**C#主导的多语言洋流模拟系统项目结构**

OceanCurrentSimulationSystem/

│

├── Documentation/ # 项目文档

│ ├── API/

│ │ ├── RestAPI\_Specification.md # REST API接口规范

│ │ ├── CppCore\_Interface.md # C++核心模块接口文档

│ │ ├── PythonEngine\_Interface.md # Python引擎接口文档

│ │ └── CrossLanguage\_Integration.md # 跨语言集成指南

│ ├── UserManual/

│ │ ├── SystemOverview.md # 系统概述

│ │ ├── SimulationWorkflow.md # 模拟工作流程

│ │ └── TroubleshootingGuide.md # 故障排除指南

│ ├── TechnicalSpecs/

│ │ ├── DataFormats.md # 数据格式规范

│ │ ├── AlgorithmSpecifications.md # 算法技术规范

│ │ ├── PerformanceRequirements.md # 性能要求文档

│ │ └── MemoryManagement.md # 内存管理策略

│ └── AlgorithmPapers/ # 算法参考论文

│ ├── LSTM-OceanPrediction-Zhu2019.pdf # LSTM海洋预测论文

│ ├── PINN-Raissi2019.pdf # 物理信息神经网络论文

│ ├── HYCOM-Bleck2002.pdf # HYCOM数值模式参考

│ ├── ParticleTracking-Paris2013.pdf # 拉格朗日粒子追踪论文

│ └── HighPerformanceComputing-MPI.pdf # 高性能计算参考

│

├── Source/ # 源代码主目录

│ │

│ ├── CharpClient/ # C#核心业务层（系统主控制器）

│ │ ├── OceanSimulation.sln # Visual Studio主解决方案

│ │ │

│ │ ├── OceanSimulation.Domain/ # 领域模型层

│ │ │ ├── Entities/

│ │ │ │ ├── OceanDataEntity.cs # 海洋观测数据实体

│ │ │ │ ├── CurrentFieldEntity.cs # 洋流速度场实体

│ │ │ │ ├── ParticleTrajectoryEntity.cs # 粒子运动轨迹实体

│ │ │ │ ├── PollutionCloudEntity.cs # 污染物扩散实体

│ │ │ │ └── SimulationSessionEntity.cs # 模拟会话实体

│ │ │ ├── ValueObjects/

│ │ │ │ ├── GeographicCoordinate.cs # 地理坐标值对象

│ │ │ │ ├── TimeRange.cs # 时间区间值对象

│ │ │ │ ├── VelocityVector.cs # 速度矢量值对象

│ │ │ │ └── ComputationGrid.cs # 计算网格值对象

│ │ │ ├── DomainServices/

│ │ │ │ ├── CurrentAnalysisService.cs # 洋流状态分析业务逻辑

│ │ │ │ ├── TrajectoryCalculationService.cs # 粒子轨迹计算服务

│ │ │ │ ├── PollutionDispersionService.cs # 污染扩散分析服务

│ │ │ │ └── PerformanceOptimizationService.cs # 性能优化服务

│ │ │ └── Interfaces/

│ │ │ ├── IDataRepository.cs # 数据仓储接口

│ │ │ ├── ICppComputeEngine.cs # C++计算引擎接口

│ │ │ ├── IPythonMLEngine.cs # Python机器学习引擎接口

│ │ │ └── IVisualizationService.cs # 可视化服务接口

│ │ │

│ │ ├── OceanSimulation.Infrastructure/ # 基础设施层

│ │ │ ├── ComputeEngines/

│ │ │ │ ├── CppEngineWrapper.cs # C++引擎包装器

│ │ │ │ ├── PythonEngineClient.cs # Python引擎客户端

│ │ │ │ └── HybridComputeOrchestrator.cs # 混合计算编排器

│ │ │ ├── DataAccess/

│ │ │ │ ├── OceanDataRepository.cs # 海洋数据仓储实现

│ │ │ │ ├── ResultRepository.cs # 结果数据仓储实现

│ │ │ │ └── ConfigurationRepository.cs # 配置数据仓储

│ │ │ ├── ExternalServices/

│ │ │ │ ├── DataProviderService.cs # 数据提供商服务

│ │ │ │ ├── WeatherService.cs # 天气服务

│ │ │ │ └── CloudComputeService.cs # 云计算服务

│ │ │ └── Caching/

│ │ │ ├── MemoryCacheService.cs # 内存缓存服务

│ │ │ └── FileCacheService.cs # 文件缓存服务

│ │ │

│ │ ├── OceanSimulation.Application/ # 应用服务层

│ │ │ ├── Commands/

│ │ │ │ ├── StartSimulationCommand.cs # 启动模拟命令

│ │ │ │ ├── OptimizePerformanceCommand.cs # 性能优化命令

│ │ │ │ └── ExportResultsCommand.cs # 导出结果命令

│ │ │ ├── Queries/

│ │ │ │ ├── GetCurrentFieldQuery.cs # 获取洋流场查询

│ │ │ │ ├── GetSimulationResultQuery.cs # 获取模拟结果查询

│ │ │ │ └── GetPerformanceMetricsQuery.cs # 获取性能指标查询

│ │ │ ├── Services/

│ │ │ │ ├── SimulationOrchestrationService.cs # 模拟编排服务

│ │ │ │ ├── ComputeResourceManager.cs # 计算资源管理器

│ │ │ │ └── CrossLanguageCoordinator.cs # 跨语言协调器

│ │ │ └── DTOs/

│ │ │ ├── SimulationParametersDto.cs # 模拟参数传输对象

│ │ │ ├── PerformanceMetricsDto.cs # 性能指标传输对象

│ │ │ └── ComputationResultDto.cs # 计算结果传输对象

│ │ │

│ │ ├── OceanSimulation.Presentation.WPF/ # WPF表示层

│ │ │ ├── App.xaml & App.xaml.cs # 应用程序入口

│ │ │ ├── MainWindow.xaml & .cs # 主窗口

│ │ │ ├── ViewModels/

│ │ │ │ ├── MainViewModel.cs # 主窗口视图模型

│ │ │ │ ├── HistoricalAnalysisViewModel.cs # 历史分析视图模型

│ │ │ │ ├── RealTimeMonitoringViewModel.cs # 实时监控视图模型

│ │ │ │ ├── PredictionSimulationViewModel.cs # 预测模拟视图模型

│ │ │ │ ├── ParticleTrackingViewModel.cs # 粒子追踪视图模型

│ │ │ │ ├── PollutionAnalysisViewModel.cs # 污染分析视图模型

│ │ │ │ └── PerformanceMonitorViewModel.cs # 性能监控视图模型

│ │ │ ├── Views/

│ │ │ │ ├── HistoricalAnalysisView.xaml # 历史分析页面

│ │ │ │ ├── RealTimeMonitoringView.xaml # 实时监控页面

│ │ │ │ ├── PredictionSimulationView.xaml # 预测模拟页面

│ │ │ │ ├── ParticleTrackingView.xaml # 粒子追踪页面

│ │ │ │ ├── PollutionAnalysisView.xaml # 污染分析页面

│ │ │ │ └── PerformanceMonitorView.xaml # 性能监控页面

│ │ │ ├── Controls/

│ │ │ │ ├── MapVisualizationControl.xaml # 地图可视化控件

│ │ │ │ ├── VectorFieldControl.xaml # 矢量场控件

│ │ │ │ ├── TimeSeriesControl.xaml # 时间序列控件

│ │ │ │ ├── ParameterInputControl.xaml # 参数输入控件

│ │ │ │ └── PerformanceGaugeControl.xaml # 性能仪表控件

│ │ │ └── Services/

│ │ │ ├── UINotificationService.cs # 界面通知服务

│ │ │ ├── DialogService.cs # 对话框服务

│ │ │ └── VisualizationService.cs # 可视化服务

│ │ │

│ │ └── Tests/ # C#测试项目

│ │ ├── OceanSimulation.Domain.Tests/ # 领域层测试

│ │ ├── OceanSimulation.Application.Tests/ # 应用层测试

│ │ ├── OceanSimulation.Infrastructure.Tests/ # 基础设施测试

│ │ └── IntegrationTests/ # 集成测试

│ │ ├── CppIntegrationTests/ # C++集成测试

│ │ └── PythonIntegrationTests/ # Python集成测试

│ │

│ ├── CppCore/ # C++高性能计算核心

│ │ ├── CMakeLists.txt # CMake构建配置

│ │ ├── include/ # 头文件

│ │ │ ├── core/

│ │ │ │ ├── ParticleSimulator.h # 粒子模拟器

│ │ │ │ ├── CurrentFieldSolver.h # 洋流场求解器

│ │ │ │ ├── AdvectionDiffusionSolver.h # 平流扩散求解器

│ │ │ │ └── PerformanceProfiler.h # 性能分析器

│ │ │ ├── algorithms/

│ │ │ │ ├── RungeKuttaSolver.h # 龙格-库塔求解器

│ │ │ │ ├── FiniteDifferenceSolver.h # 有限差分求解器

│ │ │ │ ├── ParallelComputeEngine.h # 并行计算引擎

│ │ │ │ └── VectorizedOperations.h # 向量化运算

│ │ │ ├── data/

│ │ │ │ ├── GridDataStructure.h # 网格数据结构

│ │ │ │ ├── MemoryManager.h # 内存管理器

│ │ │ │ └── DataExporter.h # 数据导出器

│ │ │ └── utils/

│ │ │ ├── MathUtils.h # 数学工具

│ │ │ ├── ThreadPool.h # 线程池

│ │ │ └── Logger.h # 日志记录器

│ │ ├── src/ # 源文件实现

│ │ │ ├── core/

│ │ │ │ ├── ParticleSimulator.cpp # 粒子模拟器实现

│ │ │ │ ├── CurrentFieldSolver.cpp # 洋流场求解器实现

│ │ │ │ ├── AdvectionDiffusionSolver.cpp # 平流扩散求解器实现

│ │ │ │ └── PerformanceProfiler.cpp # 性能分析器实现

│ │ │ ├── algorithms/

│ │ │ │ ├── RungeKuttaSolver.cpp # 龙格-库塔求解器实现

│ │ │ │ ├── FiniteDifferenceSolver.cpp # 有限差分求解器实现

│ │ │ │ ├── ParallelComputeEngine.cpp # 并行计算引擎实现

│ │ │ │ └── VectorizedOperations.cpp # 向量化运算实现

│ │ │ ├── data/

│ │ │ │ ├── GridDataStructure.cpp # 网格数据结构实现

│ │ │ │ ├── MemoryManager.cpp # 内存管理器实现

│ │ │ │ └── DataExporter.cpp # 数据导出器实现

│ │ │ └── utils/

│ │ │ ├── MathUtils.cpp # 数学工具实现

│ │ │ ├── ThreadPool.cpp # 线程池实现

│ │ │ └── Logger.cpp # 日志记录器实现

│ │ ├── bindings/ # 语言绑定

│ │ │ ├── csharp/

│ │ │ │ ├── CppCoreWrapper.h # C#包装器头文件

│ │ │ │ ├── CppCoreWrapper.cpp # C#包装器实现

│ │ │ │ └── CppCoreInterop.cs # C# P/Invoke接口

│ │ │ └── python/

│ │ │ ├── pybind\_particle\_sim.cpp # Python粒子模拟绑定

│ │ │ ├── pybind\_current\_solver.cpp # Python洋流求解器绑定

│ │ │ └── pybind\_advection.cpp # Python平流扩散绑定

│ │ ├── tests/ # C++测试

│ │ │ ├── unit\_tests/ # 单元测试

│ │ │ ├── performance\_tests/ # 性能测试

│ │ │ └── integration\_tests/ # 集成测试

│ │ ├── build/ # 构建输出

│ │ │ ├── lib/ # 动态库输出

│ │ │ ├── bin/ # 可执行文件输出

│ │ │ └── python\_modules/ # Python模块输出

│ │ └── dependencies/ # 第三方依赖

│ │ ├── eigen/ # Eigen线性代数库

│ │ ├── tbb/ # Intel TBB并行库

│ │ └── openmp/ # OpenMP并行库

│ │

│ └── PythonEngine/ # Python科学计算引擎

│ ├── requirements.txt # Python依赖包列表

│ ├── setup.py # 安装配置文件

│ ├── config.yaml # 全局配置文件

│ ├── main.py # 引擎主入口程序

│ │

│ ├── core/ # 核心功能模块

│ │ ├── \_\_init\_\_.py

│ │ ├── data\_processor.py # 核心数据处理器

│ │ ├── netcdf\_handler.py # NetCDF文件处理器

│ │ ├── interpolation\_engine.py # 插值计算引擎

│ │ ├── coordinate\_transformer.py # 坐标系转换器

│ │ ├── quality\_controller.py # 数据质量控制器

│ │ └── cpp\_interface.py # C++接口包装器

│ │

│ ├── simulation/ # 模拟算法模块

│ │ ├── \_\_init\_\_.py

│ │ ├── particle\_tracking\_wrapper.py # 粒子追踪包装器（调用C++）

│ │ ├── current\_simulation\_wrapper.py # 洋流模拟包装器（调用C++）

│ │ ├── pollution\_dispersion.py # 污染物扩散模拟

│ │ └── hybrid\_solver.py # 混合求解器

│ │

│ ├── machine\_learning/ # 机器学习模块

│ │ ├── \_\_init\_\_.py

│ │ ├── deep\_learning/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── pinn\_ocean\_models.py # 海洋PINN模型

│ │ │ ├── lstm\_current\_predictor.py # LSTM洋流预测器

│ │ │ ├── convlstm\_spatiotemporal.py # ConvLSTM时空预测器

│ │ │ └── transformer\_models.py # Transformer模型

│ │ ├── model\_training/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── training\_pipeline.py # 训练流水线

│ │ │ ├── validation\_framework.py # 验证框架

│ │ │ └── hyperparameter\_tuning.py # 超参数调优

│ │ └── classical\_ml/

│ │ ├── \_\_init\_\_.py

│ │ ├── gaussian\_process.py # 高斯过程回归

│ │ └── ensemble\_methods.py # 集成学习方法

│ │

│ ├── data/ # 数据管理模块

│ │ ├── \_\_init\_\_.py

│ │ ├── acquisition/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── external\_data\_fetcher.py # 外部数据获取器

│ │ │ ├── satellite\_data\_processor.py # 卫星数据处理器

│ │ │ └── model\_output\_downloader.py # 模式输出下载器

│ │ ├── preprocessing/

│ │ │ ├── \_\_init\_\_.py

│ │ │ ├── data\_cleaner.py # 数据清洗器

│ │ │ ├── outlier\_detector.py # 异常值检测器

│ │ │ └── format\_converter.py # 格式转换器

│ │ └── validation/

│ │ ├── \_\_init\_\_.py

│ │ ├── data\_validator.py # 数据验证器

│ │ └── consistency\_checker.py # 一致性检查器

│ │

│ ├── visualization/ # 可视化支持模块

│ │ ├── \_\_init\_\_.py

│ │ ├── field\_generators.py # 场数据生成器

│ │ ├── animation\_generator.py # 动画生成器

│ │ └── rendering\_engine.py # 渲染引擎

│ │

│ ├── api/ # API服务模块

│ │ ├── \_\_init\_\_.py

│ │ ├── rest\_server.py # REST API服务器

│ │ ├── endpoints.py # API端点定义

│ │ └── middleware.py # 中间件

│ │

│ ├── utils/ # 工具函数库

│ │ ├── \_\_init\_\_.py

│ │ ├── math\_utils.py # 数学工具函数

│ │ ├── performance\_utils.py # 性能工具

│ │ └── logging\_utils.py # 日志工具

│ │

│ └── tests/ # Python测试

│ ├── unit/ # 单元测试

│ ├── integration/ # 集成测试

│ └── performance/ # 性能测试

│

├── Data/ # 数据存储目录

│ ├── NetCDF/ # NetCDF数据文件

│ │ ├── Historical/ # 历史数据

│ │ ├── RealTime/ # 实时数据

│ │ └── Forecast/ # 预报数据

│ ├── Models/ # 机器学习模型

│ │ ├── LSTM/ # LSTM模型文件

│ │ ├── PINN/ # PINN模型文件

│ │ └── TrainingData/ # 训练数据

│ ├── Results/ # 计算结果

│ │ ├── Simulations/ # 模拟结果

│ │ ├── Predictions/ # 预测结果

│ │ └── Analysis/ # 分析结果

│ ├── Cache/ # 缓存数据

│ │ ├── CppCache/ # C++计算结果缓存

│ │ ├── PythonCache/ # Python处理结果缓存

│ │ └── VisualizationCache/ # 可视化数据缓存

│ └── Export/ # 导出数据

│

├── Configuration/ # 配置文件

│ ├── appsettings.json # 主应用配置

│ ├── database.config # 数据库配置

│ ├── cpp\_runtime.config # C++运行时配置

│ ├── python\_service.yaml # Python服务配置

│ ├── performance\_tuning.yaml # 性能调优配置

│ └── logging.config # 日志配置

│

├── Scripts/ # 脚本文件

│ ├── build\_cpp.sh # C++模块构建脚本

│ ├── build\_cpp.bat # C++模块构建脚本（Windows）

│ ├── build\_python\_bindings.py # Python绑定构建脚本

│ ├── build\_csharp\_project.bat # C#项目构建脚本

│ ├── setup\_environment.py # 环境配置脚本

│ ├── download\_dependencies.py # 依赖下载脚本

│ ├── run\_performance\_tests.py # 性能测试脚本

│ ├── deploy\_hybrid\_system.py # 混合系统部署脚本

│ └── benchmark\_comparison.py # 基准性能对比脚本

│

├── Documentation/ # 项目文档

│ ├── SystemArchitecture.md # 系统架构文档

│ ├── PerformanceOptimization.md # 性能优化指南

│ ├── CrossLanguageIntegration.md # 跨语言集成指南

│ ├── BuildInstructions.md # 构建说明文档

│ └── APIReference.md # API参考文档

│

├── Assets/ # 静态资源

│ ├── Icons/ # 图标文件

│ ├── Images/ # 图片资源

│ ├── Maps/ # 地图资源

│ └── Styles/ # 样式文件

│

├── Build/ # 构建输出

│ ├── Debug/ # 调试版本

│ │ ├── CSharp/ # C#调试版本

│ │ ├── Cpp/ # C++调试版本

│ │ └── Python/ # Python调试版本

│ ├── Release/ # 发布版本

│ │ ├── CSharp/ # C#发布版本

│ │ ├── Cpp/ # C++发布版本

│ │ └── Python/ # Python发布版本

│ └── Packages/ # 安装包

│ ├── Windows/ # Windows安装包

│ ├── Linux/ # Linux安装包

│ └── Docker/ # Docker镜像

│

├── Deployment/ # 部署配置

│ ├── Docker/ # Docker配置

│ │ ├── Dockerfile.csharp # C#应用Docker文件

│ │ ├── Dockerfile.python # Python服务Docker文件

│ │ ├── Dockerfile.cpp # C++模块Docker文件

│ │ └── docker-compose.yml # 容器编排配置

│ ├── Kubernetes/ # Kubernetes配置

│ │ ├── deployment.yaml # 部署配置

│ │ └── service.yaml # 服务配置

│ └── Cloud/ # 云部署配置

│ ├── Azure/ # Azure部署配置

│ └── AWS/ # AWS部署配置

│

├── Tools/ # 开发工具

│ ├── DataValidator/ # 数据验证工具

│ ├── PerformanceTester/ # 性能测试工具

│ ├── MemoryProfiler/ # 内存分析工具

│ └── CrossLanguageDebugger/ # 跨语言调试工具

│

└── README.md # 项目说明文档

**技术栈分工策略**

**C#核心控制层**

C#承担系统的整体架构控制和业务逻辑管理职责。这包括用户界面展示、业务流程编排、数据管理策略、配置管理、异常处理和系统监控等关键功能。C#层通过标准化接口调用C++和Python模块，确保系统的整体协调和控制。

**C++高性能计算层**

C++专门负责计算密集型任务的高效执行。主要包括粒子追踪算法、洋流场数值求解、平流扩散方程计算、大规模矩阵运算和并行计算优化等核心计算功能。通过向量化指令、多线程并行和内存优化技术，C++模块能够提供最佳的计算性能。

**Python科学计算层**

Python负责机器学习模型训练、复杂数据处理、科学计算库调用和算法原型开发等任务。Python层还承担NetCDF文件解析、数据质量控制、统计分析和可视化数据生成等功能。Python通过绑定机制调用C++模块执行高性能计算任务。

**性能优化设计**

**跨语言通信优化**

系统采用高效的跨语言通信机制，C#通过P/Invoke技术直接调用C++动态库，避免了进程间通信的开销。Python通过pybind11与C++模块进行高效数据交换，支持零拷贝的大规模数组传递。这种设计确保了不同语言模块间的高效协作。

**内存管理策略**

实施统一的内存管理策略，避免跨语言边界的内存泄漏。C++模块负责大规模数据的内存分配和释放，Python和C#通过引用机制访问共享数据。采用内存池技术减少频繁的内存分配开销，提高系统整体性能。

**并行计算架构**

C++层实现多级并行计算架构，支持CPU多核并行、SIMD向量化计算和GPU加速计算。通过OpenMP实现循环级并行，使用Intel TBB进行任务级并行，充分利用现代硬件的计算能力。Python层通过多进程机制实现数据处理任务的并行化。

这种混合架构设计充分发挥了三种编程语言的技术优势，在保证开发效率的同时实现了系统性能的最大化。

**开发执行路径**

**第一阶段：基础架构搭建（第1-3周）**

**C++核心模块开发** 首先建立C++计算核心，实现基础的数值算法和数据结构。创建CMakeLists.txt构建系统，实现ParticleSimulator和CurrentFieldSolver核心类。这些模块将成为整个系统的性能基石，必须确保算法的正确性和计算效率。

**C#主控制器开发** 并行开发C#核心业务层，建立领域模型和应用服务架构。实现CppEngineWrapper类，通过P/Invoke技术与C++模块进行通信。建立基础的用户界面框架和数据管理系统，为后续功能开发提供支撑。

**Python绑定层实现** 使用pybind11创建Python与C++的绑定接口，确保Python能够高效调用C++计算模块。实现基础的数据处理功能和NetCDF文件读取能力，为机器学习模块的开发奠定基础。

**第二阶段：核心功能实现（第4-8周）**

**高性能计算优化** 在C++层实现并行计算优化，包括OpenMP多线程并行、SIMD向量化指令和内存访问优化。开发RungeKuttaSolver和AdvectionDiffusionSolver等核心算法，确保计算精度和性能表现。

**业务逻辑完善** 在C#层实现完整的业务逻辑，包括模拟参数配置、计算任务调度和结果数据管理。开发SimulationOrchestrationService，协调不同语言模块的协作执行。

**机器学习集成** 在Python层实现LSTM和PINN等深度学习模型，建立训练和推理流水线。通过调用C++模块执行数值计算任务，实现混合计算架构的有效协作。

**第三阶段：系统集成测试（第9-12周）**

**性能基准测试** 建立全面的性能测试体系，对比纯Python实现、纯C++实现和混合架构的性能差异。验证跨语言通信的效率，确保系统整体性能达到设计目标。

**功能集成验证** 进行端到端的功能测试，验证从数据输入到结果输出的完整流程。测试不同规模数据集的处理能力，确保系统在各种工作负载下的稳定性。

**用户界面完善** 完善WPF用户界面，实现性能监控、进度显示和结果可视化功能。确保用户能够直观地了解系统运行状态和计算进度。

**性能提升预期**

**计算性能优势**

通过C++实现的核心算法预期能够获得10-50倍的性能提升，具体提升幅度取决于算法类型和数据规模。粒子追踪等计算密集型任务将获得最显著的性能改善。

**内存使用优化**

C++的手动内存管理和优化的数据结构将显著降低内存使用量，预期能够减少30-60%的内存占用。这使得系统能够处理更大规模的海洋数据集。

**并行扩展能力**

多核并行和向量化优化将显著提升系统在多核处理器上的扩展能力。在8核处理器上预期能够获得4-6倍的性能提升，充分利用现代硬件的计算能力。

**部署和维护考虑**

**跨平台兼容性**

通过CMake构建系统和标准C++代码，确保C++模块在Windows、Linux和macOS平台上的兼容性。Python模块通过conda或pip进行包管理，简化跨平台部署。

**依赖管理策略**

建立清晰的依赖管理策略，C++模块依赖Eigen、Intel TBB等高性能库，Python模块依赖NumPy、SciPy等科学计算库。通过版本锁定确保系统的稳定性和可重现性。

**调试和诊断支持**

实现跨语言的调试和诊断支持，包括性能分析、内存泄漏检测和错误追踪。建立统一的日志系统，便于系统维护和问题排查。

这种三语言混合架构为洋流模拟系统提供了最佳的性能表现和开发效率平衡。通过合理的技术分工和优化的通信机制，系统能够充分发挥每种编程语言的优势，为用户提供高性能、高可靠性的洋流模拟解决方案。