Journal of System Simulation

Volume 33 | Issue 5 Article 10

6-9-2021

A Shared Memory Based Parallel Hierarchical Interest Matching Algorithm

Wenjie Tang
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China;

Junwei Cheng
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China;

Yiping Yao
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China;

Zhu Feng
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China;

A Shared Memory Based Parallel Hierarchical Interest Matching Algorithm

Abstract

Abstract: Interest matching plays an important role in distributed simulation. However, because of huge number of simulation entities and frequent change of regions, interest matching consumes tremendous computation in large scale simulations. The ubiquity of multicore urges us to improve the performance of interest matching by parallelization. *A shared memory based parallel hierarchical interest matching algorithm is propose to solve the problem. It maps subscribe regions into a full binary tree, and compares update regions with the tree in parallel. Due to the associative relationship between adjacent nodes, unnecessary comparisons can be eliminated*. The experimental results demonstrate that the proposed algorithm has good scalability and can effectively improve the efficiency of interest matching.

Keywords

interest matching, parallel algorithm, distributed simulation, sort-based matching

Recommended Citation

Tang Wenjie, Cheng Junwei, Yao Yiping, Zhu Feng. A Shared Memory Based Parallel Hierarchical Interest Matching Algorithm[J]. Journal of System Simulation, 2021, 33(5): 1086-1094.

第 33 卷第 5 期 系统仿真学报© Vol. 33 No. 5
2021 年 5 月 Journal of System Simulation May 2021

一种基于共享存储的并行层次兴趣匹配算法

唐文杰，程俊玮，姚益平*，朱峰
(国防科技大学 系统工程学院，湖南 长沙 410073)

摘要：兴趣匹配在分布式仿真中扮演着重要的角色。然而，在大规模仿真场景中，仿真实体数量大且区域变化频繁，导致兴趣匹配的计算量很大，严重影响仿真性能。另一方面，多核处理器的普及也促使从并行视角来提升兴趣匹配算法的性能。针对上述问题，提出了一种*并行层次兴趣匹配算法，将订阅区域映射到一棵满二叉树中，由更新区域并行地与二叉树进行匹配，并利用二叉树相邻节点的区域关联关系来剔除不必要的计算*。实验结果表明：并行层次兴趣匹配算法具有较好的可扩展性，且能够有效提升兴趣匹配的效率。

关键词：兴趣匹配；并行算法；分布式仿真；基于排序的匹配
中图分类号：TP391.9 文献标志码：A 文章编号：1004-731X (2021) 05-1086-09
DOI: 10.16182/j.issn1004731x.joss.20-0019

A Shared Memory Based Parallel Hierarchical Interest Matching Algorithm

Tang Wenjie, *Cheng Junwei*, *Yao Yiping**, *Zhu Feng*
(College of Systems Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: Interest matching plays an important role in distributed simulation. However, because of huge number of simulation entities and frequent change of regions, interest matching consumes tremendous computation in large scale simulations. The ubiquity of multicore urges us to improve the performance of interest matching by parallelization. *A shared memory based parallel hierarchical interest matching algorithm is propose to solve the problem. It maps subscribe regions into a full binary tree, and compares update regions with the tree in parallel. Due to the associative relationship between adjacent nodes, unnecessary comparisons can be eliminated*. The experimental results demonstrate that the proposed algorithm has good scalability and can effectively improve the efficiency of interest matching.

Keywords: interest matching; parallel algorithm; distributed simulation; sort-based matching

引言

在分布仿真中，基于发布/订阅的通信模式能够较好地解耦和联邦成员之间的依赖关系，有效简化系统研制和集成工作。然而，如果无法进行有效的数据过滤，则会在联邦成员之间生成大量不必要的通信，严重影响系统性能。高层体系结构(High Level Architecture，HLA)标准中定义了数据分发管理服务以实现数据的过滤[1]。数据生产者可以利用数据分发管理服务来界定其发送数据的属性(更新区域)，数据消费者利用数据分发管理服务来指定他们的数据需求(订阅区域)。基于此，仿真运行支撑环境可以对更新/订阅区域进行兴趣匹配，仅将生产者的数据分发给“真正需要”的数据消费者，可大大减少无意义的数据传输。由此可见，兴趣匹配在数据分发管理中具有十分关键的作用。

然而，在大规模仿真场景中，仿真实体数量大，且更新/订阅区域变化频繁，导致兴趣匹配的

收稿日期：2020-01-07 修回日期：2020-04-20
基金项目：国家自然科学基金(61702527，61903368)
第一作者：唐文杰(1984-)，男，博士，副教授，研究方向为复杂系统建模与高性能仿真。E-mail: tangwenjie@nudt.edu.cn

计算量很大。因此，如何设计高效的兴趣匹配算法也一直是仿真平台研究中的重要课题。另一方面，多核处理器已成为当前的主流配置，也需要研究人员从并行的视角来重新审视并改进兴趣匹配算法，以获取更高的效率[2-4]。目前，兴趣匹配方法主要有蛮力匹配算法(Brute Force Matching Algorithm)，基于网格的匹配算法(Grid-based Matching Algorithm)，蛮力+网格的混合匹配算法，基于排序的匹配算法(Sort-based Matching Algorithm)。其中，蛮力匹配算法、基于网格的匹配算法、蛮力+网格的混合匹配算法相对简单且容易并行化，但算法本身的时间复杂度较高、匹配效率较低；基于排序的匹配算法利用区间端点之间的序关系，可以减少许多不必要的比较计算，性能较高但相对较难并行。

针对上述问题，本文提出了一种基于共享存储的并行层次兴趣匹配算法(Parallel Hierarchical Interest Matching Algorithm，PHIM)，将订阅区域嵌入到一棵满二叉树中，由更新区域并行地与二叉树进行匹配，并利用二叉树相邻节点的区域关联关系来剔除不必要的计算。实验表明，并行层次兴趣匹配算法具有较好的可扩展性，且能够有效提升兴趣匹配的效率。

1 问题描述

在 HLA 标准中，一个分布式仿真应用被称为一个联邦，该仿真应用中的每一个仿真组件称为一个联邦成员。作为联盟执行的基础设施，运行支撑平台(Run Time Infrastructure，RTI)是 HLA 规范的具体实现，为联邦成员之间互操作提供标准服务，如图 1 所示。联邦成员基于发布/订阅模式进行通信。例如，红方空中兵力邦员要更新导弹、飞机等实体的位置信息，蓝方侦测兵力则需要接收这些信息。在此基础上，HLA 标准还定义了数据分发管理服务以实现数据的过滤。例如，蓝方侦测兵力只接收其雷达探测范围内的飞机信息。数据分发管理服务的核心在于如何进行高效的兴趣匹配。下面先给出一些相关的定义，并解释兴趣匹配问题[1]。

定义 1 维(Dimension)：维表示一个有名字的非负整数区间，由一个有序整数对定义，其下界为 0。

定义 2 路由空间(Routespace)：由若干个维构成的空间。

定义 3 范围(Range)：范围是某个维上的连续半开区间，由一个有序整数对所确定，整数对的第 1 个数称为范围下界，第 2 个数称为范围上界，上界严格大于下界，其差至少为 1。

定义 4 区域(Region)：范围的集合，区域的维度就是其所包含范围所属的维度。一个区域的每个维度上最多只有一个范围。

定义 5 更新区域(Update region)：数据生产者用于表示所产生数据的区域。

定义 6 订阅区域(Subscribe region)：数据消费者用于表示感兴趣数据的区域。

定义 7 范围重叠(Range overlap)：同一维度上 2 个范围 $U=[U_{\text{lower}},U_{\text{upper}}]$，$S=[S_{\text{lower}},S_{\text{upper}}]$ 相交，即 $U\cap S\neq\varnothing$。

定义 8 区域重叠(Region overlap)：两区域重叠当且仅当这 2 个区域的所有公共维度上的范围重叠，如果 2 个区域没有公共维度则不重叠。

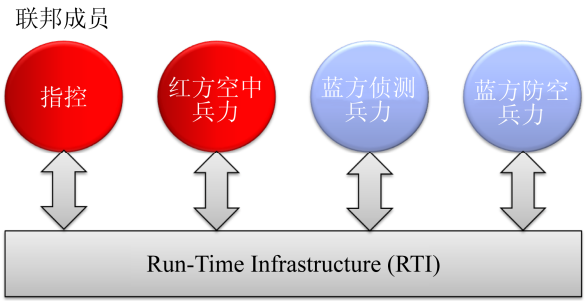

图 1 HLA 分布仿真简述
Fig. 1 HLA-based distributed simulation

兴趣匹配的本质是判断更新区域和订阅区域是否重叠。图 2 展示了二维路由空间中各个更新/订阅区域重叠的情况。其中，更新区域 U_1 与订阅区域 S_1 重叠，更新区域 U_1 与订阅区域 S_2 不重叠。

https://dc-china-simulation.researchcommons.org/journal/vol33/iss5/10
DOI: 10.16182/j.issn1004731x.joss.20-0019

第 33 卷第 5 期 系统仿真学报 Vol. 33 No. 5
2021 年 5 月 Journal of System Simulation May 2021

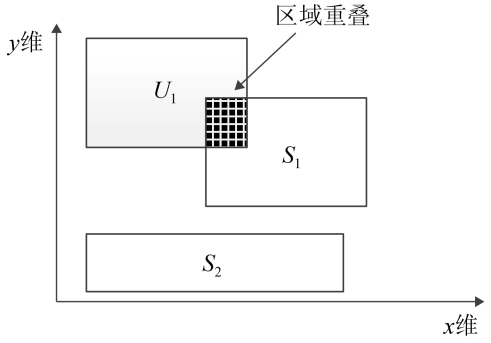

图 2 二维路由空间的区域交叠描述
Fig. 2 Region overlap in a 2-dimensional route space

2 国内外研究现状

经典的兴趣匹配方法主要有蛮力匹配算法，基于网格的匹配算法，基于排序的匹配算法。蛮力算法也称为基于区域的直接匹配，将公布区域与订阅区域逐一进行比较以获得重叠信息。该算法简单且可以获取精确的匹配结果，但由于计算复杂度是 $O(N^2)$，当更新区域和订阅区域的数目很大时，该算法的运行效率十分低下。在基于网格的算法中[4]，路由空间被划分为多个网格，更新/订阅区域都被映射到一个或多个网格中。对于处于同一网格中的更新和订阅区域视为重叠。该算法效率较蛮力算法高，但存在匹配结果不精确的问题。混合算法[3]则是蛮力匹配算法和基于网格的匹配算法的结合。在网格算法的基础上，对每个网格中的更新/订阅区域采用蛮力方法来获得精确的匹配结果。当网格数为 G 时，算法的计算复杂度为 $O(N^2/G)$。无论是网格法或是混合法，计算开销受到区域的数量、空间分布等因素影响，在实际仿真运行中难以确定合理的网格大小。基于排序的兴趣匹配算法[5]，通过将更新/订阅区域的端点进行排序，并利用端点之间的序关系来剔除掉不必要匹配计算，可大大减少兴趣匹配的计算量。上述研究工作中，蛮力匹配算法、基于网格的匹配算法、蛮力+网格的混合匹配算法相对简单且容易并行化，但算法本身的时间复杂度较高、匹配效率较低。基于排序的匹配算法时间复杂度较低但相对较难并行。除了经典的排序匹配算法外，研究人员根据兴趣匹配的动态场景需求，衍生出一系列优化算法[6-8]。

近年来，也出现了一些针对兴趣匹配并行化的工作。例如，梁洪波等[9]提出了一种基于移动相交的并行兴趣匹配算法，利用区域范围移动前后的历史信息，将匹配限定在移动区间之内，减少了大量的无关计算。由于基于历史信息，该算法更适用于部分区域变化的场景。Moreno Marzolla 等[10]提出了一种并行的区间兴趣匹配算法(Interval Tree Matching, ITM)，该算法基于一棵区间树结构来实现匹配过程的并行和不必要计算的排除，能够取得较好的性能。Liu Yanbing 等[11]提出了一种基于序关系的并行匹配算法，包括投射、排序、任务分解、内匹配和外匹配等 5 个阶段，通过 Matlab 实验表明，在区域数量较多时优于串行算法。

此外，随着云计算技术的蓬勃发展，分布仿真已不限于传统的集群环境，也逐步出现了一些云仿真研究工作[12-14]。但目前未见针对云环境的兴趣匹配算法的相关报导。

3 并行层次化兴趣匹配算法

3.1 算法框架

本小节给出算法的基本框架。PHIM 算法依次在各个维对更新区域和订阅区域进行匹配计算。根据定义，如果 2 个区域在所有公共维度上的范围都重叠，则认为 2 个区域重叠。假设整个 Routespace 包含 d 个维度，记为 $\{D_i, i \in [0, d-1]\}$。每个区域都由一个句柄 ID 来唯一标识。算法框架的伪代码如算法 1 所示，包含 3 个部分。

(1) 遍历所有维，在每个维中将订阅区域向各个维的兴趣管理树 T 上映射。后续匹配计算可借助兴趣管理树同层相邻节点的区域关联关系来剔除不必要的计算。(兴趣管理树 T 的细节和怎么映射的详细论述见 3.2 节)

(2) 对于每一个更新区域，计算其在第 0 维中会与哪些订阅区域匹配，匹配结果存储在更新区域

私有的 reglist 中。reglist 是一个位集合(bitset)，每一位对应一个订阅区域。若更新区域和订阅区域相交，设该值为 1，否则为 0。由于这一过程只涉及对兴趣管理树 T 和订阅区域的共享读操作，所以可以在共享存储体系结构下并行运行。

(3) 从第 1 维开始，依次遍历各个维，进行匹配计算。这一步和第(2)步的计算过程一致，主要区别在于要对结果进行归一，以满足在所有公共维度上的范围重叠要求。

算法 1. 并行层次化兴趣匹配算法框架

```
for   i←0 to d–1
    MapSubscribeRegiontoT(D_i);
end for
for each  U_j ∈ UpdateSet  in parallel
  U_j·reglist←Match(U_j,D_0);
end for each
for i←1 to d–1
  foreach  U_j ∈ UpdateSet  in parallel
    U_j·reglist←Match(U_j,D_i) ∩ U_j·reglist
  end for each
end for
```

无论是订阅区域映射，还是更新区域匹配，都是按照每个维独立来开展，为简化描述且不失一般性，下文仅按照一个维来进行论述。因此，不加区分地使用区域和范围，对于订阅区域 S 和更新区域 U，同样用 S 和 U 分别表示在该维上对应的订阅范围和更新范围。对于订阅范围 S 而言，记 LS 为其下界，US 为其上界；对于更新区域 U 而言，记 LU 为其下界，UU 为其上界。

3.2 订阅区域映射

3.2.1 兴趣管理树

对于路由空间 Routespace 的每一个维，PHIM 算法设计了一棵“满二叉树” T (称为兴趣管理树)进行组织管理，如图 3 所示。T 的每个节点 IMNode 管理一个半开区间。假设维的范围为[0, L)，T 的高度为 h，那么总共有 2^h–1 个 IMNode。其中，第 i 层(i=0,1,...,h–1)共有 2^i 个 IMNode。$\text{IMNode}_{i,j}$ 表示第 i 层、第 j 个节点，其管理的区间为 $[2^{-i}Lj, 2^{-i}L(j+1))$，区间长度为 $2^{-i}L$。

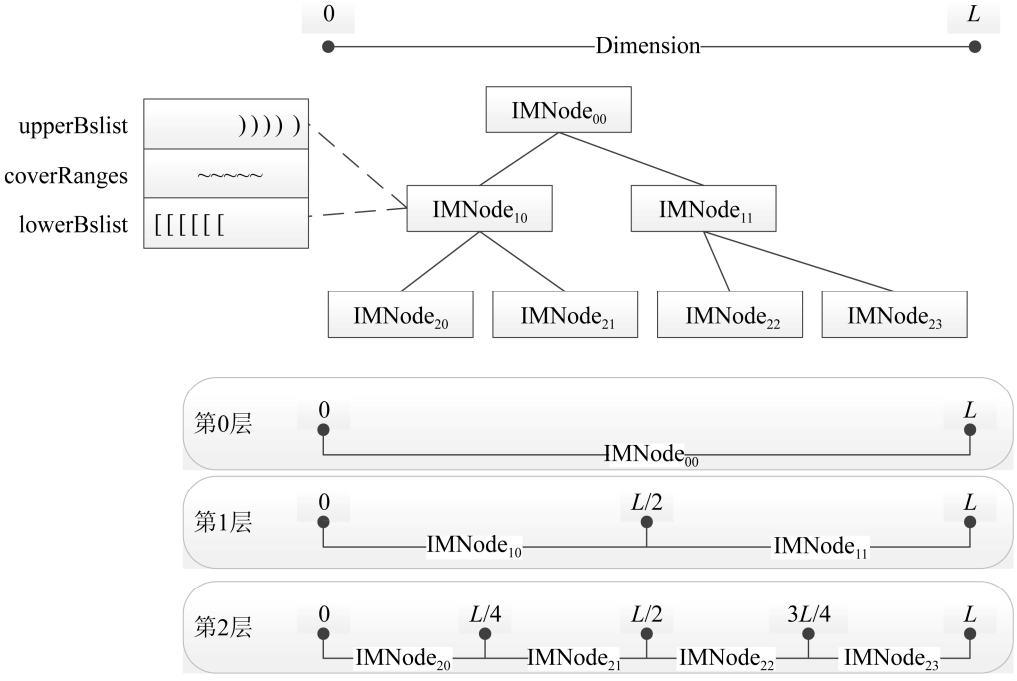

图 3 兴趣管理树结构示意

Fig. 3 Architectural view of an interest management tree

第 33 卷第 5 期 系统仿真学报 Vol. 33 No. 5
2021 年 5 月 Journal of System Simulation May 2021

每个节点 IMNode 包含 3 个数据容器，lowerBslist，upperBslist 和 coverRanges。其中，lowerBslist 是一个以订阅范围下界 *LS* 为键值、订阅区域 ID 为元素的红黑树，用于存储所有下界落入该节点的订阅范围信息；upperBslist 是一个以订阅范围上界 *US* 为键值、订阅区域 ID 为元素的红黑树，用于存储所有上界落入该节点的订阅范围信息；coverRanges 是以订阅区域 ID 为键值的哈希集合，用于存储所有覆盖该节点范围的订阅范围信息。

3.2.2 映射订阅区域

PHIM 算法根据订阅范围的长度计算出其在兴趣管理树的层次，确保订阅范围的长度大于等于该层节点所管理区间的长度，小于上一层节点的管理长度。因此，订阅范围与该层节点的有 3 种相交的情况，即订阅范围与 1 个、2 个或 3 个节点相交，如图 4 所示。一方面，由于订阅范围的长度大于等于该层节点所管理区间的长度，其下界和上界必然位于不同的 2 个 IMNode 中(S_2, S_3)，或者与某个区间的下界和上界完全重合(S_1)；另一方面，根据兴趣管理树的设定，上一层节点所管理区间的长度是本层长度的 2 倍，而订阅范围的长度小于上一层节点的管理长度，所以订阅范围不可能完全包含 2 个节点管理的区间。在确认层次后，PHIM 算法需要计算订阅范围与哪些 IMNode 相交，并把订阅区域 ID 分别加入到这些 IMNode 的 lowerBslist，upperBslist 和 coverRanges 中。

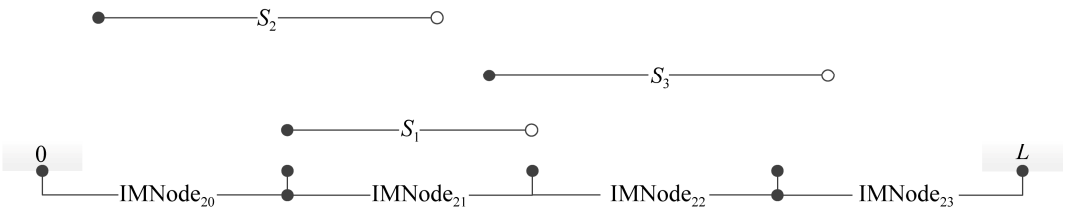

图 4 订阅范围与兴趣管理树相交的 3 种情况

Fig. 4 Three cases that a subscribe region intersects with an interest management tree

映射订阅区域的伪代码如算法 2 所示。兴趣管理树 *T* 初始时，只有一层和一个节点。因此，如果订阅范围的长度比叶子节点的区间长度小，则需要对兴趣管理树扩展。然后，算法确定订阅范围所在的层次 *l*。确保订阅范围的长度大于等于该层节点管理长度，小于上一层节点的管理长度，即

$$L/2^l \leqslant size(S_i) < L/2^{l-1}$$

接着，计算 *lsid* 和 *usid*。其中，*lsid* 表示订阅范围的下界 LS_i 所在的 IMNode 的序号，*usid* 表示订阅范围的上界 US_i 所在的 IMNode 的序号，计算方法如下：

$$lsid = \left\lfloor LS_i \cdot 2^l / L \right\rfloor,\ usid = \left\lfloor US_i \cdot 2^l / L \right\rfloor$$

根据前述相交关系的分析，可知 0≤*usid*–*lsid*≤2。将(LS_i, S_i.*ID*)加入到 $IMNode_{l,lsid}$ 的 lowerBslist 中，将(US_i, S_i.*ID*)加入到 $IMNode_{l,luid}$ 的 upperBslist 中；如果 *usid*–*lsid*=2，意味着订阅范围完全覆盖了 $IMNode_{l,lsid+1}$ 所管理的区间。因此，还需要将 S_i.*ID* 加入到 $IMNode_{l,lsid+1}$ 的 coverRanges 中。

算法 2. 订阅区域映射到兴趣管理树 *T* 上

for $S_i \in$ SubscribeSet

while $size(S_i) < L/2^h$ //扩展 *T*，直至叶子节点管理范围比订阅范围小

通过新建一层节点来扩展 *T*，*h*←*h*+1；

根据 S_i 的长度计算其在 *T* 中的层次 *l*，满足 $L/2^l \leqslant size(S_i) < L/2^{l-1}$；

计算 S_i 的下界 LS_i、上界 US_i 落入的节点编号 *lsid*，*usid*；

$$lsid = \left\lfloor LS_i \cdot 2^l / L \right\rfloor,\ usid = \left\lfloor US_i \cdot 2^l / L \right\rfloor$$

//将(LS_i, S_i.*ID*)加入到 $IMNode_{l,lsid}$ 的 lowerBslist 中

$IMNode_{l,lsid}$.addtolowerBslist(LS_i, S_i.*ID*)；

//将(US_i,$S_i.ID$)加入到 $IMNode_{l,usid}$ 的 *upperBslist* 中

$IMNode_{l,usid}$.addtoupperBslist(US_i, $S_i.ID$);

if *usid*–*lsid*=2

//将 $S_i.ID$ 加入到位于 *lsid* 与 *usid* 之间的 IMNode 的 coverRanges 中

$IMNode_{l,lsid+1}$.addtocoverRanges($S_i.ID$);

endfor

在完成订阅区域映射后，所有订阅范围的信息都被嵌入到兴趣管理树 *T* 中。后续更新范围与订阅范围的匹配计算可完全基于兴趣管理树开展。基于订阅区域的映射方式可知有如下 2 条特性。

特性 1 如果 *S* 属于 $IMNode_{i,j}$ 的 upperBslist 中，那么存在 2 种可能：① *S* 属于 $IMNode_{i,j-1}$ 的 lowerBslist；② *S* 属于 $IMNode_{i,j-2}$ 的 lowerBslist 且 *S* 属于 $IMNode_{i,j-1}$ 的 coverRanges。

特性 2 如果 *S* 属于 $IMNode_{i,j}$ 的 coverRanges 中，那么，*S* 属于 $IMNode_{i,j+1}$ 的 upperBslist 且 *S* 属于其 $IMNode_{i,j-1}$ 的 lowerBslist。

3.3 更新区域在兴趣管理树上匹配

由于兴趣管理树嵌入了订阅区域的所有信息，更新区域与订阅区域的匹配计算可以完全转化为更新区域与兴趣管理树的匹配。PHIM 算法将每一个更新区域从兴趣管理树的叶子节点层开始，在每一层中与所有相交的 IMNode 进行比较，直到根节点，如图 5 所示。基本思路是：如果更新区域 *U* 与某个 IMNode 相交，则其可能与所有由该 IMNode 管理的订阅区域相重叠。基于兴趣管理节点的设计，更新区域可按如下 3 条规则进行匹配计算：

(1) 如果 *U* 的下界与 IMNode 相交(例如，图 4 中的 *U* 与 $IMNode_{21}$)，那么 upperBslist 中键值比 *LU* 大的订阅区域与 *U* 重叠，lowerBslist 和 coverRanges 中的所有订阅区域与 *U* 重叠。

(2) 如果 *U* 包含了与 IMNode 管理的区间(例如，图 4 中的 *U* 与 $IMNode_{21}$)，那么 lowerBslist, upperBslist 和 coverRanges 中的所有订阅区域与 *U* 重叠。

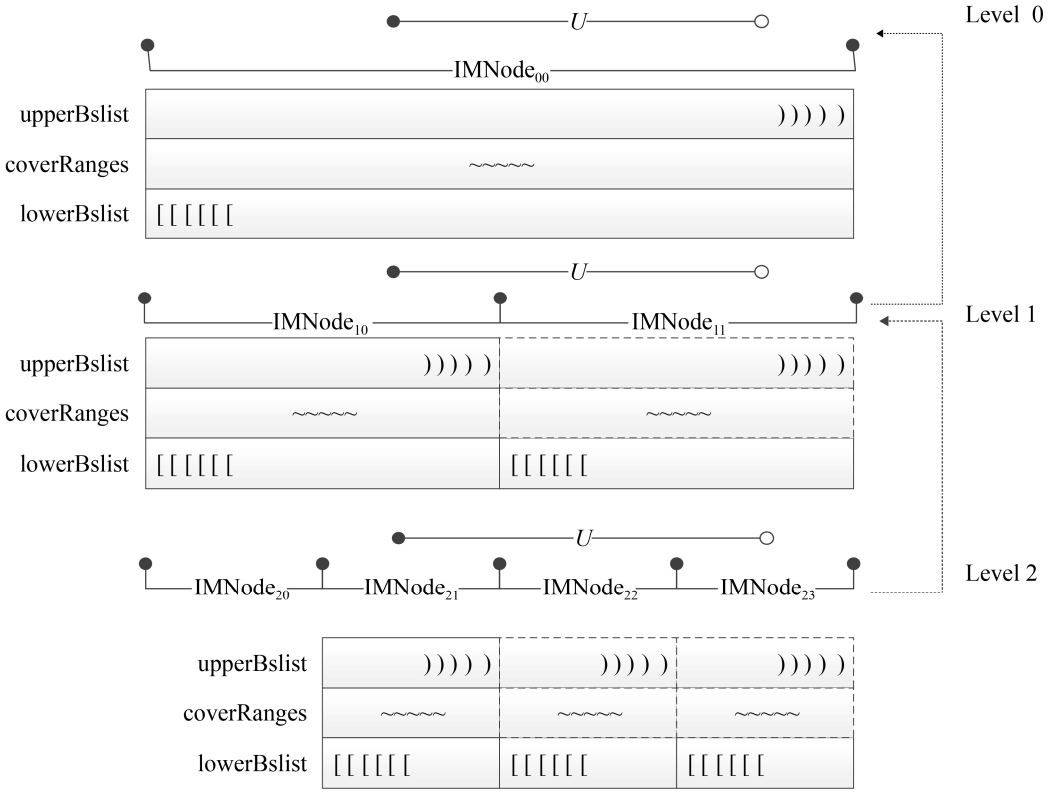

图 5 更新范围与兴趣管理树进行比较的过程

Fig. 5 Procedure that an update region matches an interest management tree

第 33 卷第 5 期 系统仿真学报 Vol. 33 No. 5
2021 年 5 月 Journal of System Simulation May 2021

(3) 如果 U 的上界与 IMNode 相交(例如，图 4 中的 U 与 $IMNode_{22}$)，那么 lowerBslist 中键值比 LU 大的订阅区域与 U 重叠，upperBslist 和 coverRanges 中的所有订阅区域与 U 重叠。

直接运用上述规则虽然能够计算出所有的重叠情况，但也存在较多不必要的重复比较。事实上，可以根据订阅区域映射的特性来消除这些重复。 图 4 中虚线框中的订阅区域都是不需要进行匹配计算的。例如，属于 $IMNode_{22}$ 的 upperBslist 的某个订阅区域 S_1，必然与 U 重叠。然而，根据特性 1，存在 2 种可能：① S_1 属于 $IMNode_{21}$ 的 lowerBslist；② S_1 属于 $IMNode_{21}$ 的 coverRanges。无论哪种情况，都 U 与 $IMNode_{21}$ 的比较中被计算过了。同样，属于 $IMNode_{22}$ 的 coverRanges 的某个订阅区域 S_2，必然与 U 重叠。然而，根据特性 2，S_2 属于 $IMNode_{21}$ 的 lowerBslist，也在 U 与 $IMNode_{21}$ 的比较中计算过了。

基于上述分析，本文设计了在消除重复比较后的匹配计算流程，参见算法 3。算法用 *left* 和 *right* 分别表示与更新范围相交的最左和最右 IMNode 的编号。因为是从叶子节点层开始，所以 *left* 和 *right* 分别为 $\lfloor LU_i \cdot 2^{h-1} / h \rfloor$ 和 $\lfloor UU_i \cdot 2^{h-1} / h \rfloor$。随着层数不断上升，*left* 和 *right* 也要随之更新。在每一层中，算法首先将 $IMNode_{level,right}$ 的 upperBslist 中所有值大于 LU_i 的订阅范围和 coverRanges 中所有订阅范围加入到 $U_i.reglist$ 中；然后，从 *left* 到 *right*–1，将 IMNode 的 lowerBslist 中所有的订阅范围加入到 $U_i.reglist$；最后，将 $IMNode_{level,right}$ 的 lowerBslist 中所有值小于 UU_i 的订阅范围加入到 $U_i.reglist$ 中。

在每一层匹配中，除了最左边节点的 upperBslist 和最右边节点的 lowerBslist 需要进行大小比较计算外，其他情况只需简单地将容器中的数据加入到 *reglist* 即可。而 lowerBslist, upperBslist 都是红黑树，故以 $O(lb\,n)$ 的计算复杂度在 lowerBslist 中找到第一个上界比 LU_i 大的订阅范围，在 upperBslist 中找到最后一个下界比 UU_i 小的订阅范围。

算法 3. 更新区域 U_i 与所有订阅区域进行匹配计算

$left \leftarrow \lfloor LU_i \cdot 2^{h-1} / L \rfloor$, $right = \lfloor UU_i \cdot 2^{h-1} / L \rfloor$; // 记录匹配计算所涉及的最左边和最右边的节点

for *level* ← *h*–1 to 0 by *level* – –

// 从 *l* 层向上直到最顶层

/*将 $IMNode_{level,left}$ 的 upperBslist 中所有值大于 LU_i 的订阅范围 ID 加入到 $U_i.reglist$ 中; */

$U_i.reglist$.insert{$IMNode_{level,left}.upperBslist$.Greater($LU_i$)};

$U_i.reglist$.insert{$IMNode_{level,left}.coverRanges$};

for j ←*left* to *right*–1

//将 $IMNode_{level,j}$ 的 lowerBslist 中所有订购范围 ID 加入到 $U_i.endlist$ 中;

$U_i.reglist$.insert{$IMNode_{level,j}.lowerBslist$.};

endfor

$U_i.reglist$.insert{$IMNode_{level,right}.lowerBslist$.Less($UU_i$)};

left ←*left*/2, *right* ←*right*/2;

endfor

4 实验分析

本节从并行算法的可扩展性和不同算法之间性能对比 2 个方面对 PHIM 算法进行评估。实验设计参考文献[10]，随机产生 N 个一维的更新区域和订阅区域，由算法判断这些更新区域和订购区域重叠的情况。区域大小由覆盖率 α 所决定，表示所有更新/订阅区域大小与整个路由空间大小的比值。在一维情况下，定义为：

$$\alpha = \frac{\sum_{i=1}^{N} size(U_i) + \sum_{i=1}^{N} size(S_i)}{L}$$

实验中，因为所有更新/订阅区域的长度都相同，所以 α 值一旦确定，可以计算出更新区域和订阅区域的长度。一旦长度确定，测试算法会随机确定区域的位置。这样，路由空间就会有大量的更新和订阅区域。通过调用兴趣匹配算法进行计算，可

以得到哪些更新区域和订购区域重叠情况。

虽然 RTI 本身是支持跨计算节点分布运行，但本文算法只是 RTI 服务的一部分，且仅适用于共享存储体系结构(如多核、SMP 等)。测试平台选用一个两路八核服务器，CPU 为 2.53 GHz Intel Xeon E5-2650 处理器，操作系统为 CentOS 6.5，gcc 版本 4.4.7，优化选项为 O3。

4.1 可扩展性分析

下面从处理核心数和更新/订阅区域数 2 个角度来分析算法的可扩展性。

图 6 展示了不同核数设定下 PHIM 算法取得的加速比(N=10^5)。可以看出，随着核数的增加，PHIM 算法的加速比也在增加。尤其是当 α=1 000 时，使用 16 核的 PHIM 算法加速比可以达到 12.24 倍。

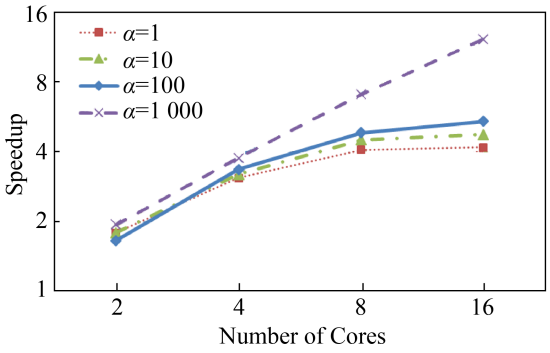

图 6 不同核数设定下 PHIM 算法的加速比
Fig. 6 Speedup of the PHIM algorithm using different number of cores

图 7 展示了不同区域数设定下 PHIM 算法取得的加速比。总体而言，算法可并行部分的比重随 N 增大而增大，所以 PHIM 算法的加速比也在增加。同时，也可以观察到，算法加速效率也随着 α 值增加而增加。这是因为，PHIM 算法是以串行方式进行订阅区域映射，以并行方式进行匹配计算。订阅区域映射的开销主要和区域的数量相关，受 α 影响相对较小。而匹配计算开销与 α 大小正相关。故 α 值越大，可并行部分的占比越高，加速效果越明显。总体而言，PHIM 算法在大 α 值时具有较好的可扩展性。

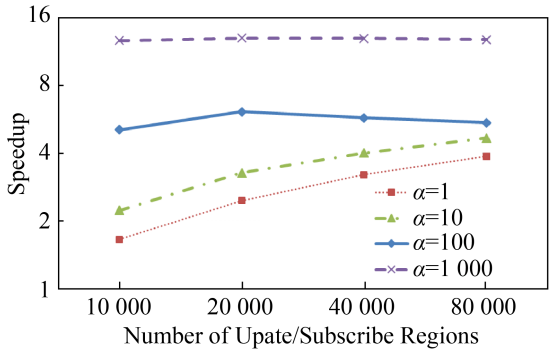

图 7 不同区域数设定下 PHIM 算法的加速比
Fig. 7 Speedup of the PHIM algorithm using different number of update/subscribe regions

4.2 性能比较

图 8 展示了不同 α 取值下 PHIM 算法与基于排序的匹配算法(Sort-based Matching Algorithm，简称 SORT 算法)的性能比较(N=10^5)。其中，PHIM-X 表示使用了 X 个线程(核)的 PHIM 算法。前期研究中，SORT 算法也被普遍认为是效率较高的串行兴趣匹配算法。可以看出，SORT 算法具有较小的时间复杂度，特别是在区域重叠较少的时候有较大的性能优势。但是，当 α=1 000 时，SORT 算法比串行的 PHIM 算法更耗时。当使用多个线程时，PHIM 算法展现出了较好的优势。例如，PHIM-16 相比 SORT 算法性能提升了 9%~2 020%。

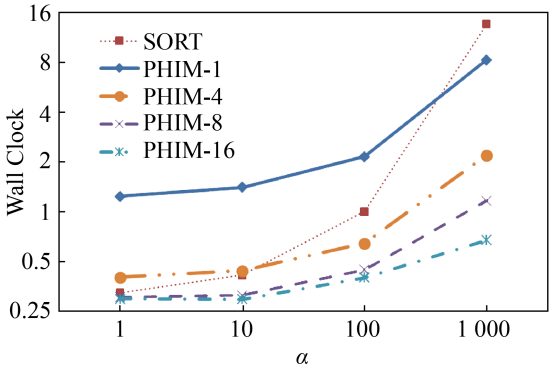

图 8 不同 α 取值下 PHIM 算法与 SORT 算法性能比较
Fig. 8 Performance comparison between the PHIM algorithm and the SORT algorithm using different α

图 9 展示了不同 α 取值下 PHIM 算法与 ITM 算法的性能比较(N=10^5)。同上，ITM-X 表示使用了 X 个线程(核)的 ITM 算法。就运行时间而言，相比于

https://dc-china-simulation.researchcommons.org/journal/vol33/iss5/10
DOI: 10.16182/j.issn1004731x.joss.20-0019

串行算法，能够并行的 ITM 算法可更充分地利用 CPU 计算资源，获得更好的性能。然而总体而言，PHIM 算法在各种情况下都优于 ITM 算法。例如，PHIM-16 相比 ITM-16 算法性能提升了 46%~351%。

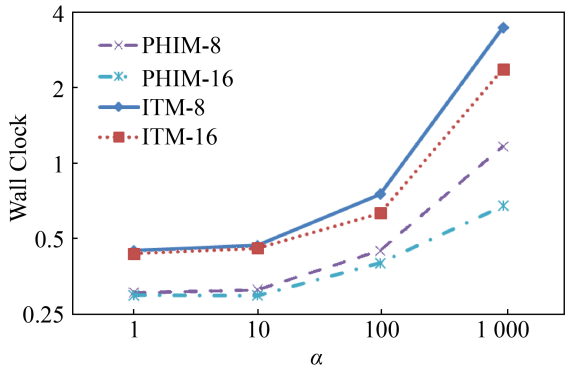

图 9 不同 α 取值下 PHIM 算法与 ITM 算法性能比较
Fig. 9 Performance comparison between the PHIM algorithm and the ITM algorithm using different α

5 结论

本文提出了一种 PHIM 算法，可充分利用多核处理器的并行计算能力，实现对大规模仿真场景下的高效兴趣匹配。相比于现有的串行、并行兴趣匹配算法，PHIM 算法均具有更好的性能。尤其在发布/订阅区域重叠较多的情况下，PHIM 算法更是有显著的优势。

参考文献：

[1] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)[S]. IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000).

[2] Sarbazi-Azad H, Zomaya A Y. Data Distribution Management[M]. Large Scale Network- Centric Distributed Systems. Hoboken, NJ: IEEE, 2014: 103-122.

[3] Tan G, Zhang Y S, Ayani R. A hybrid Approach to Data Distribution Management[C]// Proceedings 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT 2000). Piscataway, NJ: IEEE, 2000: 55-61.

[4] Boukerche A, Roy A. Dynamic Grid-based Approach to Data Distribution Management[J]. Journal of Parallel and Distributed Computing (S0743-7315), 2002, 62(3): 366-392.

[5] Raczy C, Tan G, Yu J. A Sort-based DDM Matching Algorithm for HLA[J]. ACM Transactions on Modeling and Computer Simulation (S1049-3301), 2005, 15(1): 14-38.

[6] Pan K, Turner S J, Cai W, et al. A Dynamic Sort-based DDM Matching Algorithm for HLA Applications[J]. ACM Transactions on Modeling and Computer Simulation (S1049-3301), 2011, 21(3): 1-17.

[7] Li T, Yao Y, Tang W, et al. An Exponential Search Enhanced Dynamic Sort-based Interest Matching Algorithm for Interest Management in Distributed Simulation[J]. Simulation Modelling Practice and Theory (S1569-190X), 2019, 95: 78-95.

[8] Li T, Tang W, Yao Y, et al. A Sort-based Interest Matching Algorithm with Two Exclusive Judging Conditions for Region Overlap[C]// 2018 Winter Simulation Conference (WSC). Piscataway, NJ: IEEE, 2018: 2167-2178.

[9] 梁洪波, 朱卫国, 姚益平, 等. 一种面向大规模 HLA 仿真的并行区域匹配算法[J]. 国防科技大学学报, 2013, 35(3): 84-91.
Liang Hongbo, Zhu Weiguo, Yao Yiping, et al. A Parallel Region Matching Algorithm for Large Scale HLA Simulation[J]. Journal of National University of Defense Technology, 2013, 35(3): 84-91.

[10] Marzolla M, D'angelo G, Mandrioli M. A Parallel Data Distribution Management Algorithm[C]// 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications. Piscataway, NJ: IEEE, 2013: 145-152.

[11] Liu Y, Sun H, Fan W, et al. A Parallel Matching Algorithm Based on Order Relation for HLA Data Distribution Management[J]. International Journal of Modeling, Simulation, and Scientific Computing (S1793-9623), 2015, 6(2): 1540002.

[12] Guan S, De Grande R E, Boukerche A. A Multi-layered Scheme for Distributed Simulations on the Cloud Environment[J]. IEEE Transactions on Cloud Computing (S2168-7161), 2015, 7(1): 5-18.

[13] Nägele T, Hooman J. Scalability Analysis of Cloud-based Distributed Simulations of IoT Systems Using HLA[C]// 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). Piscataway, NJ: IEEE, 2018: 1075-1080.

[14] 李伯虎, 林廷宇, 贾政轩, 等. 智能工业系统智慧云设计技术[J]. 计算机集成制造系统, 2019, 25(12): 3090-3102.
Li Bohu, Lin Tingyu, Jia Zhengxuan, et al. Smart Cloud Design Technology Applied to Intelligent Industrial System[J]. Computer Integrated Manufacturing Systems, 2019, 25(12): 3090-3102.