A Dynamic Load Distribution Algorithm for Virtual Worlds

Umar Farooq, John Glauert
School of Computing Sciences
University of East Anglia
Norwich, UK
{u.farooq, j.glauert}@uea.ac.uk

Journal of Digital Information Management

ABSTRACT: *The introduction of modern communication and computation technologies has greatly changed the traditional approaches processing computationally intensive applications such as Virtual Environments (VEs). Distributed and multi-server architectures are scalable and exploit the concepts of spatial and temporal locality for handling massive Distributed VEs (DVEs). However, using these environments require special attention for scalability, consistency, load balancing, and latency. The existing mechanisms primarily developed for games have key performance issues in managing large scale virtual worlds. In our previous work (Farooq and Glauert, 2009a), we have proposed a hierarchical infrastructure named JoHNUM for the development of scalable and consistent virtual worlds. Further, we have introduced the concept of aggregate region assignment to minimise resource utilisation and distribute the load as balanced as possible among servers (Farooq and Glauert, 2009a, 2009b). This paper presents our proposed load distribution algorithm that yields regular and contiguous areas for assignment. It examines possible combinations of regions based on proposed strategies but greatly reduces the aggregation process by using different filters. It also presents a simple communication model justifying the proposed strategies and load distribution algorithm. Simulation results with an extended set of experiments show that the proposed algorithm obtains fair distribution of load and has the capability to cope with small, medium and large scale environments while minimising communication and implementation cost.*

Categories and Subject Descriptors
H. 5 [Information Interfaces and Presentation]: I 1.2 [Algorithms]

General Terms: Virtual environment, Load distribution, Algorithms

Keywords: Virtual environment, Virtual world, Scalability, Consistency, Load balancing, Second life.

Received: 11 August 2009; Revised 19 November 2009; Accepted 20 December 2009

1. Introduction

Virtual Environments (VEs) described as simulation environments or analysis tools (Fujimoto, 2001) are one the most widely studied and investigated area of research in recent decades. Early work in simulation was led by the military with the development of computer generated synthetic environments to simulate specific applications, and train individuals for complex and dangerous tasks. However, these environments were unable to handle group activities and interaction among simulators targeting specific applications. Literature shows a number of solutions to these problems including Distributed Interactive Simulation (DIS) (Fullford, 1996) and High Level Architecture (HLA) (Dahmann et al., 1997) concentrating on the development of shared synthetic environments and interoperability among individual simulators. The DIS and HLA are well known standards for distributed simulation that have been used and extended for diverse applications by different research communities. Currently, VEs cover different application areas including Military, Games, Education, Design, and Commerce. The introduction of the latest communication and computation technologies and availability of personal computers at affordable prices are the major factors for the success of current VEs. Distributed VEs use distributed infrastructures compared with traditional client-server approaches for achieving scalability and better performance thus increasing interactive and collaborative user experience. DVEs with collaborative and social activities are encouraging people to participate in creativity while having fun. The great success of these environments has motivated researchers from different communities to develop spaces with a sense of place usually called virtual worlds (Ondrejka, 2004) (Rosedale and Ondrejka, 2003). Second Life (SL) (Second Life, 2009) is state-of-the-art in virtual worlds allowing users to live their lives according to their desires. The Games, Graphics, Internet, High Performance Computing and Social communities have leading role researching this area.

The structure of a VE is normally comprised of three components: service provider end, the Internet, and client end. This paper investigates service provider end and relevant issues with an emphasis on achieving scalable and consistent infrastructures. The Literature shows a number of infrastructures managing VEs that can be classified as Client/Server, Cluster of Server (CoS) and Distributed approaches (Ng et al., 2003). The development of an infrastructure for VEs takes different issues into account including computation power, latency, scalability, and consistency. Considering these issues, distributed approaches have shown a remarkable success for different application areas. Distributed infrastructures can be further categorized as Peer-to-Peer (P2P) and Client Multi Server (CMS) systems. However, according to Ng et al., CMS systems are best to manage DVEs that are recently extended for grid infrastructures. Grid infrastructures are more dynamic and resilient, and SL (Second Life, 2009) is hosted by a grid called SL Grid (SLG) (Second Life Grid, 2009). It is a customised form of the Butterfly Grid (BG) (IDC, 2003).

Scalability, consistency, and load distribution are the prime concerns of current large scale DVEs and therefore need powerful computation, communication and consistency strategies. (Lee et al., 2007) argue that the design of a scalable network VE needs considerations for communication architecture, interest management, concurrency control, data replication, and load distribution. Generally a VE is split into a number of regions and assigned to a dedicated set of servers for processing. Region based schemes can be categorised as static or dynamic. Both BG (IDC, 2003) and SLG (Second Life Grid, 2009) exploit region

based static assignment strategies. To eliminate problems in static configurations, a number of local, global and adaptive dynamic strategies are developed having their benefits and limitations (Ng et al., 2002) (Lui and Chan, 2002) (Lee and Lee, 2003). (Vleeschauwer et al., 2005) have achieved better load distribution but with a significant increase in communication between cells. (Shirmohammadi et al., 2008), (Chertov and Fahmy, 2006), and (Morillo et al., 2006) have also attempted the issues of scalability and load distribution. A number of hierarchal infrastructures can be found in the literature addressing the same issues (Oliveira and Georganas, 2003) (Burlamaqui et al., 2006) (Kim and You, 2006) (Balan et al., 2005). However, they have major performance issues when used to host a virtual world because these are basically developed for games. A detailed analysis and comparison of the existing mechanisms is given in (Farooq and Glauert, 2009a). We believe that virtual worlds are a step ahead of games that lack pre-defined rules. Both latency and consistency are prime concerns for these environments.

Currently, game infrastructures are hosting virtual worlds and the unavailability of a suitable infrastructure motivated us to develop a novel Joint Hierarchical Nodes based User Management (JoHNUM) infrastructure. It supports a regional server by splitting a VE into a number of regions dynamically on the basis of load and assigns it to a set of child nodes. Resource utilisation and communication is minimised by utilising the concept of aggregate assignment. The complexity of partitioning and load distribution is localised and only limited numbers of players are affected by a split compared with the load transfer mechanisms. In case of underutilisation, child servers return the regions to the parent servers and become part of the available pool of resources. Aggregate assignment is the intelligent part of our infrastructure that aims to achieve load as balanced as possible among the servers. For comparison and evaluation purposes, we have used a hierarchical middleware for games called Matrix (Balan et al., 2005) due to its similarity with our work. It achieves low latency but compromises on consistency, and has demonstrated better performance than static and dynamic strategies. It degrades interactive user experience and yields a Resource Management Tree (RMT) of many levels because each overloaded server shares its load with a new child server. To achieve better consistency, we believe that levels in RMT need to be minimised.

The paper intends to provide an abstract of our proposed JoHNUM infrastructure and presents our proposed load distribution algorithm based on the concept of aggregate assignment. The basic goal is to achieve a fair distribution of load while guaranteeing regular and contiguous areas for assignments. The algorithm uses a number of filters to minimise the effort for finding the best aggregates. It also presents a communication model justifying the use of the proposed strategies for the minimisation of communication and implementation cost. Further, it provides the results of an extended set of experiments for large scale VEs, showing that the proposed algorithm performs well with small, medium, and large scale VEs. This paper is structured as follows. Section 1 introduces the subject matter in brief. An abstract of our proposed JoHNUM infrastructure is provided in section 2. Section 3 describes the proposed load distribution algorithm and aggregation strategies. It also includes the proposed communication model. An extended set of experiments for the illustration of the proposed algorithm is presented and discussed in section 4. Conclusions are provided in section 5.

2. Background

This section briefly describes our proposed JoHNUM infrastructure (Farooq and Glauert, 2009a). The VE is assumed of approximately square shape and follows a regular square pattern for splitting overloaded regions. Further, each server handles no more than a number of users named the Maximum Server Capacity (MSC). The number of regions and their boundaries are calculated with a factor known as the Region Split Factor (RSF). The number of prospective regions is computed by squaring the RSF and regional boundaries are determined by utilising height and width of the VE.

The JoHNUM infrastructure comprises of three components: Partitioning, Assignment and Merging. Partitioning is hierarchical in nature and is triggered by a regional server as its capacity exceeds the MSC. A region is the basic unit of splitting and the concept of aggregate assignment is proposed to minimise resource utilisation and communication. Further, for minimisation of RMT levels, we have introduced two terms called provisional assignment and permanent assignment. The partitioning algorithm splits the square region handled by a server into a set of RSF^2 smaller regions. This set is then divided into two subsets, achieving as balanced a load as possible. One subset is handled by the original server while the other is passed to a new child server. At a later stage, triggered by increasing load, these subsets are re-assigned until a server is handling a single region. At that point the region will be further subdivided into yet smaller regions unless it is too small for subdivision (boundary condition). The term "*provisional assignment*" is used for aggregate assignment keeping the identity of individual regions while "*permanent assignment*" is used for a single smaller region. The server triggering a split becomes the parent of the new server. The merging algorithm implements the reverse process of partitioning. The proposed partitioning algorithm is simple and better performance is achieved by devising intelligent assignment strategies.

Our partitioning algorithm works as follows. Each server continuously monitors total players against the MSC and in case of excessive load selects an RSF value for region split. It starts with 2 that split a region into 4 regions, but, in case the players in any prospective region exceed the MSC, the next RSF value is used that divides it into 9 regions. The partitioning strategy and these cases are described in figure 1, assuming MSC of 9 players. Solid lines demonstrate actual split while dashed lines highlight prospective regions only. Figure 1(a) eases the load by splitting into 4 regions, however, the case in figure 1(b) fails to ease the load as the players in top right prospective region exceeds the MSC. Therefore, instead of splitting into 4, the region is divided into 9 regions that ease the load as shown in figure 1(c).

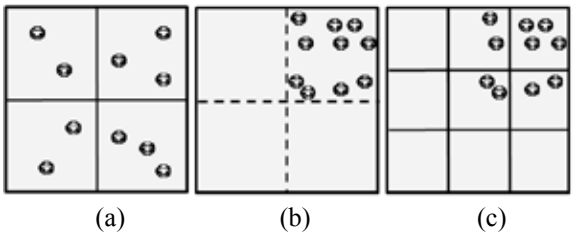

Figure 1. Illustration of JoHNUM Partitioning

The JoHNUM Assignment describes four different strategies but the last strategy outperforms Matrix and the remaining JoHNUM assignment strategies. It applies aggregation of regions at each level against the MSC and tries to achieve as balanced the load as possible. It first splits the set of regions into two distinct sets of regions and then selects another server for sharing its load named as a child server. The server keeps one set of regions and passes the other to the child server. However, it reduces the levels in RMT

by making new server a child of the server that initiated the split.

We have evaluated JoHNUM with a number of experiments by using different metrics including number of regions, resource utilisation, RMT levels, and number of times a player suffers from splitting. It is compared with Matrix and has demonstrated better performance by achieving better RMT levels and interactive user experience. Moreover, it reduces communication overhead and minimises resource utilisation, almost the same as Matrix. The concept of aggregation reduces the number of servers and requires only a single new server to become a child of the split initiated server and share the load with it. In (Farooq and Glauert, 2009b), we have investigated load distribution and proposed an aggregate region assignment algorithm for resource minimisation and load distribution. It is evaluated with some initial experiments. Here we provide a more detailed description of the filters that are used to reduce the required effort. Further, we need to develop a communication model for justifying the proposed aggregation strategies. Moreover, it needs to be evaluated for medium and large scale VEs. Therefore, in this paper we present our load balancing algorithm with an emphasis on the description of filters, a communication model, and an extended set of experiments. It is clear that the proposed algorithm is equally suitable for small, medium, and large scale VEs.

3. Proposed Load Distribution Algorithm

In this section, we propose a number of strategies achieving load as balanced as possible among the servers. This work combines only the regions sharing physical boundaries and no horizontal or vertical lines are allowed to go out of one aggregate, into the other and back again. However, a diagonal region is considered for aggregation if it shares a boundary with a region already in one of the two sets of regions during the aggregation process. The main objective is to avoid islands and peninsulas, and obtain two contiguous areas for assignment. The VE is in the form of tiled grid having square shaped regions and obtained as an output from our partitioning algorithm. The proposed load balancing algorithm works from a starting point, named as a root, in the set of regions to be partitioned. Keeping these restrictions in mind, the corner regions are chosen as roots. The proposed algorithm utilises a set of filters that greatly reduces the complexity and effort involved in the aggregation process.

3.1 Proposed Algorithm and Strategies

The work in this paper uses Top Left (TL) and Top Right (TR) regions of the grid being the roots scanning all possible and unique combinations. However, any combination of two consecutive corner regions can be chosen with minor changes to directions in strategies. Based on the restrictions, we have proposed four different aggregation strategies for each root that evaluate the complete set of combinations. However, different strategies are applicable based on the RSF value. If the RSF value is 2, the first two strategies for both TL and TR guarantee the examination of the entire set of unique combinations. But, the complete set of strategies is required for an RSF value 3. The proposed strategies for both TL and TR are described in Table 1.

The proposed algorithm distributes the load among two servers and therefore it maintains two sets of regions named as Aggregate1 and Aggregate2 during the aggregation process for each strategy. Aggregate1 and Aggregate2 are coloured black and grey for illustration purposes in the diagrams used in this paper. At first, Aggregate1 is initialised to the root node while Aggregate2 holds the rest of the regions. In each iteration, one region is moved from Aggregate2 to Aggregate1 according to the pattern described by a strategy. Figure 2(a) illustrates this process with a tiled grid of 4 regions for both roots. It is clear that the first two strategies cover the complete set of regular and contiguous areas while excluding non-contiguous cases such as diagonals. Further, it is observed that some strategies repeat a number of combinations. However, the proposed algorithm exploits a number of techniques to reduce the scanning process. Figure 2(b) provides unique combinations and the repetitions skipped by the algorithm are marked as R. There are 6 unique combinations in this case, and avoiding repetitions greatly reduces processing burden as illustrated later.

Root	Strategy No.	Aggregation strategy	Applied for RSF value of
Top Left Region (TL)	1	Left to Right, Row by Row (LRRows)	Both 2 and 3
	2	Top to Bottom, Column by Column (TBColumns)	Both 2 and 3
	3	Left to Right and Top to Bottom (LRaTB)	3 only
	4	Left to Right and Top to Bottom with Diagonal Region (LRTBwDR)	3 only
Top Right Region (TR)	1	Right to Left, Row by Row (RLRows)	Both 2 and 3
	2	Top to Bottom, Column by Column (TBColumns)	Both 2 and 3
	3	Right to Left and Top to Bottom (RLaTB)	3 only
	4	Right to Left and Top to Bottom with Diagonal Region (RLTBwDR)	3 only

Table 1. Roots and their corresponding aggregation strategies

Figures 3(a), 3(b), 3(c), and 3(d) illustrate the proposed aggregation strategies for root TL and a tiled grid of 9 regions. The LRRows strategy examines possible combinations by visiting the regions row by row from left to right. It gives a total of 8 unique combinations as shown in figure 3(a). Similarly, TBColumns shown in figure 3(b) performs scanning column by column from top to bottom yielding 6 unique combinations. The RLRows and TBColumns strategies for root TR inspect the regions row by row and column by column from right to left correspondingly. The last two strategies scan only the first row and column compared with the first two visiting every row and column in their corresponding order. The LRaTB strategy reads a single region from row and column in turn as shown in figure 3(c) giving a total of 5 but only 2 unique combinations. The RLaTB strategy is similar to LRaTB but read the regions right to left and top to bottom in turn starting with TR. It only defines 2 unique combinations with a number of repeated ones. The fourth strategy (LRTBwDR) is similar to the third strategy

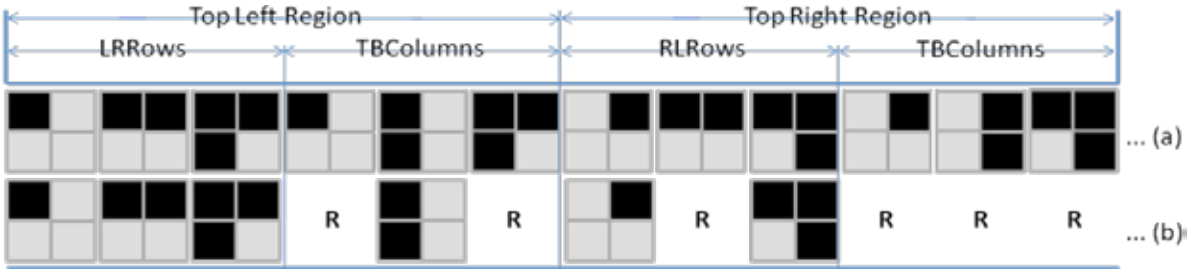

Figure 2. Illustration of Strategy 1 and 2 for RSF value of 2 and roots TL and TR

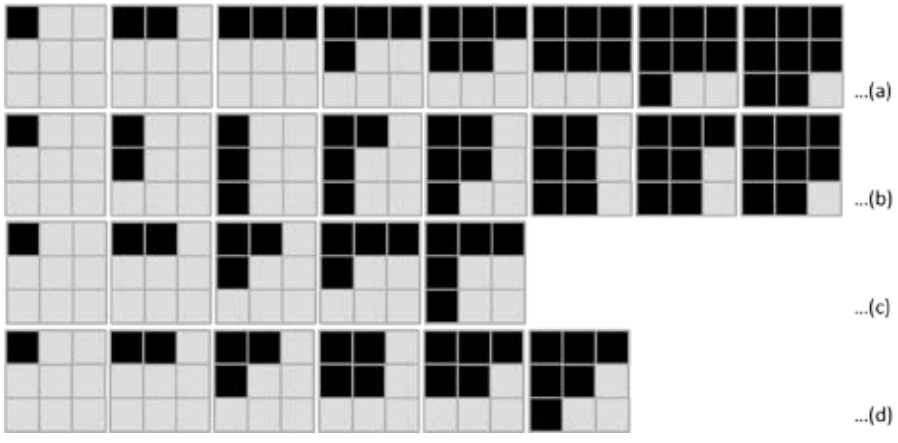

Figure 3. Illustration of Strategies 1, 2, 3 and 4 for root TL

but it reads the adjacent diagonal region of the root when the two regions sharing boundaries with root are already in an aggregate. This is depicted in figure 3(d) and also reads 2 unique combinations. The proposed mechanism cuts-off a number of repetitions if the aggregates are initialised to regions of the first unique combination. The strategies for the root TR can be visualised with the same approach.

Our partitioning algorithm splits a region into more regions compared with the existing mechanisms, and therefore we have proposed aggregation of regions to minimise resource utilisation and communication. However, it requires significant effort to balance the load while maintaining contiguous and regular areas. The following filters are proposed that greatly reduce the aggregation process and complexity:

- Regions without Players Check: adds a region to an aggregate without players per iteration until a region with players is added to the aggregate.
- Repetition Check: skips the combinations already examined by any of the previous iteration of a strategy.
- Strategy Check: passes over the remaining combinations for a strategy, if they can not achieve better distribution of the load.
- Uniform Load Check: terminates the program, if it achieves the uniform load at any point during the execution.
- Best Aggregates Check: determines whether the new combination of aggregates is better than the previous combination or not.

The above mentioned filters are embedded in the proposed load distribution technique presented in Algorithm 1. It takes Roots [], SearchStrategies, Players, RSF, and a Regional Players Matrix (RPM) holding player density for the regions as input. It reads the corresponding strategies for an RSF value and applies them in turn. The algorithm performs an exhaustive search to achieve fair load distribution but minimises the scanning process by avoiding repetitions. It investigates the unique combinations and terminates if it finds uniform load at any point. It also skips the remaining combinations of a strategy if the observed difference for a combination is greater than the difference for the previous combination, because the rest of the combinations for this strategy are unable to further improve the distribution already achieved. The proposed algorithm maintains BestAggregate1, BestAggregate2 and BestDifference where BestDifference is used for deciding best aggregates and the rest hold distinct sets of regions whose difference is the minimum one. It results in two sets of regions (BestAggregate1 and BestAggregate2) and the split initiated server then assigns one set of regions to the child server while keeping the other for its own processing. In worst cases, it must examine the entire set of possible and favourable combinations for the corresponding strategies. The favourable combinations are those that further reduce the difference and therefore must be examined.

3.2 Motivation and Discussion

This work considers two roots and eight different aggregation strategies for load balancing while obtaining contiguous and regular areas for assignment. The basic reasons for the selection of the proposed roots and strategies include content visibility and disconnection, communication, player migration and implementation concerns. These issues are relevant to each other and explained with the help of figures 4(a), 4(b), 4(c), 4(d), and 4(e) illustrating five different cases. Contents in these cases are divided into irregular and non-contiguous areas that increase communication traffic and player migrations between servers. Moreover, these increase implementation complexity as a server maintains isolated sets of regions. A player requiring content information from a neighbouring region needs to pass a request to the server

```
Data: Players, Roots [], SearchStrategies, RSF, RegionalPlayersMatrix
Result: BestAggregate1, BestAggregate2
Flag = 0;
for i = 1 to Max (Roots[]) do
    BestDifference = Players;
    Strategies [] = Read corresponding strategies for the root considering the RSF
    Value;
    for j = 1 to Max (Strategies[]) do
        Initialize Aggregate1 and Aggregate2;
        Difference = Players;
        while (All combinations are not visited) do
            Determine Aggregate1 and Aggregate2 for each successive step as
            described by the strategy;
            if (Combination not yet visited) then
                Compute AggregateTotal1 and AggregateTotal2;
                if (absolute(AggregateTotal1 - AggregateTotal2) > Difference) then
                    break;
                else
                    Difference = absolute (AggregateTotal1 - AggregateTotal2);
                end
                if (Difference < 2) then
                    Flag = 1;
                    BestAggregate1 = Aggregate1;
                    BestAggregate2 = Aggregate2;
                    BestDifference = Difference;
                    break;
                else
                    if (Difference >= BestDifference) then
                        go to the next combination;
                    else
                        BestAggregate1 = Aggregate1;
                        BestAggregate2 = Aggregate2;
                        BestDifference = Difference;
                    end
                end
            end
        end
        if (Flag == 1) then
            break;
        end
    end
    if (Flag == 1) then
        break;
    end
end
```

Algorithm 1. The Proposed Load Distribution Approach

and in case a player moves, they experience a number of disconnections and connections between the same servers.

To justify the choices described above, we propose a simple communication model showing how these excluded cases introduce extra burden in terms of communication, implementation, and user migrations. This model uses three metric parameters for an assumed and restricted number of cases: total number of shared boundaries between aggregates and the interaction capacity among players of regions in different aggregates, total number of isolated regions to manage, and number of connections and disconnections for a user. The cases discussed in (Farooq and Glauert, 2009b) are used for evaluation and comparison purposes with two additional diagonal cases. The diagonal case is the only excluded case for a tiled grid of 4 regions and is discussed for fair evaluation. This model assumes that a player of a region interacts with upto 4 neighbouring regions. Further, consider the following example mobility patterns for a player:

- **Case1:** A player at the top left region moving row-wise with an alternate left to right and right to left pattern visiting every region.
- **Case2:** A player at top left region moving column-wise with an alternate top to bottom and bottom to top pattern visiting every region.

The excluded cases are compared with equivalent aggregates having maximum number of possible regions and actual scenarios might achieve better results. The evaluation results are provided in Table 2 showing that these cases share more boundaries and therefore increase communication among inter-server regions in case of communication. The proposed mechanism significantly reduces communication and interaction compared with the excluded cases. The excluded cases greatly increase the implementation complexity by managing different isolated areas than the equivalent aggregates defined by the proposed mechanism. Further, for the selected mobility patterns, the number of connections and disconnections in excluded cases are more than the aggregates achieved by the proposed strategies except the cases shown in figure 4(a) and 4(c). These achieve a slightly better number of connection/disconnection but share more external regions and require implementation of more isolated regions. This communication model with a small number of cases shows that the proposed algorithm reduces complexity and greatly reduces communication and implementation cost while achieving load as balanced as possible. This is simply a justification for excluding the cases that will have bad communication behavior.

4. Experimental Results

In order to evaluate the flexibility of the proposed algorithm for small, medium, and large scale VEs, we extend the set of our previous experiments (Farooq and Glauert, 2009b) with three additional cases of larger dimensions. A VE for each case is represented by an n*n matrix. The dimension of each case is taken as a multiple of 6 that can be used with RSF values

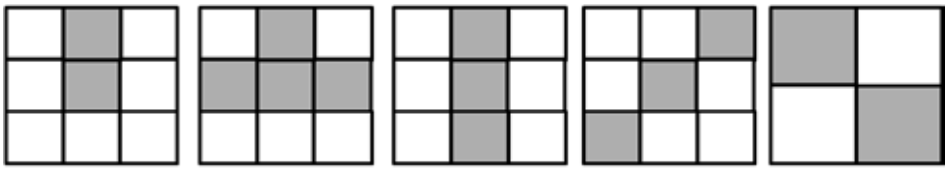

Figure 4. Cases excluded by our proposed mechanism

Serial No.	Case	Total number of shared boundaries /Interacting external regions	Total number of isolated regions	Total number of user connections/ disconnections Case1/Case2
1	Figure 4(a)	5	2	4/2
	Proposed equivalent aggregates	4	2	3/3
2	Figure 4(b)	7	4	4/6
	Proposed equivalent aggregates	4	2	3/3
3	Figure 4(c)	6	3	6/2
	Proposed equivalent aggregates	4	2	3/3
4	Figure 4(d)	8	5	6/6
	Proposed equivalent aggregates	4	2	3/3
5	Figure 4(e)	4	4	3/3
	Proposed equivalent aggregates	2	2	2/2

Table 2. Evaluation Summary of the Communication Model

Case	Dimension of the VE Matrix	MSC	Experiment Number	Distribution	RSF	VE Regional Players (RPM)
1	24*24	140	1	Uniform	2	39 26; 31 45
			2	Uniform	2	29 37; 40 35
			3	Hot Spot	3	63 30 0; 32 16 0; 0 0 0
			4	Hot Spot	3	0 0 0;0 14 31; 0 32 64
2	60*60	850	1	Uniform	2	340 143; 165 203
			2	Uniform	2	254 263; 163 171
			3	Hot Spot	3	0 180 386; 0 93 192; 0 0 0
			4	Hot Spot	3	0 0 0; 189 78 0; 393 191 0
3	90*90	1800	1	Uniform	2	395 428; 341 637
			2	Uniform	2	388 413; 427 573
			3	Hot Spot	3	744 439 0; 397 221 0; 0 0 0
			4	Hot Spot	3	0 0 0; 455 198 0; 872 276 0

Table 3. Assumption and Parametric values for the Illustrations

of both 2 and 3. The values of MSC in these cases are also greater than for the previous experiments. We have performed four experiments for each case with different uniform and hot spot distributions. Table 3 summarises the assumptions and parametric values for each experiment of the corresponding case including dimension of the matrix representing a VE, MSC, player distribution, RSF, and a matrix representing regional players for a VE described as Regional Players Matrix (RPM) holding the tiled grid. The VEs of different dimensions and player distributions are considered for fair results. The values of RSF and RPM are obtained from our proposed partitioning algorithm that takes a VE as an input. A server triggers the partitioning algorithm, when the player density exceeds the MSC in each experiment. Therefore, the total number of players initiating a split is one more than the MSC.

The proposed load balancing algorithm takes RPM as an input. The roots and their corresponding strategies are applied in the order of presentation and the summary of the step by step illustrations is presented in Table 4. The algorithm uses a number of

Case/ Experiment Number	Root	Strategy	BestAggregate1	BestAggregate2	Difference b/w Aggregates (Absolute)
			R_{11}	R_{12}, R_{21}, R_{22}	39-102=63
		LRRows	R_{11}, R_{12}	R_{21}, R_{22}	65-76=11
1/1	TL	TBColumns	**R_{11}, R_{21}**	**R_{12}, R_{22}**	**70-71=1**
			R_{11}	R_{12}, R_{21}, R_{22}	29-112=83
		LRRows	R_{11}, R_{12}	R_{21}, R_{22}	66-75=9
1/2	TL	TBColumns	**R_{11}, R_{21}**	**R_{12}, R_{22}**	**69-72=3**
1/3	TL	LRRows	**R_{11}**	**R_{12}, R_{13} R_{21},..,R_{23} R_{31},..,R_{33}**	**63-78=15**
			R_{11},.., R_{13} R_{21}, R_{22}	R_{23} R_{31},.., R_{33}	14-127=113
			R_{11},.., R_{13} R_{21},.., R_{23}	R_{31},.., R_{33}	45-96=51
1/4	TL	LRRows	**R_{11},.., R_{13} R_{21},.., R_{23} R_{31},R_{32}**	**R_{33}**	**77-64=13**
			R_{11}	R_{12}, R_{21}, R_{22}	340-511=171
2/1	TL	LRRows	**R_{11}, R_{12}**	**R_{21}, R_{22}**	**483-368=115**
			R_{11}	R_{12}, R_{21}, R_{22}	254-597=343
	TL	LRRows	R_{11}, R_{12}	R_{21}, R_{22}	517-334=183
2/2		TBColumns	**R_{11}, R_{21}**	**R_{12}, R_{22}**	**417-434=17**
			R_{11}, R_{12}	R_{13} R_{21},..,R_{23} R_{31},..,R_{33}	180-671=491
	TL	LRRows	R_{11},.., R_{13}	R_{21},..,R_{23} R_{31},..,R_{33}	566-285=281
2/3	TR	RLRows	**R_{13}**	**R_{12}, R_{11} R_{23},..,R_{21} R_{33},..,R_{31}**	**386-465=79**
			R_{11},.., R_{13} R_{21}	R_{22}, R_{23} R_{31},..,R_{33}	189-662=473
			R_{11},.., R_{13} R_{21}, R_{22}	R_{23} R_{31},..,R_{33}	267-584=317
	TL	LRRows	R_{11},.., R_{31}	R_{12},.., R_{32} R_{13},.., R_{33}	582-269=313
2/4	TR	RLRows	**R_{13},..,R_{11} R_{23},..,R_{21} R_{33}, R_{32}**	**R_{31}**	**458-393=65**
			R_{11}	R_{12}, R_{21}, R_{22}	395-1406=1011
3/1	TL	LRRows	**R_{11}, R_{12}**	**R_{21}, R_{22}**	**823-978=155**
			R_{11}	R_{12}, R_{21}, R_{22}	388-1413=1025
		LRRows	R_{11}, R_{12}	R_{21}, R_{22}	801-1000=199
3/2	TL	TBColumns	**R_{11}, R_{21}**	**R_{12}, R_{22}**	**815-986=171**
3/3	TL	LRRows	R_{11}	R_{12}, R_{13} R_{21},..,R_{23} R_{31},..,R_{33}	744-1057=313
			R_{11},.., R_{13} R_{21}	R_{22}, R_{23} R_{31},..,R_{33}	455-1346=891
	TL	LRRows	R_{11},.., R_{13} R_{21}, R_{22}	R_{23} R_{31},..,R_{33}	653-1148=495
3/4	TR	RLRows	**R_{13},..,R_{11} R_{23},..,R_{21} R_{33}, R_{32}**	**R_{31}**	929-872=57

Table 4. Summary of the Experiments

filters reducing the overall aggregation process. It skips repeated patterns and terminates if uniform load is achieved at any stage. If the difference between AggregateTotal1 and AggregateTotal2 is less than 2, it declares the achievement of uniform load. It also skips the remaining combinations for a strategy when absolute difference for a combination is greater than the difference for the previous combination, because the difference between the remaining combinations increases in further iterations. In the case of regions without players, it transfers regions from Aggregate2 to Aggregate1until a region with players in moved. If uniform load cannot be achieved, it examines the entire set of favourable combinations and determines the best possible aggregations. It assures that the best possible load distribution with contiguous patterns is achieved. It is worth mentioning that better load distribution may be possible, but not with one of our chosen aggregates. The details of the experiments are omitted and only those events are recorded that achieve better distribution of the load than the previous one. It is clear that hot spot scenarios split a VE into 9 regions compared with the uniform distribution splitting a region into 4 regions.

The VE matrix size of 24*24 is used for case 1, which is populated with different uniform and hot spot player distributions. Experiment 1 achieves uniform load on the second iteration of the second strategy for root TL by yielding aggregate totals of 70:71 for BestAggregate1:BestAggregate2. Experiment 2 of case 1 obtained the best possible load on second iteration of second strategy for root TL by getting best aggregates of 69:72. In this case, the exhaustive search was applied but the filters greatly reduced the aggregation process. The hot spot scenarios considered in experiment 3 and 4 have similar outcome to experiment 2 where the best possible load was achieved on the first and last iteration of the first strategy for root TL. They obtained best aggregates of 63:78 and 77:64 respectively. Case 2 employs a matrix size of 60*60 to represent VEs for the next four experiments based on uniform and hot spot distributions. Experiment 1 and 2 both got the best possible load distribution during the first and second strategies of root TL consecutively. These output the load distribution of 483:368 and 417:434 for BestAggregate1:BestAggregate2 in order. It is clear from experiment 2 that a uniform load is possible with aggregates based on diagonal regions. However, the proposed strategies exclude this as an odd case. Experiment 3 and 4 considers hot spots and obtain RPM of 9 regions as an input. Both achieve the best possible load during the first strategy for root TR. The best aggregates are achieved with load distribution of 386:465 and 458:393 consecutively. Case 3 with a VE matrix of size 90*90 also uses the pattern of previous cases. No uniform load is possible in the considered set of experiments and therefore perform exhaustive search to achieve load as balanced as possible. Their results are similar for both uniform and hot spot distributions and yield load distribution of 823:978, 815:986, 744:1057, and 929:872 for BestAggregate1:BestAggregate2 correspondingly.

The results of this extended set of experiments show that the proposed load balancing algorithm is flexible and can be used with small, medium and large scale VEs. It is clear that some excluded cases can balance the load better than the proposed strategies, but significantly increase communication between servers and complexity for handling isolated areas for an aggregate. The aggregations are further reduced for an RPM matrix of 9 regions.

5. Conclusions

This paper presented our proposed load distribution algorithm based on the aggregation of regions for the minimisation of resource utilisation and communication. It performs exhaustive scanning for achieving the load as balanced as possible while maintaining contiguous and regular areas for assignments. In worst cases, it examines the entire set of favourable combinations, but the proposed filters greatly reduce the aggregation process. It constitutes an integral part of our proposed JoHNUM infrastructure for the development of scalable and consistent virtual worlds. The proposed algorithmic strategies and excluding irregular and non-contiguous areas are justified with a simple communication model based on three different metrics. It demonstrated that the communication and implementation cost is significantly reduced. The algorithm is illustrated with an extended set of different cases and experiments that show that the proposed algorithm works well with small, medium, and large scale VEs. However, uniform load distribution is not always possible due to player distribution, and the proposed split and aggregation strategies.

References

[1] Balan, R. K., Ebling, M., Castro, P., Misra, A. (2005). Matrix: Adaptive middleware for distributed multiplayer games. *In:* Proc. of the Middleware 2005, ser. Lecture Notes in Computer Science, p. 390–400. New York: Springer Berlin/ Heidelberg.

[2] Burlamaqui, A. M., Oliveira, M. A. M., Goncalves, A. M. G., Lemos, G., Oliveira, J. C. De. (2006). A scalable hierarchical architecture for large scale multi user virtual environments. *In:* Proc. of the IEEE International Conference on Virtual Environment, Human Computer Interfaces and Measurement Systems, p. 114–119. La Coruna, Spain.

[3] Chertov, R., Fahmy, S. (2006). Optimistic load balancing in a DVE. *In:* Proc. of the 16th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'06), p. 1-6. Newport, Rhode Island.

[4] Dahmann, J. S., Fujimoto, R. M., Weatherly, R. M. (1997). The department of defence high level architecture. *In:* Proc. of the 1997 winter simulation conference (WSC'97), p. 142–149. Washington, USA.

[5] Farooq, U., Glauert, J. (2009a). Joint Hierarchical Nodes based User Management (JoHNUM) infrastructure for the development of scalable and consistent virtual worlds. *In:* Proc of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DSRT'09), pages 105–112. Singapore.

[6] Farooq, U., Glauert, J. (2009b). ARA: An aggregate region assignment algorithm for resource minimisation and load distribution in virtual worlds. *In:* Proc. of the 1st International Conference on Networked Digital Technologies (NDT'09), pages 404-410. Ostrava, Czech Republic.

[7] Fujimoto, R. M. (2001). Parallel and distributed simulation systems. *In:* Proc. of the 2001 winter simulation conference (WSC'01), p. 147–157. Arlington, USA, Dec.

[8] Fullford, D. A. (1996). Distributed Interactive Simulation: Its past, present, and future. *In:* Proc. of the 1996 winter simulation conference (WSC'96), p. 179–185. Washington, USA, 1996. IDC (2003). Butterfly.net: Powering next generation gaming with on-demand computing. An IDC e-business case study, Butterfly.net Inc.

[9] Kim, B. K., You, K. S. (2006). A hierarchical map partition method in MMORPG based on virtual map. *In:* Proc. of the Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops, ser. Lecture Notes in Computer Science, p. 813–822. New York: Springer-Berlin/Heidelberg.

[10] Lee, D., Lim, M., Han, S. (2007). ATLAS: A scalable network framework for distributed virtual environments. *Presence: Teleoperators and Virtual Environments* 16 (2) 125–156.

[11] Lee, K., Lee, D. (2003). A scalable dynamic load distribution scheme for multi server distributed virtual environment systems with highly skewed user distribution. *In:* Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST'03), p. 160–168. Osaka, Japan.

[12] Lui, J. C. S., Chan, M. F. (2002). An efficient partitioning algorithm for distributed virtual environment systems. *IEEE Transaction on Parallel and Distribution Systems* 13 (3) 193–211.

[13] Morillo, P., Orduna, J. M., Duato, J. (2006). A scalable synchronization technique for distributed virtual environments based on networked server architecture. *In:* Proc. of the International Conference on Parallel Processing Workshops (ICPP'06), p. 74–81. Columbus, OH, USA.

[14] Ng, B., Li, F. W. B., Lau, R. W. H., Si, A., Siu, A. (2003). A performance study on multiserver DVE systems. *Information Sciences* 154 (-) 85–93.

[15] Ng, B., Si, A., Lau, R. W. H., Li, F. (2002). A multi-server architecture for distributed virtual walkthrough. *In:* Proc. of the ACM Symposium on Virtual Reality Software and Technology *(VRST'02)*, p. 163–170. Hong Kong, China.

[16] Oliveira, J. C. de, Georganas, N. D. (2003). VELVET: An adaptive hybrid architecture for very large virtual environments. *Presence: Teleoperators and Virtual Environments* 12 (6) 555–580.

[17] Ondrejka, C. R. (2004). A piece of place: Modeling the digital on the real in Second Life. SSRN eLibrary. [Online]. Available: http://ssrn.com/paper=555883/

[18] Rosedale, P., Ondrejka, C. R. (2003). Enabling player created online worlds with grid computing and streaming. Gamasutra Resource Guide. [Online]. Available:http://www.gamasutra.com/resourceguide/20030916/rosedale 01.shtml/

[19] Second Life (2009). [Online]. Available: http://www.secondlife.com/

[20] Second Life Grid (2009). [Online]. Available: http://secondlifegrid.net/

[21] Shirmohammadi, S., Kazem, I., Ahmed, D. T., El-Badaoui, M., Oliveira, J. C. De. (2008). A visibility- driven approach for zone management in simulations.

[22] Simulation 84 (5) 215–229.

[23] Vleeschauwer, B. D., Bossche, B. V. D., Verdickt, T., Turck, F. D., Dhoedt, B., Demeester, P. (2005). Dynamic microcell assignment for massively multiplayer online gaming. *In*: Proc. of the 4th ACM SIGCOMM workshop on Network and System Support for Games, p. 1–7. Hawthorne, NY.