Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 11, No. 1, November : 2022

Designing a Cloud-Based Data Centric Publish/Subscribe System

BALACHANDRA MEENUGA
Assistant Professor
balachandra.1202@gmail.com

VELLABOYANA ARUN KUMAR REDDY
Assistant professor
varunalts@gmail.com

T. KRISHNA MURTHY
Assistant Professor
taticherla.krishnamurthy@gmail.com

Abstract

Live content and emergency applications are rising, making it difficult to provide big amounts of live material to interested viewers. Data distribution uses the publish/subscribe (pub/sub) approach because it can effortlessly scale to huge size. Cloud computing supports complicated processing and dependable connectivity. We propose and investigate the ESCC Technique, elastic and scalable content-based pub/sub systems in cloud computing. We presented and studied a cloud-based distributed two-layer pub/sub framework to accomplish ESCC.

Keywords: *ESCC technique, Cloud Computing, Two layer Publish/Subscribe System*

1. Introduction

Cloud computing provides on-demand network access to shared computer resources. These resources may be immediately supplied and unconstrained with little management or service provider engagement. Interdependent computing delivers pooled processing resources to computers and other devices on demand. It offers shared configuration computer resources (example: Networks, Servers, Storage).

Cloud computing's powerful servers and high-speed networks enable complicated processing and high-speed communication. Move, Blue Dove, and SEMAS are cloud-based pub/sub services. Cloud computing enables scalable and trustworthy third-party storage. Its pay-per-use strategy costs a fraction of in-house computer resource deployment. Any application that stores and shares data should outsource to the cloud. Cloud clients just pay for the storage, processing, and network traffic they use. They avoid in-house capital and maintenance expenditures.

2. Literature Survey

Content-based system where the focus is on properly representing the content in our social book marking data sets. Based on these representations our aim is to construct an interest profile of an active user, and then use this profile to rank-order the unseen items by similarity to the profile, thereby approximating possible interest in those items.

In software architecture, publish-subscribe is a messaging pattern where senders of messages, called publishers, do not program the messages to be sent directly to specific receivers, called subscribers, but instead characterize published messages into classes without knowledge of which subscribers, if any , there may be. Similarly, subscribers express interest in one or more classes and only receive messages that are of interest, without knowledge of which publishers, if any, there are Publish–subscribe are a sibling of the message queue paradigm, and are typically one part of a larger message oriented middleware system. Most messaging systems support both the pub/sub and message queue models in their API, e.g. Java Message Service (JMS).

This pattern provides greater network scalability and a more dynamic network topology, with a resulting decreased flexibility to modify the publisher and its structure of the data published. In the publish–subscribe model, subscribers typically receive only a subset of the total messages published. The process of selecting messages for reception and processing is called filtering. There are two common forms of filtering: Topic-based and Content-based.

Topic Based System: In a Topic-based system, messages are published to "topics" or named logical channels. Subscribers in a Topic-based system will receive all messages published to the topics to which they subscribe, and all subscribers to a topic will receive the same messages. The publisher is responsible for defining the classes of messages to which subscribers can subscribe.

Content Based System: In a Content-based system, messages are only delivered to a subscriber if the attributes or content of those messages match constraints defined by the subscriber. The subscriber is responsible for classifying the messages.

Some systems support a hybrid of the two Publishers post messages to a topic while subscribers register Content-based subscriptions to one or more topics.

2.1 A High Level Architecture of Content-based Systems

Content-based Information Filtering (IF) systems need proper techniques for representing the items and producing the user profile, and some strategies for comparing the user profile with the item representation. The high level architecture of a content-based system is depicted in Figure 1. The process is performed in three steps, each of which is handled by a separate component

• **CONTENT ANALYZER** – When information has no structure (e.g. text), some kind of pre-processing step is needed to extract structured relevant information. The main responsibility of the component is to represent the content of items (e.g. documents, Web pages, news, product descriptions, etc.) coming from in-formation sources in a form suitable for the next processing steps. Data items are analyzed by feature extraction techniques in order to shift item representation from the original information space to the target one (e.g. Web pages represented as keyword vectors). This representation is the input to the PROFILE LEARNER and FILTERING COMPONENT;

• **PROFILE LEARNER** – This module collects data representative of the user preferences and tries to generalize this data, in order to construct the user profile. Usually, the generalization strategy is realized through machine learning techniques [4], which are able to infer a model of user interests starting from items liked or disliked in the past. For instance, the PROFILE LEARNER of a Web page recommender can implement a relevance feedback method [5] in which the learning technique combines vectors of positive and negative examples into a prototype vector representing the user profile. Training examples are Web pages on which a positive or negative feedback has been provided by the user

• **FILTERING COMPONENT** – This module exploits the user profile to suggest relevant items by matching the profile representation against that of items to be recommended. The result is a binary or continuous relevance judgment (computed using some similarity metrics [6]), the latter case resulting in a ranked list of potentially interesting items. In the above mentioned example, the matching is realized by computing the cosine similarity between the prototype vector and the item vectors.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 51, Issue 11, No. 1, November : 2022

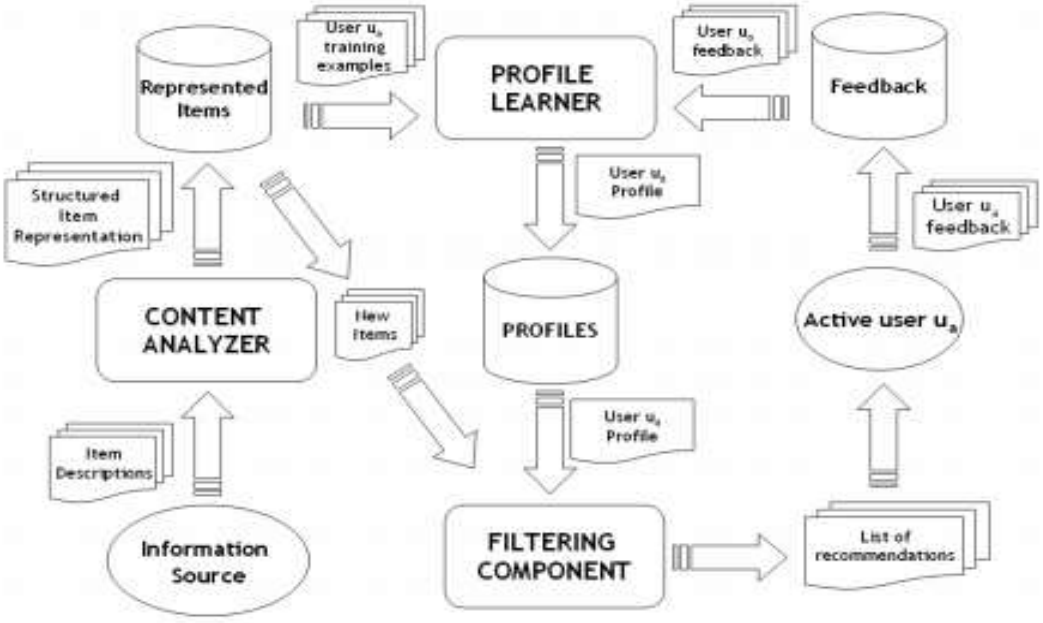

Figure 1. High level architecture of a Content-based System

2.4. Advantages of Content-based Filtering

The adoption of the content-based recommendation paradigm has several advantages when compared to the collaborative one:

- **USER INDEPENDENCE** - Content-based recommenders exploit solely ratings provided by the active user to build her own profile. Instead, collaborative filtering methods need ratings from other users in order to find the "nearest neighbors" of the active user, i.e., users that have similar tastes since they rated the same items similarly. Then, only the items that are most liked by the neighbors of the active user will be recommended
- **TRANSPARENCY** - Explanations on how the recommender system works can be provided by explicitly listing content features or descriptions that caused an item to occur in the list of recommendations. Those features are indicators to consult in order to decide whether to trust a recommendation. Conversely, collaborative systems are black boxes since the only explanation for an item recommendation is that unknown users with similar tastes liked that item

- **NEW ITEM** - Content-based recommenders are capable of recommending items not yet rated by any user. As a consequence, they do not suffer from the first-rater problem, which affects collaborative recommenders which rely solely on users' preferences to make recommendations. Therefore, until the new item is rated by a substantial number of users, the system would not be able to recommend it.
- **LOOSE COUPLING**- Publishers are loosely coupled to subscribers, and need not even know of their existence. With the topic being the focus, publishers and subscribers are allowed to remain ignorant of system topology. Each can continue to operate normally regardless of the other. In the traditional tightly coupled client-server paradigm, the client cannot post messages to the server while the server process is not running, nor can the server receive messages unless the client is running. Many pub/sub systems decouple not only the locations of the publishers and subscribers, but also decouple them temporally. A common strategy used by middleware analysts with such pub/sub systems is to take down a publisher to allow the subscriber to work through the backlog (a form of bandwidth throttling).
- **SCALABILITY** - Pub/sub provides the opportunity for better scalability than traditional client–server, through parallel operation, message caching, and tree based or network based routing, etc. However, in certain types of tightly coupled, high volume enterprise environments, as systems scale up to become data centers with thousands of servers sharing the pub/sub infrastructure, current vendor systems often lose this benefit; Scalability for pub/sub products under high load in these contexts is a research challenge. Outside of the enterprise environment, on the other hand, the pub/sub paradigm has proven its scalability to volumes far beyond those of a single data centre, providing Internet wide distributed messaging through web syndication protocols such as RSS and Atom (standard). These syndication protocols accept higher latency and lack of delivery guarantees in exchange for the ability for even a low-end web server to syndicate messages to (potentially) millions of separate subscriber nodes.

2.5. Drawbacks of Content-based Filtering

- **LIMITED CONTENT ANALYSIS** - Content-based techniques have a natural limit in the number and type of features that are associated, whether automatically or manually, with the objects they recommend. Domain knowledge is often needed, e.g., for movie recommendations the system needs to know the actors and directors, and sometimes, domain anthologies are also needed. No content-based recommendation system can provide suitable suggestions if the analyzed content does not contain enough information to discriminate items the user likes from

items the user does not like. Some representations capture only certain aspects of the content, but there are many others that would influence a user's experience. For instance, often there is not enough information in the word frequency to model the user interests in jokes or poems, while techniques for affective computing would be most appropriate. Again, for Web pages, feature extraction techniques from text completely ignore aesthetic qualities and additional multimedia information. To sum up, both automatic and manually assignment of features to items could not be sufficient to define distinguishing aspects of items that turn out to be necessary for the elicitation of user interests.

- **SPECIALIZATION** - Content-based recommenders have no inherent method for finding something unexpected. The system suggests items whose scores are high when matched against the user profile; hence the user is going to be recommended items similar to those already rated. This drawback is also called serendipity problem to highlight the tendency of the content-based systems to produce recommendations with a limited degree of novelty. To give an example, when a user has only rated movies directed by Stanley Kubrick, she will be recommended just that kind of movies. A "perfect" content-based technique would rarely find anything novel, limiting the range of applications for which it would be useful.
- **NEW USER** - Enough ratings have to be collected before a content-based recommender system can really understand user preferences and provide accurate recommendations. Therefore, when few ratings are available, as for a new user, the system will not be able to provide reliable recommendations.

3. EXISTING SYSTEM

Addressing the scalability issue a Reliable Matching Service for Content-Based Publish/Subscribe Systems is presented in which an SREM (scalable and reliable event matching service) scheme is introduced for content-based pub/sub systems in cloud computing environment. SREM connects the brokers through a distributed overlay SkipCloud, which ensures reliable connectivity among brokers through its multi-level clusters and brings a low routing latency through a prefix routing algorithm. Through a hybrid multi-dimensional space partitioning technique, SREM reaches scalable and balanced clustering of high dimensional skewed subscriptions, and each event is allowed to be matched on any of its candidate servers. To ensure the scalability, the system is analyzed by deploying on the real time cloud stack demonstrate that SREM is effective and practical, and also presents good workload balance, scalability and reliability under various parameter settings.

4. PROPOSED SYSTEM

The main objective is to design and implement elastic and scalable pub/sub system that adapt to the churn workloads, and generate high delivery latency in the face of high arrival rate of live content such that it adjust the scale of then server based on the churn workloads.

ESCC Technique: Elastic and Scalable Content based Cloud Pub/Sub System

To achieve scalable and elastic total order in content based pub/sub system, we first propose a distributed two-layer pub/sub framework based on the cloud computing environment.

At a high level, the framework consists of two layers: the matching layer and the delivery layer

- The matching layer is responsible for matching events against subscriptions and sending events with matched subscribers to the delivery layer.
- The delivery layer is responsible for orderingevents according to the total order semantics, and distributingthem to their interested subscribers

For matchingservice, the main factors limiting its scalability contain the scale of subscriptions, the distributions of subscriptionsand events, data dimensionality, and the arrival rateof events. For total ordering service, the main factors limiting its scalability contain the arrival rate of events and theprobability of ordering conflict. Thus, we can flexiblyimprove the capacities of either event matching service or total ordering service based on various workload characteristics
if both services are decoupled.

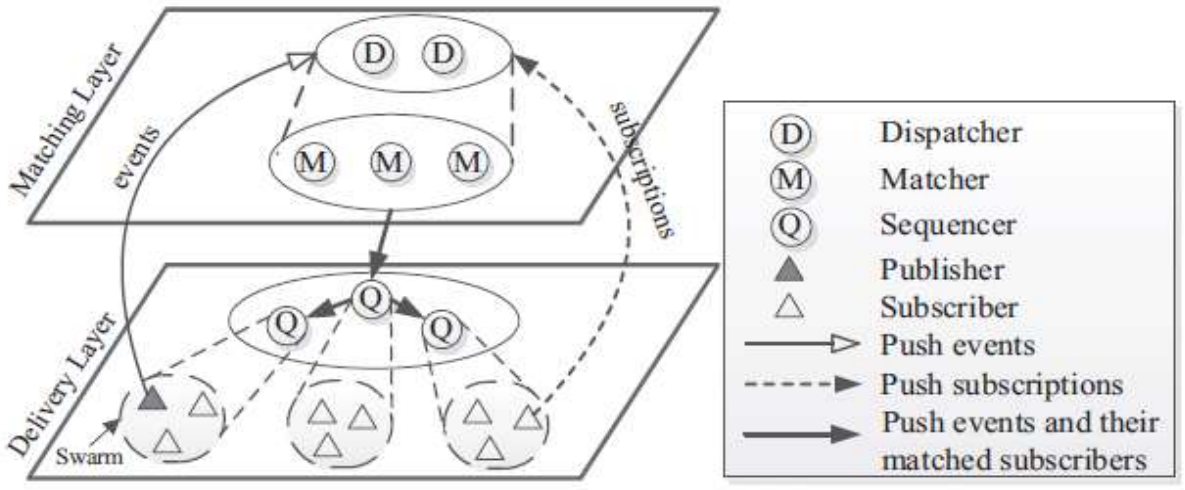

Figure.2Elastic and Scalable Content based Cloud Pub/Sub System

MATCHING LAYER

The matching layer is responsible formatching events against subscription and sending eventswith their matched subscribers to the delivery layer. Atthis layer, we employ SREM technique to implement high matching throughput. In SREM, through a hierarchicalmulti-attribute space partition technique (calledHPartition), the content space is divided into multiplehypercubes, each of which is managed by one server. Subscriptions and events falling into the same hypercubeare matched with each other, such that the matchinglatency can be greatly reduced. Besides, a performanceawaredetection technique (called PDetection) is proposedin SEMAS to adaptively adjust the scale of servers based onthe churn workloads

DELIVERY LAYER

The delivery layer is responsible for total orderingevents and delivering them to their interested subscribers. The main novelty of this layer lies in a preceding graphbuilding technique (called PGBuilder) and a performance wareprovisioning technique (called PProvision). The firstaims to reduce the total order latency in a scalable manner.In PGBuilder, subscribers are divided into multiple groups, each of which is managed by a single server. That is, all serversof PGBuilder are able to detect total order conflicts among events simultaneously, which greatly reduces thedelivery latency. Each server of PGBuilder constructs precedinggraphs among arrival events, which can quicklydetect non-conflicting events and deliver them in a parallelmanner. Besides, to ensure reliable delivery, PGBuilderprovides a series of dynamics maintenance mechanisms.

PGBuilder algorithm:

1. Initialize cluster C with Q
2. if C==0 then initialize C as new Cluster
3. Process the list L as CPL and if size (CPL)! = 0 the
4. addDPL.get (Q).

```
int I = 0;
while (i<tmpList.size ()) do
e”= tmpList.get (i);
for (int k = 0; k < e”.DPL.size (); k++) do
```

```
        tempE = e".DPl.get (k);
        if (! tmpList.contains (tempE)) then
          tmpList.add (tempE);
5.  i++;
```

Firstly, the arrival events in each sequencer are dispatchedinto multiple separated clusters, and all theseclusters are managed by a global queue (GQ). Then, e is added into the tail cluster. Clusters in the GQ are naturallyconflicted with each other, and each cluster can be treatedas a sliding window. That is, each sequencer only processesthe head cluster. After the events of the head cluster aredelivered to all their corresponding subscribers, the headcluster is removed from the global queue, and the sequencerprocesses next head cluster. Thus, each new arrivalevent only needs to detect conflicts against the events inthe tail cluster of GQ, but not all events of GQ. Throughdividing events into multiple clusters, it greatly reducesthe conflict detecting latency regardless of the event arrival rate.

Delivery Strategy:

In ESCC technique, there are mainly two roles: sequencers and subscribers. Since the joining or leaving of both roles may severely hurt the performance of total ordering, we will discuss how to keep continuous and efficient total ordering under dynamic networks.

SUBSCRIBER:

Recall that each subscriber sends its subscriptions to oneof the dispatchers in our framework as shown in Fig. 1. When a new subscriber joins the system, it is dispatchedto one sequencer whose hash value is nearest to its ownvalue according to the consistent hashing technique.To ensure reliable delivery, the sequencer starts to sendthe next event until it receives all acknowledgements fromsubscribers of the last event. Thus, sequencers need to obtain the latest view of its local subscribers. Otherwise,waiting acknowledgements from failed subscribers maylead to high delivery latency.

SEQUENCER:

When a number of new sequencers join in the system, the root sequencer assigns every new sequencer to be a child of the existing sequencers one by one. Correspondingly, each exiting sequencer should

detectwhether its every local subscriber needs to be migrated to one of the new sequencers based on the consistent hashing technique

CONCLUSION

Elastic and Reliable event matching service for Content-Based Pub/Sub Systems In Cloud Computing Environmentit is clear that although the use of cloud computing has rapidly increased, cloud computing security is still considered the major issue in the cloud computing environment. From ESCC Technique we plan to design and implement the elastic strategiesof adjusting the scale of servers based on the churn workloads. Secondly, it guarantees that the brokers disseminate large live content with various data sizes to thecorresponding subscribers in a real-time manner can be done. Where the authorized users only can access the data stored in multiple clouds based on the satisfaction of required attributes and policies and additionally re-encryption performed for more security.

REFERENCES

1.Birman, K.and Joseph, T."Exploiting virtual synchrony in distributed systems (http://portal. acm. org/citation. cfm?id=41457. 37515&coll=portal&dl=ACM&CFID=97240647&CFTOKEN=78805594) in Proceedings of the eleventh ACMSymposium on Operating systems principles (SOSP '87),1987. Pp.123 – 138.

2. J. Raymond. Mooney and R. Loriene.: Content- Based Book Recommendation Using Learning for Text Categorization. In proceedings of the fifth ACM conference on digital libraries, pages 195- 204, San Antonio, TX, June 2000.

3. P. Resnick, H. Varian: Recommender Systems. Communications of the ACM, pages 56-58 (1997)

4. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)

5. Rocchio, J.: Relevance Feedback Information Retrieval. In: G. Salton (ed.) The SMART retrieval system - experiments in automated document processing, pp. 313–323. Prentice-Hall, Englewood Cliffs, NJ (1971)

6. Herlocker, L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems 22(1), 5–53 (2004)