A General Scalable and Elastic Content Based Publish/Subscribe Service

K.VEERANGAN
PG Student, Department of MCA, GKM College of Engineering and Technology Tamil Nadu State, Chennai, INDIA

MRS,A.SELVADEVI
Assistant Professor, Department of MCA, GKM College of Engineering and Technology Tamil Nadu State, Chennai, INDIA

Abstract: In cloud computing, providing publish/subscribe as a scalable is a challenging Bluedove is used to overcome this challenge. The high performance is achieved by multiple candidate servers for each message. The message can be matched on any of its candidate servers with one hop forwarding. The performance aware forwarding in PUB/SUB systems ensures that the message is sent to the least loaded candidate server for processing, leading to low latency and high throughput. The evaluation shows that PUB/SUB systems has a linear capacity increase as the system scales up, adapts to sudden workload changes in tens of seconds, and achieves throughput multi-fold higher than techniques used in existing enterprise and peer-to-peer pub/sub systems.

Keywords: Cloud Computing, Publish/Subscribe System.

I. INTRODUCTION

Publish/Subscribe (pub/sub) is a commonly used asynchronous communication pattern among application components. Senders and receivers of messages are decoupled from each other and interact with an intermediary— a pub/sub system. A receiver registers its interest in certain kinds of messages with the pub/sub system in the form of a subscription. Messages are published by senders to the pub/sub system. The system matches messages (i.e., publications) to subscriptions and delivers messages to interested subscribers using a notification mechanism. There are several ways for subscriptions to specify messages of interest. In its simplest form messages are associated with topic strings and subscriptions are defined as patterns of the topic string. A more expressive form is attribute-based pub/sub where messages are further annotated with various attributes. Subscriptions are expressed as predicates on the message topic and attributes. An even more general form is content based pub/sub where subscriptions can be arbitrary Boolean functions on the entire content of messages (e.g., XML documents), not limited to attributes.

A. Attribute-based Pub/Sub Model

BlueDove uses a multi-dimensional attribute-based pub/sub model similar to that of. Consider k attributes $L_1, L_2, \ldots, L_k$, let $V$ be the (ordered) set of all possible values of attribute $L_i$, then $V = V_1 \times V_2 \times \cdots \times V_k$ is the entire attribute space. The attribute space is a $k$-dimensional space, and from now on we use the terms dimension and attribute interchangeably. A message is defined as a point in the attribute space, $m = (v_1; v_2; \ldots; v_k) \in V$. For instance, in a traffic monitoring application, four dimensions may be used to describe a message: longitude, latitude, speed, and timestamp. A subscription is modeled as the logical conjunction of $k$ range predicates, each along a different dimension, $(l_1 \leq v_1 < u_1) \land \cdots \land (l_k \leq v_k < u_k)$. Alternatively, a subscription can be viewed as a $k$-dimensional hyper-cuboid $S = S_1 \times S_2 \times \cdots \times S_k$, where $S_i = [l_i; u_i]$. By this definition we say a message $m$ matches a subscription $S$ if and only if $m \in S$. This form of multi-dimensional range query is common in many applications. For example, a driver interested in traffic congestion in a metro area may specify a rectangle covering his proximate area, which can be translated into a subscription, e.g., $(41 \leq \text{long} < 42) \land (70 \leq \text{lat} < 74) \land (0 \leq \text{s} < 25)$. The subscription indicates.

ISSN: 2348 – 8387 www.internationaljournalsssrg.org
vehicle speed is in the range of miles per hour, and vehicle location in a rectangular area, with longitude range and latitude range.

B. System Architecture

Being a cloud-based service, BlueDove operates under different environments than what existing pub/sub systems are designed for. In a traditional enterprise pub/sub system, scalability is achieved by using a number of brokers, each of which serves a set of locally connected clients (as shown in 2Amazon has recently offered a topic-based pub/sub service. But they do not publicize the design or implementation of their service. Dispatching Servers

III. BLUEDOVE COMPONENT DESIGN

The two-tier architecture addresses how servers should be organized in a cloud pub/sub service. There are still a number of more detailed questions that we need to answer. First, how should we assign subscriptions to matchers such that only one matcher is needed to find all matching subscriptions for any message, while still providing high performance? Obviously, a full replication approach where each subscription is stored on all matchers needs only one matcher for each message.

A matcher is responsible for one segment along each dimension; it stores subscriptions whose predicate ranges overlap with its segments. E.g., the sample subscription's range on latitude [70 74] overlaps with segment [60; 90), which is stored in the corresponding matcher F. Thus F receives a copy of the subscription to corresponding matcher(s). The copy includes not just the predicate S of dimension L, but all predicates of the subscription. In the example (shown in Figure 2), the subscription's range on latitude dimension [70 74] overlaps with segment [60; 90], which is responsible by matcher F. Thus F receives a copy of the subscription. Note that one predicate may overlap multiple segments and more than one matcher may receive the subscription along that dimension (e.g., see dimension speed in the example).

Formally, the matcher(s) receiving a subscription along dimension L takes the predicate range S, then finds which segment(s) of overlap with S and forwards a copy of the subscription to corresponding matcher(s). The copy includes not just the predicate S of dimension L, but all predicates of the subscription. In the example (shown in Figure 2), the subscription's range on latitude dimension [70 74] overlaps with segment [60; 90], which is responsible by matcher F. Thus F receives a copy of the subscription.

Given a subscription S = S^1 S^k, a matcher assigns S to matchers k times, each time along a different dimension. It takes the predicate range S^i and finds those whose responsible segments overlap with the predicate range S^i. Since all segments along each dimension cover the whole possible value space, a predicate range has to overlap with at least one segment. Thus the subscription is assigned to at least one matcher in each dimension. In total it is assigned k times to at least k matchers. With this assignment scheme, the entire set of subscriptions S = fSg is partitioned among the N matchers k times; each time along a different dimension.

A. Multi-dimensional Partitioning

In order to take advantage of the many matchers in the system, BlueDove divides the entire subscription space among the matchers, so that each matcher only handles a small subset and searches through much fewer subscriptions. In BlueDove, subscriptions are assigned to matchers using a multi-dimensional subscription space partitioning approach, called mPartition. Let N be the total number of matchers and k be the number of dimensions. For each dimension L, mPartition splits V^i, the set of all possible attribute values on L, into N continuous and non-overlapping segments IV^i_j; j = 1; : : : ; N_g. Each matcher M_j is responsible for k such segments V^i_j, one in each dimension. Figure 2 shows a traffic monitoring example, in which each of the three searchable dimensions (longitude, latitude, and speed) are split into 6 segments and assigned to 6 matchers A F

![Figure 1](https://example.com/mPartition.png)

**Fig. 1** An example of mPartition where three dimensions are each split into 6 segments. A matcher is responsible for one segment along each dimension; it stores subscriptions whose predicate ranges overlap with its segments. E.g., the sample subscription's range on latitude [70 74] overlaps with segment [60; 90], so it is stored in the corresponding matcher F. Similarly it is stored in C for longitude. Along speed dimension its range [0; 25] overlaps with two seg-ments responsible by matchers A and B. So both store a copy of the subscription.

B. Performance-aware message forwarding

As pointed out previously, given a message m, there are k candidate matchers for it, CM_i(m). Because the amount of subscriptions |Si(CM_i)| assigned to, and the workload on CM_i, may vary greatly depending on the skewness of the subscription distribution, there exists opportunity to choose a “cold spot” candidate matcher to improve the performance.

C. Content-Based Data Model

Pi specifies SREM uses a multi-dimensional content-based data model. Consider our data model consists of k dimensions A^1; A^2; : : : ; A^k. Ak. Let Ri be the ordered set of all possible values of Ai. So, Ω = R1 ∗ R2 ∗ : : : ∗ Rk is the entire content space. A subscription is a conjunction of predicates over one or more dimensions. Each predicate a continuous range for a dimension Ai, and it can be described by the tuple (Ai; vi; Oi), where vi ∈ Ri and Oi represents a
relational operator (\(\lt\), \(\leq\), \(\geq\), \(\gt\), etc.). The general form of a subscription is \(S = A_1 v_1 \land \ldots \land A_k v_k\). An event is a point within the content space \(\Omega\). It can be represented as \(k\) dimension-value pairs, i.e., \(e = (A_1 v_1, \ldots, A_k v_k)\). For each pair \((A_i v_i)\), we say it satisfies a predicate \((A_i (v_i) \circ O_i)\) if \(A_i = A_i\) and \(v_i = v_i\). By this definition we say an event \(e\) matches \(S\) if each predicate of \(S\) satisfies some pairs of \(e\).

Fig.2 system framework

To support large-scale users, we consider a cloud computing environment with a set of geographically distributed datacenters through the Internet. Each datacenter contains a large number of servers (brokers), which are managed by a datacenter management service such as Amazon EC2 or Open Stack.

We illustrate a simple overview of SREM in Figure 1. All brokers in SREM as the front-end are exposed to the Internet, and any subscriber and publisher can connect to them directly. To achieve reliable connectivity and low routing latency, these brokers are connected through a distributed overlay, called Skip Cloud. The entire content space is partitioned into disjoint subspaces, each of which is managed by a number of brokers. Subscriptions and events are dispatched to the subspaces that are overlapping.

D. Elasticity and Fault Tolerance

We evaluate BlueDove’s elasticity and fault tolerance in this section. Elasticity refers to the system’s ability to adapt to sudden changes in workload. It is essential to cloud services since the workload changes continuously. We start the experiments at a small system size of five matchers, an initial message rate of 500 message/second, and 40,000 subscriptions in the system. During the experiments, workload generators increase the message rate by 500 messages/second every five minutes. When a BlueDove dispatcher detects system saturation, it adds a new matcher to distribute the workload. When a new matcher is added, it finds the most loaded matcher (in number of subscriptions) in each dimension and takes over about half of its subscriptions.

The load information from /proc/loadavg combines CPU load and I/O load. Since matching operations only involve in-memory operations and almost no disk I/O, the load information is a reasonable reflection of CPU load.

Fig.3 Fault Tolerance

A performance-aware message forwarding technique always forwards the message to the least loaded candidate node for matching, and thus achieves low latency and high throughput. We identify the differences in the cloud environment compared to traditional enterprise or peer-to-peer pub/sub systems, and point out the implications on the pub/sub architecture.

We propose new techniques (i.e., mPartition and performance-aware forwarding) that turn data skewness into an asset for scalability and performance, and combine them with existing technique (i.e., one-hop lookup) to build a prototype for scalable, elastic and fault-tolerant pub/sub service.

B. Scalability:

Scalability is achieved by using a number of brokers, each of which serves a set of locally connected clients. Scalability of handling high message rates and large number of subscriptions. We then evaluate how many subscriptions each pub/sub system supports as the system size increases.

Modern pub/sub service must be “elastic” to quickly adapt to workload changes that may happen in a short amount of time. For instance, during rush hours, a huge volume of traffic messages and subscriptions are generated; at night, the volume of both reduces substantially.
C. Multiple Candidate matcher (Subscriber)

Number of Subscriber relies on the mapping of subscriptions to matchers along multiple dimensions to have the flexibility of multiple candidate matchers to choose from. The flexibility increases with more searchable.

D. Skewness of message distribution:
- If publisher send the message to Subscribers in ONLINE. It will access by the subscriber directly without message encryption/Decryption.
- If publisher send the message to Subscribers in OFFLINE. It will access by the subscriber only with message encryption/Decryption.

V. System Analysis

Existing enterprise pub/sub products are not adequate to meet the above requirements. In these products, servers form a cluster and each client establishes affinity with one server by connecting to it directly. Subscriptions are often replicated on all servers, such that any server can match messages and forward them to the interested subscribers. Enterprise pub/sub products are not designed with elasticity in mind because the enterprise typically over-provisions the computation resources to meet the needs of peak workload. When Publisher send the messages by Offline it will shows the all the subscribers.

VI. CONCLUSION & FUTURE WORK

We have presented the attribute-based pub/sub service that is intended to support the emerging sense and respond applications and the cloud computing model. It takes advantage of the data center environment to match publications with subscriptions in just one hop. Through multi-dimensional assignment of messages and performance-aware forwarding, it turns the challenge of skewness in data to an asset for high throughput and low latency. Our experiments show that can handle workload two orders of magnitude larger than a full replication approach that is typical in existing commercial pub/sub software, and that it can handle workload three times larger than the multi-hop overlay approach commonly used in existing peer-to-peer pub/sub systems. It also adapts to load changes and server failures in a matter of seconds.

Furthermore, different applications may use different sets of attributes. We will investigate how to alleviate these problems by partitioning the subscription space in a hierarchical manner. One possibility is to divide dispatchers and matchers into different subsets and let them handle different applications. Finally, we also want to compare its performance with a third-party research prototype, to gain better understanding of the trade-offs in load balancing and performance.

REFERENCES