from being able to batch notification messages and cDVR recordings.

Queue partitioning (Figure 6) is a pattern than can be implemented with any persistent store. We utilize Kafka as our queue primarily for the built-in partitioning, consumer groups, and log compaction features. Log compaction maintains at least the latest version of a key. Each query partition is essentially a

persistent store for user queries. This allows the Query Engines to glue onto their assigned partitions and handle re-start and failure scenarios by again consuming the set of queries.

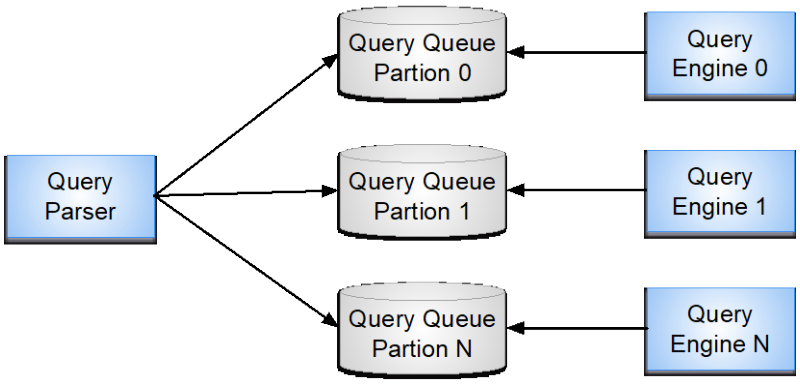

Figure 6 - Query Partitioning

3. Query Engines

Two popular streaming search libraries are Lucene based Elasticsearch-Percolators and Luwak. Pre-filtering techniques are used in Luwak, which eliminates queries that are not a possible match and presents candidate queries that may be matches. We have chosen to embed Luwak in Query Engines for stream search functionality because of the performance gains proven in lab testing.

Query Engines scale horizontally and perform in-memory search for both stream search and full program dialogue search. Figure 7 below depicts high-level workflow between components.

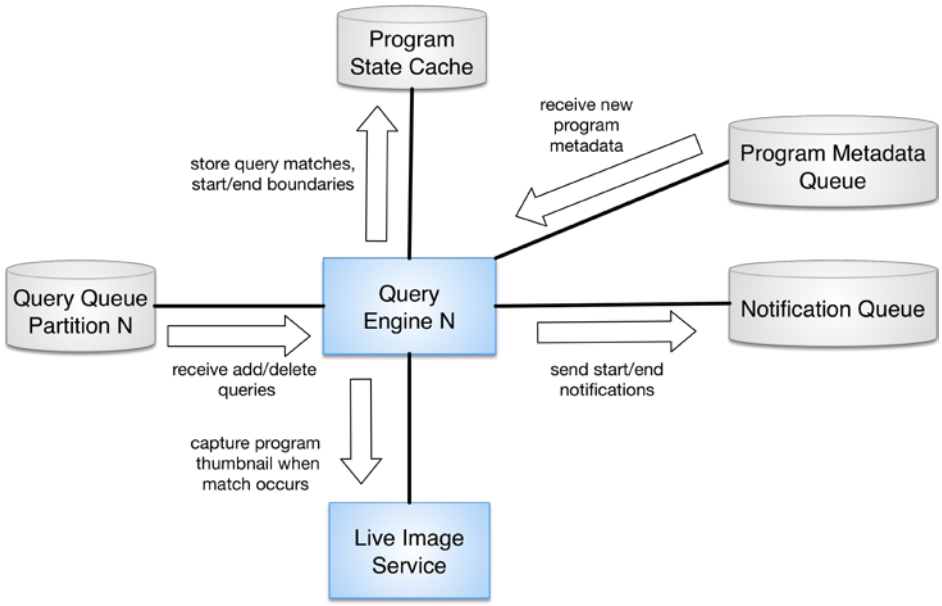

Figure 7 - Query Engine High Level Workflow

3.1. Query Filters

Queries can contain preferences such as stream or program filters for fine-grained search over desired programming. For example, a user may choose a broad search across all available linear streams rather than filtering on a single program.

Users with the same query are combined and any stream or program filtering is applied. In the example below (Figure 8), Chris and Tony are interested in "Comcast OR Netflix" but only if it is discussed on CNBC's "Squawk on the Street" or "The Closing Bell". Jen is interested in "Climate Change" if it appears on any Weather Channel program.

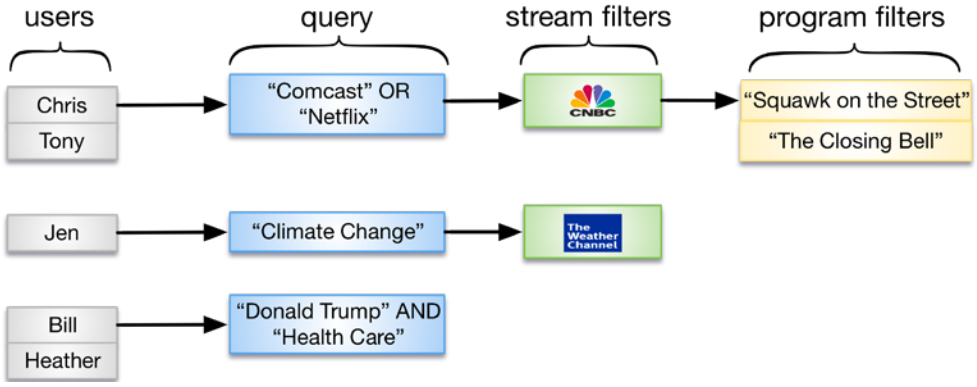

Figure 8 - Stream and Program Filters

Bill and Heather want the broadest search across any linear stream for discussions of "Donald Trump" AND Health Care".

3.2. Stream Search and Program Transcript Search

Query Engines are assigned with a query partition ID and a list of linear stream IDs. Documents for each live program are retrieved from the queue for each stream ID. Luwak monitors queries and matches new program metadata documents against them. The documents are also used to form a program transcript document for each stream.

These program transcript documents are not part of stream search, but they are searched directly with Lucene for two use cases. The first is when a query is added during a program. This allows matches to be found behind the live point of programs. Searching program transcripts are also needed for some complex queries. For example, a proximity query use distance between words or phrases, which may require a search into past dialogue.

Searching the transcript document not only provides opportunities to trigger recordings with a start boundary back-in-time, but notifications from matches can result in other non-recording actions. For example, a customer can be presented an option to tune to a point in time behind the live point where their interest appears. Tuning back-in-time is possible through Instant VOD (iVOD), a Comcast service supporting live program rewind. A user can also be presented an option to set a scheduled recording for the program's next airdate.

Just a couple minutes into a typical chatty hour-long news analysis program such as MSNBC's Hardball with Chris Mathews, an average transcript document contains less than a couple hundred words. By program end the transcript document can be over 12K words not including commercial dialogue. This can produce a roughly 20KB document size - resulting in just 10MB of RAM for 500 one-hour programs. These documents are maintained locally in-memory for the duration of the Query Engines runtime.

It should be no surprise that both stream search and transcript document search are compute bound. The frequency of program transcript document updates, which requires re-indexing, is an added burden. Both stream search and traditional search techniques have different performance considerations. We address these considerations by adjusting two key parameters in the system. (1) Total query partitions in the system and (2) and total list of linear streams consumed by each Query Engine. This also allows for a great deal of flexibility for tuning deployments for different regions with different numbers of local and national streams, running on different hardware.

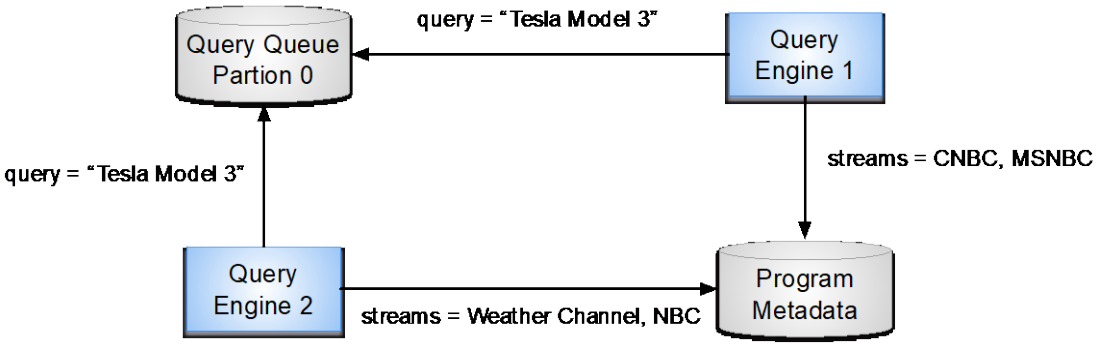

Figure 9 - Query Engines Consuming the Same Queue Partition but Different Linear Streams

3.3. Utilizing Content Transitions for Video Clip Boundaries

A content transition timeline for each program is maintained by extracting shot and scene change times from program metadata documents. Closed Captioning drifts in varying durations on all linear streams all day long. Frame accuracy search matches are not of great value due to this drift. Even if the EBP time near the matched sentence were to be used as the start boundary for the video clip, it likely would be in mid-dialogue or in the middle of scene. While this will capture content relevant to the query, it likely will not result in a great user experience.

A better quality video clip can be achieved by using content transitions before and after the time of the query match. For example, the moment of a commercial end can be used as the start boundary of the video clip – in front of the match. A scene change that occurs at some time after the match can represent the end boundary of the video clip.

The Query Engine utilizes the content transition timeline in attempt to capture a better start boundary and end boundary for a desired video clip. In Figure 10, if a search match occurs at the live point, the EBP time of the nearest scene change (walking backwards in time) is used as the start boundary for the video clip.

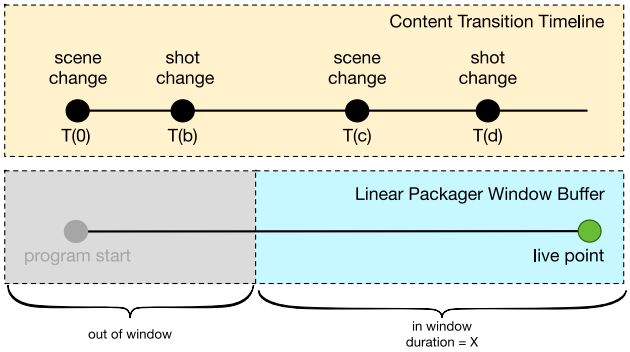

Figure 10 - Content Transition Timeline and Linear Packager Window Buffer

Scene changes are preferred over shot changes and if neither transition change is available in the past, the related EBP time of the match is used. In the above example the scene change EBP time at T(c) is selected as the start time of the recording. T(c) is within the Linear Packager window buffer allowing for a successful back-in-time start of a recording.

A more detailed example describing how basic start and end boundaries work around a content of interest is depicted in Figure 11 below.

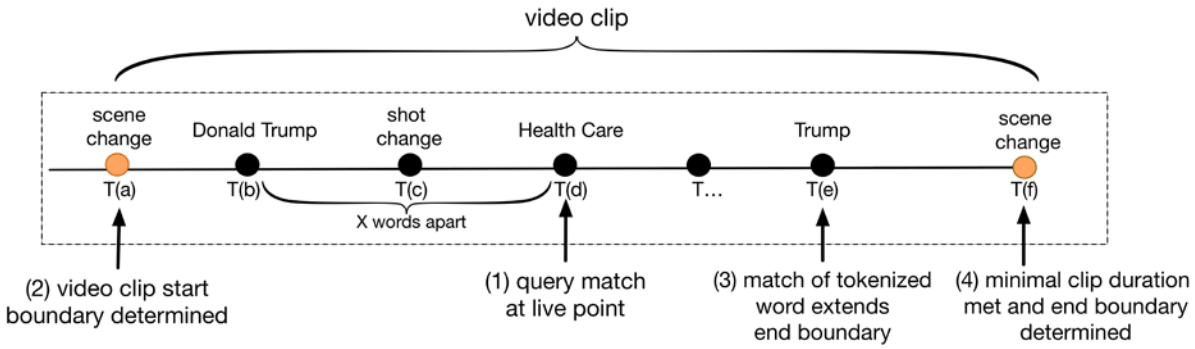

Figure 11 - Video Clip Timeline Utilizing Content Transitions

"Donald Trump" AND "Heath Care"{X} is a conversationally relevant query that will only match if the two phrases are at most X words apart. At the live point T(d), the tokenized phrase "Health Care" appears in the current sentence triggering the proximity search against the program transcript document. A match is found and the EBP time before Donald Trump appears is used to find a prior scene change EBP time in the content transition timeline cache, which results in T(a). The desire to be in front of the matched phrases eliminates shot change at T(c) because it falls within the proximity query.

If there is a preference to record the remaining program, the end time of the program is determined and set as the end boundary for the video clip. Otherwise the end boundary is set as a fixed duration and adjusted dynamically as the program progresses. At this point a notification can be fired off, resulting in the start of a recording.

Desired video clip durations are attempted and end boundaries are extended if tokenized words from the query are found in new sentence dialogue. Time progresses and at T(e) the tokenized word "Trump" is in the current dialogue and the end boundary is extended by a fixed duration. At T(f) the desired duration of

the video clip has been met and a scene change has occurred. This triggers the end boundary to be set to T(f).

Searches against the program transcript document also utilize the content transition timeline. EBP times embedded in the transcript document provide timestamps needed for content transition time lookups. For example, Figure 12 below represents a Lucene highlighter result from the query "Tesla" against a program transcript document.

```
[1498423117]Ford's market value has just been passed by <b>Tesla</b>
as the stock breaks through $350 a share.[1498423121] The first
deliveries are expected in late July for the Model 3.
```

Figure 12 - Example Program Transcript Snippet w/EBP Times (Lucene Highlighter Result)

Similar to stream search matches occurring at the live point, The EBP time in-front of "Ford's" will be used to find the nearest scene change time in the content transition timeline.

4. cDVR Supprt for Smart Recordings

A Scheduler, external to cDVR regions, manages and schedules recordings via a Recording Manager at located at each local region. Scheduling logic was not changed, but a more condensed down scheduling logic was built into the Notification system. Modifications were introduced to cDVR Manifest Agents in order to support back-in-time recordings.

Linear Packagers contain a window of content behind the live point. Manifest Agents are continuously monitoring manifests for updates and maintain a cache of manifests within a rolling window (Figure 13).

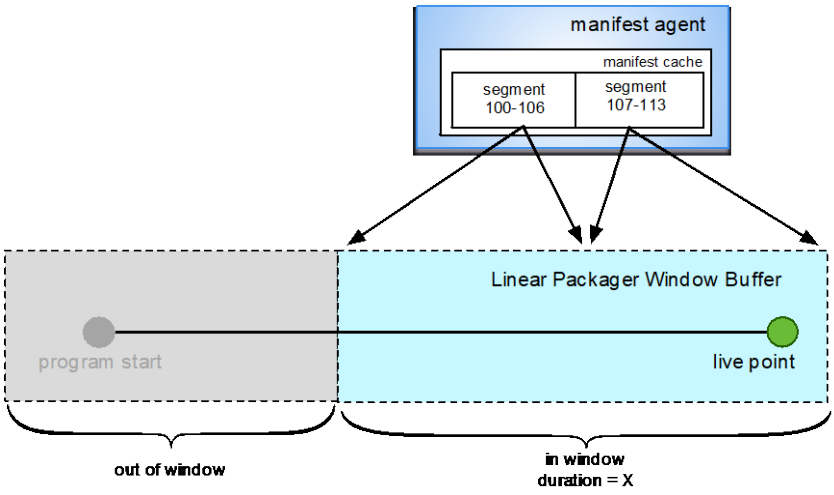

Figure 13 - Cached Manifests for Back-in-time Recordings

Maintaining a cache of manifests allows for recording video segments behind the live point that fall within the Linear Packager's buffer. Requests to record segments outside of this buffer will default to the oldest segment available in the window.

A match from a single query representing multiple users generates a batched notification. This results in a batched recording request. Copies are unique per subscriber, but batched recordings result in optimizations to the underlying network and storage system by generating a fan-out request to persist the unique video segments per user. While batched recordings are not new to cDVR, it is important to note that combining of users with the same query not only optimizes search, but also cDVR resources.

Future Considerations

The total tuners available limit concurrent recordings for a set-top box. Concurrent recording limitations also apply to cloud-based recordings. Popular topics and current news event queries can quickly reach the concurrent recording limit. As an obvious example, a simple query like “Donald Trump” is likely to appear on many channels simultaneously and can quickly consume the total active recording limit – particularly if stream and program filters are not set for each query.

In order to optimally capture the most content for a user while maximizing the concurrent recordings, many options are being considered:

1. The avoidance of recording video clips for programs already scheduled to record. For example, if a program is set as a recurring recording it is clear a customer is interested in the full program. Rather than capturing video clips of the already scheduled program, metadata can be added to the recording or notifications can inform the customer that matches appear within the recording.
2. Multi-weighted algorithms that include customer priority channel, program, and query rankings.
3. Suggested program and/or stream filters for popular queries that are likely to consume multiple concurrent recordings.

Conclusion

Identifying content transitions combined with Closed Captioning are key to dynamic searching and recording video of interest. Small modifications to cDVR that allow for back-in-time recordings provide opportunities for smooth recording starts, which also capture more context in front of a user’s matched interest. These back-in-time recording starts combined with dynamically extending end times and ending a video clip on a content transition produce a better quality video clip and user experience.

The value of content discovery on linear streams in near real-time is made possible by a combination of streaming search techniques on current sentence dialogue and search over active program transcript documents. Notifications providing opportunities for users to tune to a program back-in-time or to set scheduled recordings for the next airdate are also value adds from the system.

cDVR revolutionized how customers record and view video content. Expanding search and record capabilities using linear dialogue and video analysis is the next leap in offering additional features and value, allowing customers to never miss their interests appearing in any stream with a new personalized experience.

Abbreviations

EBP	encoder boundary point
cDVR	cloud digital video recorder
VIPER	Video Internet Protocol Engineering and Research
DASH	Dynamic Adaptive Streaming over HTTP
MPEG	Motion Pictures Expert Group
iVOD	instant VOD

Bibliography & References

Zheyun Feng, Jan Neumann, "Real Time Commercial Detection in Videos"

Apache Kafka, https://kafka.apache.org/

Luwak, https://github.com/flaxsearch/luwak

Elastic Percolator, https://www.elastic.co/guide/en/elasticsearch/reference/current/search-percolate.html

SCTE-35, Digital Program Insertion Cueing Message for Cable, ANSI/SCTE-35 2013

