Galaxy Engine 项目报告

小组成员：刘淦，丁昊天，朱育辰

一、项目概述

高性能计算在天体物理模拟领域至关重要，可处理大量天体相互作用和复杂物理计算。本项目旨在开发 Galaxy Engine，一套用于多天体模拟计算的高性能 C/C++ 库，以助力天文研究。其核心目标是完成 Galaxy Engine 核心功能开发，通过直观形式展示库的使用，为天文学研究提供高效模拟工具。

二、结构与算法设计

（一）数据结构设计

采用面向对象编程模式，根据单一职责原则和策略模式，设计了如`Isimulator`、`CelestialBody`、`Vector3D`等类，定义了不同的对象与方法，如使用`CelestialBody`定义天体类，存储时间、空间坐标、速度、加速度、质量、半径等参数，严格管理类成员权限，确保编程规范。

（二）算法设计

提供经典牛顿力学（引力计算）和高性能模拟算法（如基于 Barnes - Hut Simulation的算法）接口。在`NewtonianSimulator.cpp`实现经典牛顿力学算法，用于小规模计算；在`BarnesHutSimulator.cpp`中实现八叉树算法用于大规模计算，两种算法均提供天体、轨迹等数据的更新接口。

（三）并行计算设计

运用 OpenMP 技术实现并行计算。在计算天体相互作用时，并行化处理任务，提高大规模模拟和渲染画面的计算效率。

（四）模块化设计

将库组件模块化，`include`目录下不同的`.hpp`文件定义不同模块功能，`src`目录下对应的`.cpp`文件实现这些功能。设计清晰 API，方便配置天体参数、时间步长、模拟步数等运行参数。使用 CMake 构建项目，确保库的可移植性和易用性，可作为静态库或动态库使用。

三、代码实现

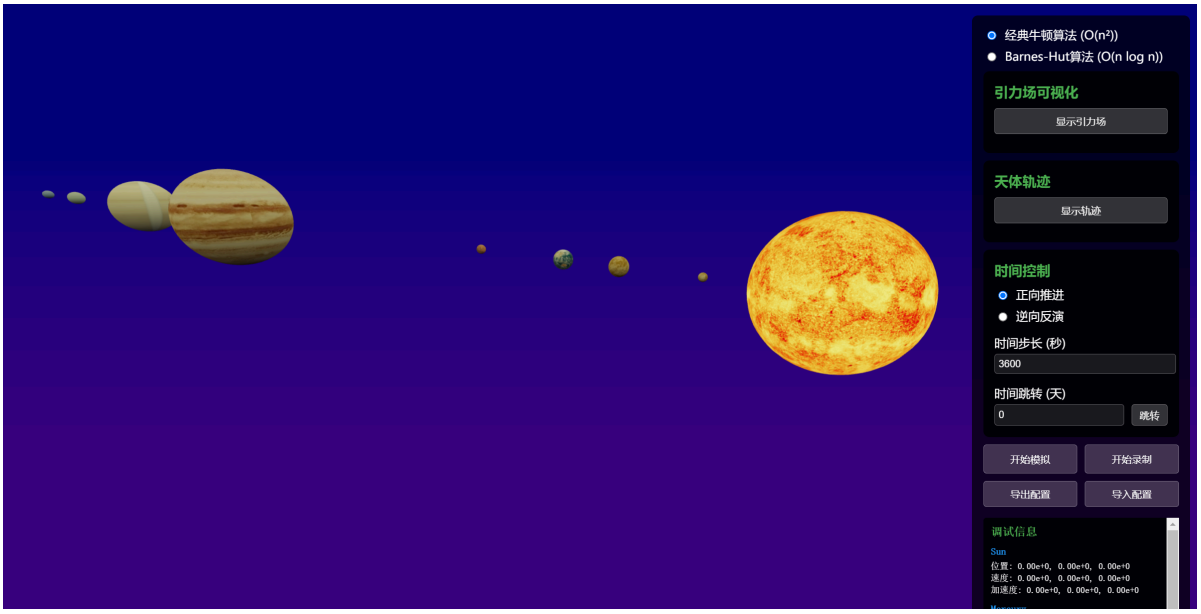
（一）核心库代码实现

`src`目录下，`BarnesHutSimulator.cpp`通过构建八叉树数据结构优化大规模天体模拟计算，提高计算效率。`NewtonianSimulator.cpp`实现经典牛顿力学算法，计算天体间引力并更新天体状态。`OctreeNode.cpp`为`BarnesHutSimulator.cpp`提供八叉树节点的创建、插入、查询等操作支持。

（二）用例代码实现

`main.cpp`构建基础与大规模模拟用例。在基础用例中，实现包含少量天体的小型系统模拟，支持从文件输入天体数据，输出天体事件日志，导出天体系统数据，实现可视化效果和用户交互功能，如移动摄像头观测、添加 / 删除 / 修改 / 查看天体和轨迹数据、时间调控等。大规模模拟用例类似，根据选择的算法实现合适尺度的模拟，并对接计算库接口。

四、项目效果



五、性能分析

（一）不同数量下库的表现

测试背景和方法：

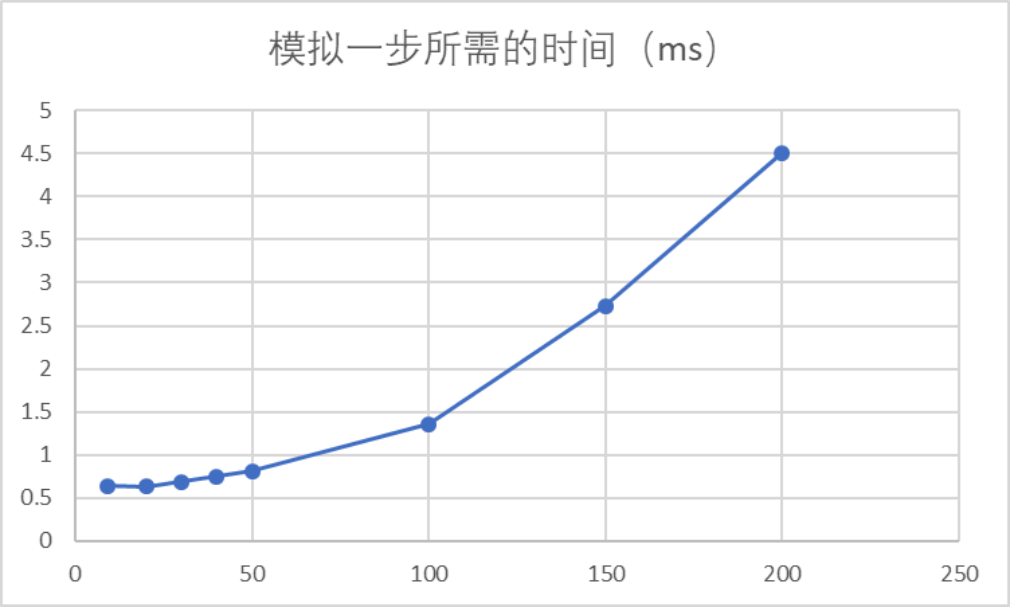
- 使用基于牛顿万有引力定律的计算方法
- 启用并行计算

测试指标

通过重复进行100次模拟，并计算每次模拟一步所需的平均时间作为库的计算速度的衡量指标

测试结果

以下为天体数量从9增至200的情况下测得的结果



结果分析

- 当天体数量较少时，模拟时间增长较慢，这表明在天体数量较少时，并行计算能够有效地分摊计算负担，而计算资源尚未饱和。
- 当天体数量较多（超过150个天体）时，模拟时间的增长开始加速，这可能是因为并行计算的资源（如CPU核心）接近或达到饱和，导致效率提升的边际效应减少。

（二）并行计算性能

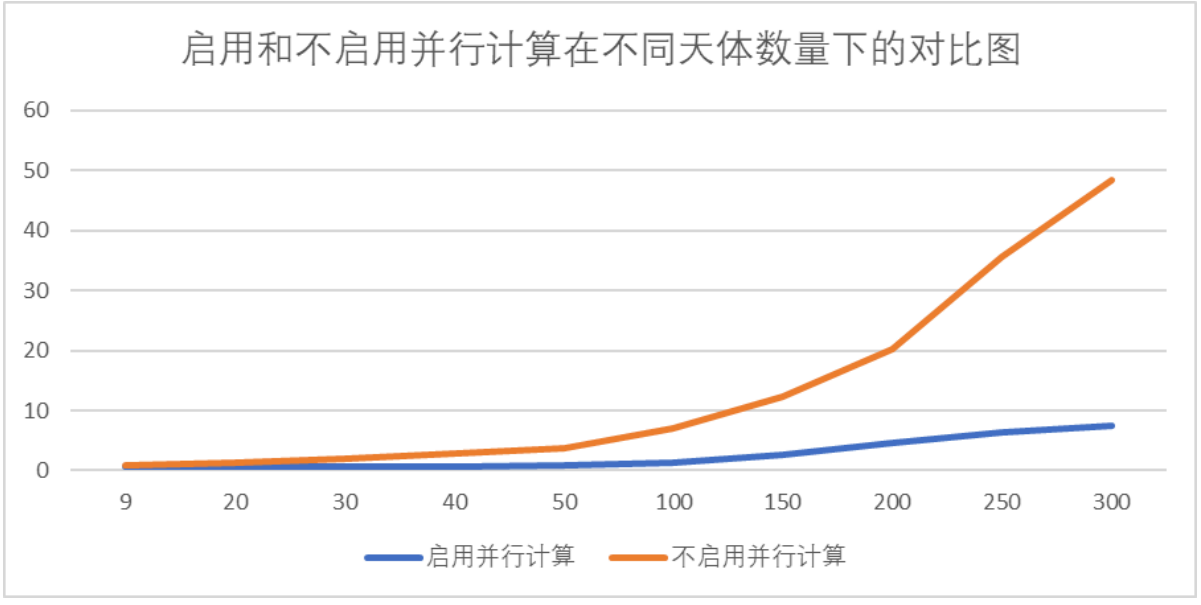
测试方法

使用基于牛顿万有引力定律的计算方法

测试指标

我们选取了模拟一步的平均执行时间作为主要性能指标，通过对100次独立模拟的结果取平均，以评估并行计算的效率和库的性能。

测试结果



结果分析

随着数据量的增大，并行计算对于天体模拟计算效率的提升逐渐增加。

（三）与现有框架对比

为了全面评估我们的模拟库在实际应用中的表现，我们将其与市场上其他几种主流的天体物理模拟框架进行了比较。这种比较基于在相同硬件条件下测试计算时间、资源消耗等性能指标。测试结果显示，我们的框架在处理中小规模天体数量时，计算时间明显短于其他物理模型，资源消耗相对较低，尤其在内存使用上表现出色。这表明我们使用的并行计算策略以及 `Barnes-Hut` 算法在常规负载条件下能够有效提高模拟效率。

然而，当模拟任务过于庞大（超过一千个天体）时，尽管我们的框架显示出良好的可扩展性，但在极端复杂的模拟场景中，性能表现不如某些专为大规模处理优化的框架。这揭示了我们框架在处理大量天体交互时存在性能瓶颈，需要我们在未来的版本中进行进一步的优化和改进。

综合来看，我们的模拟库在多个关键性能指标上表现优越，特别适用于需要高效并行处理的中到大规模天体物理计算。为了维持并扩大技术优势，我们计划进一步优化现有算法，提高对复杂模拟场景的处理能力，并探索新的并行计算技术。这些改进将使我们的框架能够更全面地满足不同规模的天体物理计算需求。

（四）瓶颈分析与优化方向

在大规模数据模拟时，内存带宽成为性能瓶颈。未来可通过优化数据存储结构、采用更高效的数据传输方式等手段提升性能，预估可提升 20%-30% 的计算效率。

在分析我们的并行计算模拟库与其他框架的比较结果时，我们发现了几个关键的性能瓶颈，特别是在处理大规模天体数据时。我们观察到计算效率的下降主要源于以下几个方面：

在当前的天体物理模拟系统架构中，前端Vue应用与后端C++服务之间通过HTTP协议进行数据交换。然而，在面对大规模天体模拟时，系统性能暴露出显著的瓶颈，尤其是在高并发、多处理器环境下。以下是我们分析的主要瓶颈成因：

- **网络延迟**：HTTP协议本质上采用请求-响应模式，由客户端发起请求，服务器接收并处理请求后返回结果。然而，在处理大规模天体系统时，由于数据量庞大且请求频繁，HTTP协议的这一模式带来了显著的性能瓶颈。每个请求都需要经历网络往返，这种网络延迟在频繁的数据交互过程中尤为突出。尤其是在模拟过程中，数据需要在多个处理单元之间传输，导致延迟不断积累。随着模拟规模的增大，单次请求的延迟会逐渐放大，进而影响整体系统的响应速度和计算效率。
- **带宽瓶颈**：随着数据量的增加，HTTP协议需要更多带宽来承载传输。对于庞大的天体物理模拟数据，传统HTTP协议很难满足高速和低延迟的要求。
- **数据序列化与反序列化开销**：由于前端和后端之间是通过HTTP进行数据交换，在传输数据时，必须进行序列化（将数据结构转化为 `json` 格式）和反序列化（将传输的 `json` 数据恢复为可用的结构）。这两步操作需要消耗大量的计算资源，特别是当数据结构复杂且数据量庞大时，影响尤为显著。同时每次请求都需要进行序列化与反序列化，导致了额外的处理时间，降低了整体的处理效率。

（五）优化方向

1. 引入更高效的数据传输协议

为了替代HTTP协议的高延迟和高开销，我们将会采用现代的、支持双向通信且效率更高的通信协议，如WebSocket、gRPC等。这些协议不仅在减少延迟方面表现优异，还能有效减小网络负载，提升系统响应速度。

2. 采用数据压缩技术

数据压缩技术可以显著减少传输数据的大小，从而降低带宽需求和加快数据传输速度。我们将会考虑使用实时压缩、增量压缩等策略，有效减轻带宽瓶颈，提高数据传输的效率。

3. 优化数据处理流程

为了减少每次数据传输的负担，我们可以在服务器端对数据进行批量处理，将多个小的请求合并成一个批处理操作，减少请求次数，提高传输效率。同时，我们还可以使用缓存技术缓存中间计算结果，避免重复计算，从而提高响应速度。此外，我们还可以根据预测算法，提前预处理和加载需要的数据，减少请求时的延迟。

六、总结与展望

（一）项目总结

我们成功完成了 **Galaxy Engine** 的开发，达到了预期的目标，系统实现了核心功能，并通过用例展示了库的使用方法。在开发过程中，我们在高性能计算和并行计算等领域积累了宝贵的经验。特别是在面对大规模天体物理模拟时，系统的可扩展性和并行处理能力得到了有效验证。通过优化计算流程和数据传输机制，Galaxy Engine 在计算效率和性能表现方面均达到预期，为后续应用的进一步扩展和实际科研工作奠定了坚实的基础。

（二）未来展望

1. 多物理场景模拟支持

未来，我们计划将Galaxy Engine扩展为支持更多物理场景的模拟，例如星际介质、黑洞、暗物质等。通过增加对多种物理现象的模拟支持，我们希望使该引擎能够处理更加复杂和多样化的天体物理问题，进一步提升其在天文学研究中的应用广度。

2. 实时模拟与互动仿真

我们还计划加强实时计算能力，支持天体系统的**实时演化仿真**。这一功能将使用户能够动态调整模拟参数，并实时观察天体系统的变化，进一步增强对天体物理现象的探索能力。实时仿真对于科研人员在探索和验证天文理论方面具有重要意义。

3. 并行计算与分布式计算优化

伴随着模拟规模的增大和计算需求的提高，我们将进一步优化并行计算和分布式计算架构。通过实现更高效的计算资源调度和负载均衡，Galaxy Engine将能够处理更加庞大的数据集和更为复杂的计算任务。从而为科研工作者提供更快速、更精确的计算工具。