Available online at www.sciencedirect.com

ScienceDirect

Simulation Modelling Practice and Theory 14 (2006) 1022–1042

SIMULATION MODELLING PRACTICE AND THEORY

www.elsevier.com/locate/simpat

Dynamically adaptive partition-based interest management in distributed simulation

Bora İ. Kumova [*]

Department of Computer Engineering, İzmir Institute of Technology, Urla, 35430 İzmir, Turkey

Received 1 December 2004; received in revised form 7 July 2006; accepted 8 July 2006
Available online 7 September 2006

Abstract

Performance and scalability of distributed simulations depends primarily on the effectiveness of the employed interest management (IM) schema that aims at reducing the overall computational and messaging effort on the shared data to a necessary minimum. Existing IM approaches, which are based on variations or combinations of two principle data distribution techniques, namely region-based and grid-based techniques, perform poorly if the simulation develops an overloaded host. In order to facilitate distributing the processing load from overloaded areas of the shared data to less loaded hosts, the partition-based technique is introduced that allows for variable-size partitioning the shared data. Based on this data distribution technique, an IM approach is sketched that is dynamically adaptive to access latencies of simulation objects on the shared data as well as to the physical location of the objects. Since this re-distribution is decided depending on the messaging effort of the simulation objects for updating data partitions, any load balanced constellation has the additional advantage to be of minimal overall messaging effort. Hence, the IM schema dynamically resolves messaging overloading as well as overloading of hosts with simulation objects and therefore facilitates dynamic system scalability.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Distributed simulation; Interest management; Load distribution; Scalability; Adaptability

1. Introduction

Establishing and maintaining network connections for the purpose of satisfying the mutual information requirements of the participants of a distributed system is called interest management (IM) [28]. Synonymous denominations are relevance filtering [11], proximity or collision detection [34], data subscription [5], data distribution management (DDM) [4] or data dissemination management [15]. The overall objective of IM is to provide applications a scalable environment in the presence of restricted resources, like local processing power, memory capacity and network bandwidth. The goal is to deliver the simulation objects the exact

[*] Tel.: +90 232 7506326; fax: +90 232 7506505.
E-mail address: borakumova@iyte.edu.tr.
URL: www.iyte.edu.te/~borakumova.

amount of data they are interested in, no more and no less data. This implies sender-side filtering, which ideally avoids sending unnecessary messages over the network. However without IM, in the worst case, every message had to be broadcast, which would not scale in the presence of network bandwidth.

An early technique for reducing network messaging is dead reckoning, in which the environment broadcasts update information on the shared data, and interested objects can predict future updates based on past updates, until the next broadcast [12]. It is a common technique in distributed simulation, such as in distributed interactive environments (DIS) [9] or collaborative virtual environments (COVEN) [29].

A more effective technique for sender-side filtering is to maintain updates and subscriptions on/to shared data and to deliver updates only to those objects that have subscribed to a particular data.

In the distributed virtual reality platform specification high-level architecture (HLA) DDM [10] shared data can be mapped onto integer values and interest on the data represented by update and subscription intervals or regions within the value range of the integer. An efficient interest matching algorithm then calculates the intersections between update and subscription regions on each integer [31].

Another technique is to sub-divide the shared data into a grid and to assign the grid cells to nodes. A move of a region between two grids then needs to be sent only to neighbouring grids, which avoids broadcasting.

It has been shown that evaluating IM combined with synchronisation and/or load balancing promises further performance improvements, as information about dependent interests and the states of the LPs may be utilised to vary synchronisation and/or load balancing policies dynamically. For instance, combining a load balancing algorithm that implements simulated annealing with conservative synchronisation [2] or combining load balancing with the optimistic time warp synchronisation [16] in Georgia Tech Time Warp (GTW) [7]. In SPEEDES the environment is sub-divided into grids, where a grid cell contains co-located sensing areas of the simulation objects [34]. Parallel processing and load balancing is facilitated via the cells. An approach for dynamically identifying clusters of interest regions is the concept of the spheres of influence (SoI) of an event. A SoI is the set of interest regions immediately effected by an event of a logical process (LP). Clusters of SoIs therefore imply clusters of LPs. Based on this technique a hierarchical cluster management is proposed [23]. The concept SoI is comparable with the concept focus [1] of an LP. Clustering interest regions by grouping common user interests is employed in distributed virtual environments (DVE) [8].

While the above techniques aim at reducing the messaging load on the network, hardware multicasting can reduce the computational load on a node, by delegating message duplication to the related hardware unit of the network. Assigning each intersection region or grid a multicast group [27,4] can reduce the computational load for replicating a data update propagation.

At the network level, yet other filtering strategies can reduce messaging overhead. For instance utilising grids for multicasting [3] or predicting continuous value changes of numerals, quantising them and sending only the significant bits [38].

A suitable combination of these techniques can significantly improve the performance of IM by facilitating load distribution, parallel processing and scalability in distributed simulation [4,18].

Principally similar techniques are employed for IM in distributed simulation of intelligent systems [14,36,32,22]. Intelligent systems possess capabilities that distinguish them from non-autonomous objects, such as non-deterministic behaviour of an agent or the multi-agent system, frequent changes of interests in the worst case in every state change, or evolution of the behaviour by adapting to the environment. Furthermore, some capabilities require high-performance computation with real-time response, such as distributed planning [17], which could be implemented as fine-grained simulation that is invoked whenever requested from within the coarse-grained simulation, such as simultaneous planning [36]. All these properties influence the performance and require the simulation platform to be highly adaptive and be able to dynamically distribute the load and scale, in order to continue with equal performance.

Dynamically adaptive partition-based DDM is a new approach for load sensitive distributed simulation that assembles and extends the above ideas. The approach is introduced here in the context of the distributed simulation platform adaptive parallel discrete event simulator (APDES) [20] for non-deterministic applications. It possesses the following capabilities:

- Matching intersecting multi-dimensional interest regions.
- Identifying and dynamically adapting multi-dimensional clusters within the multi-dimensional state variable (SV) space.
- Adapting the boundaries of a multi-dimensional SV partition to the encapsulated multi-dimensional region cluster.
- Migrating a SV partition to a less loaded neighbouring host and splitting/merging the SV partition as well as the included interest regions.
- Migrating an LP to its interest region with the highest access cost for this LP.
- Individual synchronisation of LPs, depending on their interest intersections, which allows them to continue as fine-grained distributed simulations within the overall coarse-grained distributed simulation.

These capabilities address the specific requirements for the distributed simulation of intelligent systems [21]. APDES combines some of the above discussed techniques by addressing particularly scalability through minimising the overall messaging effort against minimising the local processing overhead at each host [19].

After a brief discussion of the major concepts of interest management, the principle structure of the new IM model is introduced. Thereafter, the implementation of the model in the ADPES prototype is sketched. Finally, the work is summarised and concluded with directions for the future.

In this text we shall use interchangeable terms for SV, such as interest, shared data and data dimension and refer to as *d*. Interchangeable terms for the logical process (LP) shall be federate and simulation object, although sometimes an LP may represent several simulation objects.

2. Interest management

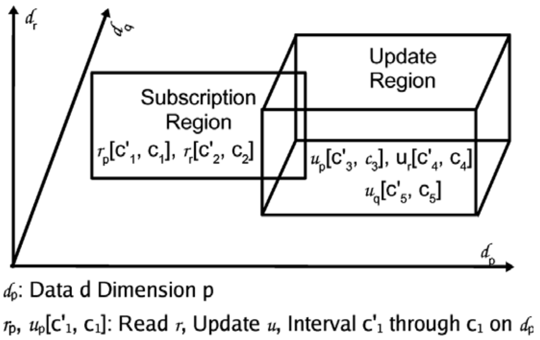
The major factors for high performance and scalability of interest management (IM) are the effective representation of interest in the form of shared data and its effective distribution.

2.1. Shared data

In discrete event simulation, a user model is mapped onto a set of logical processes (LP) and a set of state variables (SV), which can be represented as integer values. LPs produce a set of events that may change the SVs. Access to the shared data is granted LPs through publishing update regions on SVs or through subscribing to regions. An update may be interpreted as a modification of the application environment in a particular region and a read as sensing within a region (Fig. 1). Depending on SV semantics, a particular property of the user model may be mapped onto one or more SVs.

2.2. Representation of interest

Interest on a particular subject can be projected onto a linear finite dimension d_i, also known as scalar.



d_r
d_q
d_p
Subscription Region
$r_p[c'_1, c_1], r_r[c'_2, c_2]$
Update Region
$u_p[c'_3, c_3], u_r[c'_4, c_4]$
$u_q[c'_5, c_5]$
d_p: Data d Dimension p
$r_p, u_p[c'_1, c_1]$: Read r, Update u, Interval c'_1 through c_1 on d_p

Fig. 1. Sample update/subscription regions in the multi-dimensional hyperspace as of HLA DDM 1516.

In addition to interest on our physical world that consists of the dimensions length, breadth, depth, time, radio waves, ultraviolet waves, etc., further interest may be expressed in metaphysical dimensions. For instance, interest in the virtual objects of a specific knowledge area or in their properties. Since a given user model may consist of a large number of metaphysical objects, a scalable and efficient representation for interest is required. One such model is the multi-dimensional hyperspace of the HLA DDM 1516 [30] standard (Fig. 1). In the course of a simulation, objects of the application may update regions, subscribe to regions, modify their region boundaries or withdraw from such declarations. An interval on a scalar represents a region of interest and an intersection represents matching interests. Depending on the semantics of an application domain, inter-relationships may be declared between two or more dimensions, enabling for multi-dimensional regions. Consequently, the intersections of multi-dimensional regions may be multi-dimensional as well.

2.3. Interest management tasks

Common IM tasks are interest specification, interest matching, data distribution, synchronisation and load distribution. The first two are concerned with optimal calculation of interests on the shared data with minimal local processing effort. Whereas data distribution is concerned with maintaining consistency of the shared data with minimal messaging effort. Synchronisation is necessary for guaranteeing consistent shared data between global time advances. Load distribution helps resolving local bottlenecks and facilitates scalability. Synchronisation and load distribution may be seen as subtasks of data distribution, as they share some objectives. However their primary objectives are distinct. That will be discussed in the next chapter.

2.3.1. Interest specification

The specification of interest bears the greatest potential for filtering data. For instance, selecting for each simulation object a subset from the data dimensions of the application and declaring regions on the dimensions can reduce each simulation object's interest space dramatically within the interest hyperspace of the whole application. Further filtering can be achieved by specifying a list of classes and subclasses of simulation objects, in which a particular LP is interested in. This aspect however is less suitable for application domains with non-deterministic simulation objects, as unpredictably changing or evolving interests may require sharing data with previously uninterested simulation objects. Therefore, in applications with non-deterministic simulation objects, the process of filtering simulation objects is non-deterministic as well. Hence a non-deterministic simulation object should decide on filtering other simulation objects rather dynamically, as soon as its interests change.

2.3.2. Interest matching

Following interest matching alternatives are known:

- *Region:* An LP declares an interest area on a SV. Interests of different LPs are matched by calculating intersections between the areas that were declared on a particular SV.
- *Grid:* No area of interest of an LP is explicitly declared. Matching interest is implied by co-location of LPs within the same grid cell.
- *Hybrid of grid and region:* In case of large grid cell size, relative to the size of the LP's interest area, LPs within the same cell may explicitly declare their interest area. Matching interests could then be calculated by intersecting the interest areas of all LPs that share the same cell.
- *Partition:* Is like a grid, but with variable size.
- *Hybrid of partition and region:* Like the hybrid of grid and region, but with variable partition size.

In case of regions, the upper bound time complexity of the HLA DDM 1516 model is $O(n^2)$ for comparing in the worst case all n subscription regions with all n update regions of the multi-dimensional data space. A reduction to $O(n \log n)$ is possible with recursive interest matching algorithms [30,31].

The partition technique and the hybrid technique of partition and region are the new interest management approaches that will be discussed in conjunction with the APDES implementation.

2.3.3. Data distribution

The last filtering level in IM is concerned with distributing the shared data and propagating updates with minimal messaging effort. Although efficient data distribution techniques, such as data replication and fragmentation are common in distributed simulation, their optimal management in terms of synchronisation and load distribution for achieving scalable simulations and optimal performance is still matter of research. The effectiveness of the data transmission can be measured with the ratio $E = S/I$, where S is sent and I is interested data [27]. Usually $E > 1$ holds, due to inaccurate storage of interest and redundant network transmission of data [26].

2.3.4. Synchronisation

Shared data in coarse-grained distributed simulation need to be synchronised in a way, such that the system can respond to all user transactions in real-time and synchronous with the wall-clock time of all users. This requirement implies global conservative synchronisation of all peers. Logically independent LPs however may be synchronised optimistically, in order to enable them to advance as fast as possible, like in fine-grained distributed simulation.

2.3.5. Load distribution

Orthogonal to the objective of minimising the data network transfer is the objective of minimising the processing overload at each peer. Since, administrative processing as well as application processing increases the system load, distributing the load can reduce the risk for bottlenecks, hence enable for continuous system performance. Administrative load can be distributed by fragmenting the shared data to the hosts and performing interest matching independently. Alternatively by delegating the replication of a message from the host to a multicasting unit of the network. Application load can be distributed through migration of LPs.

2.4. Interest management objectives

User requirements imply a multi-objective optimisation problem with conditional priorities between the objectives.

2.4.1. Primary objective

Real-time response and scalability are basic user requirements for distributed systems. Since IM enables scaling the shared data, improvement suggestions to IM approaches are dominated by performance and scalability issues [18].

2.4.2. Secondary objectives

In order to achieve these primary objectives though a concerted application of the above tasks, following secondary objectives need to be resolved as a multi-objective optimisation problem:

- Minimising network bandwidth usage through interest specification and interest matching.
- Minimising network latency through data distribution.
- Minimising total number of data replica.
- Minimising processing overload at each peer through load distribution.
- Minimising overall processing load of an application through minimising roll-backs. This implies minimising the duration of inconsistent shared data through synchronisation.

The first two objectives result from the heuristic “sender-site filtering”, which states that as much data should be filtered as possible, before sending it into the network.

2.4.3. Tertiary objectives

There is a potential for further performance improvement that is implied by the secondary objectives, which can be achieved through load-sensitive synchronisation of updates on interest regions:

- Optimistically synchronised update regions, matching regions and all directly related LPs should be considered less for migration.
- If an update region is remote, then less optimistic advance of the LP should be considered and migration of the related partition to the LP's local host should be given less priority.
- If an update region is intersecting with another, then it should be synchronised the less optimistically the higher the access frequency of the matching region.

The effectiveness of these general rules may be improved by adapting them to the related properties of a particular domain application.

3. Principle structure of the partition model

The principle idea of the new IM model is to distribute the shared data of an application over a given set of hosts, by assigning each host one partition of each data dimension and by preserving adjacency information between partitions. Since one partition has at most two adjacent partitions, changing interest region boundaries need to be communicated to at most one host, as we assume continuous value changes on SVs. Partition sizes are dynamically calculated based on emerging data clusters and distributed depending on the contained regions' access costs.

3.1. State variable conceptualisation

The following discussion summarises the SV concept of the IM model. SVs are normalised within the value range $[0,\ldots,1]$, as this facilitates homogeneous and variable-sized partitioning over varying numbers of hosts. If other value ranges are used within an application, then related normalisation functions need to be employed for bi-directionally mapping values between $[0,\ldots,1]$ and the required value range within the application. Value changes of a SV may be principally continuous or steady.

3.1.1. Continuous value changes of SVs

Values may increase or decrease only by a given granularity, which is a fraction of the normalised value range. For the purpose of increasing performance, a value may change multiple times of the granularity, but at most to the maximum extent of the local data partition size. SV partitioning is discussed below.

3.1.2. Steady value changes of SVs

Values may increase/decrease more than the local partition size in one step, for instance a "jump" within the environment. This does not make sense for the physical environment, but could be a property of some metaphysical SVs. However, for SVs with such characteristics, the application should provide a projection onto continuous SVs, for instance by projecting a steady SV onto one or more logically dependent, continuous SVs.

3.1.3. Update/subscription on/to SVs

A LP^e may declare at most one update region $u_i^e[c_1', c_1]$ and at most one subscription region $r_i^e[c_2', c_2]$ per SV d_i, where $c_1' \leqslant c_1, c_2' \leqslant c_2$ and $c_1', c_1, c_2', c_2 = [0,\ldots,1]$.

3.2. Data access evaluation

Statistics over the access to SV regions provide qualitative information about regions and enable the IM to dynamically adapt to the application behaviour, thus facilitating load distribution and scalability. Hence every access region is attributed with the following statistical values, which are additionally stored at the owner LP's site.

3.2.1. Region access frequency

In the course of a simulation the access frequency of SVs may change dynamically, depending on changing interests and interactions between LPs. The access frequency $F(a_i^e)$ of an update or subscription region

$a_i^e[c_1', c_1]$ of LP^e on SV i, gives information about how often it was accessed in the most recent time interval $[t_{0-\Delta f}, t_0]$

$$F(a_i^e) = \sum_{[t0-\Delta f]}^{[t0]} a_i^e[c_1', c_1]$$

t_0 is the last global state change and Δf the number of some most recent global state changes. Δf may be adapted to the overall region access frequency of a particular application. This information is used for optimising the distribution of the shared data and the LPs. It is provided to interrelated LPs as well, for monitoring the activities of other LPs. Hence it facilitates the dynamic adaptation of an LP to its environment.

3.2.2. *Region access latency*

The average time to access a region is updated with every access to that region with

$$L(a_i) = (L'(a_i)F(a_i) + |a_i|_{\text{time}})/(F(a_i) + 1)$$

a_i is an abbreviation for $a_i[c_1', c_1]$, L' is the previous access frequency and $|a_i|_{\text{time}}$ is the time required for the most recent access. The access latency of local regions is always set to zero. In case of remote regions, the time stamp of the incoming message is compared with local time.

3.3. *Data cluster detection*

If the access regions of some LPs are closely located in a particular value range of a SV or even overlap each other, then this area could be recognised as a data cluster on that SV. Overlapping update and association regions indicate possible communication between the related LPs. Placing data clusters together with the related LPs on the same host can therefore reduce messaging and thus bears potential for performance improvement, load distribution and scalability.

A cluster C_i is identified as an ordering relation of access region boundaries on SV d_i:

$$C_i[c_{z'1}, c_{z1}] = \{(a_i[c_{z'2}, c_{z2}], b_i[c_{z'3}, c_{z3}])|c_{z'1} \leqslant c_{z'2} \wedge c_{z'2} \leqslant c_{z2} \wedge c_{z'3} \leqslant\leqslant c_{z3} \wedge c_{z'3} \leqslant c_{z2} \wedge c_{z3} \leqslant c_{z1} \wedge c_{z'1}, c_{z1}, c_{z'2}, c_{z2}, c_{z'3}, c_{z3} = [0, \ldots, 1] \wedge i = [1, \ldots, s]\}$$

For any access region a_i, b_i, where two clusters $C_i[c_{z'1}, c_{z1}]$ and $C_i[c_{y'1}, c_{y1}]$ on the same SV d_i do not intersect, iff $c_{z1} < c_{y'1}$. The number of regions in a cluster $|C_i[c_{z'1}, c_{z1}]|_{\text{size}}$ is stored separately. The smallest possible cluster is a single region without intersections $|C_i[c_{z'1}, c_{z1}]|_{\text{size}} = 1$. Cluster detection is processed simultaneously with interest matching, as both share a similar algorithmic structure.

3.4. *Interest matching*

Matching interest is calculated by intersecting two boundaries of two access regions $a_i[c_1', c_1]$ and $b_i[c_2', c_2]$ on SV d_i, such that

$$a_i[c_1', c_1] \cap b_i[c_2', c_2] \neq \emptyset \rightarrow c_2' \leqslant c_1$$

Only c_1 and c_2' need to be compared for each region, as the boundary values c_1', c_1, c_2', c_2 are sorted. Additionally, the algorithm simultaneously calculates the clusters C_i. For this purpose, region boundaries on each SV are kept sorted in ascending order, which requires an additional computational effort. The time complexity for inserting one region into the sorted list of regions is O(log$n$).

3.5. *Basic data partitioning concepts*

Data management starts initially with Cartesian partitioning the shared data in form of a table and continuous with some primitive operations on the table.

Table 1
Host look-up table with initial assignment of data partitions to hosts

$d_{i,j}$ at h_j	h_1	...	h_k	...	h_n
d_1	$d_{1,1}$	...	$d_{1,k}$	...	$d_{1,n}$
...	...	...	...	...	...
d_s	$d_{s,1}$	...	$d_{s,k-}$	...	$d_{s,n}$

3.5.1. Initial partition distribution

A data dimension d_i is sub-divided into n equal-sized partitions $d_{i,j}$, which are referenced by the running variable j, and each partition assigned to a different host h_k of the distributed system that consists of n hosts (Table 1).

The look-up table at entry $h(i,j)$ contains the actual network address of the hosts h_k with partition $d_{i,j}$. This schema allows to find the network location of a partition $d_{i,j}$ with a single table access. Depending on the semantics of a data dimension and the location of LPs, adjacent partitions $d_{i,j}$ and $d_{i,j+1}$ may be stored at geographically close hosts h_j and h_{j+1} in the network, as region boundaries of access regions can move only to adjacent partitions.

Note that in the initial configuration, the partition index j in $d_{i,j}$ and the host index k in h_k are the same, hence $h(i,j) = h_j$. In Section 3.7, an extension to this schema is introduced with variable-size partitioning and partition migration, where the equality $h(i,j) = h_j$ does not necessarily hold any more.

3.5.2. Primitive partitioning operations

In the course of a simulation, a partition $d_{i,j}$ may be merged or split with adjacent partitions $d_{i,j-1}$ or $d_{i,j+1}$. This is exemplified below on a disengaging/engaging host, where a column j that consists of partition j on all SVs d_i with $i = [0,\ldots,s]$, is merged (Table 2) or split (Table 3) with adjacent partitions respectively.

Splitting and merging partitions is performed by default on neighbouring hosts. Since the operations are the primary means for re-distributing the shared data, they could be used for shrinking and growing the system's platform as well. Growing the system is exemplified for the case of a newly engaging host (Table 3).

Table 2
Merging data partitions from a disengaging host with neighbouring partitions

$d_{i,j}$ at h_j	...	h_{k-1}	h_k	h_{k+1}	...
d_1	...	$d_{1,k-1}$; $d_{1,k}$ $\leftarrow$	$d_{1,k}$	$d_{1,k+1}$	...
...	...	... $\leftarrow$	...	...	...
d_s	...	$d_{s,k-1}$; $d_{s,k}$ $\leftarrow$	$d_{s,k}$	$d_{s,k-+1}$	...

Table 3
Splitting data partitions and re-assigning to a new engaging host

$d_{i,j}$ at h_j	...	h_{k-1}	h_k	h_{k+1}	...
d_1	...	$d_{1,k-1}$; $d_{1,k}$	$\rightarrow$ $d_{1,k}$	$d_{1,k+1}$	...
...	...	...	$\rightarrow$ ...	...	...
d_s	...	$d_{s,k-1}$; $d_{s,k}$	$\rightarrow$ $d_{s,k}$	$d_{s,k-+1}$	...

3.5.3. Partition size

The size of data partitions are directly affected by the number of available hosts. Depending on the context of a SV d_i and performance considerations during a particular application, initially equal-sized partitions $d_{i,j}$ may become variable or zero-sized. Variable size partitioning allows adapting to clusters of intersecting access regions. A cluster that is split over more than one partition can cause additional network messaging. Zero partition size of $d_{i,j}$ indicates that host $h(i,j)$ does currently not maintain any value range of that SV, which may change during a particular application for load distribution purposes.

A performance issue is the ratio between partition length and access region length, respectively:

$$m_j = |a_i^e[c_1', c_1]|/|d_{i,j}|$$

For $m \leqslant 1$ one message is required. For $1 < m$, m split regions of a_i^e need to be maintained at m hosts, which implies an additional network overhead of $m - 1$ messages and an additional local overhead at $m - 1$ hosts for pointing to the host with the owner LP^e of a_i^e. At the site of LP^e an additional local overhead of $m - 1$ pointers to further $m - 1$ hosts is required.

3.5.4. Access region replication

For the purpose of minimising messaging effort, for each subscribed remote SV region, a local copy is maintained. The objective of IM is to minimise the total number of replica within a simulation, as each replica requires one message for synchronising with each update on the original remote data.

3.6. Multi-dimensionality of state variables and data clusters

By default all SVs are homogeneous and no semantics are defined between value ranges of a SV or between different SVs. However supporting such semantics helps identifying multi-dimensional data clusters in the context of user model semantics. Consider for instance the state variables d_p and d_r in (Fig. 2). If they are evaluated independently, then following LP clusters would result: (LP^1, LP^4) and (LP^2, LP^3) on d_p and (LP^1, LP^2, LP^4) and (LP^3) on d_r. A somehow logically combined evaluation of d_p and d_r however, for instance as physical area, apparently results in different LP clusters (Fig. 2). In this case, LP^2 does no more appear in the cluster with LP^1 and LP^4, as it is not intersecting with them on the physical area.

3.6.1. Defining compound SVs

A group of SVs $d_1, \ldots, d_\mathrm{q}$, with $1 \leqslant q \leqslant s$, where s is the total number of SVs of the application can be introduced as one compound SV $(d_1, \ldots, d_\mathrm{q})$ by declaring

$$d_{(1,\ldots,q)} = \{(d_1, \ldots, d_q) | q \leqslant s\}$$

In general, the more SVs are bound to each other by grouping relationships, the smaller the probability for involving access regions incorrectly in multi-dimensional data clusters. Hence, the smaller the probability for LPs to depend on each other over shared data, where a particular SV d_k may be grouped multiple times.

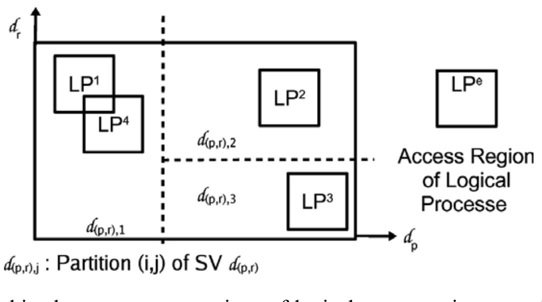

d_r
LP^1
LP^4
LP^2
LP^e
$d_{(p,r),2}$
Access Region of Logical Processe
$d_{(p,r),3}$
LP^3
$d_{(p,r),1}$
d_p
$d_{(p,r),j}$: Partition (i,j) of SV $d_{(p,r)}$

Fig. 2. Sample relationships between access regions of logical processes in a two-dimensional data space.

Resulting consistency relationships between groups however is in the responsibility of the user application. For instance whether updates on d_k over one group should have consistent interpretation in an other group that includes d_k as well.

3.6.2. Detecting multi-dimensional data clusters

A multi-dimensional data cluster $C_{(1,\ldots,q)}[c_{z'1}, c_{z1}, \ldots, c_{z'q}, c_{zq}]$ is detected in two steps: Firstly, the above described cluster detection algorithm is applied to all SVs, independently from whether a SV is bound or not. Secondly, for each compound SV $d_{(1,\ldots,\mathrm{q})}$, related access regions $a^{e}_{(1,\ldots,q)}$ and $a^{f}_{(1,\ldots,q)}$ are compared pairwise, if they intersect on at least one SV $d_1, \ldots, d_{\mathrm{q}}$, such that

$$[c_{z'1}, c_{z1}] \cap [c_{y'1}, c_{y1}] \neq \emptyset \vee \cdots \vee [c_{z'q}, c_{zq}] \cap [c_{y'q}, c_{yq}] \neq \emptyset \rightarrow C^{e,f}_{(1,\ldots,q)}[c_{z'1}, c_{z1} \vee c_{y1}], \ldots, [c_{z'q}, c_{zq} \vee c_{yq}]$$

3.6.3. Time complexity for detecting multi-dimensional data clusters

The time complexity for detecting q-dimensional clusters is $O(q * g) = O(n^2)$, for comparing in the worst case all g LPs' access regions on all q SVs. With a recursive algorithm, the complexity reduces to $O(n \log n)$.

3.6.4. Space complexity for maintaining variable-size, multi-dimensional partitions

On one hand, less dependent LPs require less messaging effort and can be re-located more independently. On the other hand, bound SVs increase the space requirement at each host. For instance, the number of adjacent multi-dimensional rectangular partitions increases exponentially with $s^3 * 2^s$ for increased number of s dimensions, which has a space complexity of $O(n^3 2^n)$. It enables however encapsulating an s-dimensional cluster within an s-dimensional SV partition of variable edge size. For instance, for a three-dimensional partition, the addresses of $3 * 2^3 - 3 = 21$–24 hosts need to be stored at each host, depending on the location of a partition at SV boundaries. The number increases with the number of adjacent partitions at each edge or surface.

3.6.5. Declaring compound SVs

For the purpose of long-term adaptation to evolving SV inter-relationships, an application may require dynamically re-grouping SVs. This possibility makes sense for SVs that represent metaphysical dimensions whose inter-relationships may evolve over time. The control over which SVs are to be grouped, when and how however is domain-dependent and therefore left to the user application.

3.7. Optimisation of partition distribution

Partition distribution over the hosts is dynamically optimised through variable-size partitioning, for adapting to cluster formations, and partition migration, for distributing the load.

3.7.1. Variable-size partitioning

A single partition $d_{i,j}$ is split between two clusters $C_i[c_{z'1}, c_{z1}]$ and $C_i[c_{z'2}, c_{z2}]$ at $(c_{z1} + c_{z'2})/2$. A multi-dimensional partition $d_{(1,\ldots,\mathrm{q})}$ is split between two clusters $C_{(1,\ldots,q)}[c_{z'1}, c_{z1}, \ldots, c_{z'q}, c_{zq}]$ and $C_{(1,\ldots,q)}[c_{y'1}, c_{y1}, \ldots, c_{y'q}, c_{yq}]$ at each SV partition $(1, \ldots, q)$, such that $(c_{z1} + c_{y'1})/2, \ldots, (c_{zq} + c_{y'q})/2$. Consequently, variable-size partitions may emerge in the course of a simulation (Table 4).

Note again the two kinds of indexing a host with h_j for referring to host j and with $h(i,j)$ for referring to partition j of the SV i. Accordingly, index j of two partitions $d_{i1,j}$ and $d_{i2,j}$ may be equal for different SVs that are indexed by the running variables $i1 \neq i2$, but not necessarily located on the same host $h(i1,j) \neq h(i2,j)$.

3.7.2. Partition migration

Variable-size partitioning allows adapting to data clusters, but can cause growing clusters in several partitions $d_{i,j}$ to concentrate on one host $h(i,j)$, which in turn can over load that host. In order to distribute the

Table 4
Sample variable-size data partitions

$d_{i,j}$ at $h(i,j)$	$h(i,j)$				
d_1	$d_{1,1}$ at h_1	$d_{1,2}$ at h_4	$d_{1,3}$ at h_3	$d_{1,4}$ at h_5	
d_2	$d_{2,1}$ at h_3	$d_{2,2}$ at h_2	$d_{2,3}$ at h_1	$d_{2,4}$ at h_4	$d_{2,5}$ at h_5
d_3	$d_{3,1}$ at h_2	$d_{3,2}$ at h_3	$d_{3,3}$ at h_5	$d_{3,4}$ at h_1	$d_{3,5}$ at h_4

load in such a case, clusters on that host are migrated to hosts $h(i,j-1)$ or $h(i,j+1)$ and merged with their adjacent partitions $d_{\mathrm{i,j}-1}$ or $d_{\mathrm{i,j}+1}$, respectively. Candidates for migration are chosen from the list of all clusters $C_i[c_{z'1}, c_{z1}]$ on host h_j, which is sorted in descending order after cluster access cost $\mathrm{sort}_{\mathrm{desc}}(A(C_i[c_{z'1}, c_{z1}]))$. Clusters with relative high access cost are those, whose owner LPs are located remote, thus migrating them will at least not increase the overall messaging effort of the application. Note that in this process a cluster is not distinguished from a single access region that has no intersections. The migration is continued for all clusters, until the accumulated cluster access costs of neighbouring hosts are equal, relative to their local load:

$$\Delta H_{j-1} * H_{j-1} = \Delta H_j * H_j = \Delta H_{j+1} * H_{j+1}$$

for hosts $h_{\mathrm{j}-1}$, h_j and $h_{\mathrm{j}+1}$, respectively, where

$$H_j = \sum_i^s \sum_z A(C_i[c_{z1}, c_{z2}])$$

and z denotes the number of regions on SV d_i. Partition migration is initiated for all hosts in every second phase of $2 * \Delta f$ state changes, alternating with the task for optimal placement of LPs. This schema distributes the load in a simulated annealing fashion, where the grade of annealing is adapted over Δf for a given application. ΔH_j is the load factor of host j and is re-calculated based on the local processor load.

3.7.3. Cluster splitting

In cases where the above equality for load distribution cannot be achieved on the overloaded host, the next cluster in the list that exceeds the equality is split, just like a partition is split. A split cluster is principally undesired, as it increases messaging effort, but is unavoidable, if overloading need to be resolved. This capability enables the DDM to adapt to applications that may develop increasingly growing clusters or, in the worst case, even a single all-one-cluster.

3.8. Optimal placement of logical processes

For the purpose of distributing the load to other hosts, each LP^e is placed on host k with the highest access cost A_k^e for LP^e:

$$\max_v \left(\sum_i^s (A_k(r_{i,k}^e) + A_k(u_{i,k}^e)) \right) \quad \text{iff } r_{i,k}^e \cap u_{i,k}^f \neq \emptyset$$

$v \leqslant s$ is the total number of regions of LP^e, $e, f = [1, \ldots, g]$ and $e \neq f$, and $A_k^e = F_k^e * L_k^e$ with $i \leqslant s$ of s SVs. A subscription region without any intersections with update regions is excluded from this calculation, as it has always zero access cost, wherever LP^e is located. Consequently, an LP is placed on that host where the most significant region for that LP is located.

A sample distribution of access regions and LP placement for hypothetical access costs is sketched in (Fig. 3). Note that, for instance subscription region $r_{s,1}^f$ was not included in the calculation for the placement

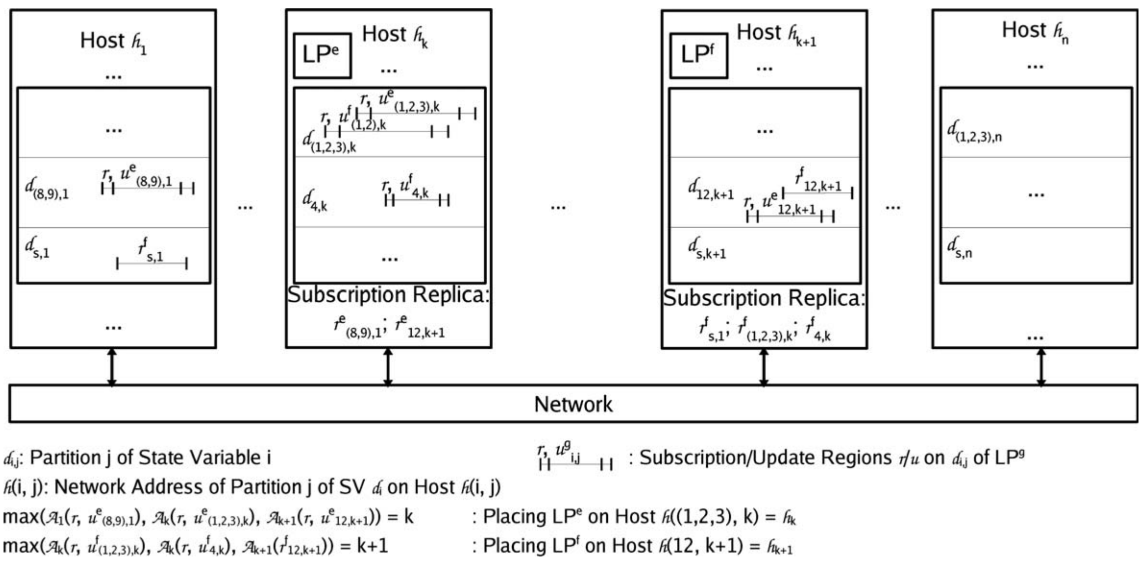

Host h_1
Host h_k
LP^e
Host h_{k+1}
LP^f
Host h_n
Subscription Replica: $r^e_{(8,9),1}$; $r^e_{12,k+1}$
Subscription Replica: $r^f_{s,1}$; $r^f_{(1,2,3),k}$; $r^f_{4,k}$
Network
$d_{i,j}$: Partition j of State Variable i
$r, u^g_{i,j}$: Subscription/Update Regions r/u on $d_{i,j}$ of LP^g
$h(i, j)$: Network Address of Partition j of SV d_i on Host $h(i, j)$
$\max(A_1(r, u^e_{(8,9),1}), A_k(r, u^e_{(1,2,3),k}), A_{k+1}(r, u^e_{12,k+1})) = k$: Placing LP^e on Host $h((1,2,3), k) = h_k$
$\max(A_k(r, u^f_{(1,2,3),k}), A_k(r, u^f_{4,k}), A_{k+1}(r^f_{12,k+1})) = k+1$: Placing LP^f on Host $h(12, k+1) = h_{k+1}$

Fig. 3. Sample data distribution and optimised placement of two logical processes for given data access schema.

of LP^f, as it does not intersect with any update region. Also, compound SV $d_{(1,2,3)}$ is depicted as one-dimensional region, for simplicity, but is to be interpreted as cubical region.

3.9. Primitive operations on shared data

LPs can access the shared data over three primitive operations: reading or updating a region and changing region boundaries. Each operation initiates a well defined sequence of messages for keeping the shared data globally consistent, by complying with the above sketched data distribution schema.

3.9.1. Message content

The message sent for synchronising a remote SV is uniform. It contains the new SV value and a local time stamp. In order to reduce the physical number of messages sent to the same host, each update on a SV that occurs until the next time advance request is packed as logical message into one physical message.

3.9.2. Reading subscribed data

When LP^e subscribes to a region $r_i^e[c_1', c_1]$ within partition $d_{i,j}$ on host $h(i,j)$, then the region is copied to the host where LP^e is located. Any following read operation from LP^e within $r_i^e[c_1', c_1]$ is then performed locally. Thus two messages are required for each new subscription, if the partition is remote.

3.9.3. Updating associated data

If an update within a region $u_i^e[c_1', c_1]$ is remote, with respect to the associated LP^e, then one message is required. If the update region intersects with a total number of v remote subscription regions, then further v messages are required to propagate the update on the original data to v replica, where $v = [0, \ldots, s]$ and s is the total number of SVs. Thus at most $v + 1$ messages are required.

3.9.4. Updating region boundaries

Changing region boundaries, caused for instance by a moving simulation object, requires a new calculation of intersections with other regions. If the boundary change is within this partition, then no messages are required. Otherwise, one message is sent to the host with the related adjacent partition. If the boundaries of a subscription region change and the partition is remote, then one message is required. One further message is sent back to the region owner, if the new boundary has entered an adjacent partition on another host. Thus at most two messages are required for this operation.

3.10. Time management

Time management involves synchronising updates on the shared data with respect to local time advance of the LPs and global time advance of the simulation.

3.10.1. Synchronisation

SV updates are synchronised either conservatively or optimistically, depending on the context of the update. Accordingly, the following synchronisation policies are applied:

- Conservative synchronisation of an LP that owns remote located interest regions.
- Conservative synchronisation of interest region updates, if the regions intersects with another region.
- Optimistic synchronisation in all other cases.

3.10.2. Time advance

The employed optimistic synchronisation is a moderate optimistic variant of the time warp algorithm [16]. Time advance in local virtual time (LVT) is calculated according the following schema [19]: Initially, all local LPs are allowed to advance in their LVT as far as possible. But in case of a roll-back, the LVT of an LP^e is restricted to a maximum allowed time advance $T_{\max}$, for the purpose of reducing the risk for roll-backs. If no straggler message was received, then LP^e may advance up to $T_{\max}$ in LVT. If no roll-back is processed, then after each $T_{\max}$ time advance, T^e_{advance} is accumulated by some constant small increment, like $0.1 * T_{\max}$:

$$\text{if rollBack then } T^e_{\text{advance}} = T_{\max} \quad \text{else } T^e_{\text{advance}} = T^e_{\text{advance}} + T_{\text{increment}} * T_{\max}$$

T^e_{advance} is the allowed time advance for LP^e to be calculated in $T_{\max}$ time intervals and $T_{\text{increment}}$ is the accumulation factor.

3.11. Complexity of DDM techniques

The overall performance of the region, grid, partition techniques and their hybrid combinations in terms of messaging, time and space complexity of the primitive operations on shared data is now analysed for the following two cases (Table 5):

Table 5
Messaging, time and space complexity of operations of DDM techniques for evenly distributed versus single-clustered interests with and without load distribution

Technique	Shared data distribution	Operation				Shared data connectivity (space]
		Read or update (messaging)		Move (messaging/time)		
		Evenly distributed	Single cluster	Evenly distributed/ matching	Single cluster/ matching	
Any	Server	$O(n)$	$O(n)$	$O(n)/O(n\log n)$	$O(n)/O(n\log n)$	$O(0)$
Region	Peer	$O(n^2)$	$O(n^2)$	$O(n^2)/O(n\log n)$	$O(n^2)/O(n\log n)$	$O(0)$
Grid	Peer	$O(n^2)$	$O(n)$[a]	$O(n^2)/O(0)$	$O(n)/O(0)$[a]	$O(n2^n)$
Partition	Peer	$O(n^2)$	$O(n^2+B)$[b]	$O(n^2)/O(0)$	$O(n^2+B)/O(0)$[b]	$O(n^3 2^n)$
Grid with region	Peer	$O(n^2)$	$O(n)$[a]	$O(n^2)/O(\log n)$	$O(n)/O(n\log n)$[a]	$O(n2^n)$
Partition with region	Peer	$O(n^2)$	$O(n^2+B)$[b]	$O(n^2)/O(\log n)$	$O(n^2+B)/O(\log n)$[b]	$O(n^3 2^n)$

[a] Without load distribution: single cluster of interest regions located on a single host (interest region server).
[b] With load distribution cost B: single cluster of interest regions split and scattered over all peers (interest region peers).

- *Evenly distributed interest:* interest regions and the overall access cost of the LPs are evenly distributed over the hosts (interest region peers).
- *Single-clustered interest:* interest regions and the overall access cost of the PLs are concentrated within a single grid/partition (interest region server).

All techniques perform equal for the case that all *s* SVs are maintained at a server. For read operations, the worst case is assumed, where each read is preceded by an update on the original data and therefore requires an additional message for replicating the update. Messaging complexity is only O(n), as in the worst case all n hosts may send a message to the server in one state change. Time complexity is O($n\log n$), since interest matching is performed only on the server. For instance, the region technique combined with the grid on a server has been proposed in [35].

If all SVs are fully replicated at all peers, peer-to-peer communication is required with the region technique. Interest has to be matched at all n hosts over all s SVs, no matter whether all interest regions are single-clustered or not. The replicated world of most DIS simulations [4] approaches this case.

In case of grid and partition-based distribution of the SVs, peer-to-peer communication complexity is O(n^2) for read/update operations. It reduces to O(n) in the case of a single cluster that is located at a single peer in the grid, but increases to O($n^2 + B$), if the cluster is split and scattered over all peers, where B is an average cost for load distribution.

In case of move operations, peer-to-peer communication complexity is principally equal for the grid and partition techniques, if interest regions are evenly distributed over all peers. Interest matching reduces to O($\log n$) at each host. If an application develops a single cluster, then messaging reduces to O(n) in the grid, assuming that moving interest regions will remain within the cell on that peer. If the single cluster is split and scattered over all peers within partitions however messaging increases to O($n^2 + B$). Interest matching increases to O($n\log n$) within the single grid cell and reduces to O($\log n$) at each partitioned peer.

4. The APDES implementation

This discussion introduces the sample implementation of the hybrid technique of partitioning the shared data and matching of interest regions in the distributed simulation environment APDES.

4.1. Relationships between components

In this distributed IM schema, every host may be engaged with every other host in peer-to-peer communication, in fulfilling the above discussed functionality. This functionality is structured in form of modules (Fig. 4), which is discussed below.

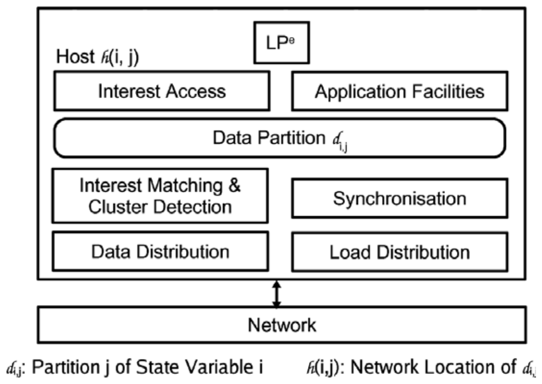


Fig. 4. Task-oriented modules of the APDES architecture.

4.1.1. Interest access

This module provides functionality to local LPs for declaring and accessing SVs, where a global view on the shared data is provided, so that all shared data appears local to the LPs.

4.1.2. Interest matching and cluster detection

This is performed on each host locally and independently from other hosts. This processes can run in parallel and does not require any network communication, as only local matches and clusters are calculated. Even in case of a split cluster or split access region no communication is required with the related hosts for the other part of the cluster or access region. Therefore interest matching does not require re-unifying split clusters or regions.

4.1.3. Data distribution

This module is responsible for the global consistency of the shared data, which is implemented over links to related data partitions and hosts, and by synchronising the access to SVs. A local access request to a region is forwarded to related hosts, if the region is remote. At receiver site, the access on the original data is performed and multicast to the related hosts.

4.1.4. Application facilities

This module includes components that provide some domain-independent services to LPs, such as dynamically enabling/disabling an LP's migration, grouping of SVs, providing an LP the clusters it is involved in, searching for SVs or other LPs that are outside but close to LP's sense area on a given SV. Application or domain-specific components can also be collected here, such as extensions to interest declaration for filtering classes and subclasses of objects.

4.1.5. Control flow

The generic scenarios below sketch the control flow between functional units for the three primitive operations, when initiated by an LP.

- Reading subscribed data:
  1. LP^e calls *read* within $r_i^e[c_1', c_1]$ at host h_1.
  2. Data distribution module returns value from the local replica.

- Updating associated data:
  1. LP^e calls *update* within $u_i^e[c_1', c_1]$ at host h_1.
  2. Data distribution module locates region.
  3. If region is remote, then message is sent to related host h_2.
  4. h_1 performs update on original data.
  5. h_1 forwards the update to all v subscribers and h_1.
  6. All v subscribers and h_1 call-back local LP requests, after having received update.
  7. If first Δf state changes have passed, then all hosts. initiate partition migration
  8. If second Δf state changes have passed, then re-location is initiated for all LPs.

- Updating region boundary:
  1. LP^e calls update of region boundary for $a_i^e[c_1', c_1]$ at host h_1.
  2. Data distribution module locates region.
  3. If region is remote, then message is sent to related host h_2.
  4. h_2 updates region boundary and initiates interest matching with cluster detection.
  5. If update region is outside of local partition size, then, either host $h(i,j-1)$ or $h(i,j+1)$ is notified, which also initiates interest matching with cluster detection and notifies back host h_1.

4.2. Performance evaluation

The performance of APDES has been evaluated for a test application that has some statistical properties. The objective of the test was to observe the load distribution functionality when clustering on a single host was given.

4.2.1. Test environment

The tests were run on nine machines, out of a Beowulf cluster of 128 machines. Each machine had a 2.66 GHz CPU with 1 GB RAM and 80 GB local HD. The network switch had a 720 giga bits per second (Gbps) bandwidth and 400 million packets per second (Mpps) switching performance.

4.2.2. Application properties

The application is a hypothetical model that allows testing the load distribution and the performance of the simulation. Following parameters were constant in the course of a simulation and during all test runs:

- Number of hosts: 9.
- Number of LPs: 72.
- Number of SVs: 1 two-dimensional compound SV.
- Number of interest regions: 72 = 72 LPs ∗ 1 SV.
- Hypothetical peer overload threshold: 95 ms local processing for each LVT advance.
- Hypothetical peer load reduction: 10% of the hypothetical peer overload threshold.
- Number of messages per region and LP: normal-distributed over all regions and LPs.
- Message increase per GVT advance: 100 messages.
- Total application size: guaranteed to fit into local RAM.
- 2 ∗ LP processing = 1 ∗ region processing: in the average, each message was processed with double the amount at a region than by an LP.

4.2.3. Start constellation

In the start constellation of each test, 24 interest regions and 24 LPs were clustered and placed at host 5. The other regions and LPs were evenly distributed over the remaining hosts. This constellation continued, until the total number of the messages of the application reached 7000 (Fig. 5 and Table 1).

4.2.4. Region migration from the cluster

Each following table (Fig. 5) shows the constellation after a completed load distribution, where only region migration was allowed. As the total number of messages of the application increased, the load at each peer, like expected, increased as well. The hypothetical peer overload threshold at 7000 messages was detected at host 5 (Fig. 6). Than the load distribution module migrated six regions from host 5 to the first six neighbouring hosts; each host 1 region (Fig. 5 and Table 2). At approximately 11 500 messages, all hosts exceeded their hypothetical peer overload threshold and the simulation stopped.

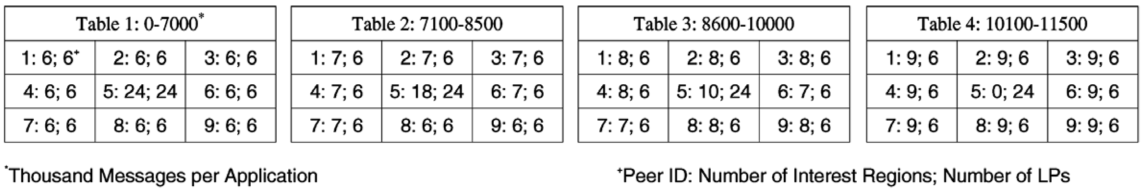


Table 1: 0-7000*		
1: 6; 6⁺	2: 6; 6	3: 6; 6
4: 6; 6	5: 24; 24	6: 6; 6
7: 6; 6	8: 6; 6	9: 6; 6

Table 2: 7100-8500		
1: 7; 6	2: 7; 6	3: 7; 6
4: 7; 6	5: 18; 24	6: 7; 6
7: 7; 6	8: 6; 6	9: 6; 6

Table 3: 8600-10000		
1: 8; 6	2: 8; 6	3: 8; 6
4: 8; 6	5: 10; 24	6: 7; 6
7: 7; 6	8: 8; 6	9: 8; 6

Table 4: 10100-11500		
1: 9; 6	2: 9; 6	3: 9; 6
4: 9; 6	5: 0; 24	6: 9; 6
7: 9; 6	8: 9; 6	9: 9; 6

*Thousand Messages per Application

⁺Peer ID: Number of Interest Regions; Number of LPs

Fig. 5. Re-distribution of the number of interest regions from the region and LP clustering at host 5.

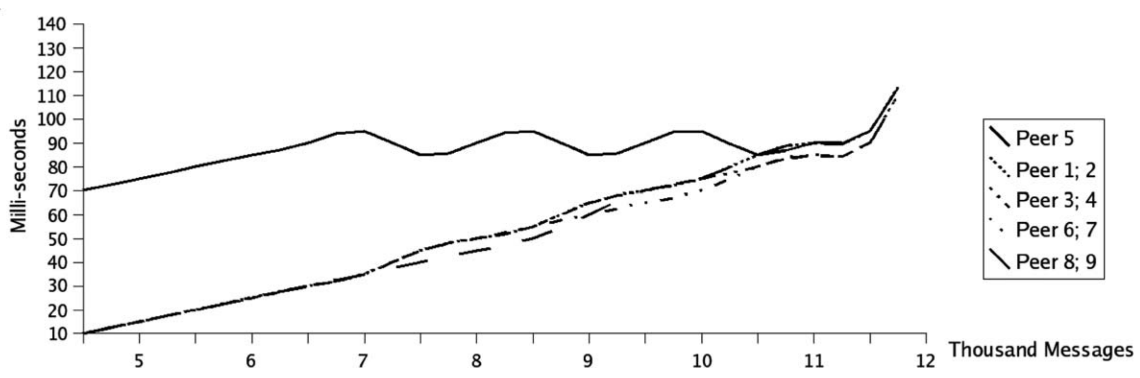


Fig. 6. Response time of APDES at each peer for increased message load (with partition migration; without LP migration).

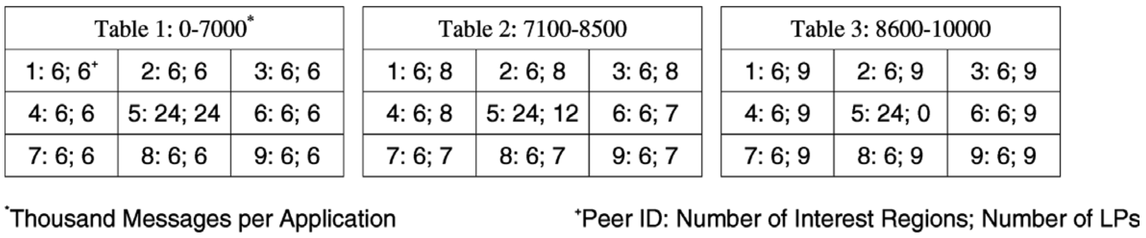


Table 1: 0-7000*		
1: 6; 6+	2: 6; 6	3: 6; 6
4: 6; 6	5: 24; 24	6: 6; 6
7: 6; 6	8: 6; 6	9: 6; 6

Table 2: 7100-8500		
1: 6; 8	2: 6; 8	3: 6; 8
4: 6; 8	5: 24; 12	6: 6; 7
7: 6; 7	8: 6; 7	9: 6; 7

Table 3: 8600-10000		
1: 6; 9	2: 6; 9	3: 6; 9
4: 6; 9	5: 24; 0	6: 6; 9
7: 6; 9	8: 6; 9	9: 6; 9

*Thousand Messages per Application

+Peer ID: Number of Interest Regions; Number of LPs

Fig. 7. Re-distribution of the number of LPs from the region and LP clustering at host 5.

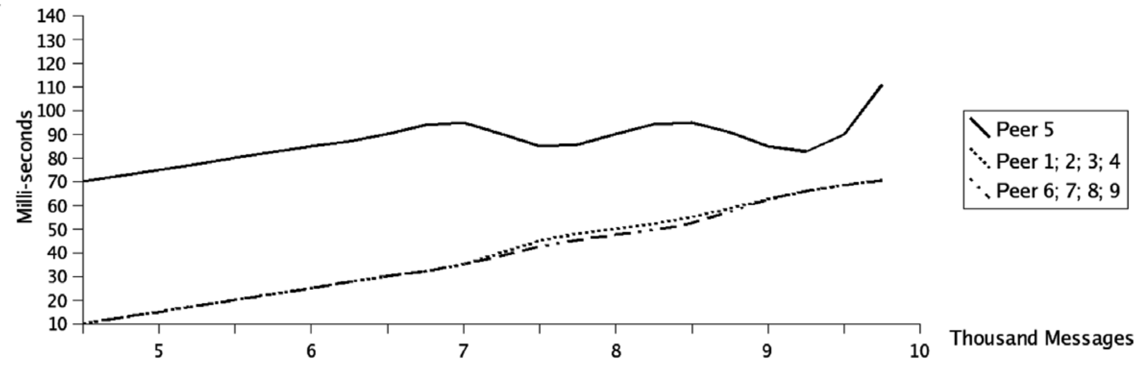


Fig. 8. Response time of APDES at each peer for increased message load (without partition migration; with LP migration).

4.2.5. LP migration from the cluster

In this case only LP migration was allowed. The hypothetical peer overload threshold at 7000 messages was detected at host 5 (Fig. 7). The load distribution module migrated than 12 LPs from host 5 to the neighbouring hosts (Fig. 8 and Table 2). At approximately 10 000 messages, host 5 exceeded its hypothetical peer overload threshold and the simulation stopped. A free processing capacity of approximately 25% was recorded at the other hosts.

5. Discussion

Based on the above discussed performance results, performance issues for some special application constellations are discussed and related work compared.

5.1. Performance issues

The performance of the IM changes dynamically, depending directly on the evolved spatial relationships between the simulation objects of an application. The adaptation of the IM to these dynamics is recognised as a multi-objective optimisation problem defined as:

- Minimisation of overall messaging effort against.
- Minimisation of local overhead at each host.

The first is approached through data clustering and maximisation of partition size against partition and LP migration for minimisation of the second. The following discussion on some special cases of the optimisation problem exemplifies the adaptability of the model and the dynamically achievable performance improvements.

5.1.1. General performance

The messaging complexity is bound by $O(n^2)$, in the worst case for all g LPs sending each a message to all n hosts in one state change of the system. Since logically independent messages that are within one time advance of the simulation are packed into one physical message, at most n^2 messages are sent between n hosts. The administrative overhead in case of fixed-size partitioning is zero, as no partition administration is required. The network overhead for maintaining variable-size partitions is diminishing, as it is performed in intervals of $2 * \Delta f$ state changes and only for a subset of n hosts, depending on the simulated annealing factor ΔH. These special cases come into effect, when evaluated over a period of state changes, such as Δf, hence reveal the higher achievable performance of the system in practice. The total time complexity for simultaneously performing interest matching, keeping region boundaries on SVs sorted and detecting multi-dimensional clusters, respectively, is

$$O(n \log n) + O(\log n) + O(n \log n) = O((2n+1) \log n) \rightarrow O(n \log n)$$

5.1.2. Performance in large-scale environments

In case of an application with less clustering probability, interest regions would rather scatter in the environment. In this case the performance can be expected to approach $O(nv) \rightarrow O(n)$ for $v \ll g$, where n is the number of hosts and v the total number of remote subscriptions of the application that intersect with update regions.

5.1.3. Partition distribution

Optimal partition distribution is achieved when no data cluster is split, which avoids additional administrative overhead. Worst case partition distribution is a constellation where partitions split clusters and single access regions, which introduces additional local overhead on each host. Splitting clusters can cause additional network overhead between hosts.

5.1.4. LP distribution

Optimal LP distribution is achieved when each LP is placed on the host, to which otherwise its access cost would be the highest. Non-intersecting access regions of that LP are all on that host, which also minimises the replica. Thus in this scenario the messaging effort is minimised. Worst case LP distribution is a constellation, where all access regions are located remote from their owner LPs. This scenario approaches the maximum administrative overhead of $O(n^2)$.

5.1.5. Performance improvement

The restriction to have one partition per host and per SV can be relaxed, in order to allow non-adjacent partitions to be located on one host. This would enable for migrating also those regions and clusters with second highest, third highest and so forth, access costs to the owner LP. Hence reducing further messaging effort and facilitating further load distribution. This extension however would prioritise splitting partitions over merging them, resulting in one partition for each access region or cluster in the worst case. Applications with frequently moving access regions would then change relative frequently between partitions as well, which would cause a considerable administrative overhead. In addition, a partition de-fragmentation algorithm may also be required, if increasingly smaller partitions scatter even more.

5.2. Contrasting related work

In addition to the discussion in Section 1, some additional aspects that have been implemented in ADPES are contrasted here to related work.

The maintenance of each interest region's access frequency is comparable with the SoI of an event [23], however no implementation of this concept is known.

Dynamic load balancing has been proposed in various forms for HLA simulations, mostly for LPs or whole federates. Variable partition size and partition migration has not been proposed before.

Maintaining information about remote interest intersections of an LP at the LP's site is implemented in APDES and has not been reported by now from other implementations. For instance current HLA implementations of the DDM module, namely run time infrastructure (RTI), such as DMSO RTI [10,33,37], a modification on the boundary of an update region is broadcast. Because a host does not maintain remote objects' regions that do not intersect with local objects' regions [13].

Region-oriented IM is common to most distributed virtual environments (DVE), where the complete environment is maintained on a server [14] or a central event queue [32] and therefore are restricted scalable.

Grid-oriented IM is implemented in the RTI next generation (NG) versions and parallel proximity detection in synchronous parallel environment for emulation and discrete event simulation (SPEEDES) operating system [34] arranges the three physical dimensions into three grids and updates are unicast. Local grid cells are addressed via hash tables. Further dimensions can be handled by plugging in an HLA DDM component [25].

Similar to SPEEDES is the partitioning concept of some synthetic theater of war (STOW) simulations, with the difference that conservative synchronisation and multicasting within each grid is employed [12].

Parallel discrete event simulators for fine-grained simulation, like Georgia Tech Time Warp (GTW) [7], Rensselaer's optimistic simulation system (ROSS) [6] or WARPED [24] do not employ IM. Instead, performance improvements are explored with various optimistic synchronisation techniques and, in case of GTW and WARPED, additionally with load distribution.

The concept of dynamic partitioning should not be confused with the concept of dynamic grids [3], which refers to dynamically calculating multicast groups for grids. Hence dynamic grids do not address message overloading within a grid cell.

6. Conclusion

The introduced IM model inherits its adaptability from the dynamics of the data distribution schema, particularly the partition adaptation to cluster formations and the optimal placement of LPs. This is achieved by dynamically calculating network latencies for updating the shared data. The model benefits from the fact that no receiver-side filtering is required, as all LPs receive only the data they have subscribed for. A further benefit is the propagation of updates on interest region boundaries, whose administrative maintenance is, with at most one message, diminishing. The administrative overhead is restricted to a few network messages for maintaining partition adjacency information. These capabilities of the model facilitate further load distribution, hence scalability. Although there is a notable local processing overhead for interest matching, cluster detection and adjacency maintenance, the scalability is not influenced by this, as load distribution is employed. The adopted heuristic here is that less messaging load can be achieved against more local processing, where local processing overload in turn can be re-distributed.

Although the model possesses capabilities to dynamically overcome some worst case constellations, the restriction to a single partition per host, prevents from optimising further some special cases. For instance update regions that are not on adjacent partitions and that do not intersect with others' update regions, cannot be migrated to the owning LP. This could be optimised by allowing multiple non-adjacent partitions on one host. Which would allow also migrating a whole cluster to the LP that owns a region of the cluster with the highest access cost.

The trade-off between this extension and the additional messaging effort however is to be tested, as frequent moving interest regions can reduce the system performance.

An interesting property of the model is that it can dynamically adapt to application constellations and hence is suitable for coarse-grained as well as fine-grained simulations that are invoked within a coarse-grained simulation. This adaptability has been shown with the performance evaluations for dynamic region or LP migration. In cases where LPs start unexpected and relative intensive processing, such as planning, it will also be interesting to observe this adaptability.

Besides these dynamics of ADPES that facilitate the simulation of intelligent systems, further support for such systems is provided, such as homogeneous and scalable schema for shared data and meta-data, which can be utilised by LPs for adapting to their environment.

Our next goal is to examine the performance of the hybrid partition-region-based DDM technique for some application constellations that are common to intelligent systems, such dynamic interest change in a swarm.

As a by product of this examination, we shall present a benchmark application and scenarios that represent significant properties of representative application domains.

Acknowledgements

The numerical calculations reported in this paper were carried out partially at the Beowulf cluster of the High Performance Computing Centre (ULAKBİM) of the Turkish Scientific and Technical Research Council (TÜBİTAK) and the local cluster of the International Computer Institute at Ege University.

References

[1] Steve Benford, Lennart Fahlen, A spatial model of interaction in virtual environments, in: Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93), 1993.
[2] Azzedine Boukerche, A static partitioning and mapping algorithm for conservative parallel simulations, in: 8th Workshop on Parallel and Distributed Simulation (PADS, ACM, 1994.
[3] Azzedine Boukerche, Amber Roy, Dynamic grid-based approach to data distribution managementWorkshop on Distributed Simulation and Real-Time Applications, Elsevier, 2002.
[4] Azzedine Boukerche, Amber Roy, In search of data distribution management in large scale distributed simulations, in: Summer Computer Simulation Conference, 2002.
[5] James O. Calvin, Carol J. Chiang, Daniel J. Van Hook, Data subscription, in: Twelfth Workshop on Standards for the Interoperability of Distributed Simulations, 1995.
[6] Christopher D. Carothers, David Bauer, Shawn Pearce, ROSS: a high-performance, low memory, modular time warp system, in: 14th Workshop on Parallel and Distributed Simulation (PADS), ACM, 2000.
[7] S.R. Das, R. Fujimoto, K. Panesar, D. Allision, M. Hybinette, GTW: a time warp system for shared memory multiprocessors, in: Proceedings of the Winter Simulation Conference, 1994.
[8] Han Seunghyun, Mingyu Lim, Dongman Lee, Scalable interest management using interest group based filtering for large networked virtual environments, Virtual Reality Software and Technology, ACM, 2000.
[9] J. Hardt, K. White, Distributed interactive simulation (DIS), EEL 4781 Computer Networks, University of Central Florida, 1998.
[10] HLA 1.3, US DMSO, HLA Data Distribution Management Design Documentation Version 0.5, 1997, 1996. Available from: <http://dmso.mil/public/transition/hla>.
[11] Daniel J. van Hook, Steven J. Rak, James O. Calvin, Approaches to relevance filtering, in: Workshop on Standards for the Interoperability of Distributed Simulations (DIS), 1994.
[12] Daniel J. van Hook, David P. Cebula, Steven J. Rak, Carol J. Chiang, Paul N. DiCaprio, James O. Calvin, Performance of STOW RITN application control techniques, in: 14 th Workshop on Standards for the lnteroperability of Distributed Simulations, 1996.
[13] Daniel J. van Hook, James O. Calvin, Spring, data distribution management in RTI 1.3, Simulation Interoperability Workshop (SIW), IEEE, 1998.
[14] Bryan Horling, Roger Mailler, Victor Lesser, Farm: a scalable environment for multi-agent development and evaluationAdvances in Software Engineering for Multi-Agent Systems, Springer, 2004.
[15] Bryan Horling, Victor Lesser, Data dissemination techniques for distributed simulation environments, in: Winter Simulation Conference (WSC 2004), 2004.
[16] David R. Jefferson, Virtual Time, ACM Transactions on Programming Languages and Systems 7 (3) (1985).
[17] Bora İ. Kumova, CEVOP: a co-evolutionary planning methodology, in: MALCEB'02, 2002.
[18] Bora İ Kumova, Interest management in distributed simulation: a survey, in: Symposium on Design, Analysis, and Simulation of Distributed Systems (DASD'04), The Society for Modelling & Simulation International (SCS), 2004.

[19] Bora İ Kumova, Design concepts of a distributed simulation kernel, in: EuroSim Congress on Modelling and Simulation (EuroSim'04), Federation of European Simulation Societies (EuroSim), 2004.
[20] Bora İ Kumova, Scalable interest management in distributed simulation of intelligent systems, in: Symposium on Design, Analysis, and Simulation of Distributed Systems (DASD'05), The Society for Modelling & Simulation International (SCS) 2005.
[21] Bora İ Kumova, Dynamically adaptive partition-based data distribution management, in: Principles of Advanced and Distributed Simulation (PADS'05), IEEE Computer Society, 2005.
[22] Jong S. Lee, Bernard P. Zeigler, Space-based communication data management in scalable distributed simulation, Journal of Parallel and Distributed Computing (2002).
[23] B. Logan, G. Theodoropoulos, Dynamic interest management in the distributed simulation of agent-based systems, in: Proceedings of 10th AI, Simulation and Planning Conference (RAE2001), Society for Computer Simulation International, 2000.
[24] D.E. Martin, T.J. McBrayer, P.A. Wilsey, WARPED: a time warp simulation kernel for analysis and application development, in: 29th Hawaii International Conference on System Sciences (HICSS-29), 1996.
[25] Metron Inc., SPEEDES Usre's Guide, 2003. Available from: <www.speedes.com>.
[26] Katherine L. Morse, Jeffrey S. Steinman, Data distribution management in the HLA multidimensional regions and physically correct filtering, in: Spring Simulation Interoperability Workshop (SIW), IEEE, 1997.
[27] Katherine L. Morse, Lubomir Bic, Kevin Tsai, Multicast group for dynamic data distribution management, in: Proceedings of the 31st Society for Computer Simulation Conference (SCSC), 1999.
[28] Katherine L. Morse, Lubomir Bic, Michael Dillencourt, Interest management in large scale virtual environments, MIT PRESENCE – Teleoperators and Virtual Environments, 2000.
[29] Véronique Normand, Christian Babski, Steve Benford, Adrian Bullock, Stéphane Carion, Nicolas Farcet, Emmanuel Frécon, Nico Kuijpers, Nadia Magnenat-Thalmann, Soraia Raupp-Musse;
Tom Rodden, et al., The COVEN project: exploring applicative, technical, and usage dimensions of collaborative virtual environments, Presence-Teleoperators and Virtual Environments, MIT Press, 1999.
[30] Mikel D. Petty, Comparing high level architecture data distribution management specifications 1.3 and 1516Simulation Practice and Theory, Elsevier, 2001.
[31] Mikel D. Petty, Katherine L. Morse, The computational complexity of the high level architecture data distribution management matching and connecting processes, Simulation Modelling Practice and Theory, Elsevier, 2004.
[32] Patrick Riley, MPADES: middleware for parallel agent discrete event simulation, RoboCup-2002: Robot Soccer World Cup VI, Springer, LNAI, 2002.
[33] pRTI. Available from: <http://www.pitch.se>.
[34] Jeff S. Steinman, Frederick Wieland, Parallel proximity detection and the distributed list algorithm, in: Workshop on Parallel and Distributed Simulation (PADS), IEEE, New York, 1994.
[35] Gary Tan, Yu Song Zhang, Rassul Ayani, A hybrid approach to data distribution management, in: Workshop on Distributed Simulation and Real-Time Applications (DS-RT), IEEE Computer Society, 2000.
[36] Adelinde M. Uhrmacher, M. Krahmer, A conservative approach to the distributed simulation of multi-agent systems, in: European Multi-Conference on Simulation (ESM'2001), 2001.
[37] Douglas D. Wood, Len Granowetter, Rationale and design of the MÄK real-time RTI, in: Simulation Interoperability Workshop (SIW), IEEE, 2001.
[38] Bernard P. Zeigler, George Ball, Hyup Cho, J.S. Lee, Hessam Sarjoughian, Bandwidth utilization/fidelity tradeoffs in predictive filtering, in: Simulation Interoperability Workshop (SIW), IEEE, 1999.