# DS-Index: A Distributed Search Solution for Federated Cloud

Yongqing Zhu, Quanqing Xu ✉, Haixiang Shi, Juniarto Samsudin
Data Storage Institute, A*STAR (Agency for Science, Technology and Research), Singapore.
{ZHU_Yongqing, XU_Quanqing, SHI_Haixiang, Juniarto_SAMSUDIN}@dsi.a-star.edu.sg

***Abstract*—Federated cloud facilitates data sharing across multiple organizations. Existing distributed storage systems cannot support efficient data search with complicated query requirements. This paper proposes a DS-index solution to provide distributed search in federated cloud, targeting to support multi-attribute range search with secondary index. DS-index includes a three-layer architecture with a multi-attribute index overlay, a tree-based P2P network layer, and a federated cloud layer. In addition, a dynamic mapping algorithm has been presented for DS-index to map between different layers. We have also defined a Markov Chain-based cost model to facilitate node selection and mapping. The proposed DS-index solution is capable to support multi-attribute range queries in the federated cloud. The experiments have shown that, the DS-index solution with cost model can save the computation resource and reduce network bandwidth consumption by around 30% comparing to the one without cost model. It can also reduce the splitting/ merging times by around 20% comparing to the state-of-the-art solution.**

***Keywords—distributed search; federated cloud; secondary index; mapping algorithm; cost model; peer-to-peer network***

## I. INTRODUCTION

Cloud computing is the paradigm to establish large-scale infrastructure and provision computation, storage, and high-level services to end users on demand. Many applications [1] [2] have been deployed in cloud to leverage the availability and scalability provided by cloud. With the growth of cloud usage, the resource and scalability provided by a single cloud provider are approaching the limits [3]. Federated cloud has been proposed to joint multiple external and internal clouds together to achieve extended resources and improved scalability. Federated cloud facilitates data sharing across geographically dispersed organizations. Storing of large-scale data can be achieved by deploying distributed storage system in cloud, e.g. Amazon's Simple Storage Service (S3) [4], etc. Data can be retrieved from huge dataset via file ID, object ID, or primary key. However, query requirements are more complicated in real world. Users may want to search data based on combined attributes with specific ranges from the cloud. Existing distributed storage systems have no support for secondary index. They cannot provide efficient data search for multi-attribute range queries.

This paper has proposed a DS-index solution to provide distributed search in federated cloud, targeting to support multi-attribute range queries with secondary index. Here DS stands for Distributed Search. DS-index includes a three-layer architecture with a multi-attribute index overlay, a tree-based P2P network layer, and a federated cloud layer. In addition, a dynamic mapping algorithm has been presented for DS-index to map between different layers. We have also defined a Markov Chain-based cost model to facilitate node selection and mapping. Our work is based on the BAlanced Tree Overlay Network (BATON) [5] network and combines with multidimensional secondary index to support multi-attribute range search. The works that are most similar to ours is RT-CAN [6] and CG-index [7]. RT-CAN integrates CAN-based Peer-to-Peer (P2P) network and the R-tree based indexing scheme to support multi-dimensional queries. CG-index organizes computer nodes into a BATON network and builds $B^+$-tree indexes to support one-dimensional queries.

Section II presents the system overview of the proposed DS-index solution. Dynamic mapping algorithm and Markov chain-based cost model are elaborated in Section III. Section IV shows the experiments and evaluation results, and Section V concludes the whole paper.

## II. SYSTEM OVERVIEW OF DS-INDEX

We propose a DS-index solution in this paper to provide distributed search in federated cloud, targeting to support multi-attribute range queries with secondary index. In order to support multi-attribute range search across dispersed locations, a three-layer search architecture has been proposed for DS-index solution. The three layers include: 1) a multi-attribute index overlay, providing data indexing and multi-attribute range search capabilities within a private cloud, 2) a tree-based P2P network layer, supporting query forwarding and routing across different clouds, and 3) a federated cloud layer, providing physical connection within a cloud and between clouds.

### *A. Multi-attribute Index Overlay*

The multi-attribute index overlay is dedicated to indexing data and providing complicated search features in each private cloud. The data is indexed by a multidimensional tree in each private cloud, so that multi-attribute range queries can be achieved within the cloud. To build a multidimensional tree, the data space is partitioned along different dimensions (attributes). Many kinds of multidimensional trees can be deployed here, e.g. KD-tree [8]. We call these multidimensional trees the *distributed search trees* as they are dispersed in the federated cloud with each private cloud having one search tree. Search trees for different clouds may have different fan-outs and depths, because the data in different clouds have different scales and distribution. The search tree is updated when there are data updates in the cloud, including new data insertion, old data deletion and update. A search tree consists of a group of nodes, including internal nodes (the root node and intermediate nodes) and leaf nodes. Each node contains a subset of data that are located in the node and its descendants. More specifically, each node is coupled with a group of data value boundaries, which are corresponding to the multiple attributes of data.

### B. *Tree-based P2P Network*

The tree-based P2P network layer takes charge of query forwarding and routing between clouds. We adopt tree-based P2P network because the tree structure is capable to support range queries with each peer being mapped (or managing) some data with a range of values. Here we deploy P2P BATON tree [5] in this layer. A BATON tree consists of a group of peers that are built from servers across the federated cloud. Each peer keeps routing information to the related peers, whose corresponding servers may come from the same cloud or different clouds. In our design, each peer in the BATON tree is corresponding to a node from the distributed search trees. When a node from a distributed search tree is mapped to a peer in the BATON tree, the data value boundaries of that node are published to the peer. Each peer maintains a routing table to keep data value boundary information for itself and the related peers. When a peer receives a query command, it will look up the routing table to decide which peers to forward the query command by comparing the data value boundaries and the search range. With the mapping and query forwarding, desired data can be searched across the federated cloud. The mapping between search tree nodes and BATON peers is described in Section IV. The construction and update of BATON tree follow the processes described in [5].

### C. *Federated Cloud Layer*

The federated cloud layer provides physical connection within a private cloud and between the clouds. A federated cloud is the deployment and management of multiple external and internal cloud computing services to match business needs. Federated cloud can enable data sharing across different private clouds for scientific research and collaborations. Data is owned by individual organization/private cloud in the federation. To facilitate distributed search, servers from different private clouds are connected based on the P2P BATON BATON protocol [5], corresponding to the BATON peers. If two peers are routing neighbors in BATON, a TCP/IP connection is kept between the two corresponding servers. The mapping from a P2P network to a federated cloud layer is straightforward, as peers are built from servers across the federated cloud. Each private cloud has at least one server corresponding to the BATON peer.

## III. Mapping Algorithm and Cost Model

### A. *Dynamic Mapping between Index Overlay and Tree-based P2P Network*

To facilitate data search across the federated cloud, some nodes from search trees in the index overlay are selected and mapped to the peers in tree-based P2P network (Fig. 1). When a node is selected for mapping (in dark color in Fig. 1), its data value boundaries are published to the corresponding peer in the BATON tree accordingly. With new data insertion and old data deletion/update, the search tree will be updated by splitting/ merging nodes or rebalancing itself. The corresponding peers in the BATON tree will need to send messages to inform the related peers about the changes of data value boundaries. The message exchanges will lead to network cost eventually.

A dynamic mapping algorithm is proposed for DS-index solution, to select nodes from distributed search trees and map to the peers in P2P BATON tree. The algorithm targets to reduce network cost incurred by message exchanges between peers. In a cloud environment, network cost is one of the main concerns for cloud deployment and maintenance. Dynamic mapping is proposed here to adjust node selection for the case of search tree updates and query pattern changes.

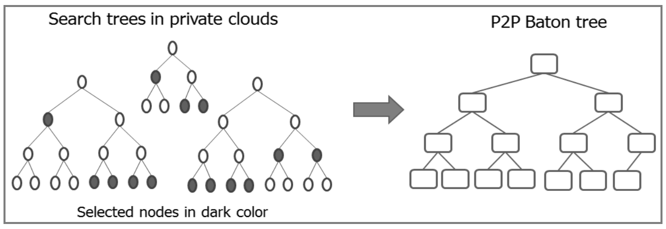


Fig. 1. Illustration of mapping between index overlay and P2P network

Initially, the algorithm selects the nodes in the 2nd level (children of the root node) of each search tree and maps them to the peers in P2P BATON tree. The reasons to choose this level of nodes for mapping are: 1) the cost of publishing data value boundaries of nodes in the 2nd level is relatively low compared to publishing those in the lower levels; and 2) mapping nodes in this level leads to fewer false positives in search than mapping the root node.

The node selection strategy always guarantees index completeness and index uniqueness for the distributed search trees. For a set of nodes *S* selected for mapping, the index is considered complete if and only if any data in the local dataset is contained by one node in *S*. Index completeness guarantees the correctness of query processing. In the other hand, the index is considered unique if for any node belonging to *S*, its descendant nodes are not belonging to *S*, and vice-versa. Index uniqueness minimizes the total cost of the published index. In order to reduce network cost, the nodes selected for mapping are dynamically adjusted during the operation of federated cloud. For each of the selected nodes, the cost of selecting the current node is calculated and compared with the cost of selecting its children nodes. The calculation and comparison continue until the leaf nodes, and the node selection strategy with the least cost will be opted. The adjustment of node selection happens when the tree updates or query pattern changes meet some pre-defined criteria, A cost model is defined (in the next part) to calculate network cost and facilitate node selection and adjustment.

With the proposed mapping algorithm, DS-index can enable federated data search across clouds with dynamic mapping between distributed search trees and P2P BATON tree. Network cost can be reduced with fewer message exchanges and less network bandwidth consumption, which is the main advantage in a cloud environment.

### B. *Cost Model*

We define a cost model and apply in the mapping algorithm to map between index overlay and tree-based P2P network.

Two types of network cost are considered in the model: 1) index maintenance cost: incurred by node splitting and merging, and tree rebalancing; and 2) node selection cost: regarding selecting and mapping nodes to a BATON tree. A Markov chain model [9] is applied to define the cost. We propose a three-state Markov chain model, in which three states are splitting, merging and rebalancing.

*1) Index Maintenance Cost:* Index maintenance is triggered by the search tree update including three scenarios: splitting, merging and rebalancing. Network cost is incurred when node splitting/merging and tree rebalancing lead to removal/addition of corresponding peers from/to the P2P BATON tree. It is measured by the number of messages exchanged. The maintenance cost of node n is defined as follows:

$$C_M(n) = 3logN(p_s(n)+p_m(n)) + T(n)\times logN \times p_b(n)$$
$$= logN(3(p_s(n)+p_m(n)) + T(n)\times p_b(n) \qquad (1)$$

where $N$ is total number of peers in a P2P BATON tree, $logN$ is average routing cost in a P2P BATON tree, $p_s(n)$ is probability of splitting node $n$ in the multidimensional search tree, $p_m(n)$ is probability of merging node $n$ with another node in the multidimensional search tree, $p_b(n)$ is probability of rebalancing the multidimensional search tree caused by splitting or merging of node $n$, and $T(n)$ is the total number of node removal and mapping from/to the P2P BATON tree due to rebalancing.

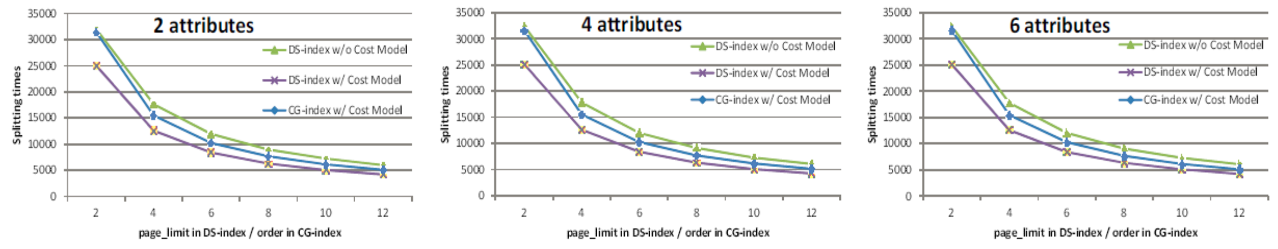
*2) Node Selection Cost:* Node selection cost includes the cost of selecting a set of nodes $S$ and map to a P2P BATON tree. It mainly considers false positive cost and index maintenance cost. False positive happens when high-level nodes are selected and mapped to the BATON tree, and some sub-queries are routed to the corresponding peer, but cannot get any results. Cost of selecting a set of nodes $S$ is defined as follows:

$$C(S) = \sum_{n\in S} c(n)$$
$$= \sum_{n\in S}(C_{FP}(n,Q) + C_M(n)) \qquad (2)$$
$$= \sum_{n\in S} logN(|Q_{FP}(n)| + 3(p_s(n)+p_m(n)) + T(n)\times p_b(n))$$

where $C(n)$ is the indexing cost of node $n$, $C(n)=C_p(n)+C_M(n)$, $C_{FP}(n)$ is the false positive cost of node $n$, $Q_{FP}(n)=\{q|q.range \cap n.range \neq \emptyset \wedge |f(q,n)| = 0\}$, $C_{FP}(n)=logN|Q_{FP}(n)$.

## IV. Evaluation

In this section, we evaluate the performance for the proposed DS-index solution. The performances are compared among the DS-index with cost model, DS-index without cost model, and CG-index with cost model. CG-index [7] presents a solution based on B$^+$-tree to address the cloud search problem, which can support single-dimension range search. CG-index also deploys a P2P BATON tree for cloud search, which is similar to our solution. To support multidimensional indexing and search, we use Universal B+-tree [10] to replace B+-tree in CG-index and all other mechanisms maintain the same as in [7]. For DS-index, we deploy multi-way KD trees for local multidimensional data search. All experiments are conducted on a Linux Server with Intel(R) Xeon(R) CPU X5650 @ 2.67GHz and 64GB of RAM, running 64-bit Ubuntu 10.10. The parameters used in the experiments are listed in Table 1. Due to the space constraint, we do not show the results of 3, 5, and 7 attributes for node splitting and merging.



(a) 2 attributes (b) 4 attributes (c) 6 attributes

Fig. 2. Performance comparison on node splitting

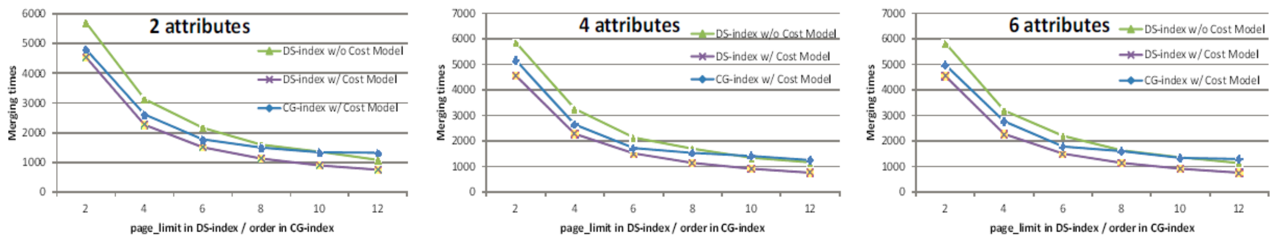


Fig. 3. Performance comparison on node merging

TABLE I. PARAMETERS USED IN EXPERIMENTS

| Parameter | Setting |
|---|---|
| Dimensionality | 2, 3, 4, 5, 6, 7 |
| Warmup | 1,000 points |
| Insertions | 50,000 points |
| Deletions | 10,000 points |
| Page limit of multi-way KD tree | 2, 4, 6, 8, 10, 12 |
| Order of Universal B+ tree | 2, 4, 6, 8, 10, 12 |
| Number of peers | 100, 200, …, 1,000 |

## *A. Splitting*

Performance comparison of splitting times is presented in Fig. 2 using three different approaches: DS-index with cost model, DS-index without cost model and CG-index with cost model. DS-index with cost model is much more efficient than the other two approaches for data with 2 ~ 6 attributes because of its excellent cost model, as shown in Fig. 2 (a) ~ Fig. 2 (c). Compared with DS-index without cost model, DS-index with cost model causes less splitting times because its node selection criteria guarantees index completeness and index uniqueness. There are not many splitting times in DS-index with cost model. DS-index with cost model uses less splitting times than CG-index with cost model because the former is more flexible than the latter. The latter explores Universal $B^+$-tree based on z-order [7], which is somehow difficult to adapt to frequent data insertions.

## *B. Merging*

Fig. 3 demonstrates performance comparison of merging times using three different approaches. Obviously, DS-index with cost model is much more efficient than the other two approaches: DS-index without cost model and CG-index with cost model for data with 2 ~ 6 attributes, as shown in Fig. 3 (a) ~ Fig. 3 (c). DS-index with cost model causes fewer merging times than DS-index without cost model because its node selection guarantees index completeness and index uniqueness, and there are not many merging times in DS-index with cost model. Compared with CG-index with cost model, DS-index with cost model uses fewer merging times because it is more flexible than CG-index with cost model.

## *C. Network cost*

Network cost is measured by the number of messages exchanged between peers. The more messages are exchanged, the higher network cost they cause. Network costs are shown in Fig. 4 for DS-index with cost model and DS-index without cost model. Compared with DS-index without cost model, DS-index with cost model reduces 27.65% in maximum and 26.40% in minimum in terms of the number of messages exchanged, respectively. Average reduced percentage is 27.32%. DS-index with cost model is better than DS-index without cost model in network overhead because its excellent cost model reduces splitting times, merging times and rebalancing times compared to DS-index without cost model. In addition, DS-index is scalable as shown in Fig. 4. DS-index with cost model has better scalability than DS-index without cost model, and the message exchanges are increased at smaller sub-linear rate when the number of peers increases.

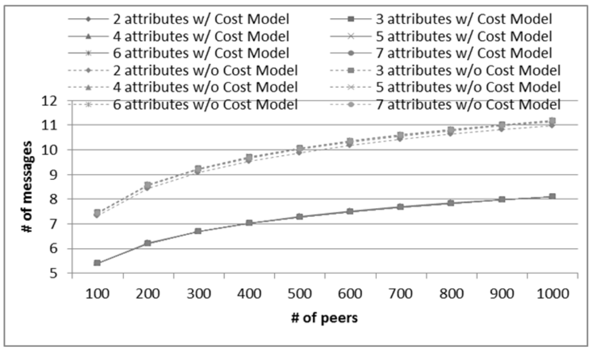


Fig. 4. Performance comparison on network cost

## V. CONCLUSIONS

This paper has proposed a DS-index solution to provide distributed search in federated cloud, with target to support multi-attribute range search with secondary index. DS-index includes a three-layer architecture, and we have presented a dynamic mapping algorithm for the DS-index solution to map between different layers. We have also defined a Markov Chain-based cost model to facilitate node selection and mapping. The proposed DS-index solution is capable to support multi-attribute range queries in the federated cloud. With the dynamic mapping algorithm and cost model, the proposed DS-index solution is more cost-effective than the traditional P2P networks as it can reduce the network cost in terms of index maintenance cost and node selection cost. The experiments have shown that, the DS-index solution with cost model can save the computation resource and reduce network bandwidth consumption by around 30% comparing to the one without cost model. It can also reduce the splitting/merging times by around 20% comparing to the state-of-the-art solution.

## REFERENCES

[1] R. Lea and M. Blackstock, “City hub: A cloud-based iot platform for smart cities,” in IEEE CloudCom 2014, 2014, pp. 799–804.

[2] Q. Xu, K. M. M. Aung, Y. Zhu, and K. L. Yong, “Building a large-scale object-based active storage platform for data analytics in the internet of things,” The Journal of Supercomputing, pp. 1–19, 2016.

[3] Rochwerger, B.; et al. "Reservoir - When One Cloud Is Not Enough," Computer , vol.44, no.3, pp.44-51, March 2011

[4] Amazon S3, http://aws.amazon.com/s3/.

[5] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. “BATON: A Balanced Tree Structure for Peer-to-Peer Networks”. In VLDB, pages 661–672, 2005.

[6] J. Wang, S. Wu, H. Gao, J. Li, and B. C. Ooi. “Indexing multi-dimensional data in a cloud system”. SIGMOD 2010, pp. 591–602.

[7] Sai Wu, Dawei Jiang, Beng Chin Ooi, Kun-Lung Wu. “Efficient B-tree based indexing for cloud data processing”, VLDB 2010, pp. 1207-1218.

[8] J. L. Bentley, “Multidimensional Binary Search Trees Used for Associative Searching”, Communications of the ACM, vol. 18, no. 9, pp. 509-517, 1975.

[9] Meyn, S. Sean P., and Richard L. Tweedie. Markov chains and stochastic stability, Cambridge University Press, 2009.

[10] Rudolf Bayer, “The Universal B-Tree for Multidimensional Indexing: general Concepts”, In WWCA '97, pp. 198-209.