软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007238] [CSTR: 32375.14.jos.007238]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向天河新一代超算的大规模平行城市交通仿真*

何贤浩[1], 胡逸騵[1], 李毅晨[1], 严宇威[1], 吕宜生[2], 廖 清[3], 李 勇[4], 李肯立[1]

[1](湖南大学 信息科学与工程学院, 湖南 长沙 410082)

[2](中国科学院 自动化研究所, 北京 100190)

[3](哈尔滨工业大学 (深圳) 计算机科学与技术学院, 广东 深圳 518055)

[4](清华大学 信息科学技术学院, 北京 100084)

通信作者: 胡逸騵, E-mail: yikunhu@hnu.edu.cn

摘 要: 随着城市规模不断增加, 城市交通系统面临着越来越多的挑战, 如交通拥堵、交通安全等问题. 交通仿真是一种解决城市交通问题的方法, 其采用虚实结合的计算技术, 以处理实时交通数据、优化城市交通效率, 是平行城市理论在智能交通的重要实现方法. 然而, 传统的计算系统在运行大规模城市交通仿真中会出现计算资源不足、仿真延迟过长等问题. 针对上述问题, 基于平行城市理论, 结合天河新一代超算的异构体系结构, 提出一种平行城市交通仿真并行算法. 该算法能够精确模拟车辆、道路、交通信号等交通要素, 并采取路网划分、车辆并行化行驶、信号灯并行化控制等方法, 以实现高性能交通仿真. 该算法运行在 16 节点、超过 2.5 万核心的天河新一代超算平台, 并针对北京市五环内 240 万辆车、7797 个路口和 17 万条车道的真实交通场景进行仿真. 相比于传统的单节点仿真, 每步仿真时间从 2.21 s 减少到 0.37 s, 取得近 6 倍的加速效果, 在国产超算异构平台上成功实现百万车辆规模的城市交通仿真.

关键词: 平行城市; 数字孪生; 高性能计算; 交通仿真

中图法分类号: TP301

中文引用格式: 何贤浩, 胡逸騵, 李毅晨, 严宇威, 吕宜生, 廖清, 李勇, 李肯立. 面向天河新一代超算的大规模平行城市交通仿真. 软件学报. http://www.jos.org.cn/1000-9825/7238.htm

英文引用格式: He XH, Hu YK, Li YC, Yan YW, Lü YS, Liao Q, Li Y, Li KL. Large-scale Traffic Simulation for Parallel Cities Based on New-generation Supercomputer Tianhe. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000 -9825/7238.htm

Large-scale Traffic Simulation for Parallel Cities Based on New-generation Supercomputer Tianhe

HE Xian-Hao[1], HU Yi-Kun[1], LI Yi-Chen[1], YAN Yu-Wei[1], LÜ Yi-Sheng[2], LIAO Qing[3], LI Yong[4], LI Ken-Li[1]

[1](College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China)

[2](Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China)

[3](School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen 518055, China)

[4](School of Information Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: As the scale of cities continues to increase, urban transportation systems are facing more and more challenges, such as traffic congestion and traffic safety. Traffic simulation is a method to solve urban traffic problems. It uses virtual and real computing technologies to process real-time traffic data and optimize urban traffic efficiency. It is an important method to achieve the parallel city theory in intelligent transportation. However, traditional computing systems often encounter problems such as insufficient computing resources and long simulation delays when running large-scale urban traffic simulations. To solve the above problems, this study proposes a parallel algorithm for traffic simulation of parallel cities based on the parallel city theory and the heterogeneous architecture of China's

* 基金项目: 国家重点研发计划 (2020YFB2104000); 国家自然科学基金青年基金 (62102143)
收稿时间: 2023-09-13; 修改时间: 2024-02-01, 2024-03-25; 采用时间: 2024-06-11; jos 在线出版时间: 2024-11-18

new-generation supercomputer, Tianhe. This algorithm accurately simulates traffic elements such as vehicles, roads, and traffic signals, and applies methods such as road network division, parallel driving of vehicles, and parallel control of signal lights to achieve high-performance traffic simulation. The algorithm runs on Tianhe, a supercomputing platform with 16 nodes and more than 25 000 cores, and simulates real traffic scenarios involving 2.4 million vehicles, 7 797 intersections, and 170 000 lanes within the Fifth Ring Road in Beijing. Compared with traditional single-node simulation, the proposed algorithm reduces the simulation time of each step from 2.21 s to 0.37 s, achieving nearly 6 times acceleration. An urban traffic simulation with a scale of one million vehicles has been successfully implemented on a domestic heterogeneous supercomputing platform.

Key words: parallel city; digital twins; high-performance computing; traffic simulation

随着城市化进程的推进, 城市人口活动日益频繁[1,2], 给城市交通带来了前所未有的压力, 如交通拥堵和交通安全等问题. 为了应对这些挑战, 人们开始积极探索和推动智能交通的建设[3,4]. 城市交通仿真是一种高效构建智能交通的方法, 它将先进的信息技术、通信技术和数据分析技术应用于城市交通管理和运行中, 通过实时的数据采集、分析和交互, 实现城市交通智能化管理和优化[5−7]. 平行城市是一种构建智慧城市的思路和理论框架[8,9], 它将城市中的人、物、事件、基础设施等要素进行数字化和软件化定义, 实现了从物理城市到数字化城市的映射. 平行城市理论通过精确描述和建模物理城市系统, 构建了人工城市系统, 并采用虚实结合的方式对物理城市进行仿真模拟、推演和预测. 目前, 平行城市理论已被北京、青岛、新加坡、法国雷恩等多个城市应用于构建虚实互动的交通系统[8,9]. 平行城市的优势在于其能够提供高度可控的实验环境, 通过对人工城市系统的建模和仿真, 能够对城市系统进行多种情景和策略的测试和评估, 从而帮助决策者制定更科学、更有效的城市规划和管理策略.

随着城市规模的不断扩大, 城市交通道路网络变得错综复杂, 并且车辆数量也在迅速增长, 目前大型城市的核心区域车辆同时运行的数量已达百万级规模. 针对大规模城市的交通仿真, 传统的计算系统会面临以下几个问题: (1) 计算性能不足, 特别是在单节点仿真过程中, 导致计算大规模交通数据的时间过长, 不能满足虚实互动的城市交通仿真的延迟要求; (2) 通信开销过大, 主要因为在并行分布式计算环境下, 随着仿真车辆达到百万级规模, 通信开销会远大于计算开销. 因此, 实现大规模的城市仿真需要使用高性能计算作为重要的技术支撑[10,11]. 在当前的后摩尔时代, 以 CPU 为主导的传统计算系统性能已逐渐成为瓶颈, 许多高性能计算系统纷纷采用加速器以提高计算性能. 针对这种异构体系结构, 需要设计更为复杂的并行方法以充分挖掘其性能潜力[12−14]. 特别地, 国产天河新一代超级计算集群采用由多核处理器和加速器组成的异构计算系统[15], 成功实现了 10^{18} 数量级的计算规模. 利用天河超级计算系统能很好地解决传统计算系统在交通仿真中所遇到的问题. 针对计算性能不足的问题, 根据天河新一代超算的多核 CPU 和加速器的异构体系结构, 设计并行算法以实现高效的路网分析和运行车辆计算; 针对通信开销的问题, 可以设计高效的任务划分方式, 并利用天河新一代超算节点间高速互联网络技术, 以实现更低的传输延迟.

基于上述分析, 本文以平行城市为理论基础, 依托天河新一代超级计算平台的强大算力, 结合城市交通的真实场景, 提出了一种平行城市交通仿真并行算法. 该并行算法的主要目标是利用天河新一代超算的计算能力以实现大规模城市交通模拟, 并设计了一种负载感知的两阶段路网划分策略、车辆行驶的并行策略, 以及信号灯控制的并行策略. 通过这些并行优化方法, 该算法实现了多节点异构计算加速, 完成了北京五环内百万级车辆在天河新一代超算平台上的交通仿真.

综上所述, 本文在交通仿真领域做出了以下贡献.

1) 针对传统计算系统在运行大规模交通仿真遇到的计算资源不足、仿真延迟过长等问题, 提出了一种面向天河新一代超算的平行交通仿真并行算法. 该算法基于平行城市理论, 能够精确模拟车辆、道路、交通信号等交通要素, 实现了对城市交通系统的虚实互动.

2) 针对天河新一代超算的异构体系结构, 该并行算法提出了负载感知的两阶段路网划分策略、车辆并行化行驶策略, 以及信号灯并行化控制策略, 充分利用超算的计算性能以高效地完成仿真任务.

3) 通过在北京五环内 240 万辆车辆、7797 个路口和 17 万条车道的真实交通场景模拟, 展示了该并行算法的强大性能, 在超过 2.5 万核心的 16 个计算节点进行的交通仿真, 相较于传统的单节点仿真, 每步仿真所需时间从 2.21 s 缩短至 0.37 s, 取得了近 6 倍的加速效果.

本文第 1 节介绍城市交通仿真相关问题的定义. 第 2 节介绍天河新一代超算的系统架构. 第 3 节介绍面向天河新一代超算的平行交通仿真并行算法. 第 4 节通过对比实验验证该算法在天河新一代超算进行交通仿真的有效性. 第 5 节总结全文并展望未来工作.

1 问题定义

定义 1. 平行城市. 平行城市的核心在于构建与实际城市等价的人工城市. 平行城市使用城市大数据和物联网感知技术, 对实际城市进行物理建模, 从而构建多尺度、多分辨率的城市运行模拟系统. 另外, 平行城市采用人工智能和统计等方法对城市采样数据进行法分析和预测, 最终实现城市运行过程中的问题诊断和管控.

因此, 平行城市的 3 大特点为: 精准描述、智能预测和主动引导. 首先, 对于构建与实际城市等价的人工城市, 需要将城市运行中的人、环境、基础设施等要素进行“数字化”. 这涉及将城市中的各个实体和属性转化为适当的数据结构和软件模型, 以便在虚拟环境中进行建模和仿真. 这种数字化过程包括对城市物理空间和数字空间之间的关联关系进行建立, 以确保模拟的结果能够准确地反映实际城市的行为和特征.

其次, 平行城市可以借助机器学习、统计等模型, 根据历史城市运行数据来预测未来城市的状态, 例如未来交通状况. 这些预测方法的应用能够为城市规划、交通管理和资源分配等领域提供有价值的信息和决策支持.

最后, 平行城市系统能够根据预测结果和当前城市状态, 主动进行引导和管控. 通过虚实结合的方式, 平行城市系统能够在数字空间中模拟城市运行状态, 并针对不同场景和情况制定合理的规划方案. 例如, 在预测到交通拥堵即将发生时, 平行城市可以通过智能交通信号控制系统调整信号灯配时, 优化交通流动, 减少拥堵现象. 类似地, 在突发应急事件发生时, 平行城市可以迅速响应, 调配资源和人力, 以最佳方式应对危机并最小化影响.

在基于平行城市理论进行交通仿真过程中, 根据平行城市“精准描述”的特点, 需要将现实世界的城市道路交通网络抽象成能用计算机表示的数据结构, 因此, 本文将城市路网转化为图数据结构, 并根据分布式交通仿真任务, 定义了路网划分.

定义 2. 城市路网. 对于城市道路网络, 可将其抽象为一个连通无向图 $G(V,E,A)$, 其中 $V=\{v_1,v_2,\ldots,v_n\}$, $n\in\mathbb{N}$ 表示图 G 的顶点集, $E=\{e_1,e_2,\ldots,e_m\}$ 表示图 G 的边集, $A\in\mathbb{R}^{n\times n}$ 为图 G 的邻接矩阵, $A[i,j]=1$ 表示顶点 v_i 与 v_j 有一条边相连. 在实际应用中, 顶点集 V 表示为交叉口或道路的端点, 边集 E 表示道路, 因此, $A[i,j]=1$ 转化为交叉口 i 和 j 之间有一条道路, 并且对于任意的交叉口 i 和 j, 存在一个路径 $Path=\{e_{a_0},e_{a_1},\ldots,e_{a_s}\}$ 使得 i 和 j 可达. 为了简单起见, $G(V,E,A)$ 可省略边集 E, 从而表示为 $G(V,A)$.

定义 3. 路网划分. 假设连通无向图 $G(V,A)$ 表示为一个城市的道路网络, 一个集合函数: $\pi(G,k)=\{G_1,G_2,\ldots,G_k\}$ 将路网 $G(V,A)$ 划分为不同的子图 $G_i(V_i,E_i,A_i)$ $(i=1,2,\ldots,k)$, 且 $V_i\subseteq V, E_i\subseteq E, A_i=A[V_i]$. 此外, 对于任意两个子图 G_i 和 G_j, 有 $G_i\cap G_j=\varnothing$.

当路网 $G(V,A)$ 被划分为多个子图时, 每个子图 G_i 会分布到不同的计算节点, 并且 G_i 与其他子图 G_j 存在多个边的“连接”, 这是因为存在于 G_i 与 G_j 的顶点在路网 $G(V,A)$ 可能有边相连, 而在传输的数据量相同时, 该“连接”可近似作为 G_i 与 G_j 之间的通信代价.

定义 4. 划分代价. 假设连通无向图 $G(V,A)$ 表示为一个城市的道路网络, 且被集合函数 $\pi(G,k)$ 划分为 $\{G_1,G_2,\ldots,G_k\}$, 现定义两个子图 G_i 和 G_j 的划分代价为:

$$cut(G_i,G_j)=\sum_{\substack{v_p\in V_i\\ v_q\in V_j}}A(p,q) \tag{1}$$

由于路网 $G(V,A)$ 为连通的无向图, $\forall i,j,\ A[i,j]=A[j,i]$, 因此 $cut(G_i,G_j)$ 具有对称性, 即 $cut(G_i,G_j)=cut(G_j,G_i)$.

当路网 $G(V,A)$ 被划分为多个子图时, 定义 4 的划分代价 $cut(G_i,G_j)$ 提供了一种评估划分通信开销的量化方法, 但在大规模分布式交通仿真过程中, 每个计算节点的计算负载也需要进行分析评估. 在实际应用中, 需要根据每个计算节点的体系结构特点, 确保每个节点的负载尽可能地均衡, 在同构节点的条件下, 定义的 ϵ-Balancing 如下.

定义 5. ϵ-Balancing. 假设路网 $G(V,A)$ 被集合函数 $\pi(G,k)$ 划分为 $\{G_1,G_2,\ldots,G_k\}$, 这些划分的子图集合 $\{G_1,G_2,\ldots,G_k\}$ 是 ϵ-Balancing 的, 当且仅当 $\exists\epsilon\geqslant 0,\forall i,|\frac{|V|}{k}-|V_i||\leqslant\epsilon$. 当 ϵ=0 时, 表示 k 个子图的计算负载是均衡的.

2 天河新一代超算架构

随着目前进入百亿亿次计算时代, 超级计算机的架构呈现出多元化的发展趋势. 在后摩尔时代, 通用 CPU 的计算能力已经达到了瓶颈, 大多数高性能计算机采用了加速器作为新的计算资源. 当前市面上的加速器种类较多, 例如英伟达公司的通用 GPU (general-purpose graphic processing unit, GPGPU)[16−19]、英特尔公司的集成众核 (many integrated core, MIC)[20,21]和通用数字信号处理 (general purpose digital signal processing, DPDSP)[22,23]等, 都被广泛地使用以实现高性能计算技术. 由于引入了加速器, 整个高性能计算系统采用的是异构模式, 同时也使得高性能计算系统的体系结构多样化和复杂化, 对实现高性能应用程序以充分挖掘异构系统的算力带来了挑战.

天河新一代超算同样采用异构的体系结构, 这种架构的优势在于能够充分发挥加速器的计算性能, 从而提高应用程序的执行效率. 相较于通用 CPU, 加速器拥有更多的计算单元和更高的内存带宽, 具备更强的计算能力. 通过将主机端的数据传输到加速器上执行计算, 可以快速处理大规模并行任务, 减轻主机端的负载压力. 同时, 加速器计算单元之间采用类似于 SPMD (single program multiple data) 的并行模式, 进一步提升了并行计算效率. 这种异构架构使得国产天河新一代超算能够更好地满足大规模并行计算需求, 提供高性能的计算能力.

天河新一代超算采用的异构系统由多核处理器和 DSP 加速器组成, 不同的计算节点之间采用高速网络进行互联. 超算每个计算节点采用的是 MT-3000 的处理器, 该处理器采用分层结构和多并行的特性, 并具有异构内存一致性模型. 图 1 展示的是天河新一代超算一个节点的示意图, 从图中可以看到, 超算的节点划分了两个不同的域: 通用域和加速域. 通用域包含了 16 个通用 CPU, 并且实现了缓存的一致性机制. 而加速域包含了 96 个控制核和 1 536 个加速核, 并且从图中可以看到, 加速域的核心被划分成了 4 个互相独立的簇, 每个簇拥有自己的全局共享内存 (global shared memory, GSM)、高速带宽共享内存 (high bandwidth shared memory, HBSM) 以及片外 DDR 内存映射区域. 通用域的核心可以访问片外 DDR 内存的所有区域和所有簇的高速带宽共享内存, 而加速域的核只能访问各自的片上内存以及相对应的片外 DDR 内存区域. 此外, 每个加速簇中的 24 个控制核可以独立执行指令和访问内存, 因此每个簇可以独立运行程序.

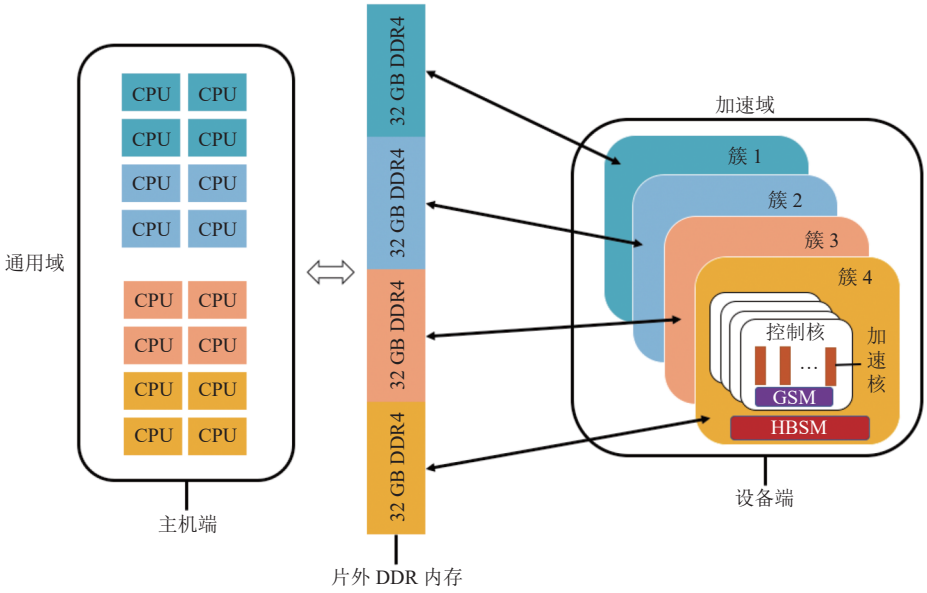


图 1 天河新一代计算节点架构图[24]

3 面向天河新一代超算的平行交通仿真

面向天河新一代超算的平行交通仿真框架如图 2 所示, 该框架主要包括以下几个模块: (1) 将物理空间映射到

数字空间的空间映射模块; (2) 在数字空间进行交通模拟的仿真引擎; (3) 根据仿真结果对实际交通状况进行决策和管控的模块. 从图 2 中可以看到, 平行交通仿真框架利用平行城市理论, 每个模块互相关联且形成了闭环, 实现了虚实结合的城市交通仿真.

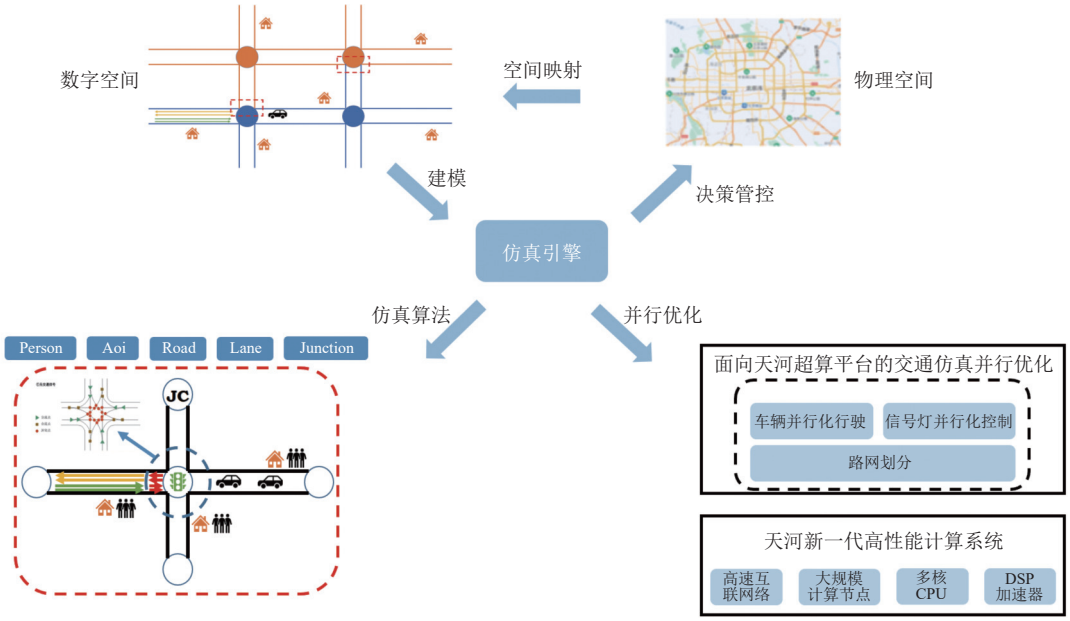


图 2　面向天河新一代超算的平行交通仿真框架图

仿真引擎的核心是微观交通仿真算法, 该算法用于模拟个体车辆的行为和交通流动, 并按照真实的物理场景, 将整个路网划分出了 5 大模块.

- Person: 模拟行人通行, 在仿真引擎中通过开车或行走的方式穿梭于各个 Aoi 之中.
- Aoi: 模拟物理场景中的建筑物, 在仿真引擎中作为车辆出发点或终点, 可连接多条 lane.
- Road: 模拟物理场景中某条道路, 包括正向和反向一条或多条 lane.
- Lane: 模拟道路中的其中一条车道, 车辆行驶在各条 lane 中, lane 根据交叉口位置包含方向信息.
- Junction: 模拟交通中的交叉口, 和 road 相似, 包括正向和反向一条或多条 lane, 同时作为信号灯管控的载体.

根据这 5 个模块, 共同组成了整个路网中的静态位置信息, 当车辆从 Aoi 中行驶上路后, 可根据各个模块对城市交通中的行为进行模拟, 利用信号灯调控交通网络等.

对于像北京、上海等特大型城市, 其交通网络过于复杂且庞大, 如果使用单个计算节点或传统计算系统进行模拟仿真, 节点的计算负载过大, 增大了每一步仿真的延迟. 天河新一代超算由于异构的体系结构特性, 可以提供高性能的计算能力, 并大幅提高仿真计算效率. 为了能够实现大型城市交通仿真在天河新一代超算上运行, 且发挥其高性能的特性, 还需要设计一种高效的并行算法. 本文提出的并行交通仿真算法如算法 1 所示, 该并行算法以交通仿真应用的需求为切入点, 设计并实现了包括路网划分、车辆并行化行驶、信号灯并行化控制等方法, 使得城市交通仿真能运行在大规模的计算节点上, 且尽可能地保持并行效率.

算法 1. 交通仿真并行算法.

输入: 车辆、道路等信息, 计算节点数 n;
输出: 仿真结果.

1. 构建相应的数据结构
2. 将任务划分为 n 个子任务 T_i $(i=1,\ldots,n)$, 且每个子任务分配给一个计算节点
3. **do**
4. // 任务并行
5. **parallel_for** all T_i
6. **parallel_for** all vehicles in T_i
7. 车辆行驶算法
8. **end for**
9. **parallel_for** all lanes in T_i
10. 车道状态更新
11. **end_for**
12. **parallel_for** all junctions **in** T_i
13. 向量化计算交叉口每个相位的压力
14. 根据压力值, 向量化执行交叉口每个相位的信号灯调整策略
15. **end_for**
16. // 任务之间通信同步
17. synchronize T_i
18. **end for**
19. **until** 仿真结束
20. **return** 仿真结果

3.1 路网划分

算法 1 的第 2 行对任务进行了划分, 每个子任务分配到不同的节点上, 从而降低了每个节点的计算负载. 值得注意的是, 在该并行模式中, 不同的计算节点往往需要进行数据通信以实现协同计算 (如算法 1 的第 17 行), 进而使得任务划分需要考虑如何降低通信开销.

一种较为直观的思路是将车辆数进行平均划分, 每个节点保存相同数量的车辆状态, 考虑交通仿真是按照相同步长执行的特点, 即每经历一个步长, 所有仿真对象执行一次, 每一步仿真时需要进行全量的车辆状态信息同步. 如图 3 所示, 假设此时有 4 辆车, 将它们划分到 4 个计算节点, 每个节点保存完整的路网信息并且有一辆车执行仿真任务, 每一步仿真结束后, 每一个节点的车辆 (如图 3 左上) 需要将其状态信息 (如车速、位置等) 同步到其他节点, 以保证其他节点的车辆能感知它的状态.

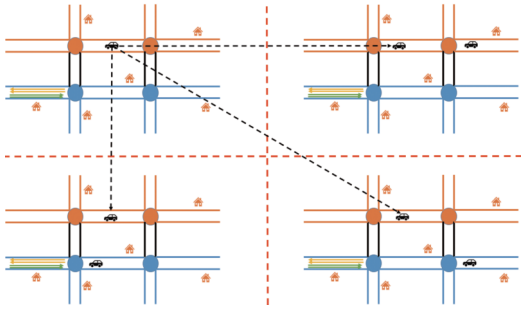


图 3 车辆均匀划分策略

这一种任务划分方法的优点在于简单直接, 不需要考虑复杂的路网结构, 只需要将车辆进行均匀划分即可. 但该方法的缺点也很明显, 每次仿真都需要进行全节点的车辆状态同步, 这样会造成巨大的网络通信开销, 因此可以

考虑路网的划分方案. 具体而言, 在交通仿真过程中, N 个交叉口的坐标为 $J_i(x_i,y_i)\ (i=1,2,\ldots,N)$, 交叉口作为道路之间车辆数据交互的节点, 可将交叉口 J_i 作为图顶点 v_i, 道路作为边, 从而将路网转化为图结构 $G(V,A)$, V 表示顶点集合, A 表示邻接矩阵, 如果 $A[i,j]=1$, 则表示图中顶点 v_i 和 v_j 有边相互连接.

将路网 G 被划分成 n 个子图 $S=\{G_1,G_2,\ldots,G_n\}$, 需要考虑以下两个问题: (1) 每个子图的车辆数据通信开销尽可能小; (2) 每个子图的计算负载尽可能均衡. 假设 $C(G_i)$ 表示子图 G_i 的计算负载, $T(G_i,G_j)$ 表示子图 G_i 和 G_j 的通信开销, 每个顶点要传输的车辆数据量相同, 因此路网划分的优化目标为:

$$S_1=\mathop{\arg\min}_{\{G_1,G_2,\ldots,G_n\}}\sum_{i=1}^{n}\sum_{j=1}^{n}T(G_i,G_j)+\left||V_i|-\frac{|V|}{n}\right| \tag{2}$$

其中, 第 1 项表示子图之间的通信开销, 第 2 项 $\left||V_i|-\frac{|V|}{n}\right|$ 为 ϵ-Balancing 的定义, 用于表示计算的负载.

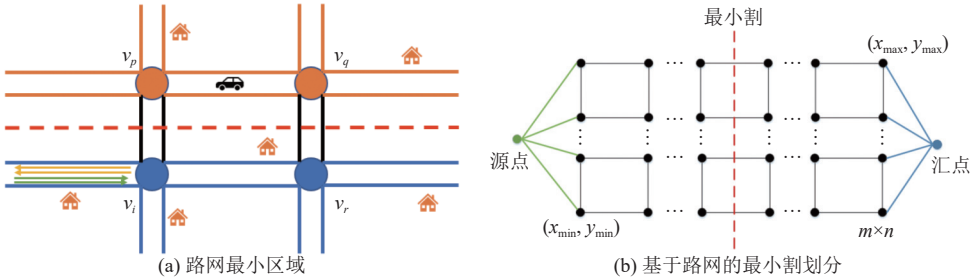
对于优化计算负载, 尝试尽可能保证 n 个子图 $\{G_1,G_2,\ldots,G_n\}$ 的顶点数相同, 由于天河超算平台的每个节点的体系结构相同 (多核 CPU+DSP), 这样每个子图 G_i 在超算节点的负载基本相同. 而对于通信开销的优化, 需要从图内通信和图间通信两部分考虑, 即:

$$T(G_i,G_j)=T_{\text{in}}(G_i)+T_{\text{in}}(G_j)+T_{\text{out}}(G_i,G_j) \tag{3}$$

其中, $T_{\text{in}}(G_i)$ 和 $T_{\text{in}}(G_j)$ 表示 G_i 和 G_j 内部交叉口之间车辆数据的通信开销, 而 $T_{\text{out}}(G_i,G_j)$ 表示车辆数据在 G_i 和 G_j 之间进行通信的开销. 为了最小化 $T_{\text{in}}(G_i)$, 可以在划分阶段使 G_i 为一个连通图, 这样 G_i 内部交叉口之间车辆数据交互不需要跨网络通信, 极大减少通信开销, $T_{\text{out}}(G_i,G_j)$ 的通信开销取决于 G_i 和 G_j 之间的连接拓扑结构, 即 G_i 和 G_j 连接的边数越少, G_i 和 G_j 需要通信的开销越少. 因此, 根据定义 4 的划分代价, 优化目标 S_1 可以转化为:

$$\begin{aligned}S_2&=\mathop{\arg\min}_{\{G_1,G_2,\ldots,G_n\}}\sum_{i=1}^{n}\sum_{j=1}^{n}cut(G_i,G_j)+\left||V_i|-\frac{|V|}{n}\right|\\&=\mathop{\arg\min}_{\{G_1,G_2,\ldots,G_n\}}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{v_p\in G_i\\v_q\in G_j}}A[p,q]+\left||V_i|-\frac{|V|}{n}\right|\end{aligned} \tag{4}$$

公式 (4) 的 $\sum_{v_p\in G_i,v_q\in G_j}A[p,q]$ 表示 G_i 和 G_j 的连接数. 现有的主流图划分算法能较好地实现划分策略, 但这些算法并未考虑路网 G 中每个顶点的地理位置. 目前, 城市大部分的道路采用“网格”结构连接, 如图 4(a) 所示, 这种“网格”结构可定义为一个路网最小区域.



(a) 路网最小区域　　(b) 基于路网的最小割划分

图 4　路网划分基本策略

定义 6. 路网最小区域. 对于“网格”结构的 G, $\forall v_i\in V,\exists v_p,v_q,v_r$, 使得 $A[i,p]=A[p,q]=A[q,r]=A[r,i]=1$, 且 $x_i\approx x_p,\ x_r\approx x_q,y_i\approx y_r,y_p\approx y_q$.

路网 G 可视为多个路网最小区域组成的结构, 因此可以采用“水平”或“竖直”的边切分方式, 将 G 划分为不同子图.

定理 1. 由多个路网最小区域组成的路网 G, 采用“水平”或“竖直”的边切分方式划分为 G_1,G_2, 则 $\sum_{v_p\in G_1,v_q\in G_2}A[p,q]$ 最小.

证明: 假设路网 G 由 $m\times n$ 个路网最小区域组成, 则 G 是连通的. 如图 4(b) 所示, 构建源点 $s(x_s,y_s)$ 和汇点 $e(x_e,y_e)$, 且 $\forall v_i\in V,\ x_s<x_i<x_e$, 令 $x_{\min}=\min\{x_i\}$, $x_{\max}=\max\{x_i\}$, 构建顶点集 V_1 和 V_2, 使得 $\forall v_p\in V_1,\ \forall v_q\in V_2$, 有 $x_p\approx x_{\min}, x_q\approx x_{\max}$, 将源点 s 与 V_1 的每个顶点建立一条边, 而汇点 e 与 V_2 的每个顶点建立一条边, 权重都为 1. 根据最大流最小割定理, 图 4 中“竖直”虚线为最小割边, 最大流为 $m+1$. 此外, 源点 $s(x_s,y_s)$ 和汇点 $e(x_e,y_e)$ 的位置还能满足条件: $\forall v_i\in V,\ y_s<y_i<y_e$, 令 $y_{\min}=\min\{y_i\}$, $y_{\max}=\max\{y_i\}$, 此时顶点集 V_1 和 V_2 的顶点满足 $\forall v_p\in V_1,\ \forall v_q\in V_2$, 有 $y_p\approx y_{\min}, y_q\approx y_{\max}$, 同样地, 将源点 s 与 V_1 的每个顶点建立一条边, 而汇点 e 与 V_2 的每个顶点建立一条边, 权重都为 1. 同理, 根据最大流最小割定理, 存在某条“水平”线为最小割边, 且最大流为 $n+1$. 此时, $\sum_{v_p}\in G_1,\ v_q\in G_2 A[p,q]=\min\{m,n\}+1$.

定理 1 提供了一种“水平”或“竖直”的划分方法使得通信开销最小, 接下来只需要每个划分的子图节点数相同即可完成 S_2 的优化. 目前, 四叉树 (quad tree) 和 KD 树 (KD-tree) 等划分方法沿着“水平”和“竖直”方向将路网划分为若干相等区域, 可以满足通信开销最小的条件, 但对于每个划分的区域, 并不能保证其计算负载是相同的. 为此, 提出了一种两阶段划分方案将路网划分成 $(\sqrt{n}\times\sqrt{n})$ 个分块, 其主要算法包括以下 6 个步骤.

(1) 确定每个顶点 (交叉口) 的二维坐标.

(2) 将每个顶点按照 x 坐标排序.

(3) 按照排序后的 x 坐标将路网分成 $\sqrt{n}$ 个子图, 每个子图包含相同数量的顶点.

(4) 每个子图的顶点按照 y 坐标排序.

(5) 按照排序后的 y 坐标对每个子图再次分成更小的 $\sqrt{n}$ 个子图, 每个更小的子图包含相同数量的顶点.

(6) 每个子图分配到一个计算节点.

两阶段划分算法第 1 次先将顶点 (交叉口) 按照 x 坐标排序, 并其划分成 $\sqrt{n}$ 个子图, 每个子图的顶点数为 $\frac{|V|}{\sqrt{n}}$, 第 2 次在每个子图的顶点按照 y 坐标排序, 并将每个子图再次分成更小的 $\sqrt{n}$ 个子图, 此时总共分成了 n 个子图, 每个子图的顶点数为 $\frac{|V|}{n}$, 根据 ϵ-Balancing 的定义, $\forall i<n,\ |V_i|-\frac{|V|}{n}=0$, 因此该算法是 0-Balancing 的. 而对于四叉树和 KD 树的划分, $\exists i<n,\ |V_i|-\frac{|V|}{n}>0$, 从而存在 $\epsilon>0$, 使得 $\forall i<n,\ -\epsilon\leqslant|V_i|-\frac{|V|}{n}\leqslant\epsilon$. 综上, 两阶段划分算法相较于四叉树和 KD 树具有更优的计算负载.

3.2 车辆行驶及其并行优化

车辆在道路上行驶主要遵从 Krauss 模型 (跟驰模型)[25]. Krauss 模型是安全距离类模型, 当一辆车突然进行刹车或减速时, 安全距离模型假设后面的车辆能够迅速察觉到前车的行为, 并有足够的时间来做出反应. 这包括驾驶员注意到前车制动的信号、反应时间内减速以及最终停车, 以避免发生碰撞. 具体而言, 假设前车 (before) 与后车 (after) 之间的车距为:

$$g=x_b-x_a-l \tag{5}$$

其中, l 表示车本身长度, 如果要求后车不撞前车, 需要满足:

$$L(v_a)+v_a\tau<L(v_b)+g \tag{6}$$

其中, $L(v_a)$ 表示后车刹车距离, $L(v_b)$ 表示前车刹车距离, v_a 表示后车速度, τ 表示刹车时间, g 表示车间距, 为了计算 v_a, 假设在极短时间内, 近似速度可表示为 $\bar{v}=(v_a+v_b)/2$, 泰勒展开后对 $\bar{v}$ 求一阶导数即可得到:

$$L'(\bar{v})v_a+v_a\tau<L'(\bar{v})v_b+g \tag{7}$$

假设刹车时加速度为 $-b(v)$, 则有:

$$L'(v)=\frac{\mathrm{d}}{\mathrm{d}v}\int_0^v\frac{s}{-b(s)}\mathrm{d}s=\frac{v}{b(v)} \tag{8}$$

公式 (8) 的积分项表示为刹车距离. 在模拟过程中, 车辆会根据跟驰模型自行判断当前加速和减速, 保证最大安全跟车速度, 并根据交叉口信号灯模块, 合理规划行车路线, 模拟行驶. 因此, 车辆行驶算法的主要包括以下几个步骤.

(1) 获取驾驶环境, 如周围车辆的信息、车道信息 (如限速、是否在交叉口等)、车辆自身的状态.

(2) 计算车辆驾驶行为, 如前进的加速度、是否变道, 以及变道长度等.

(3) 根据步骤 (2) 得到的驾驶行为进行车辆状态更新, 如位置、车速.

而对于车辆变道的行为, 主要包括以下几个步骤.

(1) 获取目标车道和距离, 判断是否存在目标车道.

(2) 获取前后车辆信息, 并计算前后车辆的相对距离.

(3) 如果车辆准备进行车道变换, 判断是否直接变道会导致碰撞或者被后车追尾. 如果是, 则设置车辆状态为等待. 否则, 允许车辆进行变道.

(4) 如果车辆变换车道, 则更新前后车辆的信息.

在大规模交通仿真任务中, 车辆的运行状态时刻在改变, 由于每个节点内的车辆数量较多, 需要利用高性能计算系统的多核特点, 设计并行处理算法. 根据上述的车辆模拟流程, 可以观察到车辆在当前运行时的状态仅与其上一个时间点的状态相关, 而与其他车辆的状态是相互独立的. 基于这种特性, 可以在每个计算节点内为每辆车分配一个执行线程. 在具体实现过程中, 采用 OpenMP 框架以完成并行化的车辆状态计算. 每个执行线程负责处理一个车辆的状态更新, 并且线程之间可以独立地执行计算任务. 这样可以充分利用计算节点上多核 CPU 的多个计算核心, 并行地计算车辆状态, 从而提高整体计算性能.

此外, 在车辆执行的过程中, 车道也需要根据车流的变化更新其状态信息, 如当前车道中车辆的位置信息. 与行车算法相同, 每个车道状态的更新流程同样与其他车道相互独立, 采用行车并行算法的思路, 将每个车道的更新分配到一个执行线程. 同样地, 采用 OpenMP 框架, 完成并行化的车道状态更新算法.

3.3 信号灯控制及其并行优化

本交通仿真以相位 (phase) 作为信号灯的控制粒度, 信号灯相位主要是指交通信号灯在不同时间段内显示的不同信号状态. 通常, 信号灯相位包括绿灯、黄灯和红灯, 用于控制交通流量和行车安全. 八相位模式是一种高级的交通信号灯控制系统, 用于优化交叉路口的交通流量, 按照空间位置即东西南北方向进行划分. 该模式将信号灯划分为 8 个不同的相位, 每个相位对应于特定的交通流动模式. 通过精确控制每个相位的时间间隔和顺序, 八相位模式可以根据交通需求和优先级来动态调整交通信号, 以最大程度地提高交通效率和安全性. 此外, 采用最大压力法[26,27]计算交叉口各相位的压力值, 以调整信号灯的显示控制. 交通仿真中关于信号灯控制的模块有: 压力计算模块和策略调整模块.

对于压力计算模块, 具体来说, 若将从车道 l 穿过交叉口 i 到车道 m 的交通运动表示为 (l,m), 交通运动 (l,m) 的运动信号表示为 $a(l,m)$, 其中, 运动信号包括绿灯允许通行的允许动向信号和红灯禁止通行的禁止动向信号, $a(l,m)=1$ 表示运动信号为允许动向信号, $a(l,m)=0$ 表示运动信号为禁止动向信号. 交叉口中一个相位从绿灯切换为红灯再切换为绿灯的循环为一个控制阶段, 将一个控制阶段表示为 $p=\{(l,m)|a(l,m)=1\}$. 交通运动 (l,m) 的压力可以通过公式 (9) 进行计算:

$$w(l,m)=\frac{x(l)}{x_{\max}(l)}-\frac{x(m)}{x_{\max}(m)} \tag{9}$$

其中, $w(l,m)$ 为交通运动 (l,m) 的压力, $x(l)$ 为车道 l 上的车辆数量, $x(m)$ 为车道 m 上的车辆数量, $x_{\max}(l)$ 为车道 l 上允许承载的最大车辆数量, $x_{\max}(m)$ 为车道 m 上允许承载的最大车辆数量, $x(l)/x_{\max}(l)$ 为车道 l 上的车辆密度, $x(m)/x_{\max}(m)$ 为车道 l 上的车辆密度. 交叉口 i 的压力可以通过公式 (10) 进行计算:

$$P_i=\left|\sum_{(l,m)\in i}w(l,m)\right| \tag{10}$$

其中, 交叉口 i 的压力 P_i 为交叉口 i 内所有交通运动的绝对压力之和.

对于策略调整模块, 用于根据所述最大压力的计算结果, 动态调整所述交通信号灯的显示策略. 在实际应用中, 通过控制决策时间, 根据以下动态调整公式对当前通行相位进行动态调整:

$$D_n(k_n)=\begin{cases}\text{延长1},\ \sigma_{\max}(k_n)=\sigma^*(k_n)\\ \text{延长2},\ \sigma_{\max}(k_n)\neq\sigma^*(k_n)\ \text{and}\ (1+\delta)P_{n,\sigma^*(k_n)}\geqslant P_{n,\sigma_{\max}(k_n)}\\ \text{激活},\ \sigma_{\max}(k_n)\neq\sigma^*(k_n)\ \text{and}\ (1+\delta)P_{n,\sigma^*(k_n)}<P_{n,\sigma_{\max}(k_n)}\end{cases}\tag{11}$$

其中, $D_n(k_n)$ 表示交叉口控制器的决策, $\sigma^*(k_n)$ 表示当前通行相位, 延长 1 为将绿灯显示时间延长第 1 预设时间, 第 1 预设时间可以设置为绿灯可延长时间的最大值. 延长 2 为将绿灯显示时间延长第 2 预设时间, 第 2 预设时间小于第 1 预设时间. $P_{n,\sigma^*(k_n)}$ 为交叉口 n 当前通行相位的压力值, $P_{n,\sigma_{\max}(k_n)}$ 为交叉口 n 最大压力所在相位压力值, δ 为预设百分比, 当 $\sigma_{\max}(k_n)\neq\sigma^*(k_n)$ 时, 说明当前通行相位不为最大压力所在相位, 此时判断 $(1+\delta)P_{n,\sigma^*(k_n)}$ 和 $P_{n,\sigma_{\max}(k_n)}$ 的压力值大小, 若 $(1+\delta)P_{n,\sigma^*(k_n)}\geqslant P_{n,\sigma_{\max}(k_n)}$, 说明当前通行相位的压力值仍然较大, 可继续对绿灯显示时间进行延长. 如果 $(1+\delta)P_{n,\sigma^*(k_n)}<P_{n,\sigma_{\max}(k_n)}$, 说明当前通行相位的压力值较小, 可以结束当前通行相位, 并发送控制信号激活下一相位进行绿灯显示. 因此, 信号灯控制算法的主要步骤如下.

(1) 按照压力从大到小的顺序排列所述最大压力计算线程的计算结果, 得到最大压力所在相位.

(2) 判断当前时刻交叉口的通行相位是否为所述最大压力所在相位.

(3) 若当前时刻交叉口的通行相位为所述最大压力所在相位, 将所述通行相位的交通信号灯显示时间调整为最大绿灯时间.

(4) 若当前时刻交叉口的通行相位不为所述最大压力所在相位, 将所述通行相位的交通信号灯显示时间调整为最小绿灯时间.

(5) 若当前时刻交叉口的通行相位为最大压力所在相位, 在所述通行相位的交通信号灯显示时间等于最大绿灯时间时, 根据各相位的压力值以及相位通行等待时间确定待通行相位, 控制所述通行相位的交通信号灯切换为黄灯显示, 控制所述待通行相位的交通信号灯切换为绿灯显示.

(6) 若当前时刻交通信号灯的通行相位不为所述最大压力所在相位, 在所述通行相位的交通信号灯显示时间等于所述最小绿灯时间时, 控制所述通行相位的交通信号灯切换为黄灯显示, 控制所述最大压力所在相位的交通信号灯切换为绿灯显示.

在大规模交通仿真任务中, 每一步仿真迭代都采用上述的压力计算模块和策略调整模块对信号灯的状态进行修改, 为了能及时地调整信号灯以减少对整体系统延迟的影响, 需要对信号灯控制进行并行处理. 目前节点内的车辆行驶算法已采用多核 CPU 进行了多线程处理, 使得车辆的模拟能够高效地并行执行. 而在并行算法 1 的实现过程中发现, 信号灯控制可以与车辆行驶并行执行, 从而可以利用 DSP 对交通信号灯的控制算法进行加速. 天河超算平台的加速器 DSP 采用 SPMD 的并行模式, 在具体实现中, 将分配一个 DSP 的加速核依次计算一个交叉口中一个相位的最大压力值 (算法 1 的第 13 行) 和该信号灯控制策略 (算法 1 的第 14 行). 因此, 需要将交叉口数据按照 DSP 的簇进行划分, 每个簇的交叉口数据存在高速带宽共享内存中, 且其存储格式如后文图 5 所示.

为了保证每个 DSP 加速核心访存的连续性, 在交通仿真系统中针对每个交叉口的数据采取了一系列策略来优化数据的存储和访问方式. 具体而言, 考虑到交叉口内部的相位、交叉口内部车道、前驱和后继车道以及车辆数目等关键信息, 将这些数据进行紧凑的存储, 确保相邻的数据项在内存中的物理位置是连续的, 同一交叉口的数据在访问时可以连续地加载到 DSP 加速核心中, 避免了频繁的访存操作和数据拷贝, 减少访存的跳跃和碎片化, 提高访存的效率和性能. 通过上述内存布局的优化策略, 在保证数据连续性的前提下, 能够高效地计算出每个交叉口各相位的压力值和信号灯控制策略.

3.4 数据存储与同步

根据天河新一代的架构, 每个计算节点中车道和路口等位置不变的数据均匀分配到不同加速簇的内存, 而车辆运动过程中产生的动态数据 (如车道的车辆数、车速以及车辆所处的车道等), 则动态地更新至其所处车道或路口对应簇的内存. 因此, 在每一步的仿真过程中, 考虑到动态数据在内存的同步问题, 设计了如图 6 所示的数据同步方法.

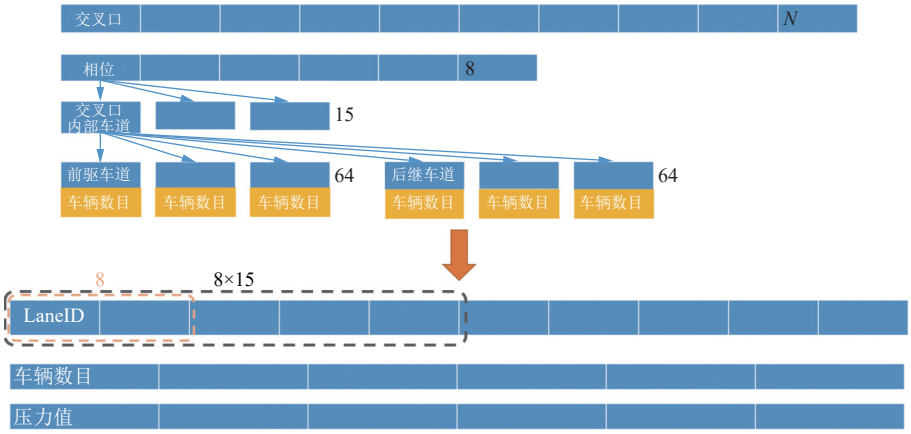


图 5　交叉口数据存储结构

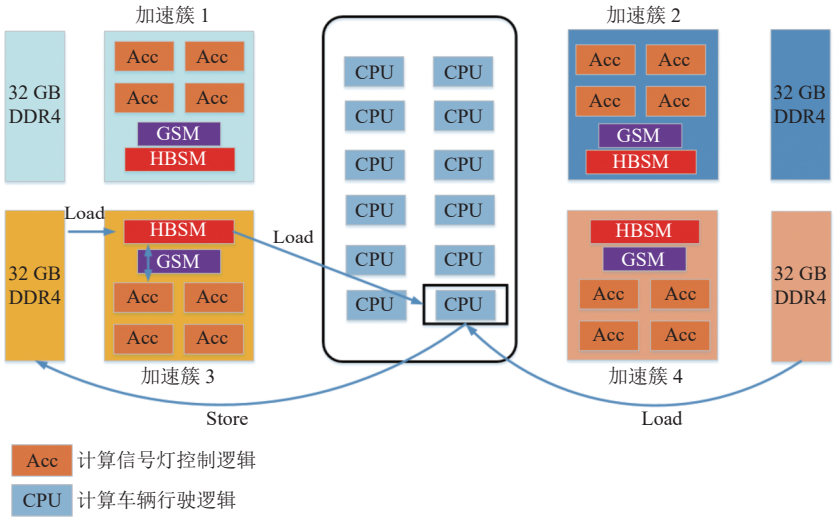


图 6　数据同步过程

根据第 3.3 节描述, 每个路口的信号灯控制计算逻辑分配到一个加速核进行单独处理, 然而天河新一代架构的特性使得加速簇无法直接访问远端的内存[15]. 为了实现数据同步, 当车辆被调度到 CPU 时, 首先进行当前时刻的仿真, 根据是否变道或进入下一个路口, 选择新的车道, 并将当前的车辆运动状态信息更新到新道路对应簇的内存. 如果新的车道与当前车道不属于同一个加速簇, 则 CPU 产生一次跨簇的内存更新操作. 此外, 每个加速核等待簇内存中相关数据更新, 并将更新后的数据加载到高速带宽共享内存进行信号灯控制处理. 信号灯控制的结果写回高速带宽共享内存, 因此在 CPU 中仿真的车辆可以直接通过访问共享内存获得信号灯数据, 并根据该数据完成下一步动作. 通过 CPU 直接访问加速簇中共享内存的数据传输方式, 极大地提高交通仿真的性能.

3.5　性能分析

对于上述提出的算法 1, 结合第 3.1–3.3 节实现的并行计算策略, 现对其进行理论上的性能分析. 假设 M 台车辆在 P 个节点进行模拟, 且路网的大小为 N (表示车道和交叉口数量规模), M 台车辆在单节点的每一步模拟计算时间 $T(M,N)$ 的公式如下:

$$T(M,N)=\max\{C(M)+L(N),D(N)+T'(N)\} \tag{12}$$

其中, $C(M)$、$L(N)$ 和 $D(N)$ 分别表示车辆并行行驶、车道并行更新和信号灯并行控制的时间, 而 $T'(N)$ 表示将

交叉口数据从内存拷贝到 DSP 高速带宽共享内存以及将计算结果从 DSP 共享内存拷贝到内存的时间开销. 上述计算和传输的时间开销与车辆、车道和交叉口的规模相关.

简单起见, 假定每个节点的车辆数为 M/P, 且根据提出的路网划分策略, 每个节点的路网大小为 N/P, 根据公式 (12), 可以得到每个计算节点进行每一步模拟的时间为 $T(M/P,N/P)\approx T(M,N)/P$. 在算法 1 的第 17 行中, 多个节点下每一步仿真车辆可能需要进行通信, 假设每个节点平均有 m 辆车需要将其大小为 k 的状态信息发送到其他节点, 则每个节点的通信时间为:

$$C(m)=mk/B \tag{13}$$

其中, B 为节点之间的传输带宽. 因此, 可以得到算法 1 的加速比为:

$$S=\frac{T(M,N)}{T(M/P,N/P)+C(m)}=\frac{T(M,N)}{T(M,N)/P+mk/B}=\frac{1}{\frac{1}{P}+\frac{mk}{B\cdot T(M,N)}} \tag{14}$$

则其并行效率为:

$$E=S/P=\frac{1}{\frac{1}{P}+\frac{mk}{B\cdot T(M,N)}}/P=\frac{1}{1+\frac{mkP}{B\cdot T(M,N)}} \tag{15}$$

根据公式 (14) 和公式 (15), 算法加速比 S 与节点数 P 和计算时间 $T(M,N)$ 相关, 当节点数增加或增加车辆或路网规模时, 都能提升算法加速性能. 而并行效率 E 与计算时间 $T(M,N)$ 和节点数 P 有关, 当加大模拟的车辆规模和路网规模时, 计算时间 $T(M,N)$ 也会随之增加. 当节点数 P 增加时, 为了保证并行效率 E 不会降低, 可以增加模拟车辆或路网的规模. 此外, 当节点数 P 不变时, 增加车辆和路网规模也能提高并行效率.

4 测试分析

4.1 软硬件设置

平行交通仿真框架按照上述的并行优化方案部署并运行在国产天河超算平台的多个计算节点, 每个节点包括 16 个通用 CPU、96 个控制核心和 1 536 个加速核心, 节点内 DDR 内存大小为 32 GB, 带宽达到 204 GB/s, 而 DSP 内高速带宽共享内存大小为 48 MB, 带宽达到 307 GB/s. 仿真框架采用的是 C++编程语言以及 HThread 编程框架进行开发, 该框架是由国防科技大学自主研发的面向数字信号处理的编程框架, 而节点间采用 MPI 进行通信, 节点内并行计算采用 OpenMP 共享内存的多线程程序方式. 仿真程序模拟了北京五环内 7 797 个路口和 17 万条车道的真实交通状况, 测试了在不同计算节点下 (1、2、4、9 和 16 节点), 执行不同车辆规模 (40 万、80 万、160 万和 240 万车辆) 的仿真性能. 对于不同数量节点的路网划分, 将按照 1×2 、 2×2 、 3×3 、 4×4 的划分方式将子图分配到 2 节点、4 节点、9 节点和 16 节点.

4.2 测试结果与分析

图 7 展示了在不同计算节点下, 执行计算的车辆数量逐渐增加时, 每个时间步长所需的计算时间. 可以观察到随着计算车辆数量的增加, 仿真执行时间呈现出一定的加速效果. 这是因为在节点数较少的情况下, 每个计算节点能够有效地分担计算负载, 实现并行化计算. 随着节点数的增加, 可以看到执行时间呈现出更显著的下降趋势. 值得注意的是, 在节点数达到 16 并且计算车辆数量达到 240 万时, 可以观察到明显的加速效果. 每个时间步长仅需要 0.37 s 的计算时间, 相比单个节点下 240 万辆车的执行时间减少了 1.8 s. 此外, 对于相同数量的计算节点, 由于增加了车辆数量, 模拟的规模增大, 仿真的执行时间也会随之增加.

图 8 反映了图 7 中的性能加速比情况, 对于不同数量的车辆模拟, 增大计算节点的规模, 都能提高仿真的性能, 而对于固定的计算节点, 增加车辆数也可以提高仿真的性能, 这一规律可以从公式 (14) 中体现. 对于车辆和路网规模固定时, 增加节点数, 式中 $1/P+mk/B\cdot T(M,N)$ 减小, 从而可以算法提高加速性能 (图 8 中每一行的测试结果); 对于节点数固定时, 增加车辆数, 计算时间 $T(M,N)$ 增大, 导致 $1/P+mk/B\cdot T(M,N)$ 减小, 同样也能提高算法的

加速性能 (图 8 中每一列的测试结果). 从图中可以观察到, 在节点数达到 16 并且计算车辆数量达到 240 万时的加速比最大, 接近 6 倍的性能提升.

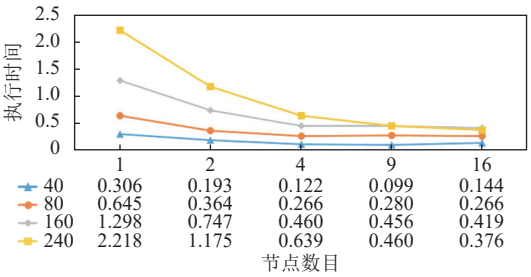


图 7 交通仿真执行时间分析图

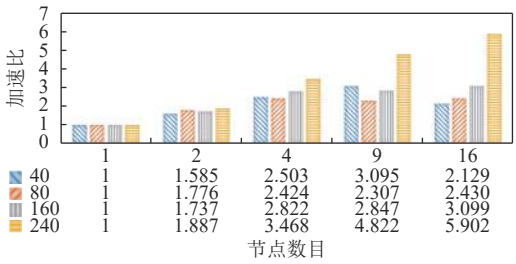


图 8 交通仿真执行加速比分析图

根据图 8 的加速比分析, 描绘图 9 以展示在不同节点数时算法的并行效率, 并行效率是评估并行计算系统的一个重要指标, 它衡量了并行计算在多个处理节点上的实际性能相对于理论最优性能的比例. 在图 9 中, 可以清晰地看到随着节点数的增加, 系统整体的并行效率逐渐下降的趋势, 同样当节点数固定时, 增加车辆模拟的规模, 可以提升算法的并行效率, 该变化规律可以通过公式 (15) 进行分析. 对于车辆和路网规模固定时, 增加节点数, 公式 (14) 中的 $mkp/B\cdot T(M,N)$ 会增大, 导致算法的并行效率降低 (图 9 中每一行的测试结果); 而对于节点数固定时, 增加车辆模拟的规模, 式中的 $mkp/B\cdot T(M,N)$ 会减小, 导致算法的并行效率增加 (图 9 中每一列的测试结果). 总之, 结合公式 (15) 和图 9 的变化规律, 当节点数增加时, 为了保持算法的并行效率, 可以适当地增加仿真数据的规模.

4.3 消融实验

本节采用不同的路网划分方法, 以验证提出的两阶段路网划分方法的有效性, 并且仅使用天河超算的 CPU 运行交通仿真并行算法, 以验证充分利用天河超算的异构体系结构对于实现高性能交通仿真的重要性.

(1) 两阶段路网划分与车辆平均划分对比

将车辆均匀划分方法和两阶段路网划分方法在 80 万辆车模拟场景下的每一步仿真所需时间进行了分析, 结果如图 10 所示. 从图中可以看到, 两阶段路网划分对于车辆平均划分的优势较为明显, 其主要原因还是在于通信开销, 车辆平均划分的策略较为简单, 但每一步仿真结束后, 需要将每一个节点的车辆状态信息同步到其他节点, 其带来的通信开销难以承受, 尤其是在节点数量较多的场景.

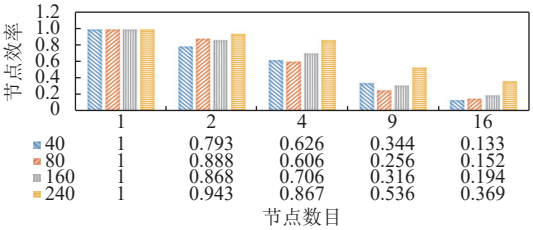


图 9 交通仿真执行并行效率分析图

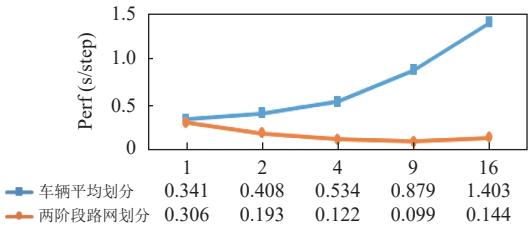


图 10 划分方法性能分析图

(2) 两阶段路网划分与 METIS 划分对比

使用 METIS[28]将路网 G 按照计算节点的数量进行划分, 得到不同的子图分配到不同节点上进行仿真计算, 将 METIS 的划分方法与两阶段路网划分方法在 80 万、160 万和 240 万辆车模拟场景下的每一步仿真时间进行了对比, 结果如表 1 所示. 从表中可以看到, 两阶段路网划分对于 METIS 划分具有一定优势, 当计算节点数目较少时, METIS 划分后每一步仿真时间更短; 而当节点数增大时, 两阶段路网划分后每一步仿真的时间更短.

表 1 METIS 与两阶段路网划分在不同车辆数、不同计算节点下每一步的仿真时间

车辆数 (万)	划分	节点数路网 (s)				
		1	2	4	9	16
80	METIS	0.646	0.383	0.307	0.294	0.276
	两阶段路网划分	0.645	0.363	0.266	0.279	0.265
160	METIS	1.297	0.711	0.467	0.442	0.421
	两阶段路网划分	1.297	0.741	0.459	0.455	0.418
240	METIS	2.214	1.167	0.612	0.474	0.401
	两阶段路网划分	2.217	1.175	0.639	0.459	0.375

(3) 异构计算与同构计算对交通仿真的性能影响

天河超算采用的异构体系结构主要是 CPU 和 DSP 加速器. 将使用纯 CPU 运行的交通仿真算法和 CPU 结合 DSP 加速器运行的交通仿真算法进行对比, 结果如表 2 所示. 可以看出, 使用了 DSP 加速器能大幅缩短每一步仿真时间, 因此充分利用超算的异构体系结构是实现高性能交通仿真的关键.

表 2 同构计算和异构计算在不同车辆数、不同计算节点下每一步的仿真时间

车辆数 (万)	划分	节点数路网 (s)				
		1	2	4	9	16
80	CPU	1.416	0.921	0.829	0.814	0.627
	CPU+DSP	0.645	0.363	0.266	0.279	0.265
160	CPU	3.031	2.223	1.782	1.512	1.083
	CPU+DSP	1.297	0.741	0.459	0.455	0.418
240	CPU	5.504	3.563	2.428	2.126	1.311
	CPU+DSP	2.217	1.175	0.639	0.459	0.375

5 总结与展望

本文基于平行城市理论, 提出了一种平行城市交通仿真并行算法, 该算法针对天河新一代超算的异构体系结构, 实现了路网划分、车辆并行化行驶、信号灯并行化控制等方法, 并运行在 16 节点、超过 2.5 万核心的天河新一代超算平台, 进行了北京市五环内 240 万辆车、7797 个路口和 17 万条车道的真实交通场景模拟, 相比于传统单节点的仿真, 每步仿真时间从 2.21 s 减少到 0.37 s, 取得了近 6 倍的加速效果.

在未来的研究工作中, 拟开展以下几个方面的研究: (1) 将交通仿真框架应用于其他不同城市的交通模拟仿真, 并进一步扩大仿真规模, 以涵盖多个城市间的复杂交通体系, 进而实现从单个城市到城市群的交通仿真; (2) 目前交通仿真算法只对车辆流动进行了仿真, 未来还可以结合居民出行和暴雨天气等场景, 以实现从单一场景的仿真到复杂场景的交通仿真; (3) 针对平行城市交通仿真算法提出一种自动化的并行方法, 能够根据不同异构系统架构自动地进行并行优化, 以降低仿真算法在不同平台的并行优化难度, 并进一步在百万核心规模的超算平台进行交通仿真.

References:

[1] Dharwal M, Agarwal N, Kumar S, Anand S, Vatsa M. Building smarter smart cities for sustainable development through artificial intelligence. In: Proc. of the 5th Int'l Conf. on Electronics, Communication and Aerospace Technology. Coimbatore: IEEE, 2021. 1185–1187. [doi: 10.1109/ICECA52323.2021.9676140]

[2] Ni AM, Ruff DA, Alberts JJ, Symmonds J, Cohen MR. Learning and attention reveal a general relationship between population activity and behavior. Science, 2018, 359(6374): 463–465. [doi: 10.1126/science.aao0284]

[3] Javaid S, Sufian A, Pervaiz S, Tanveer M. Smart traffic management system using Internet of Things. In: Proc. of the 20th Int'l Conf. on Advanced Communication Technology. Chuncheon: IEEE, 2018. 393–398. [doi: 10.23919/ICACT.2018.8323770]

[4] Lanke N, Koul S. Smart traffic management system. Int'l Journal of Computer Applications, 2013, 75(7): 19–22. [doi: 10.5120/13123-0473]

[5] Shao ML, Cao E, Hu M, Zhang Y, Chen WJ, Chen MS. Traffic light optimization control method for priority vehicle awareness. Ruan Jian Xue Bao/Journal of Software, 2021, 32(8): 2425–2438 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6191.htm [doi: 10.13328/j.cnki.jos.006191]

[6] Ghazal B, ElKhatib K, Chahine K, Kherfan M. Smart traffic light control system. In: Proc. of the 3rd Int'l Conf. on Electrical, Electronics, Computer Engineering and their Applications. Beirut: IEEE, 2016. 140–145. [doi: 10.1109/EECEA.2016.7470780]

[7] Levinson D. Perspectives on efficiency in transportation. Int'l Journal of Transport Management, 2003, 1(3): 145–155. [doi: 10.1016/j.ijtm.2004.01.002]

[8] Lü YS, Wang FY, Zhang Y, Zhang XD. Parallel cities: Framework, methodology, and application. Chinese Journal of Intelligent Science and Technology, 2019, 1(3): 311–317 (in Chinese with English abstract). [doi: 10.11959/j.issn.2096-6652.201932]

[9] Zhang XD, Xu DD, Wang L, Liang H, Lü YS, Wang FY. Model architecture and urban computing for parallel cities based on complex adaptive systems. Journal of Command and Control, 2021, 7(1): 28–37 (in Chinese with English abstract). [doi: 10.3969/j.issn.2096-0204.2021.01.0028]

[10] Zhu FH, Lv YS, Chen YY, Wang X, Xiong G, Wang FY. Parallel transportation systems: Toward IoT-enabled smart urban traffic control and management. IEEE Trans. on Intelligent Transportation Systems, 2020, 21(10): 4063–4071. [doi: 10.1109/TITS.2019.2934991]

[11] Fu ZS, Yu J, Sarwat M. Building a large-scale microscopic road network traffic simulator in apache spark. In: Proc. of the 20th IEEE Int'l Conf. on Mobile Data Management. Hong Kong: IEEE, 2019. 320–328. [doi: 10.1109/MDM.2019.00-42]

[12] Zhu XM, He C, Wang JJ, Jiang JQ. An elastic energy-aware scheduling strategy for heterogeneous computing systems. Chinese Journal of Computers, 2012, 35(6): 1313–1326 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.01313]

[13] Wang YR, Li LS, Wang JT, Tian R. Acceleration of smoothed particle hydrodynamics method on CPU-GPU heterogeneous platform. Chinese Journal of Computers, 2017, 40(9): 2040–2056 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2017.02040]

[14] Bian Y, Yuan F, Guo JX, Li Z, Zhao RL. CPU+GPU heterogeneous computing orientated multi-objective test case prioritization. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 943–954 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4968.htm [doi: 10.13328/j.cnki.jos.004968]

[15] Lu K, Wang YH, Guo Y, Huang C, Liu S, Wang RB, Fang JB, Tang T, Chen ZY, Liu BW, Liu Z, Lei YW, Sun HY. MT-3000: A heterogeneous multi-zone processor for HPC. CCF Trans. on High Performance Computing, 2022, 4(2): 150–164. [doi: 10.1007/s42514-022-00095-y]

[16] Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC. GPU computing. Proc. of the IEEE, 2008, 96(5): 879–899. [doi: 10.1109/JPROC.2008.917757]

[17] Rofouei M, Stathopoulos T, Ryffel S, Kaiser W, Sarrafzadeh M. Energy-aware high performance computing with graphic processing units. In: Proc. of the 2008 Conf. on Power Aware Computing and Systems. San Diego: USENIX Association, 2008. 11–15.

[18] Shi L, Chen H, Sun JH, Li KL. vCUDA: GPU-accelerated high-performance computing in virtual machines. IEEE Trans. on Computers, 2012, 61(6): 804–816. [doi: 10.1109/TC.2011.112]

[19] Xu S, Wang W, Zhang J, Jiang JR, Jin Z, Chi XB. High performance computing algorithm and software for heterogeneous computing. Ruan Jian Xue Bao/Journal of Software, 2021, 32(8): 2365–2376 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6008.htm [doi: 10.13328/j.cnki.jos.006008]

[20] Duran A, Klemm M. The Intel® many integrated core architecture. In: Proc. of the 2012 Int'l Conf. on High Performance Computing & Simulation. Madrid: IEEE, 2012. 365–366. [doi: 10.1109/HPCSim.2012.6266938]

[21] Heinecke A, Klemm M, Bungartz HJ. From GPGPU to many-core: Nvidia Fermi and Intel many integrated core architecture. Computing in Science & Engineering, 2012, 14(2): 78–83. [doi: 10.1109/MCSE.2012.23]

[22] Ibrahim D, Davies A. The evolution of digital signal processors. In: Proc. of the 6th IEEE History of Electrotechnology Conf. Glasgow: IEEE, 2019. 25–29. [doi: 10.1109/HISTELCON47851.2019.9040130]

[23] Zhang QR, Xie QS, Duan KF, Liang B, Wang M, Wang GX. A digital signal processor (DSP)-based system for embedded continuous-time cuffless blood pressure monitoring using single-channel PPG signal. Science China Information Sciences, 2020, 63(4): 149402. [doi: 10.1007/s11432-018-9719-9]

[24] Yang J, Yang WD, Qi RX, Tsai Q, Lin SL, Dong FK, Li KL, Li KQ. Parallel algorithm design and optimization of geodynamic numerical simulation application on the Tianhe new-generation high-performance computer. The Journal of Supercomputing, 2024, 80(1): 331–362. [doi: 10.1007/s11227-023-05469-9]

[25] Wang DH, Jin S. Review and outlook of modeling of car following behavior. China Journal of Highway and Transport, 2012, 25(1): 115–127 (in Chinese with English abstract). [doi: 10.19721/j.cnki.1001-7372.2012.01.018]

[26] Mercader P, Uwayid W, Haddad J. Max-pressure traffic controller based on travel times: An experimental analysis. Transportation

Research Part C: Emerging Technologies, 2020, 110: 275–290. [doi: 10.1016/j.trc.2019.10.002]

[27] Zoabi R, Haddad J. An advanced max-pressure traffic controller based on travel time delays. In: Proc. of the 25th IEEE Int'l Conf. on Intelligent Transportation Systems. Macao: IEEE, 2022. 1662–1667. [doi: 10.1109/ITSC55140.2022.9922414]

[28] Karypis G, Kumar V. METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Technical Report, 97-061, Minneapolis: University of Minnesota, 1997. 1–33.

附中文参考文献:

[5] 邵明莉, 曹鹍, 胡铭, 章玥, 陈闻杰, 陈铭松. 面向优先车辆感知的交通灯优化控制方法. 软件学报, 2021, 32(8): 2425–2438. http://www.jos.org.cn/1000-9825/6191.htm [doi: 10.13328/j.cnki.jos.006191]

[8] 吕宜生, 王飞跃, 张宇, 张晓东. 虚实互动的平行城市: 基本框架、方法与应用. 智能科学与技术学报, 2019, 1(3): 311–317. [doi: 10.11959/j.issn.2096-6652.201932]

[9] 张晓东, 许丹丹, 王良, 梁弘, 吕宜生, 王飞跃. 基于复杂系统理论的平行城市模型架构与计算方法. 指挥与控制学报, 2021, 7(1): 28–37. [doi: 10.3969/j.issn.2096-0204.2021.01.0028]

[12] 朱晓敏, 贺川, 王建江, 江建清. 异构计算系统中弹性节能调度策略研究. 计算机学报, 2012, 35(6): 1313–1326. [doi: 10.3724/SP.J.1016.2012.01313]

[13] 王迎瑞, 黎雷生, 王景焘, 田荣. 光滑粒子流体动力学方法的高效异构加速. 计算机学报, 2017, 40(9): 2040–2056. [doi: 10.11897/SP.J.1016.2017.02040]

[14] 边毅, 袁方, 郭俊霞, 李征, 赵瑞莲. 面向 CPU+GPU 异构计算的多目标测试用例优先排序. 软件学报, 2016, 27(4): 943–954. http://www.jos.org.cn/1000-9825/4968.htm [doi: 10.13328/j.cnki.jos.004968]

[19] 徐顺, 王武, 张鉴, 姜金荣, 金钟, 迟学斌. 面向异构计算的高性能计算算法与软件. 软件学报, 2021, 32(8): 2365–2376. http://www.jos.org.cn/1000-9825/6008.htm [doi: 10.13328/j.cnki.jos.006008]

[25] 王殿海, 金盛. 车辆跟驰行为建模的回顾与展望. 中国公路学报, 2012, 25(1): 115–127. [doi: 10.19721/j.cnki.1001-7372.2012.01.018]

何贤浩(1995－), 男, 博士生, CCF 学生会员, 主要研究领域为高性能计算, 机器学习.

吕宜生(1983－), 男, 博士, 研究员, 博士生导师, 主要研究领域为智能交通, 无人驾驶.

胡逸鼐(1988－), 男, 博士, 助理研究员, 主要研究领域为高性能计算, 并行计算, 智能计算.

廖清(1988－), 女, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为数据挖掘, 机器学习.

李毅晨(1999－), 男, 硕士生, 主要研究领域为智慧城市, 机器学习.

李勇(1985－), 男, 博士, 副教授, 博士生导师, CCF 杰出会员, 主要研究领域为城市科学与计算.

严宇威(1999－), 男, 硕士生, CCF 学生会员, 主要研究领域为边缘智能, 高性能计算, 分布式推理.

李肯立(1971－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为高性能计算系统与应用.