Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

A BINARY PARTITION-BASED MATCHING ALGORITHM FOR DATA DISTRIBUTION MANAGEMENT

Junghyun Ahn
Changho Sung
Tag Gon Kim

Korea Advanced Institute of Science and Technology (KAIST)
373-1 Kuseong-dong, Yuseong-gu
Daejoen, Korea

ABSTRACT

Data Distribution Management (DDM) is one of the High Level Architecture (HLA) services that reduce message traffic over the network. The major purpose of the DDM is to filter the exchange of data between federates during a federation. However, this traffic reduction usually suffers from higher computational overhead when calculating the intersection between update regions and subscription regions in a matching process. In order to reduce the computational overhead for the matching process, this paper proposes a binary partition-based matching algorithm for DDM in the HLA-based distributed simulation. The new matching algorithm is fundamentally based on a divide-and-conquer approach. The proposed algorithm recursively performs binary partitioning which divides the regions into two partitions that entirely cover those regions. This approach promises low computational overhead, since it does not require unnecessary comparisons within regions in different partitions. The experimental results show that the proposed algorithm performs better than the existing DDM matching algorithms and improves the scalability of the DDM.

1 INTRODUCTION

Data Distribution Management (DDM) is one of the most important filtering mechanisms regarding large-scale distributed simulations (Morse and Steinman 1997). The DDM has been successful in reducing the network traffic in some aspect, but either its capability is limited by the computational overhead for matching update regions and subscription regions which represent interest of data producers and data consumers (Petty and Morse 2004). There are already proposed several DDM matching algorithms which will be described in Section 3. Even though there are several matching algorithms, they do not take into account how much regions are generated and distributed in the multidimensional space. As the previous matching algorithms do not consider a characteristic of the region distribution, it is difficult to select a matching algorithm which is appropriate for some situations of region distribution.

Therefore, in this paper, we propose a binary partition-based matching algorithm in order to reduce the computational overhead of the matching process in the DDM. The algorithm recursively performs binary partitioning which divides the regions into two partitions that entirely cover those regions. Our approach focuses on improving overall performance by the ordered relation of partitions previously not considered. The proposed algorithm reduces the computational overhead and decreases the execution time for the matching process, since it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within regions in different partitions. We experiment with different numbers of regions, overlap rates, and distribution of regions for various situations to reflect the region distribution. Our experimental results show that the proposed algorithm performs better than the previous DDM matching algorithms across a variety of workloads. The proposed algorithm outperforms significantly the previous matching algorithms when many regions are highly overlapped. The higher

regions are overlapped, the better performance of the proposed algorithm. In more complex scenarios where the distribution of regions is clustered, not uniform, the proposed algorithm is much better than the others. Moreover, the proposed algorithm improves the scalability of DDM implementation for large-scale simulations.

The remainder of this paper is organized as follows: Section 2 describes an overview of the DDM in the HLA and Section 3 briefly provides the related work of this paper. In Section 4, we present a binary partition-based matching algorithm which is based on a divide-and-conquer strategy. Section 5 shows experimental results across a variety of workloads. Finally, Section 6 includes concluding remarks.

2 OVERVIEW OF DDM IN HLA

DDM services, one category of the High Level Architecture (HLA) management services (IEEE Std 1516.1-2000 2000), reduce message traffic by sending the data only to those federates who need the data for system scalability. The DDM performs filtering based on interest expressions of federates in a federation and allows the federation to control the routing and delivery through user-defined information (i.e., regions). It then distributes objects and interactions from data producers to data consumers. Also, the DDM allows a federate to receive the subscribed attributes after the subscription region is intersected with the publishing federates' update region.

The DDM services allow a *value-based filtering*, (Morse and Steinman 1997). This type of filtering provides the most precise filtering mechanisms which ensure the federates receive the minimal set of data they interest. In the large-scale federation, it is necessary to filter more elaborate for data exchange during the federation execution. Therefore the DDM which provides the value-based filtering can be required for the large-scale simulations with numbers of data exchange.

Table 1: Terminology Definitions in the DDM.

Terminology	Definition
Dimension	An named coordinate axis with non-negative integers
Multidimensional space	A coordinate system whose dimension is d (where d is a fixed natural number)
Range	A continuous semi-open interval on a dimension (lower bound, upper bound]
Region	A set of ranges for any given dimension
Update region	A specified set of region instance for which is associated by a publishing federate
Subscription region	A specified set of region instance for which is associated by a subscribing federate
Overlap	All ranges of dimensions that are contained in the update region and subscription region put one upon another pairwise
Intersection	An existence when the corresponding region sets overlap
Matching process	A process to calculate the intersection between update and subscription regions

Table 1 presents the definitions of terms used in this paper. These definitions originate from the HLA standard (IEEE Std 1516-2000 2000). Although the definitions of terms are provided in Table 1, the terms usage is illustrated and described in Fig. 1. This figure shows an example of region intersection within 2-dimensional space for the *value-based filtering*. Because our interest lies in the simple exemplification of DDM terms, we assume that the multidimensional space is a 2-dimensional space. In the figure, there are four update regions and four subscription regions. An update region is the defined set of data declared by a publishing federate, whose information is delivered to subscribing federates. A subscription region is the area of interest declared by the subscribing federates. To check the overlapping of update and subscription regions is called *intersection*. When an intersection exists, data exchange occurs between the publishing federate and the subscribing federate. In this case, there are 2 intersections between update and subscription regions. A process to identify these intersections between update and subscription regions is a *matching*

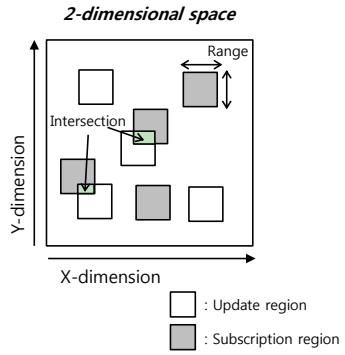

Figure 1: Example of region intersection in the 2-dimensional space.

process. The main role of DDM is to reduce the volume of data exchanged through the matching process during a federation.

3 RELATED WORKS

3.1 Region-based Algorithm

The region-based algorithm *exhaustively* compares the intersection of all the pairs of regions. This algorithm is quite straightforward while computing the intersection, as every update region is directly checked against every subscription region. If there are N update regions and M subscription regions, there are $N \times M$ pairs to check in the worst case. However, a considerable amount of computational overhead occurs in the matching process.

3.2 Grid-based Algorithm

The grid-based algorithm involves dividing the multidimensional space into a grid of cells, and update and subscription regions are allocated into each cell of the grid which is associated with a multicast group (Boukerche, Roy, and Thomas 2000), (Boukerche and Dzermajko 2004). In this algorithm, as an exact evaluation of the matching process is not implemented, update and subscription regions are assumed to overlap if and only if they share at least one common grid cell. Although the grid-based algorithm requires much less computation than the region-based algorithm, the overlapping information is not exact. This incorrect matching creates irrelevant data communication and additional receiver-side filtering is required. Another problem is that it is hard to define the appropriate size of grid cells (Ayani, Moradi, and Tan 2000).

3.3 Hybrid Approach

The hybrid approach improves performance by exploiting the advantages of the two previous approaches (i.e., region-based algorithm and grid-based algorithm) and minimizing the drawbacks of the grid-based algorithm (Tan, Zhang, and Ayani 2000). The overhead of the matching process is lower than that of the region-based algorithm, and the overlapping information is exact. It also produces a lower number of irrelevant messages than the grid-based algorithm. However, the major issue of this approach is selecting the optimal size of grid cells which the performance of the hybrid approach depends on.

3.4 Sort-based Algorithm

The sort-based algorithm detailed in (Raczy, Tan, and Yu 2005) and (Pan, Turner, Cai, and Li 2007) computes the intersection between update and subscription regions using a sorting algorithm. It projects the regions on each dimension. The end points in each dimension of all regions are sorted for each direction in order to determine the overlap information. When the sorted lists of end points are scanned, the sort-based

algorithm can maintain the set of subscription regions before and after the current position. Therefore, it is possible to know exactly, for each update region, which subscription regions match on each dimension. However, the sort-based algorithm's performance is degraded when the regions are highly overlapped and it is needed to optimize the sorting data structure for the efficient matching operation.

4 BINARY PARTITION-BASED MATCHING ALGORITHM

In this section, we present the binary partition-based matching algorithm. Our approach takes a *divide-and-conquer* approach similar to the one used for the quicksort (Cormen, Leiserson, and Rivest 1990). This approach consists of two main processes, the repetitive binary partitioning process and the matching process. First, in the binary partitioning process, the algorithm recursively performs binary partitioning which divides the regions into two partitions that entirely cover those regions. Second, in the matching process, the algorithm uses the concept of an ordered relation which represents the relative location of partition. The region is represent as $\{i, d, (r_d.l, r_d.u)\}$ where i is an identifier of region, d is a specified dimensions, and $(r_d.l(i), r_d.u(i))$ is the range of region whose upperbound is $r_d.u(i)$ and lowerbound is $r_d.l(i)$ with i region handle on d dimension. The proposed algorithm promises low computational overhead, since it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within regions in different partitions which are located in the ordered relation of partition.

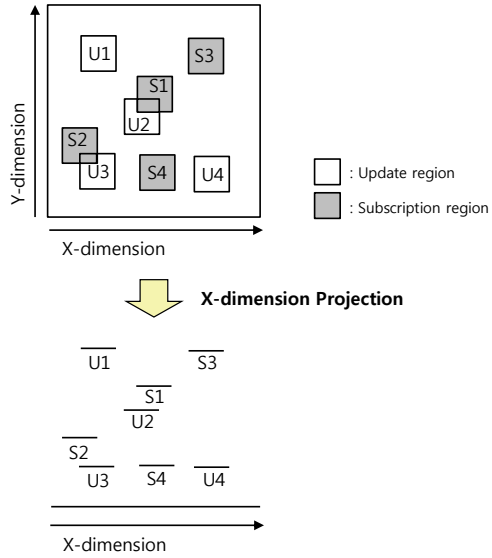

Figure 2: Dimension projection with the X dimension.

4.1 Dimension Projection

To facilitate a meaningful interpretation of the multidimensional space, our approach uses a well-known dimension projection algorithm, (Raczy, Tan, and Yu 2005). The dimension projection algorithm is being used to determine the overlap information in multidimensional space, spatially. The previous works for the regions in the multidimensional space is not intuitive, so the dimension projection algorithm is being used. The algorithm first works by projecting all dimensions in the multidimensional space using the above definition. It reduces the multidimensional problem to a one-dimensional problem and intuitively finds out the pairs of regions that are potentially overlapped. It then carries out the partition process, as will be presented in Section 4.2. Fig. 2 shows the dimension projection of eight regions located in 2-dimensional

space. In the figure, we project each region to the x-dimension. Next, the x-dimension is recursively partitioned into a binary partition. After performing one dimension, the algorithm repeats this procedure for y-dimension. The overall overlap information can be obtained by combining the information of each dimension: two regions overlap if and only if they overlap for x-y dimensions.

4.2 Proposed Binary Partition-based Matching Algorithm

As our algorithm divides the regions into two partitions, the left and right partitions, it guarantees the avoidance of checking the intersection between regions in the different partitions. The left and right partitions are located in the ordered relation which is described in the later section.

Algorithm 1 Binary Partition Algorithm

Input : The projected *Regions* $\top\mathbb{R}_d$, pivot value p
Output : Regions set, S_{lu}, S_{ru}, S_{pu}, S_{ls}, S_{rs}, S_{ps}

```
1: procedure BinaryPartition(⊤ℝ_d, p)
2: for Each region R_i ∈ Region ⊤ℝ_d do
3:    if R_i = update region then
4:       if r_d.u(i) ≤ p then
5:          S_lu ← {i} ∪ S_lu
6:       else if r_d.l(i) > p then
7:          S_ru ← {i} ∪ S_ru
8:       else
9:          S_pu ← {i} ∪ S_pu;
10:      end if
11:   else if R_i = subscription region then
12:      if r_d.u(i) ≤ p then
13:         S_ls ← {i} ∪ S_ls
14:      else if r_d.l(i) > p then
15:         S_rs ← {i} ∪ S_rs
16:      else
17:         S_ps ← {i} ∪ S_ps;
18:      end if
19:   end if
20: end for
```

In the binary partitioning, the projected regions are partitioned in approximately two equal sized sets, which are assigned to the left and right partitions, $\mathscr{P}_l$ and $\mathscr{P}_r$. Regions are assigned to the left partition if the coordinate is smaller than the pivot value, otherwise to the right partition. Additionally, if regions fall inside the boundaries of these partitions, the region are assigned to the pivot partition, $\mathscr{P}_l$. The detailed binary partition algorithm is presented as Algorithm 1. Those regions which the coordinate are greater than the pivot value are put in $\mathscr{P}_r$, otherwise they are put in $\mathscr{P}_l$. The $\mathscr{P}_l$ consists of two subsets of regions S_{lu} and S_{ls}, i.e. update and subscription regions for the left partition, respectively. Also, $\mathscr{P}_r$ consists of two subsets of regions S_{ru} and S_{rs}, respectively. A conceptual description shown in Fig. 3 demonstrates the binary partition. After the dimension projection, the algorithm uses the binary partitioning the regions around a pivot value and then dealing with the 2 smaller partitions, separately. As a result, the x-dimension is partitioned into two partitions by the pivot value and a pivot partition. They are the left partition of the pivot value, the right partition of the pivot value, and the partition with the pivot value (i.e., pivot partition). In this figure, from the left partition $\mathscr{P}_l$, we find that S_{lu} is $\{$U1, U3$\}$ and S_{ls} is $\{$S2$\}$, respectively. In this way, the other set in each partition is also arranged by the binary partition.

Through the binary partition, if two regions are different partitions, they do not overlap through the ordered relation of partition. As can be seen the above definition, the *ordered relation* between different partitions plays an important role in deciding the overall overlap information of regions. The algorithm

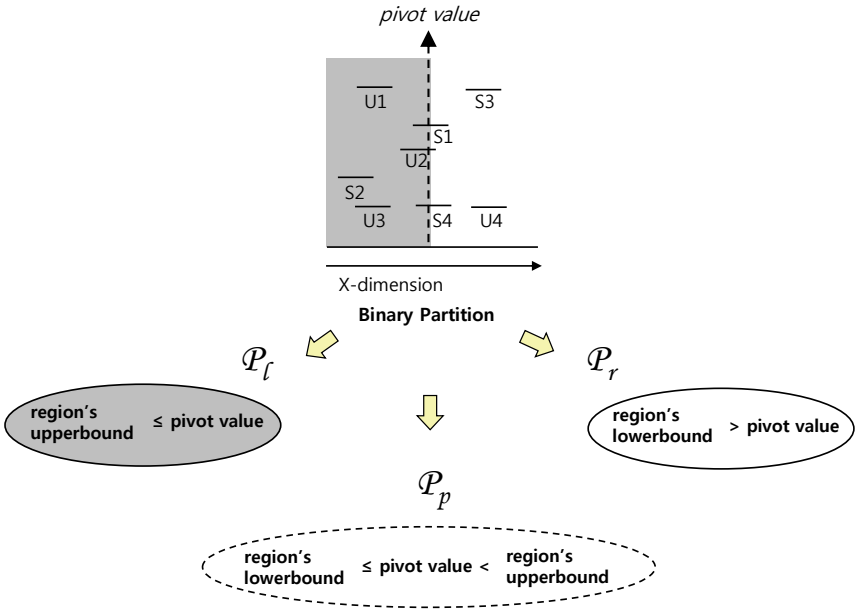

Figure 3: Binary partition into three partitions, $\mathscr{P}_l$, $\mathscr{P}_r$, and $\mathscr{P}_p$.

constructs partitions of regions. As there is no need to determine the overlap information between the left and right partition's region, the algorithm does the matching operation the left and right partition with the pivot partition. In other words, it is not necessary to compare and match in the left and right partitions through the ordered relation in the binary partition, but it is necessary to compare and match the pivot partition with the left and right partitions.

Algorithm 2 IntersectionCalculation Algorithm

Input : Regions set, S_l, S_r
Output : n-by-n matrix, $OM = (om_{ij})$, where $i, j \in n$

```
1: procedure IntersectionCalculation(S_l, S_r)
2: for Each region R_i ∈ S_l do
3:     for Each region R_j ∈ S_r do
4:         if r_d.u(i) > r_d.l(j) then
5:             om_ij++;
6:         end if
7:     end for
8: end for
```

From the binary partition, we can extract regions which fall inside the boundaries of partitions by using *BinaryPartition()*. These regions are collected into a pivot partition, $\mathscr{P}_p$. Also, the pivot partition consists of two subsets of regions S_{pu} and S_{ps}, respectively. This partition is used for the matching process to compare the intersection between regions in the pivot partition, $\mathscr{P}_p$, and regions in the left and right partition, $\mathscr{P}_l$ and $\mathscr{P}_r$.

For the matching process, it is checked by the *IntersectionCalculation()* algorithm which is presented in Algorithm 2. As the left and right partitions are the ordered relation in the binary partition, there is no need to determine the overlap information between the left and right partitions. Therefore, the algorithm does one matching operation when comparing the left and right partition with the pivot partition.

The *IntersectionCalculation()* algorithm determines whether subscription regions in S_{ps} overlap with update regions in S_{lu} and S_{ru} or update regions in S_{pu} overlap with the subscription regions in S_{ls} and

S_{rs} because of the ordered relation. Additionally, all update regions and subscription regions in the pivot partition are overlapped, since their range includes the pivot value. As the regions in the boundary of the pivot partition will be all overlapped, the proposed algorithm finds the calculation of intersection in the matching process. If there is with a high overlapped region's distribution, the calculation of intersection is easily obtained. Therefore, this approach is more efficient to compute the high overlapping regions on each partition. This matching process results in a matrix with overlap information, *OM*. The overlap matrix is the $n \times n$ matrix (i.e., n = the number of regions). If two regions whose handle are (i, j) have overlapped on each dimension, om_{ij} is added (+1). After the whole matching process, if om_{ij} is same as the number of dimension, d, two regions $\mathscr{R}_i$ and $\mathscr{R}_j$ is exactly overlapped. In the example explained in Fig. 3, we obtain $OM = \{o_{u1,s2}, o_{u2,s1}, o_{u2,s4}, o_{u3,s2}, o_{u4,s3}\}$.

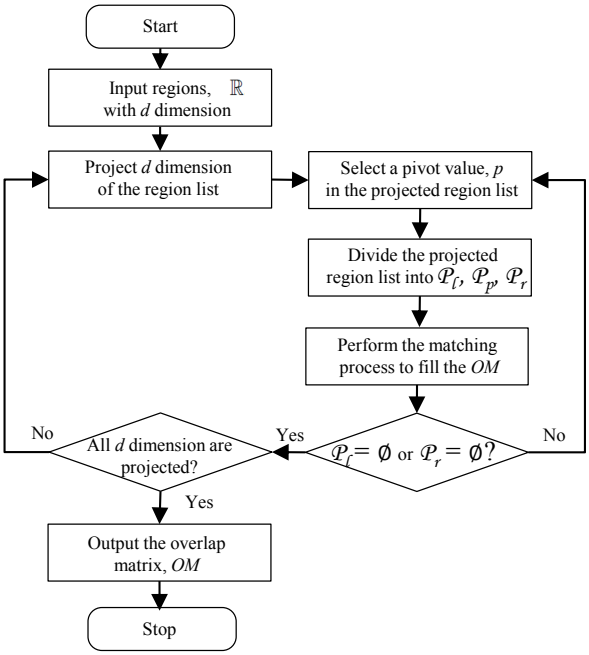

Figure 4: Flowchart of the proposed algorithm.

The proposed algorithm decomposes the multidimensional space into a number of partitions through the binary partition. The algorithm is adopted the divide-and-conquer strategy based on the ordered relation. The flowchart of the proposed algorithm is presented in Fig. 4. The process of the flowchart is executed in the following steps.

1. Project all dimensions of regions with the multidimensional space into one-dimension through the dimension projection.
2. Set a midpoint as the pivot value in the projected regions.
3. Divide the projected regions into twp partitions, $\mathscr{P}_l$ and $\mathscr{P}_r$, and the pivot partition, $\mathscr{P}_p$, with six sets, S_{lu}, S_{ls}, S_{ru}, S_{rs}, S_{pu}, and S_{ps}, through the binary partition.
4. Perform the matching process in the pivot partition. Also, perform the matching process in the left and right partitions to compare update regions in S_{lu} and S_{ru} with subscription regions in S_{ps} and subscription regions in S_{ls} and S_{rs} with update regions in S_{pu}. From this matching process, our algorithm determines *OM* which stores the overlap information into n-by-n matrix (where n is the number of regions).
5. Iterate the pivot selection and matching process in the $\mathscr{P}_l$ and $\mathscr{P}_r$ until the remain is never partitioned.

The binary partition-based matching algorithm is shown Algorithm 3. It uses two procedures, *BinaryPartition()* and *IntersectionCalculation()*, for the partition process and the matching process.

Algorithm 3 Binary Partition-based Matching Algorithm

Input : *Regions* $\mathbb{R}$
Output : n-by-n matrix, $OM = (om_{ij})$, where $i, j \in n$

```
 1: procedure BinaryPartitionbasedMatchingAlgorithm(Region ℝ)
 2: for Each dimension, d do
 3:    g1 ← the minimum lower bound of d dimension;
 4:    g2 ← the maximum upper bound of d dimension;
 5:    Set a pivot value, p ← (g1 + g2)/2;
 6:    Use the BinaryPartition(⊤ℝ_d, p) to find six sets S_lu, S_ru, S_pu, S_ls, S_rs, S_ps;
       {For the pivot partition}
 7:    for Each region R_i ∈ S_pu do
 8:       for Each region R_j ∈ S_ps do
 9:          om_ij++;
10:       end for
11:    end for
       {For the left and right partition}
12:    IntersectionCalculation(S_ls, S_pu);
13:    IntersectionCalculation(S_pu, S_rs);
14:    IntersectionCalculation(S_lu, S_ps);
15:    IntersectionCalculation(S_ps, S_ru);
16:    S_l ← S_lu ∪ S_ls;
17:    S_r ← S_ru ∪ S_rs;
18:    BinaryPartitionbasedMatchingAlgorithm(Region S_l);
19:    BinaryPartitionbasedMatchingAlgorithm(Region S_r);
20: end for
```

4.3 Theoretical Analysis on Computational Complexity

To analyze the computational complexity of the proposed algorithm, we suppose that there are N regions. For the simple analysis provided here, we assume that the number of dimensions, d is 2 in the multidimensional space. The process of the binary partitioning divides regions into the two partitions. Steps 24-43 in the *BinaryPartition()* procedure require $\mathcal{O}(N)$ computation using the binary partitioning, because d is a constant. To find the exact position of a region in the partitions, Steps 12-13 require $N \times \mathcal{O}(log\ N)$ computation.

Moreover, the matching process of comparing the intersection of regions between partitions requires $\mathcal{O}(n^2)$ computation (where n is the number of regions in each partition). The complexity of the *IntersectionCalculation()* procedure is proportional to n. It seems that the most important points are the binary partitioning. It is obvious that the number of regions, n, is a *determinant* factor. Because the overlap information of all regions is obtained by the pivot partition, it is not necessary to compare their overlap information in the left and right partitions. In cases where there is no overlapping at all in these partitions, the binary partitioning with the pivot partition and the left and right partitions is the most time-consuming operation which requires n^2 computation. So the computational complexity of the proposed algorithm is $n^2 \times \mathcal{O}(log\ N)$ computation. If the number of regions, n, is normally very small in a large-scale spatial environment, so the proposed algorithm should be very efficient. Therefore, the actual computational complexity depends on how the binary partition is well achieved.

5 EXPERIMENTAL EVALUATION

This section presents an experimental evaluation of the proposed method. We compare the proposed algorithm with the previous DDM matching algorithms in the literature, the region-based algorithm, hybrid approach (Ayani, Moradi, and Tan 2000), Raczy's sort-based algorithm (Raczy, Tan, and Yu 2005). We implemented these algorithms with C++ language and processed in the HLA-based distributed simulations. We assume that our experimental environment has a 2-dimensional space for the sake of clarity as described in Section 4. Since the performance of the hybrid approach depends on the number of grid cells, we used configurations of both 10×10 and 100×100 grid cells which are used in (Raczy, Tan, and Yu 2005) and (Ayani, Moradi, and Tan 2000). As stated before, our goal is to show the practicability of the binary partition-based matching algorithm in the DDM. As the performance of the DDM *execution time* for the matching process is measured, our experiments were conducted using Microsoft Windows 7 with a 2.80GHz Intel(R) Core(TM) i7 CPU and 8GB memory.

One of the important experimental parameter is the number of regions. This parameter is capable of characterizing the entire situation of regions for the matching process. It represents the scalability of the system. We also used two different methods to generate the location of each region in the space. These methods have a uniform distribution and clustered distribution. As these synthetic regions can be useful in a variety of situations, our experiment provides realistic - but not real - regions distribution to third parties for testing the DDM matching algorithms. Therefore, the experiment uses the uniform distribution and clustered distribution as the synthetic region distribution which can achieve simple forms of the realistic data set. In the uniform, the regions are distributed randomly across the space, whereas the clustered distribution has regions around $k \geq 1$ clusters. Finally, the overlap rate is defined as the proportion of the scene volume occupied by the regions.

$$Overlap\ rate = \frac{\sum area\ of\ region}{area\ of\ space} \tag{1}$$

If the space is 100×100 and one region is 1×1, where the number of region is fixed at 100, the overlap rate is 0.01 = $\frac{100 \times (1 \times 1)}{100 \times 100}$. Note that higher overlap rate implies greater probability of region overlap. For enhanced accuracy and reliability, we repeated the experiment more than 30 times. A 95 percent confidence interval was estimated for all experimental results. For enhanced accuracy and reliability, we repeat the experiment more than 30 times. A 95 percent confidence interval was estimated for all experimental results.

5.1 Uniform Distribution

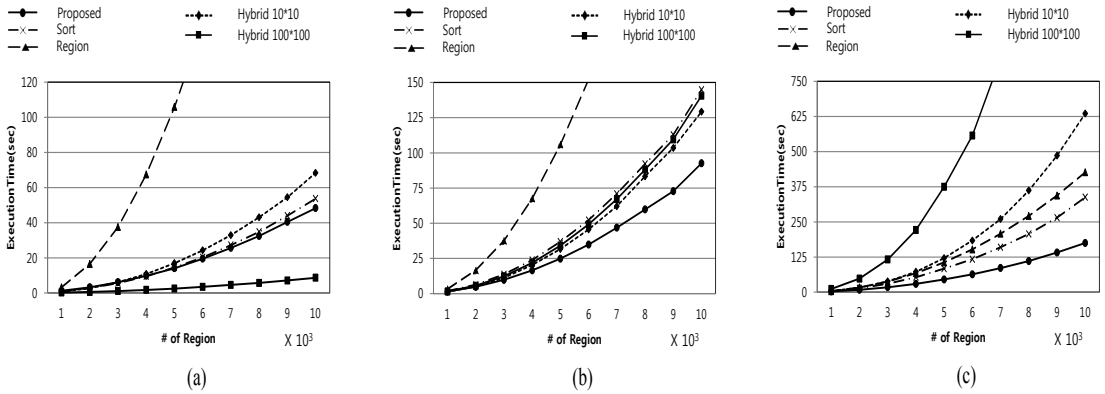

Figure 5: Performance Comparisons (a) Overlap rate = 0.01, (b) Overlap rate = 0.1, and (c) Overlap rate = 1 under uniform distribution.

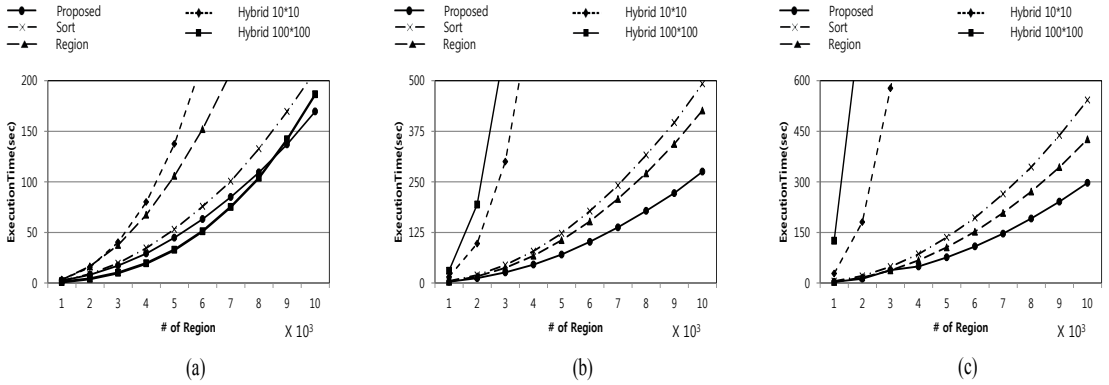

Figure 6: Performance Comparisons (a) Overlap rate = 0.01, (b) Overlap rate = 0.1, and (c) Overlap rate = 1 under clustered distribution.

Fig. 5(a) shows the execution time for the matching process in four algorithms when the overlap rate is 0.01. It seems that the hybrid approach with 100 $\times$ 100 grid cells always has the best performance because the regions in one cell are small in the overlap rate = 0.01. However, the hybrid approach did not perform well in situations where the size of the grid cells changed (i.e., the performance at 10 $\times$ 10 grid cells of the hybrid approach is degraded). The proposed binary partition-based matching algorithm outperforms the other matching algorithms when the overlap rate is 0.01 under the uniform distribution (except the hybrid approach which the overlap rate is low and the optimal 10 $\times$ 10 grid cells are used).

Fig. 5(b) shows the execution time for the matching process in the overlap rate = 0.1. According to this figure, well in this situation where the overlap rate is 0.1. The performance of the other algorithms are similar in the overlap rate = 0.01. As the hybrid approach has the computational overhead in the matching process with more regions, the performance is significantly degraded.

With the overlap rate = 1, according to Fig. 5(c), the proposed algorithm performs well. The proposed algorithm is not the best choice when the overlap is relatively low, but it has the advantage when the overlap rate is high. The more regions that are overlapped with the pivot partition, the less matching overhead is incurred in the whole matching process. It means that the higher regions are overlapped, the better performance of the proposed algorithm. On the other hand, the sort-based algorithm performs better, except our proposed algorithm in the overlap rate = 1. When the number of regions increases and the overlap rate is high, the performance of the region-based algorithm becomes increasingly better than the other overlap rate. The hybrid approach has a higher computational overhead, particularly with 100 $\times$ 100 grid cells. This approach degrades performance when the overlap rate is high. From all of the figures, we know that the hybrid approach with 100 $\times$ 100 grid cells has an extremely big computational overhead for the matching process.

5.2 Clustered Distribution

We next varied the overlap rate under the clustered regions distribution. The second set of experiments compared the execution time for the matching process under the clustered distribution. When the regions were distributed under the clustered distribution, there was a greater probability of region overlap than the uniform distribution in the *fixed* number of regions. Fig. 6(a) shows the performance comparison of the four algorithms. When the number of regions was more than 9000, the proposed algorithm performed better than the hybrid approach with 100 $\times$ 100. This finding suggests that the proposed algorithm is scalable with a large-scale spatial environment.

As described in the Fig. 6(b), the hybrid approach is not much more scalable. Even worse, they will not perform well in the clustered distribution, since most regions are collected in some points. The region-based

algorithm and the sort-based algorithm perform well when there is a high probability of finding a pair of regions that intersect. The proposed algorithm is the best choice in the clustered distribution, since it separates the clustered regions into several partitions for the matching process.

Fig. 6(c) shows the execution time for the matching process. All experimental results are similar performances when the overlap rate is 0.1. The proposed algorithm shows the best performance compared with the other algorithms, as expected. This shows that the proposed algorithm is scalable and performs well in the clustered region distribution.

5.3 Discussions

With complicated overlap situations, the proposed algorithm has worked more robustly than the other matching algorithms. As the proposed algorithm avoids checking any regions between different partitions, it is the best choice (except the hybrid approach which the overlap rate is low and the optimal grid size is used). Furthermore, it has additional advantages compared with the other algorithms. First, the proposed algorithm is practical and efficient because it reduces the computational overhead and improves overall performance for the matching process. Since the proposed algorithm reflects the region distribution by using the concept of the ordered relation, it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within regions in different partitions. Second, it is less affected by the overlap rate. The algorithm recursively performs binary partitioning which divides the regions into two partitions that entirely cover those regions. This means there is no need to determine the overlap information between partitions, because they already consider whether the regions are intersected through the ordered relation. This requires minimal computational overhead in some partitions, especially at high overlapping regions. This is attributed to the same reason as the case of the clustered distribution. Overall, from the experimental results, we found that our algorithm performs better than the others in a given set of scenarios, especially at the high overlap rate and in the clustered distribution. And it improves the scalability of DDM implementation for large-scale simulations. Furthermore, to complete the study, we tested all the algorithms in more complex situations, for instance, scenarios where the distribution of regions is not uniform (i.e., clustered distribution).

6 CONCLUSIONS

This paper proposes a binary partition-based matching algorithm which is based on the divide-and-conquer approach. The algorithm recursively performs binary partitioning which divides the regions into two partitions that entirely cover those regions. The proposed algorithm is, to the best of our knowledge, the first attempt to facilitate the binary partitioning and obtain the overlap information by using the concept of an ordered relation previously not considered. The algorithm reduces the computational overhead for the matching process, since it easily calculates the intersection between regions on partition boundaries and does not require unnecessary comparisons within regions in different partitions. According to the theoretical analysis, the proposed algorithm has the advantage in the scalability. We have developed the binary partition-based matching algorithm in the DDM and compared its performance with the previous matching algorithms in different situations of region distribution. Our experimental results show that the proposed algorithm performs better than the previous matching algorithms across a variety of workloads. Especially, the proposed algorithm significantly outperforms in a given set of scenarios of the high overlap rate. In more complex scenarios where the distribution of regions is clustered, not uniform, the proposed algorithm is much better than the others.

REFERENCES

Ayani, R., F. Moradi, and G. S. H. Tan. 2000. "Optimizing cell-size in grid-based DDM". In *14st International Workshop on Principles of Advanced and Distributed Simulation (PADS 2000)*, 93–100.

Boukerche, A., and C. Dzermajko. 2004. "Performance evaluation of Data Distribution Management strategies". *Concurrency - Practice and Experience* 16 (15): 1545–1573.

Boukerche, A., A. Roy, and N. Thomas. 2000. "Dynamic Grid-Based Multicast Group Assignment in Data Distribution Management". In *4th International Workshop on Distributed Simulation and Real-Time Applications (DS-RT 2000)*, 47–54.

Cormen, T. H., C. E. Leiserson, and R. L. Rivest. 1990. *Introduction to Algorithms*. MIT Press.

Morse, K., and J. S. Steinman. 1997. "Data Distribution Management in HLA: Multidimensional Regions and Physically Correct Filtering". In *Proc. Spring Simulation Interoperability Workshop*, no. 97S-SIW-053.

Pan, K., S. J. Turner, W. Cai, and Z. Li. 2007. "An Efficient Sort-Based DDM Matching Algorithm for HLA Applications with a Large Spatial Environment". In *21st International Workshop on Principles of Advanced and Distributed Simulation (PADS 2007)*, 70–82.

Petty, M. D., and K. L. Morse. 2004. "The computational complexity of the high level architecture data distribution management matching and connecting processes". *Simulation Modelling Practice and Theory* 12 (3-4): 217–237.

Raczy, C., G. S. H. Tan, and J. Yu. 2005. "A sort-based DDM matching algorithm for HLA". *ACM Trans. Model. Comput. Simul.* 15 (1): 14–38.

Tan, G. S. H., Y. Zhang, and R. Ayani. 2000. "A Hybrid Approach to Data Distribution Management". In *4th International Workshop on Distributed Simulation and Real-Time Applications (DS-RT 2000)*, 55–61.

IEEE Std 1516-2000 2000. IEEE standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.

IEEE Std 1516.1-2000 2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification.

AUTHOR BIOGRAPHIES

JUNGHYUN AHN is a Ph D. candidate at the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST). His research interests include methodology for M&S of discrete event systems (DEVS) and DDM in HLA/RTI. His email address is jhahn@smslab.kaist.ac.kr.

CHANGHO SUNG is a Ph D. candidate at the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST). His research interests include discrete event systems modeling, collaborative M&S, distributed simulation, and hybrid simulation. His email address is chsung@smslab.kaist.ac.kr.

TAG GON KIM is a Professor at the Department of Electrical Engineering Korea Advanced Institute of Science and Technology (KAIST). He was the Editor-In-Chief for Simulation: Transactions for Society for Computer Modeling and Simulation International(SCS). He is a co-author of the text book, Theory of Modeling and Simulation, Academic Press, 2000. He published about 200 papers in M&S theory and practice in international journals and conference proceedings. He is very active in defense modeling and simulation in Korea. His e-mail address is tkim@ee.kaist.ac.kr.