计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210100109

基于 MPI 的分布式并行 Gazebo 仿真优化与测试

蒋化南[1]　张　帅[2]　林宇斐[2]　李　豪[2]
1 天津(滨海)人工智能军民融合创新中心　天津 300280
2 军事科学院　北京 100000
(jianghua_nan@163.com)

摘　要　Gazebo 作为机器人通用仿真平台,能够在复杂的室内和室外环境中准确模拟机器人行为,并在单节点上支持多机器人的协同仿真。但执行仿真任务中如果启动成百上千台机器人,通常会发现 Gazebo 性能参考值(RTF)仿真实时比会降低两个数量级,甚至出现仿真错误的情况,仿真性能会成为其主要制约因素。为了实现机器人集群的高性能仿真,探索了基于 MPI 的跨节点 ROS+Gazebo 仿真平台搭建方法,核心过程是针对确定的仿真任务进行并行划分,可采用编号划分或区域划分,将划分好的各个子任务部署到各计算节点的 Gazebo 上进行仿真,最后通过 Gazebo 之间的 MPI 进程通信保证仿真的同步和一致性,以此实现机器人分布在不同计算节点上的协同仿真。同时编写了固定翼和四旋翼同构和异构的仿真测试案例,通过脚本程序读入 world 配置文件和 roslaunch 文件来实现,设计了对用户友好的与 ROS 类似的启动方式,进行了单节点和跨节点的性能测试,验证了分布式并行仿真的优越性。

关键词:MPI;Gazebo;ROS;机器人集群;高性能仿真;
中图法分类号　TP391.9

Simulation Optimization and Testing Based on Gazebo of MPI Distributed Parallelism

JIANG Hua-nan[1],ZHANG Shuai[2],LIN Yu-fei[2] and LI Hao[2]
1 Tianjin Artificial Intelligence Innovation Center,Tianjin 300280,China
2 The Academy of Military Science,Beijing 100000,China

Abstract　Gazebo,as a general robot simulation platform,can simulate robot behavior accurately in the complex environment of indoor or outdoor,and support multi-robot collaborative simulation on single computer node. But when the simulation task contains hundreds of robots,it is usually found that the RTF (Gazebo simulation real-time performance) will reduce two orders of magnitude,some errors even appear in the simulation. The simulation performance will become the critical limiting factor. In order to realize high-performance simulation,the across node simulation platform based on MPI and ROS+Gazebo is explored. The core process is to divide the simulation tasks in parallel,which can be divided by number or region. The divided sub tasks are deployed to the Gazebo of each computing node for simulation. Finally,the MPI process communication between the Gazebo ensures the synchronization and consistency of the simulation,so as to realize the collaborative simulation of robots distributed on different computing nodes. At the same time,two types of cases including homogeneity and heterogeneity about fixed wing and quadrotor are writed,which are realized by reading the world configuration file and roslaunch file through the script program. The user-friendly starting mode similar to ROS was designed,and the single-node and cross-node performance tests are carried out to verify the advantage of distributed parallelism simulation.

Keywords　MPI,Gazebo,ROS,Robot swarm,High-performance simulation

1　引言

21 世纪以来,国内外对机器人技术的发展越来越重视,成为衡量一国工业化水平的重要标志,机器人可以做一些重复性高或是高危险的工作,也可以做一些因为尺寸限制人类无法从事的工作,甚至可以工作在外太空或深海等不适合人类生存的环境中[1]。机器人在越来越多方面可以取代人类,已被逐步地应用到工业生产、仓储物流、搜索探测、家庭服务、军事行业等领域[2]。作业环境复杂,完美协同性和自主可控性是机器人能否在这些应用领域正常工作所面临的重大挑战。面对这种挑战,人们提出机器人集群的概念,它们通过相互之间的简单合作可以完成单体机器人所无法实现的任务,比如多个机器人协同搬运单个机器人无法搬运的大型货物、多个低成本机器人可以以更高的性能完成单个高成本机器人所要实现的任务等[3]。机器人集群具有简单性、可扩展性、容错性、平行性等优势,是未来无人系统的突破口,具有非常可观的发展前景。由于机器人集群系统是一个具有较多特性未知的复杂被控对象,需要进行大量的实验来测试系统的稳定

基金项目:国家自然科学基金青年基金(61902425)
This work was supported by the National Natural Science Foundation of China(61902425).
通信作者:张帅(zhangshuai9999@outlook.com)

性，而实物试验时间长、风险高、成本高，而仿真天然具备安全可靠、快速迭代等特性，所以通常需要仿真平台来对系统进行仿真试验以检测其控制逻辑和功能[4]。目前常用的机器人仿真软件有 Webots，Mojuco，OpenAI Gym，Gazebo 等[6]。其中，Gazebo 是一个开源的机器人 3D 物理仿真环境，它能够在复杂的室内和室外环境中准确模拟机器人行为，既能和通用机器人操作系统 ROS 配合来使用，也可单独用来仿真。它包含了强大的高精度物理引擎、高质量的图像渲染和友好的图形界面[7]。因此，从逼真度、可扩展性、开发简易性和成本方面来考量[6]，本文选取 Gazebo 作为仿真平台。目前 Gazebo 可以在单节点上支持多机器人的协同仿真，但随着机器人群体规模的不断上升，性能瓶颈越来越凸显，由于基于时间步长的仿真耦合了高精度物理引擎，如何在保证仿真精度的前提下提高仿真性能成为当今的研究热点。通过研究发现，执行仿真任务中如果启动成百上千台机器人，Gazebo 性能参考值 RTF 会降低两个数量级，甚至出现仿真错误的情况。基于这些问题，本文针对 Gazebo 进行了机器人集群仿真性能分析，根据分析结果实现了基于 MPI 的分布式并行优化，并测试验证了优化效果。

2 平台简介与性能分析

Gazebo 本身拥有高精度的物理引擎，可以精确定义机器人在仿真过程中所需的静力学、动力学环境，实现复杂环境下机器人的规划控制仿真[1]。Gazebo 仿真软件使用了分布式架构，拥有独立的库，可用于物理模拟、呈现、用户界面、通信和传感器的生成[8]。图 1 给出了 Gazebo 的主要仿真模块，其中 World::LoadPlugin 主要加载一些世界插件和模型插件，比如 gazebo_ros 接口插件；Server::ParseArgs 主要进行参数解析和世界文件的加载。图 2 展示了逻辑运算模块 gzserver 运行流程，从图中可以看出循环调用的核心模块为 WorldUpdateMutex 模块，此模块更新的内容主要包括模型预处理过程（动力学、传感器等插件挂载遍历计算更新）、模型更新、碰撞检测更新、物理更新等，本文基于 MPI 的分布式 Gazebo 仿真的主逻辑运行主要基于 gzserver 模块进行更改。

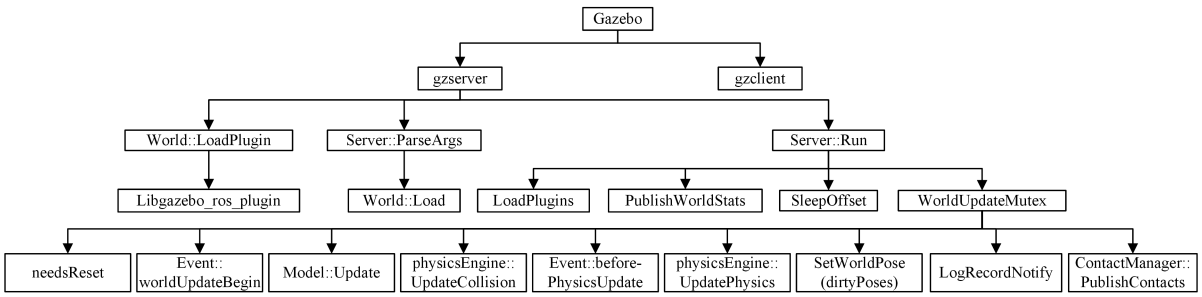

图 1 Gazebo 仿真模块图

Fig. 1 Gazebo module diagram

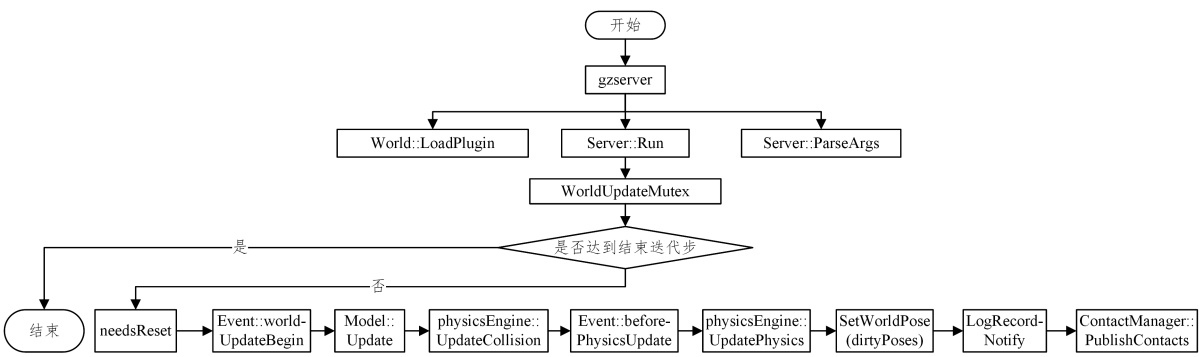

图 2 gzserver 模块运行流程图

Fig. 2 Operation flow diagram of gzserver module

ROS 是由斯坦福大学人工智能实验室和 Willow Garage 合作研发并于 2007 年公开发布的次级操作系统，是针对机器人研究与开发所设计出来的一套系统结构[9]。这套系统集成了大量的代码库、通信协议等工具，能够提供操作系统功能，包括硬件的抽象描述、底层驱动管理、多进程通信等基础功能，为开发越来越复杂的机器人系统提供了巨大的便利，极大提高了机器人的研发速度，使开发成本变得更低廉[10]。目前 ROS 主要基于 Ubuntu 操作系统来实现操作，ROS 的架构由文件系统级（Filesystem level）、计算图级（Computation Graph level）、社区级（Community level）3 部分组成，其中计算机图级主要包括功能包（package）、消息（message）、服务（message）、话题（topic）等[11-12]。ROS 支持多种编程语言，如 Python，C++，Lisp 等[4]。Gazebo 是与 ROS 无缝兼容的，ROS 官方为其提供了一个强大的离线数据库，包含各式各样的能够模拟真实环境的模型，用户也可在软件中自由创建模型。同时，Gazebo 可以与 ROS 中的 Rviz 产生联动，使得开发者可以从多个角度观察仿真的过程、结果，提高了科研人员的研发效率[13]。因此本文主要基于 Ubuntu16.04 操作系统采用 C++语言进行开发，选择 ROS kinetic 版本，以 Gazebo＋ROS 的方式进行分布式并行仿真设计实现。

MPI 是由工业、科研和政府部门等联合建立的消息传递并行编程标准。MPI1.0 版本于 1994 年发布，1998 年 MPI2.0 版本发布。在 MPI2.0 中，共有 287 个调用接口，MPI 以语言独立的形式来定义这个接口库，并提供了与 C 和 Fortran 语言的绑定，这个定义不包含任何专用于某个特别的制造商、操作系统或硬件的特性，由于这个原因，MPI 在并行计算界被广

泛的接受[14-15]。MPI 凭借其标准化、可移植、可扩展等优点，成为目前高性能计算的主要模型，也是本文采取的计算模型。

在确定 Gazebo 并行方案前，首先需定位 Gazebo 的仿真热点，针对这些热点进行并行计算。因此，以 rosplane 固定翼和 hector 四旋翼无人机为例，采用时间插针的方式进行热点分析，采用 ROS+Gazebo 方式启动 60 架无人机迭代 100 000 步。表 1 为仿真核心模块内各子模块的时间占比，从表中可以看出模型预处理模块(worldUpdateBegin)、碰撞检测模块(updateCollision)和物理更新模块(updatePhysics)是主要的仿真性能热点，针对这些仿真热点，MPI 并行方案会将这些模块计算任务分到多个节点上。

表 1 Gazebo 仿真模块的运行时间表

Table 1 Gazebo simulation module running schedule

仿真模块	四旋翼案例平均仿真时间	固定翼案例平均仿真时间
模型预处理模块	2.1506(88.59%)	0.6989(54.22%)
碰撞检测模块	0.0513(2.11%)	0.0454(3.53%)
物理更新模块	0.0884(3.64%)	0.1820(14.17%)
其他模块	0.1375(5.66%)	0.3580(27.88%)
整体更新	2.4277(100%)	1.2843(100%)

3 分布式并行 Gazebo 仿真优化

3.1 方案设计

群体无人系统并行分布式可扩展仿真架构如图 3 所示。首先确定群体无人系统仿真任务，然后针对仿真任务进行并行划分，将划分好的各个子任务部署到各计算节点的 Gazebo 上进行仿真，最后通过模拟器之间的通信保证仿真同步和一致性。基于架构设计，模拟器在 Gazebo7.14.0 版本的基础上进行修改，针对模型预处理模块、碰撞检测模块、物理更新模块这 3 个仿真性能热点，将这些模块计算任务分到多节点上，总仿真任务分配给多个 Gazebo 的示意如图 4 所示，其中傀儡实体做虚化处理。

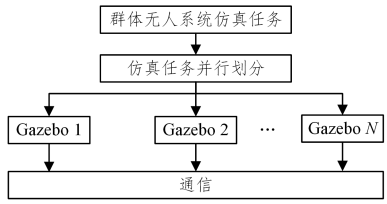

图 3 群体无人系统仿真分布式并行可扩展架构示意图

Fig. 3 Robot swarm simulation of distributed parallelism extensible architecture schematic diagram

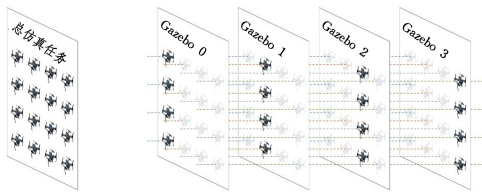

图 4 总仿真任务分配给多个 Gazebo 的示意图

Fig. 4 Total simulation tasks assigned to multiple Gazebos schematic diagram

跨节点 Gazebo 仿真方案设计如图 5 所示，以 4 架四旋翼无人机 2 节点仿真为例，每个节点中的 Gazebo 相对独立地作为本节点的 ROS 节点，实现 ROS 通信，每个 Gazebo 主要仿真其中 2 架无人机，模型预处理模块并行后，每个节点仅计算部分虚拟无人实体(仿真实体)所挂载的插件，同步接收本节点以外的其他节点虚拟无人实体(傀儡实体)的位姿信息并设置傀儡模型的位姿为该值，检测碰撞并计算本节点模型物理模块；在每个仿真步最后利用 MPI 通信汇总所有虚拟无人实体位姿信息并将其发送，使每个虚拟无人实体保持正确的位姿。

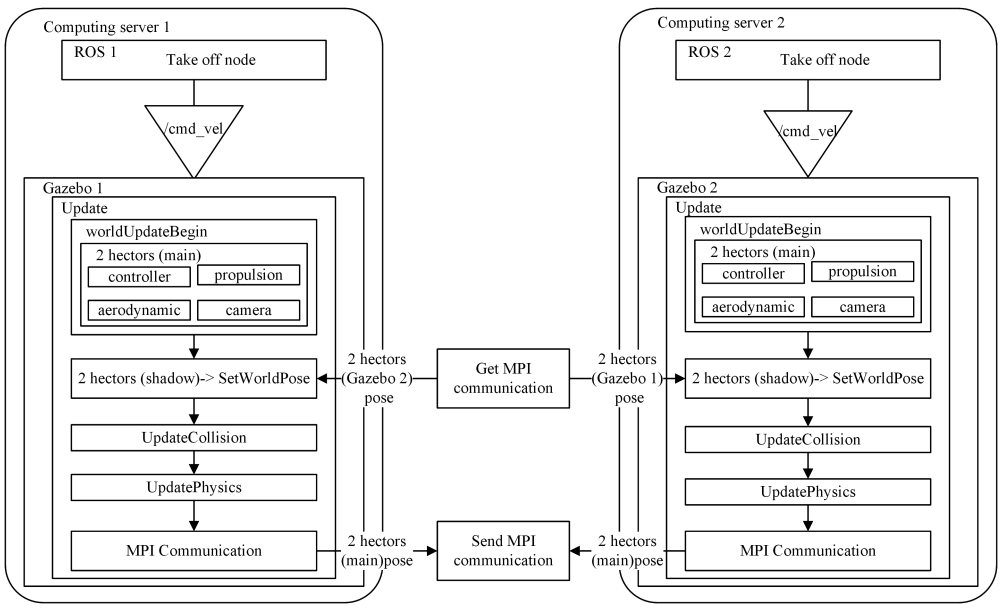

图 5 基于四旋翼无人机案例的跨节点 Gazebo 仿真方案

Fig. 5 Gazebo simulation scheme across nodes based on quadrotorcase diagram

代码结构设计方面，通过分析源码主要修改了 World 类，增加了一些 MPI 相关内容的接口，设计了 Distribution 类和 GazeboID 类，实现了分布式信息的输入，增加了 CommunicationData 结构体变量，包含了模型的位姿数据，还修改了 Gazebo 与 ROS 的接口 gazebo_ros_pkg 包支持 MPI 节点信息的标签功能，并通过 World 类新增加的接口函数实

现模型从属节点信息的输入。

3.2 仿真流程实现

跨节点仿真模块如图 6 所示，整个流程基于 Gazebo 原始逻辑，本文主要更改了其中的模型加载、参数解析和世界更新子模块，图中用虚线框标识的内容均进行了修改或者添加。修改后的 gzserver 运行逻辑如图 7 所示，在 gzserver 启动过程中增加了 MPI 环境的设置，实现多进程的开启。

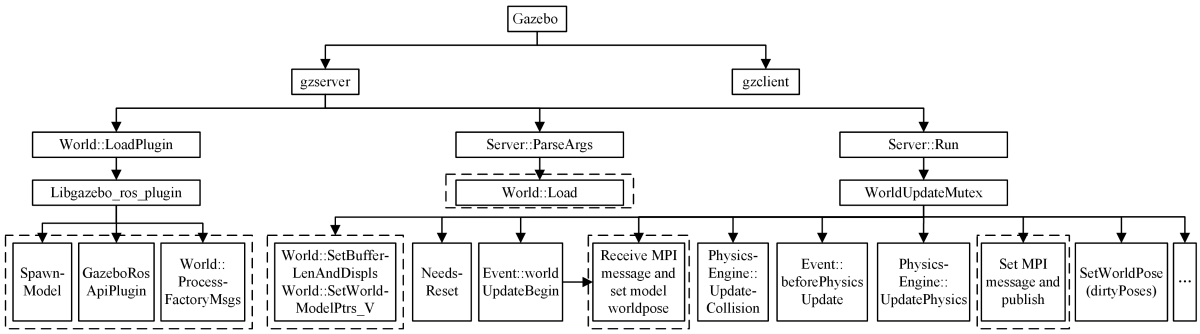

图 6 分布式并行 Gazebo 仿真模块图

Fig. 6 Gazebo of distributed parallelism module diagram

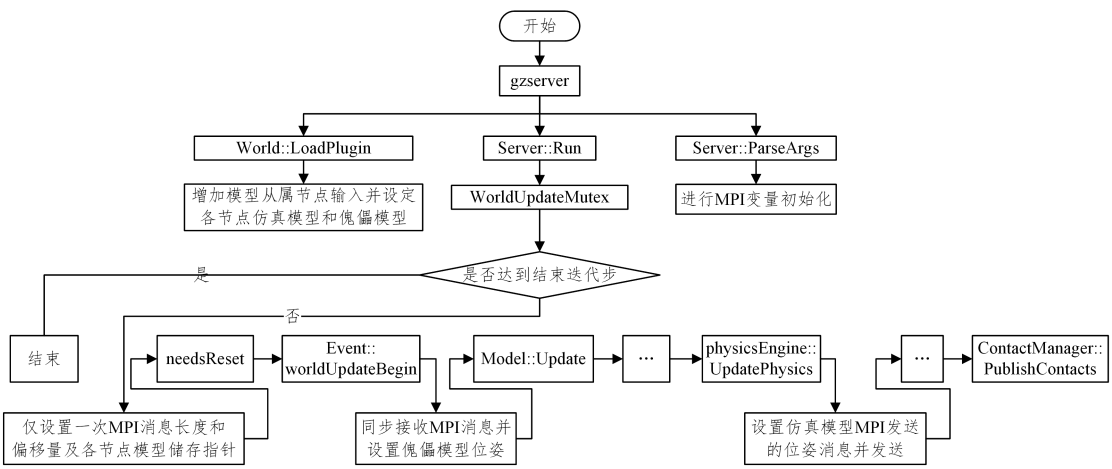

图 7 分布式并行 gzserver 运行流程图

Fig. 7 Operation flow diagram of gzserver of distributed parallelism

下面分别介绍各模块的主要内容。

(1)模型加载模块，修改 launch 启动文件中的 SpawnModel 类，增加标签信息实现模型从属节点消息输入。修改 GazeboRosApiPlugin 插件，指定仿真模型运行节点、设置 MPI 信息。当通过 launch 文件进行模型加载时，World::ProcessFactoryMsgs 会监听是否有模型加载，若有则调用 World::LoadModel 函数进行模型加载，在加载过程中通过判断模型从属节点号和本节点号是否一致来区分仿真实体和傀儡实体。仿真实体需要加载模型插件进行物理运算，傀儡实体设置为静态，不加载模型插件，只是作为同步显示和检测碰撞。因此在 SpawnModel 模块下增加傀儡模型信息的伪代码如下。

/ * 各节点模型加载逻辑 * /
输入：launch 文件加载 SpawnModel 模块
输出：加载仿真模型和傀儡模型
1. Read model Gazebo ID from launch file
2. World::LoadModel
3. if(gazeboLocalID == gazebo_id) //若模型从属于本节点则将其添加入指针向量
4. then model add to ownModels
5. else
6. model is static，not calculated //否则作为傀儡模型
7. World::ProcessFactoryMsgs
8. for (i←0 to modelCounts) //循环所有的模型包括仿真模型和傀儡模型
9. if model is in ownModels //判断是否为本节点仿真模型
10. then model loads plugins //仿真模型加载模型插件

(2)参数解析模块，World::Load 函数中增加了 sdf 文件中 distribution 内容的读入和 MPI 相关变量的初始化，增加内容的伪代码如下所示。

/ * 解析 distribution 标签内容，初始化 MPI 变量 * /
输入：sdf 文件(world 文件)
输出：变量初始化
1. if (sdf has Element("distribution") && flag is 1) //判断是否有分布式标签信息
2. then //初始化 MPI 相关变量
gazeboCounts (启动的 Gazebo 节点个数)，sendBufferLen (MPI 发送消息缓冲区中的数据个数)，modelCounts (仿真任务模型总数量)，bufferLen(MPI 接收数据个数)，displs(MPI 接收数据偏移)，mpi_flag (MPI 消息传输与否标记)等变量初始化
3. call SetSendBufferData() // 根据仿真模型个数为发送数据开辟内存空间
4. call SetReceiveBufferData() // 为接收数据开辟内存空间

(3)世界更新模块，主要进行 MPI 进程间传递消息的设

定和收发。在 needsReset 前会首先设置消息长度和偏移量及仿真模型指针容器，此设置仅会在迭代开始时调用一次。模型指针容器 worldModelPtrs_V 的主要作用是存储从属于各节点上的仿真模型指针，此容器存在于每个节点并完全一致，有了模型指针就可以使用 Model 类进行模型信息的读取和设置。在进行 Model::Update 过程前，先进行 MPI 消息接收，为保证各节点仿真步调一致，在接收消息前需进行 MPI_Barrier 进程同步，接着通过指定接收数据长度和偏移量得到其他节点的 CommunicationData 消息，并通过 worldModelPtrs_V 容器中的模型指针设置 CommunicationData 数据中的位姿消息。在 physicsEngine::UpdatePhysics 过程后，采用 MPI_Iallgatherv 方式进行 MPI 消息的发送，MPI 发送数据内容通过本节点仿真模型指针获取的 CommunicationData 数据(模型位姿)来给定，在发送完数据后进行下一轮更新。因此在 load 函数下增加 MPI 变量设定值的伪代码如下。

```
/*仿真模型和傀儡模型位姿同步更新*/
输入:MPI 相关变量,仿真模型和傀儡模型指针向量
输出:模型位姿信息
1. World::Update
2.   if((1 ==flag) && (iteration is 1)) // 判断分布式标签以及是否为计算初始
3.   then
       /*设置消息长度和偏移量及各节点仿真模型指针容器,该设置只在开始进行*/
4.     call SetBufferLenAndDispls() // 设置接收数据个数和偏移量
5.     call SetWorldModelPtrs_V() // 设置各节点存储的模型指针向量
6. World::needsReset
7. Event::WorldUpdateBegin // 为仿真模型进行预处理过程
8.   for (i←0 to gazeboCounts ) // 循环所有 Gazebo 节点
9.     if (gazeboLocalID !=i) // 设置除本节点以外所有节点模型位置
10.    then
11.      MPI_Barrier // MPI 同步
12.      Receive MPI msgs and SetWorldPose //将接收的位姿信息赋予相应模型中
13. Model::Update
14. PhysicsEngine::UpdateCollsion
15. Event::beforePhysicsUpdate
16. physicsEngine::UpdatePhysics
17. SetWorldPose(dirtyPoses)
18. for (i←0 to local gazebo modelcounts) //循环本节点内的仿真模型
19.    set sendBufferData[i] ← get modelpose[i]
20. MPI_Iallgatherv //MPI 消息发送
```

3.3 启动和结束仿真方式设计

分布式仿真案例的启动方式主要通过脚本程序读入 world 配置文件和 roslaunch 文件来实现。为了对用户友好，设计了与 ROS 类似的启动方式，启动命令如下所示：

./MPI_Gazebo_launch.sh [ros_pkg_name] [roslaunch file] [world file] [hostfile] [rankfile]

MPI_Gazebo_launch.sh 脚本程序主要完成如下内容：1)读入节点 hostfile 文件信息以便实现跨节点 ssh 登录；2)跨节点启动各自 roscore 节点；3)读入节点 rankfile 文件实现指定核数运行，以 mpiexec 方式启动 gzserver 进程加载 world 文件；4)登录各节点加载 roslaunch 文件。

world 文件中增加分布式仿真开关标志和需要使用的节点数目内容，通过 flag 值判定是否开启跨节点仿真，通过指定 gazebo_id 来指定要启动的节点。增加内容的代码示例如下所示。

```
〈distribution flag=1〉
  〈gazebo_id num=0 〉
  〈/gazebo_id〉
  〈gazebo_id num=1〉
  〈/gazebo_id〉
    ……
〈/distribution〉
```

roslaunch 文件在 spawn 节点里增加 gazebo_id 标签，gazebo_id 用于指定该模型的从属节点，增加内容的代码示例如下所示。

```
〈launch〉
  〈group ns="model_name"〉
    〈include file="$(find ros_pkg_name)/launch/spawn.launch"〉
    ……
    〈arg name="gazebo_id" value="0" /〉
    ……
    〈/include〉
  〈/group〉
〈/launch〉
```

当仿真任务结束时，执行 MPI_Gazebo_end.sh 脚本，该脚本主要以结束进程的方式来结束仿真，结束后 roscore，roslaunch，mpiexec，gzserver 进程均会被终止，命令如下所示：

./MPI_Gazebo_end.sh [hostfile] [rankfile]

4 分布式并行 Gazebo 仿真性能测试

为了更好地验证分布式 Gazebo 的性能优势，使用两台一样的 DELL 台式机，针对四旋翼 Hector 案例、固定翼 rosplane 案例和异构无人机案例比较单节点和跨节点仿真性能方面的差异。

计算机配置方面，CPU 配置为 Intel Core i7-8700 CPU @ 3.20 GHz × 12，32 GB 内存，图形显卡为 GeForce GTX 1050Ti。两个案例分别启动 30 架、60 架、100 架无人机进行测试，仿真步长为 1 ms，迭代步数为 100 000(即仿真时间 100 s)，统计各模块和总的实际执行时间，得到 1 个仿真步长的平均执行时间和仿真实时率，再进行不同架次的横向比较。同构无人机测试结果如表 1－表 4 所列。通过测试发现，无论是四旋翼还是固定翼无人机案例，分布式仿真均有较显著的加速效果，尤其在无人机数量较多的情况下在性能方面得到了较大的提升，整体加速比为 1.4～1.8。从模型预处理模块的对比可以看出，通过并行分布式仿真能够有效降低该模块执行时间，模块加速比近乎呈线性增长。从表 1 和表 3 的数据中发现，四旋翼(挂载相机)和固定翼(挂载相机)仿真案例(60 架和 100 架)中的模型预处理模块和物理更新模块加速比接近线性比 2，甚至固定翼案例超过了线性比 2，作为对比测试了未挂载相机的固定翼无人机案例和四旋翼案例。从表 2 和表 4 数据可以看出，未挂载相机的情况下模型预处理模块和物理更新模块加速比均未超过 2，这是由于无人机如果全部携带了相机传感器，在 Gazebo 中的仿真中会对每个相

机传感器开启单独的线程进行计算，分布式并行仿真后，原来相机传感器的线程计算平均分到了两个计算节点上，降低了线程数量，节省了计算资源，造成超线性加速比的现象。引入并行分布式仿真后，增加了 MPI 通信模块，需将本计算节点负责仿真的无人机位姿消息传递给另外的计算节点并保持仿真同步，从测试也可以看出分布式仿真并未达到理论加速效果，尤其是从 30 架无人机测试案例可以看出，当无人机数量较低时，MPI 的通信模块耗费的时间会抵消计算模块的加速效果，后续可通过通信模块优化、松耦合等方法进一步提升并行分布式仿真的性能。

同时为了进一步验证分布式仿真的通用性，如图 8 所示设计了固定翼和四旋翼同时存在的异构无人机仿真场景。整个场景中包含了 30 架固定翼无人机和 30 架四旋翼无人机，分别采用两种方式进行任务划分即区域划分和编号划分。区域划分是将 1 区四旋翼无人机在节点 1 上进行仿真，2 区固定翼无人机在节点 2 上进行仿真；编号划分是将 1 区的 15 架四旋翼飞机和 2 区的 15 架固定翼飞机放入节点 1 进行仿真，剩余的无人机放入节点 2 进行仿真。从表 5 可以看出无论任务是区域划分还是编号划分，分布式仿真都较显著地提高了仿真性能，整体加速比在 1.6 至 1.8 之间，由此进一步验证了 MPI 分布式仿真的可行性和优越性。

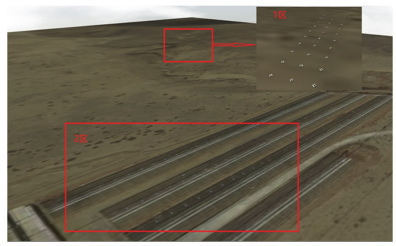

图 8　异构无人机仿真案例

Fig. 8　Heterogeneous UAV case diagra

表 1　四旋翼无人机(挂载相机)测试结果

Table 1　Quadrotor(mounted camera) test results

案例描述	30 架			60 架			100 架		
	单节点	两节点	加速比	单节点	两节点	加速比	单节点	两节点	加速比
模块更新处理模块	1.099	0.575	1.911	2.873	1.450	1.981	5.333	2.679	1.990
碰撞检测模块	0.025	0.022	1.136	0.067	0.059	1.136	0.125	0.095	1.316
物理更新模块	0.044	0.036	1.222	0.116	0.088	1.318	0.217	0.185	1.173
MPI 通信模块	0	0.102	—	0	0.152	—	0	0.202	—
其他模块	0.070	0.048	1.458	0.176	0.112	1.571	0.293	0.208	1.410
整体更新	1.238	0.783	1.581	3.232	1.861	1.737	5.968	3.369	1.771
平均 RTF	0.807	1.275	—	0.310	0.538	—	0.168	0.298	—

表 2　四旋翼无人机(未挂载相机)测试结果

Table 2　Quadrotor(unmounted camera) test results

案例描述	30 架			60 架			100 架		
	单节点	两节点	加速比	单节点	两节点	加速比	单节点	两节点	加速比
模块更新处理模块	0.930	0.502	1.853	2.151	1.113	1.933	4.405	2.271	1.940
碰撞检测模块	0.019	0.016	1.188	0.051	0.046	1.109	0.100	0.090	1.111
物理更新模块	0.034	0.029	1.172	0.088	0.077	1.143	0.171	0.145	1.179
MPI 通信模块	0	0.110	—	0	0.143	—	0	0.259	—
其他模块	0.056	0.041	1.366	0.138	0.11	1.255	0.261	0.205	1.273
整体更新	1.038	0.697	1.489	2.428	1.487	1.633	4.936	2.967	1.664
平均 RTF	0.963	1.476	—	0.412	0.673	—	0.203	0.349	—

表 3　固定翼无人机(挂载相机)测试结果

Table 3　Fixed wing rosplane(mounted camera) test results

案例描述	30 架			60 架			100 架		
	单节点	两节点	加速比	单节点	两节点	加速比	单节点	两节点	加速比
模块更新处理模块	0.222	0.124	1.790	0.699	0.324	2.157	2.509	1.167	2.150
碰撞检测模块	0.016	0.014	1.143	0.045	0.031	1.452	0.164	0.099	1.657
物理更新模块	0.070	0.044	1.591	0.182	0.089	2.045	0.623	0.284	2.194
MPI 通信模块	0	0.150	—	0	0.194	—	0	0.330	—
其他模块	0.137	0.082	1.671	0.358	0.210	1.705	1.125	0.658	1.710
整体更新	0.444	0.412	1.078	1.284	0.848	1.514	4.412	2.538	1.738
平均 RTF	2.250	2.428	—	0.779	1.190	—	0.204	0.358	—

表 4　固定翼无人机(未挂载相机)测试结果

Table 4　Fixed wing rosplane(unmounted camera) test results

案例描述	30 架			60 架			100 架		
	单节点	两节点	加速比	单节点	两节点	加速比	单节点	两节点	加速比
模块更新处理模块	0.179	0.094	1.904	0.485	0.250	1.940	0.947	0.483	1.961
碰撞检测模块	0.015	0.013	1.154	0.040	0.028	1.429	0.079	0.052	1.519
物理更新模块	0.071	0.040	1.775	0.184	0.102	1.804	0.337	0.173	1.948
MPI 通信模块	0	0.135	—	0	0.190	—	0	0.310	—
其他模块	0.085	0.050	1.700	0.242	0.173	1.399	0.426	0.302	1.411
整体更新	0.350	0.332	1.054	0.951	0.743	1.280	1.789	1.320	1.355
平均 RTF	2.857	3.028	—	1.051	1.356	—	0.559	0.760	—

(下转第 692 页)

Networks, 2020, 107: 102202.
[12] LU W, XU X, YE Q, et al. Power Optimization in UAV-Assisted Wireless Powered Cooperative Mobile Edge Computing Systems[J]. IET Communications, 2020, 14(15): 2516-2523.
[13] LIN H, ZEADALLY S, CHEN Z H, et al. A survey on computation offloading modeling for edge computing[J]. Journal of Network and Computer Applications, 2020, 169: 102781.
[14] LV Z, QIAO L. Optimization of collaborative resource allocation for mobile edge computing [J]. Computer Communications, 2020, 161: 19-27.
[15] RAUSCH T, RASHED A, DUSTDAR S. Optimized container scheduling for data-intensive serverless edge computing[J]. Future Generation Computer Systems, 2021, 114: 259-271.

LUAN Ling, born in 1978, BE, deputy senior economist. Her main research interests include intelligent cost control and edge computing.

(上接第 677 页)

表 5 异构无人机测试结果

Table 5 Heterogeneous UAV test results

案例描述 / 模块名称	按区域划分			按编号划分		
	单节点	两节点	加速比	单节点	两节点	加速比
模型预处理模块	4.097	2.133	1.921	4.097	2.120	1.933
碰撞检测模块	0.137	0.113	1.212	0.137	0.114	1.202
物理更新模块	0.339	0.275	1.233	0.339	0.240	1.413
MPI 通信模块	0	0.091	—	0	0.180	—
其他模块	0.483	0.317	1.524	0.483	0.430	1.123
整体更新	5.056	2.929	1.726	5.056	3.084	1.639
平均 RTF	0.200	0.348	—	0.200	0.320	—

结束语 本文为了解决机器人集群仿真在 Gazebo 平台上存在性能瓶颈问题，通过 MPI 进程通信的方式使得 Gazebo 形成跨节点分布式仿真，并进行了单节点和两节点的性能测试对比，验证了跨节点分布式仿真的性能优越性。但是目前形成的跨节点 Gazebo 版本虽然解决了多进程协同仿真的问题，但由于集群仿真通信频率高、通信量大等问题，其仿真性能依然存在较大瓶颈，如何提高通信性能依然是亟待解决的问题。另外，目前的开发是基于 ROS+Gazebo 仿真，对于单独 Gazebo 仿真需通过另外一套逻辑实现，如何将其统一也是需要解决的问题。目前开发的分布式 Gazebo 也不支持模型动态增删和跨节点动态调整的功能，为更好地支持仿真任务同样需要开发此类功能。因此后续的开发可继续完善这些不足，最终实现稳定、高效、功能强大的跨节点 Gazebo 仿真平台。

参 考 文 献

[1] REN F J, SUN X. Present Situation and Development of Intelligent Robots[J]. Science & Technology Review, 2015, 33(21): 32-38.
[2] LIU Y, XU H, GENG C X, et al. Research on Manipulator Motion Planning in ROS/Gazebo[J]. Coal Mine Machinery, 2018, 39(3): 42-44.
[3] YAO W J. Distributed Multi-robot Circumnavigation Formation Control[D]. Changsha: National University of Defense Technology, 2017.
[4] CHEN X, XIA Y C, HUANG X H. A Digital Hard-in-the-Loop Real Time Simulation System for Flight Control[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2001, 33(2): 200-203.
[5] ZHAO Z J, ZHONG S J, LI J L, et al. Ground Simulation of UVA Flight Control[J]. Ordnance Industry Automation, 2013, 32(8): 32-34.
[6] JIANG Y, YUANG M T. Construction of Robot Flexible Operation Control Platform Based on Gazebo[J]. Industrial Control Computer, 2018, 31(12): 44-46.
[7] TAN Z. Some Problems in Formation Control of Quadrotor [D]. Hangzhou: Hangzhou Dianzi University, 2019.
[8] HAO P P, YAO L F. Mobile Robot Path Planning Simulation based on Player/Gazebo[J]. Education Circle, 2010(10): 138.
[9] ZHANG J W, ZHANG L W, HU Y, et al. Open source operating system-ROS[M]. Beijing: Science Press, 2012.
[10] XIE M. Vehicle remote control and navigation simulation based on ROS[D]. Hefei: University of Science and Technology, 2019.
[11] WANG D G, WANG B H. Seven-degree-of-freedom robot arm Simulation and Motion Planning based on ROS[J]. Science and Technology & Innovation, 2018(2): 116-117.
[12] QUIGLEY M, CONLEY K, GERKEY B P, et al. ROS: an open-source Robot Operating System[C] //ICRA Workshop on Open Source Software. 2009.
[13] YU W L. A Mapping Simulation Design of Mobile Robot SLAM Based on ROS[J]. Software Engineering and Applications, 2020, 9(4): 294-301.
[14] FAN P Q, ZHANG L, TANG S, et al. Impletmentation and Performance Analysis of MPI+Open MPI Hybrid Programming [J]. Computer Science and Application, 2019, 9(10): 1859-1866.
[15] ZHANG Z H. The Study of Parallel Computing Based on MPI [D]. Beijing: China University of Geosciences, 2006.

JIANG Hua-nan, born in 1988, master. His main research interests include high-performance computing, robotics and finite element simulation.

ZHANG Shuai, born in 1986, Ph.D. His main research interests include robotics and CFD simulation.