A Parallel Interest Matching Algorithm for Distributed-Memory Systems

Elvis Liu

Georgios Theodoropoulos

UNIVERSITY OF BIRMINGHAM

IBM®

Outline

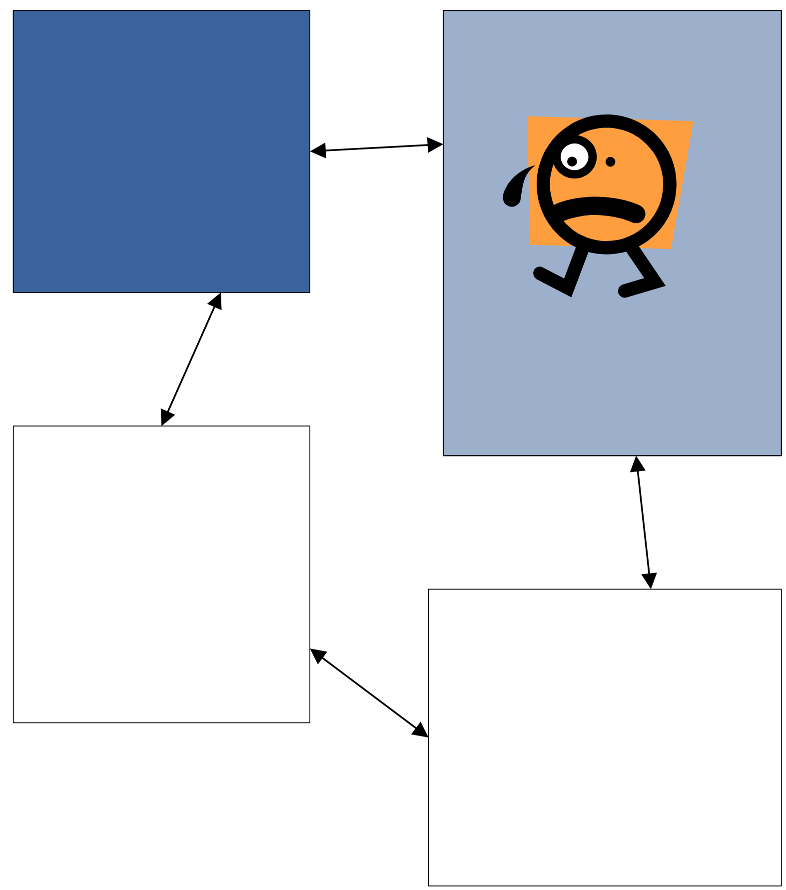
- Introduction
  - Distributed Virtual Environments (DVE)
  - Interest Management
- Parallel Interest Matching
  - Algorithm
  - Load-balancing
- Experimental Results
- Conclusions

Distributed Virtual Environments (DVE)

- Allow multiple users interact in real-time even though they are in different physical locations
- Commercial Application – Massively Multiplayer Online Games (MMOGs)
- Academic/Military Application – HLA compliant systems
- Scalability
- Message broadcasting
- Interest Management

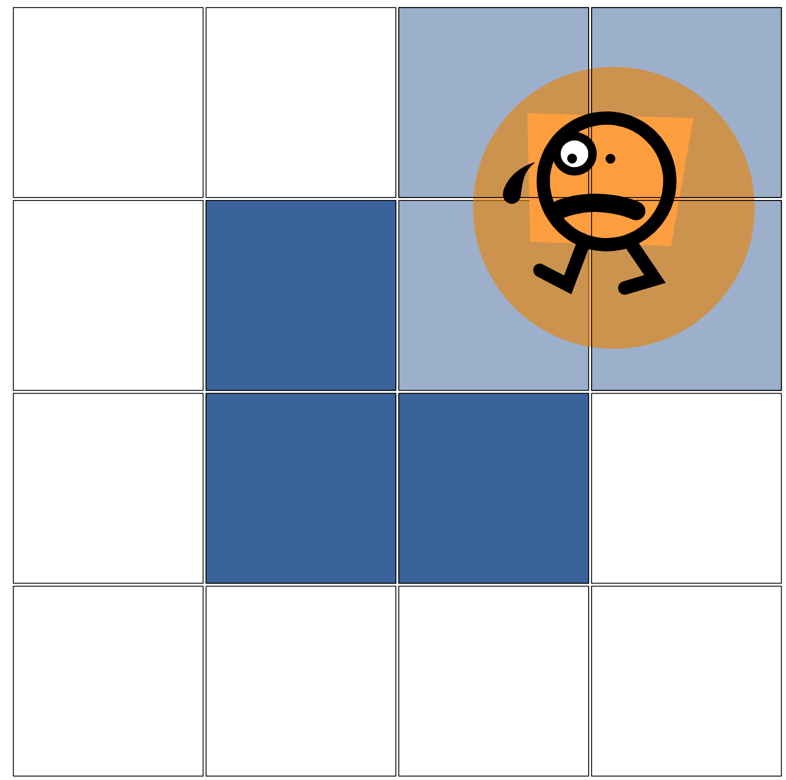
Seamed Zone-based Schemes

- Used by most MMOGs: FFXI, Everquest, GuildWars
- Divide the virtual world into zones
- Only receive update from one zone
- No interest matching is required



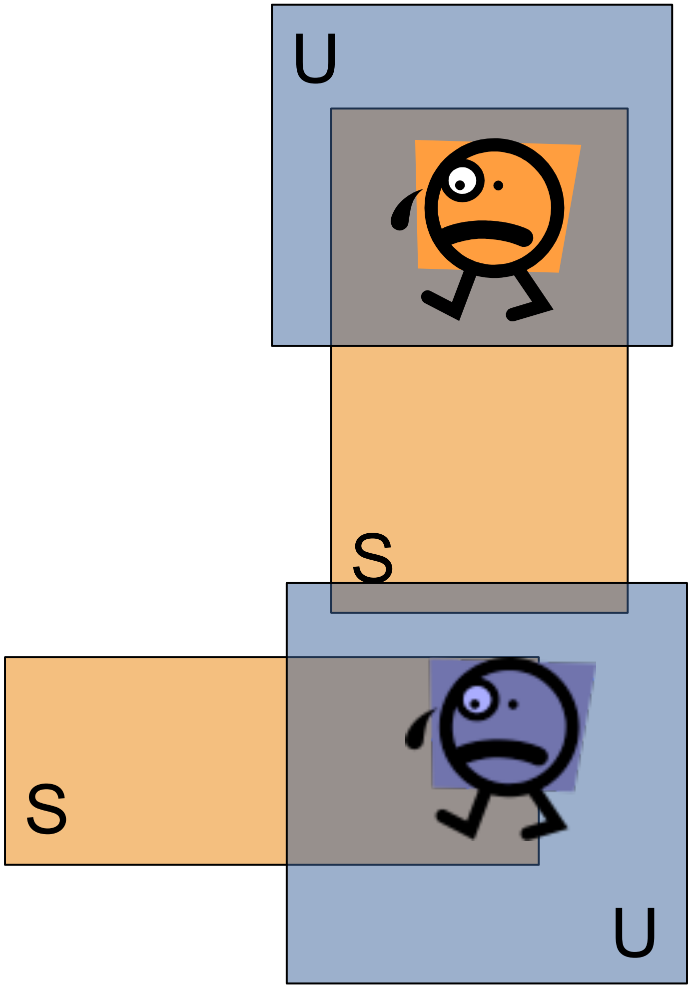
Seamless zone-based Schemes

- Used by NPSNET
- Divide the virtual world into zones
- Invisible border
- Area of Interest (AOI)
- Interest Matching – *O(n)* for *n* AOIs



Aura-based Schemes

- Used by MASSIVE
- Higher filtering accuracy than zone-based schemes
- Interest Matching – *O(nm)* for *n* update regions and *m* subscription regions



Filtering Precision vs. Runtime Efficiency

- A trade-off
- Zone-based schemes: Good runtime efficiency but poor filtering precision
- Aura-based schemes: Good filtering precision but poor runtime efficiency
- Existing interest matching algorithms try to deal with this problem

Existing Interest Matching Algorithms

- Try to improve the runtime efficiency of interest matching
- Multidimensional Binary Trees (Van Hook 1997)
- Collision Detection Algorithm (Morgan 2004)
- Sort-based (Raczy 2005, Pan 2007, Liu 2005)
- All of the above are serial algorithms

Parallel Processing

- Serial algorithms – Poor workload sharing
- Need of parallel interest matching algorithm
- Commercial applications (e.g. MMOGs) usually use shared-memory multiprocessors as servers
- Parallel Processing revolution – multicore processors becoming mainstream
- Heterogeneous platforms

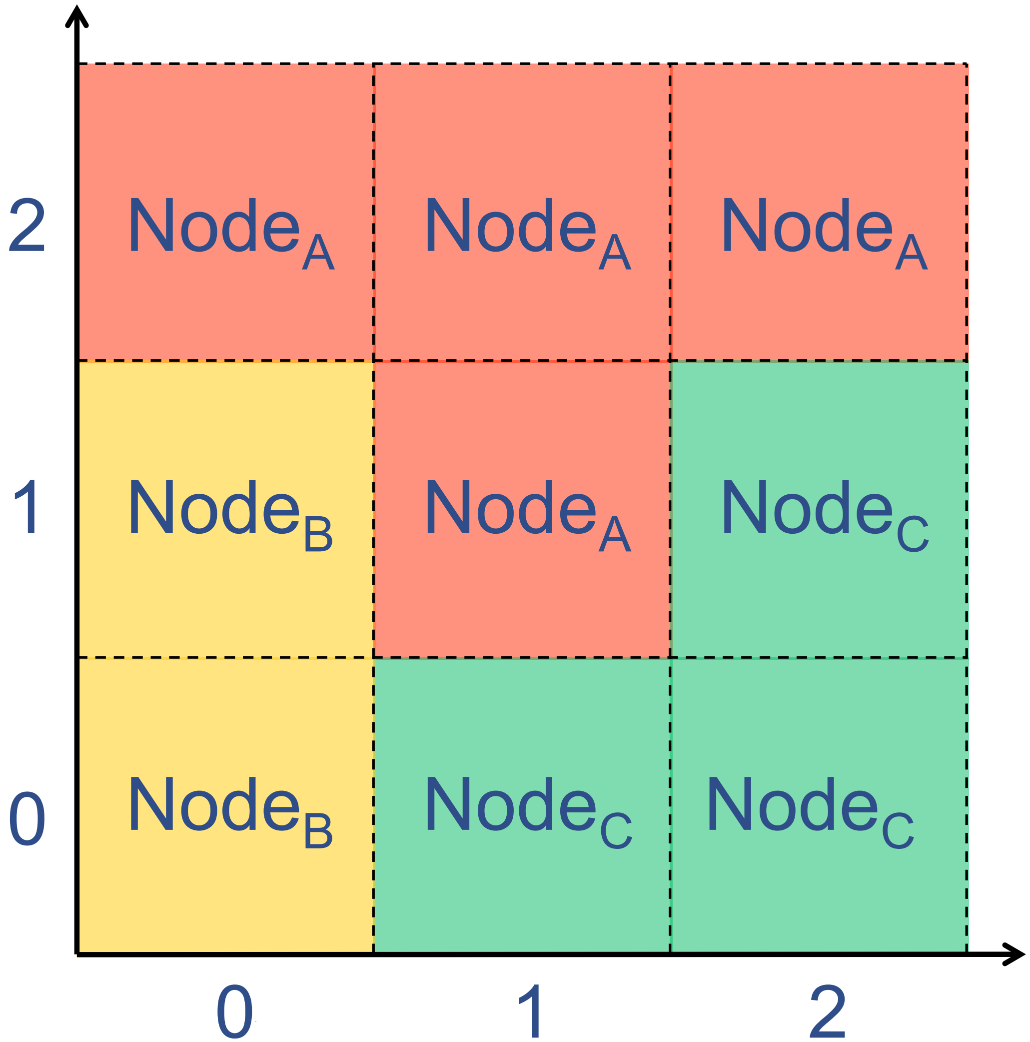
Parallel Interest Matching

- Enhance runtime efficiency by parallel processing
- Two phases
  - First Phase: Spatial Decomposition
  - Second Phase: Sorting and Matching

Space Decomposition

- Decompose the multidimensional virtual space into “flat subdivisions”
- Determine the index for each subdivision
- Work Unit (WU): the interest matching process within a space subdivision
- WU-Node map: contains the information of the space subdivisions that are currently being processed by a node

WU-Node Map



- Node$_A$: WU(0,2), WU(1,1), and WU(1,2)
- Node$_B$: WU(0,0) and WU(0,1)
- Node$_C$: WU(1,0), WU(2,0), and WU(2,1)

Space Decomposition (cont.)

- At the initialisation stage, an equal number of WUs is assigned to each node
- Regions are distributed to different nodes according to the space subdivisions they reside in
  - If a region lies in multiple space subdivisions that are owned by different nodes, it would be distributed to all of them.

Spatial Hashing

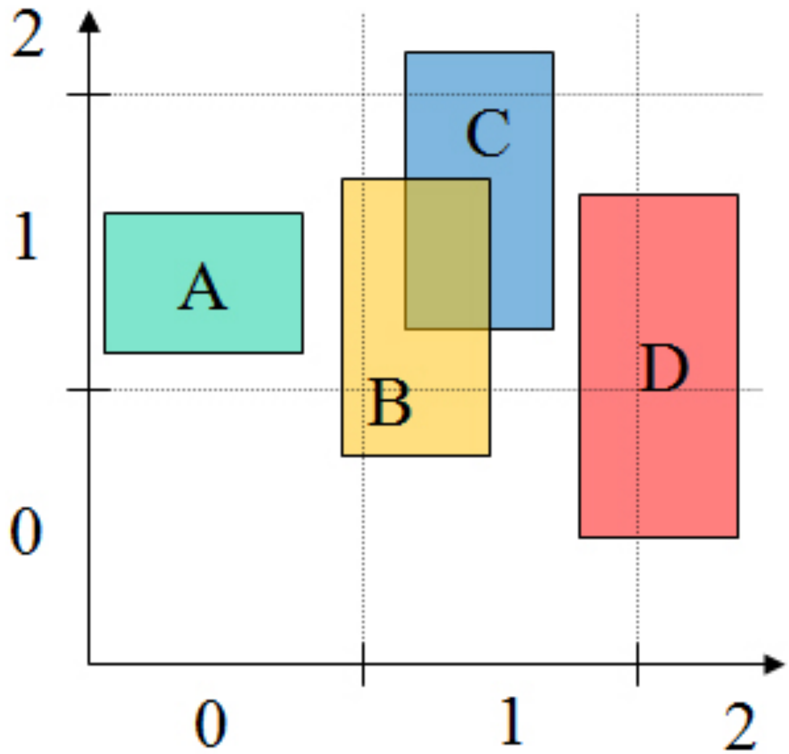
- Position and size of a region may be modified dynamically during simulation
- Owner node is responsible to determine whether the region in question is changing spaces
- Construct a hash table with the indices
- Hash all update regions and subscription regions into the hash table
  - Compute hash value *H(v)*, for each vertex *v* of a region

Hash Function

$$H : \mathbf{R}^n \rightarrow \mathbf{Z}^n, H(x_i) = \lfloor \frac{x_i}{l_i} \rfloor, i = 1, 2, ..., n$$

- x_i: coordinate of vertex on dimension i
- l_i: Length of subdivision on dimension i

Hashing for Space Subdivisions



- A is hashed into (0,1)
- B is hashed into (0,0), (0,1), (1,0) and (1,1)
- C is hashed into (1,1) and (1,2)
- D is hashed into (1,0), (1,1), (2,0) and (2,1)

Hash Table

Table Slot

(0,0)	B
(0,1)	A,B
(0,2)	
(1,0)	B,D
(1,1)	B,C,D
(1,2)	C
(2,0)	D
(2,1)	D
(2,2)	

After Hashing

- Hash table collision => at least two regions are in the same subdivision
- Each slot of the hash table (with collision) represents a WU

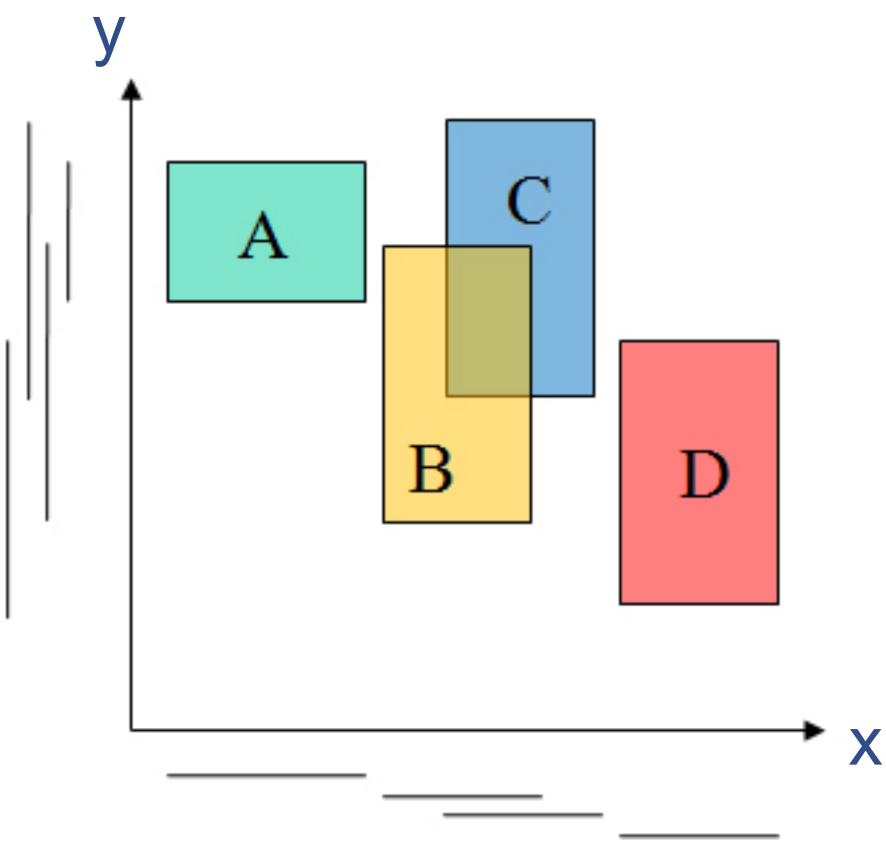
Load Balancing

- Two algorithms
  - (1) Redistribute the WUs of an overloaded node to the least loaded node
  - (2) Redistribute the WUs of an overloaded node to the least loaded neighbour node
- Isolated WUs
  - All adjacent WUs are owned by different nodes
  - Increases the communication overhead of border crossing
- Algorithm (2) decreases the chance of creating isolated WUs

The Second Phase

- A sorting algorithm based on dimension reduction is used to determine the overlapping status of the regions

Dimension Reduction



- X-axis overlaps: B-C
- Y-axis overlaps: A-C, A-B, B-C, B-D, C-D
- 2D overlaps: B-C

Two regions overlap iff their extents overlap on all dimensions

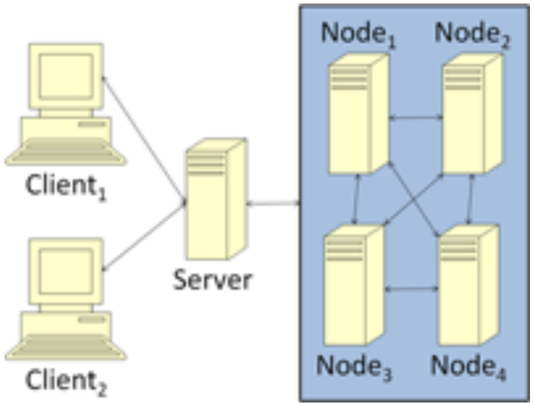
Sorting and Matching

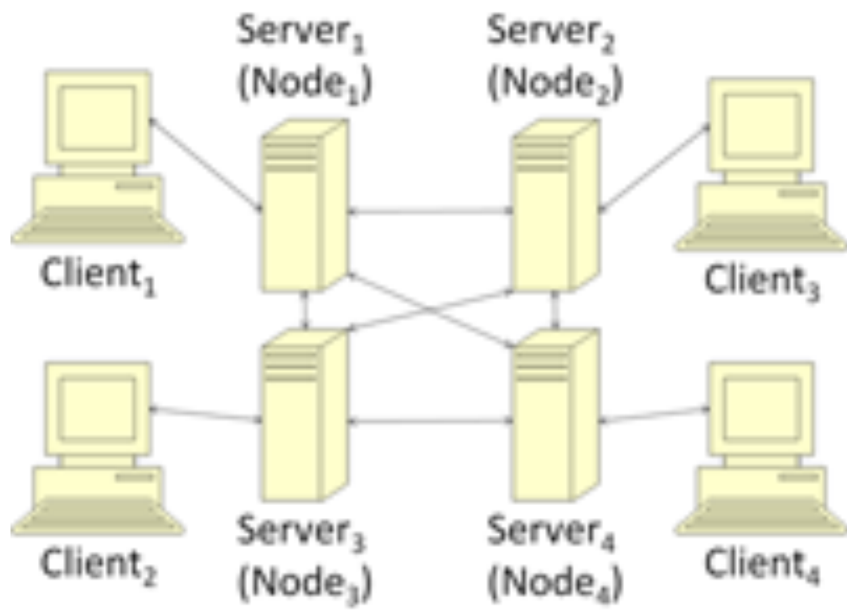
- Construct a list of end-points for each dimension
- Determine which extents overlap by sorting the lists
- Re-sort the lists using insertion sort during runtime

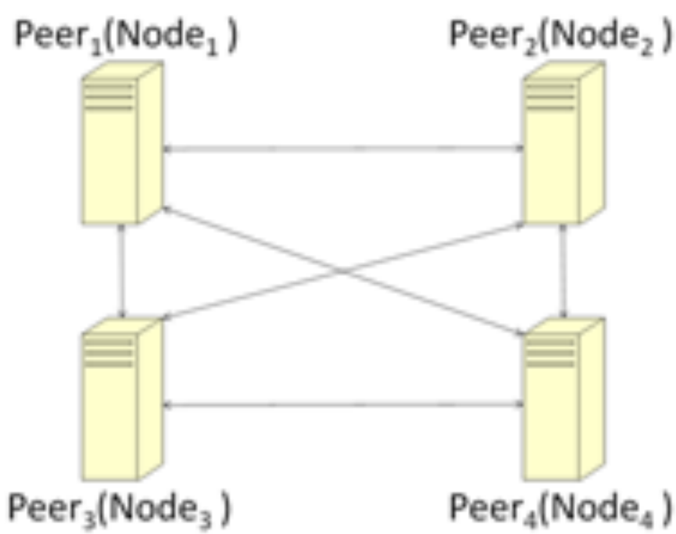
Temporal Coherence

- Assumption: Time-steps are small enough that entities do not travel large distance
  - i.e. Before re-sorting the lists of end-points, they would be nearly sorted
- Insertion sort (with original complexity $O(n^2)$) can be done in linear time

Configuration Scenarios



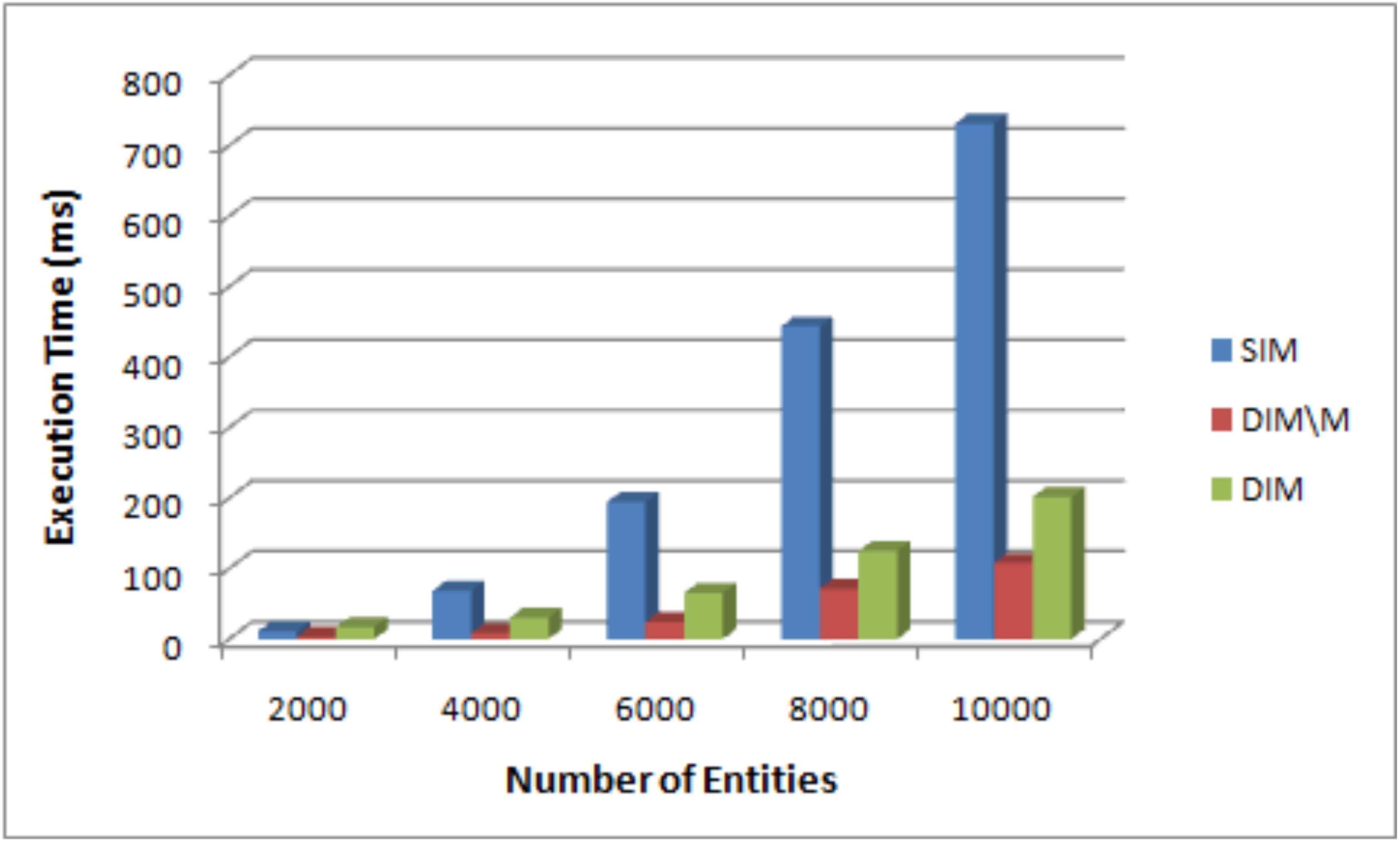




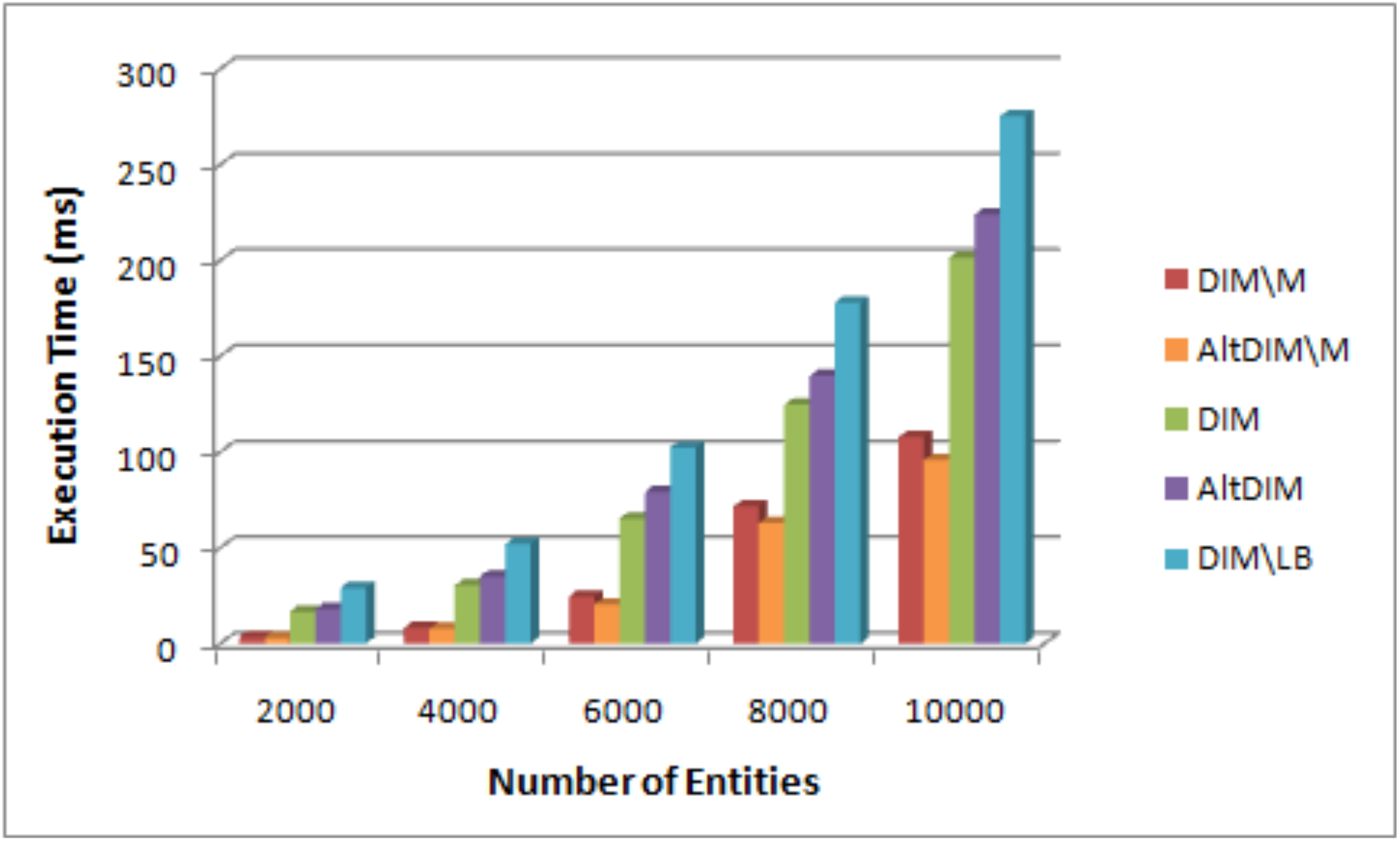
Experiments

- Serial interest matching by sorting algorithm (SIM)
- Parallel interest matching with load-balancing algorithm (1) (DIM)
- Parallel interest matching with load-balancing algorithm (2) (AltDIM)
- Parallel interest matching without load-balancing (DIM\LB)
- DIM without communication overhead (DIM\M)
- AltDIM without communication overhead (AltDIM \m)

Results


Execution Time (ms)
800
700
600
500
400
300
200
100
0
2000
4000
6000
8000
10000
Number of Entities
SIM
DIM\M
DIM

Results



Execution Time (ms)
300
250
200
150
100
50
0
2000
4000
6000
8000
10000
Number of Entities
DIM\M
AltDIM\M
DIM
AltDIM
DIM\LB

Conclusions

- A parallel interest matching approach
- Suitable for distributed-memory systems
- More computationally efficient than existing (serial) sorting algorithms
- High filtering accuracy
- HLA DDM compatible
- Two load-balancing algorithms