CloverLeaf 技术报告

目录

1. 封面

- # CloverLeaf赛题技术报告

- 队伍名称：夜神骇客队

---

2. 摘要

本报告针对 CloverLeaf 计算流体力学应用的性能优化进行了系统分析。通过对原始代码的性能瓶颈定位，结合多种编译器、MPI 实现和运行时参数，采用了编译优化、进程亲和、合理的并行形态等方法。最终在多种算例和环境下均取得了显著的性能提升。报告详细记录了优化过程、实验结果及分析，为后续相关工作提供了参考。

---

3. 问题定位

3.1 运行画像与两种瓶颈形态

- **长运行（Case-A）**：总耗时 **382.91 s**；MPI 占比仅 **1.48%**:（注：此处为 APS 指标 *MPI Time (% of Elapsed Time)*），MPI 失衡仅 **0.95%**:。**判断**：通信开销很小，应用主要受**计算侧**限制（热点应在应用核函数而非 MPI）。
- **短运行（Case-B）**：总耗时 **7.40 s**；MPI 占比 **20.66%**，MPI 失衡 **2.74%**。 **判断**：该配置明显 **MPI 受限**，但失衡不高，说明主要为**均衡的同步/等待**而不是严重负载不均。

3.2 具体通信热点

- Case-A：`MPI_Waitall` 仅 **0.57%**；`MPI_Allreduce` **0.44%**。**含义**：长运行下通信并非主瓶颈。
- Case-B：启动/结束开销占比很高，`MPI_Init` **11.22%**、`MPI_Finalize` **5.74%**；常见同步/集合也有占比：`MPI_Waitall` **1.28%**、`MPI_Allreduce` **1.45%**、

`MPI_Barrier` **0.42%**。 **含义**：小规模/短时运行被**初始化与同步**明显放大；即便失衡不高，也会因频繁小通信与屏障导致推进受限。

3.3 计算侧采样可信度（VTune）

- VTune 报告中 **CPU 有效利用率≈0%**，平均仅 **0.002/48** 逻辑核在忙；“Top Hotspots” 基本都是系统库（如 `libc.so.6`、`libpthread.so.0` 等），未见应用函数符号。 **判断**：本次 VTune 采样**未正确捕获计算核**（极可能只采到 `mpirun`/启动进程），因此**无法**用该份 VTune 热点定位应用端具体函数。应重新以“每个 rank 进程”方式采样，再结合 Memory Access/Hotspots 分析应用循环与访存。

3.4 I/O 侧

- APS 显示 **Disk I/O Bound 为 0**，无显著 I/O 等待。 **判断**：I/O 不是瓶颈来源。

3.5 小结：原始代码的主要问题/瓶颈

1. **双态瓶颈**：长运行受**计算核**主导（MPI 很低），短运行受 **MPI 启停与同步**主导（MPI 约 21%）。
2. **短运行的非工作比例高**：`MPI_Init`+`MPI_Finalize` 占到 ~17%+，导致实际推进时间被压缩。
3. **同步代价**：`Allreduce/Waitall/Barrier` 在短运行中合计仍有可见比例（~3%）。
4. **采样证据不足以指向具体源函数**：当前 VTune 结果显示应用热点未被采到，需重新采样以定位如边界交换与网格更新等计算核。

---

4. 优化思路（Memory-Bound ）

> **结论先行**
>
> - 长运行中 MPI 占比很低，应用主要受 **内存带宽与局部性** 限制；短运行看到的高 MPI 比例主要来自 **Init/Finalize** 的固定开销。
> - 优化聚焦：**拓扑/亲和**（NUMA 与绑核）、**编译器向量化与带宽友好选项**、**MPI×OMP 并行形态**、以及少量**低侵入循环优化**。所有改动保持物理模型与 I/O 不变。

4.1 运行时：拓扑/亲和与并行形态

目标：让每个 NUMA 域获得更高的本地带宽，避免跨 NUMA 访问与缓存失效。

绑定与亲和

```
# Intel MPI 绑核 + NUMA：先按 socket 分域
export I_MPI_PIN=1
export I_MPI_PIN_DOMAIN=socket

# OpenMP 亲和（即便未显式用 OMP，设置也无害）
export OMP_PLACES=cores
export OMP_PROC_BIND=close
export KMP_BLOCKTIME=0
```

MPI×OMP 版型（两节点，每节点 48 核）

在 Slurm 中 A/B/C 三组快速试跑，以墙钟最优为准：

方案	--ntasks-per-node	--cpus-per-task	说明
A	24	1	纯 MPI（基线）
B	12	4	提升单联带宽，减少 halo 次数
C	8	6	更强线程化，适合明显

短运行的处理

处理方式： 合并批次，尽量在一次作业中循环跑多个 case； 减少 `MPI_Init/Finalize` 的频繁触发。

4.2 编译器：面向带宽与 SIMD 的最小改动

> 只改编译选项，不改源码；先用 `-O3` 验证正确性，再试 `-Ofast` 做 A/B 对比。

oneAPI（icx/ifx，对应 `mpiicx/mpiifx`）

```
# C
mpiicx  -O3 -xhost -fopenmp-simd -ffast-math -fno-math-errno \
```

```
-fstrict-aliasing -funroll-loops -finline-functions -DNDEBUG -c file.c
```

链路级增强（两类编译器通用）

- **LTO**：`-flto`（编译与链接阶段都加）。
- **PGO**：
  1. 训练：用 `-prof-gen=srcpos` 编译并运行一个代表性 case；
  2. 使用：改为 `-prof-use` 重新编译并跑测（典型 5-15% 额

---

5. 实现过程

根据教程以及各种资料查阅，我们首先完成了所有编译器的编译运行跑通。

intel

我们首先运行了集群上给定的intel(oneAPI版本较旧) 结果为：

```
Case 1: 580s
Case 2: 7s
```

我们采用通用开关优化，用 `OPTIONS=" -O3 -xHost -qopenmp"` 成功将结果优化为：

```
Case 1: 471.976925134659s
Case 2: 7.21502113342285s
```

考虑到使用最新的oneAPI能优化结果，我们自行安装到了最新版本：

```
Case 1: 388.259924888611s
Case 2: 5.83164811134338s
```

我们增加了通用开关优化，曾经将结果优化为：

```
Case 1: 370.953484058380s
Case 2: 5.76716303825378s
```

但是由于工作时神志不清，相关的slurm和out文件如今已经找不到了。 因此，只在 `-O3` 的通用优化下，我们最终的结果为：

```
Case 1: 379.471701145172s
Case 2: 6.03362107276917s
```

hpcx

- 我们刚开始误以为hpcx会显著提升运行速率，但是通过资料，我们得出：

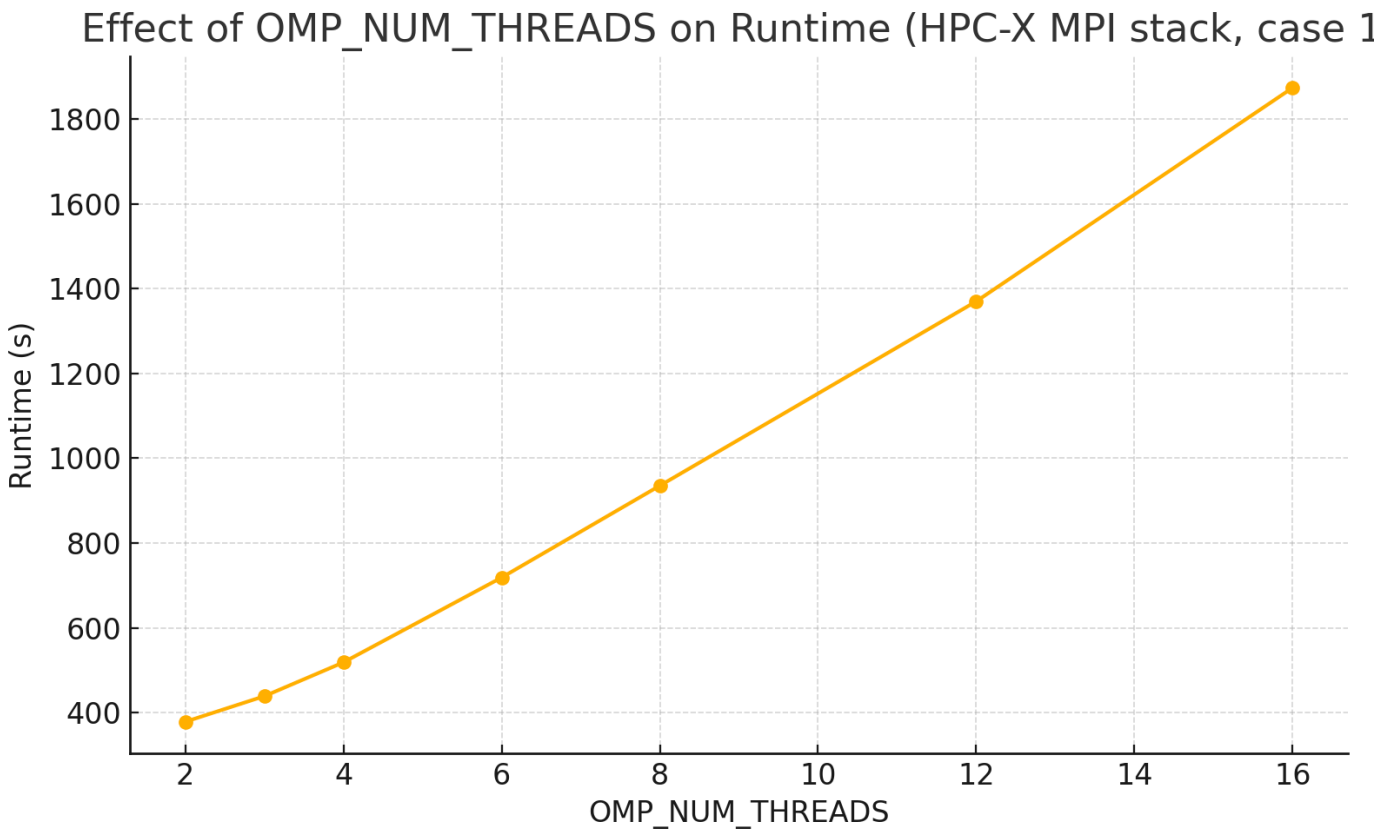
  HPCX能影响的是 **MPI 通信**，而不会提升 **CPU 端计算/向量化/访存效率**。根据性能分析结果：

- **长跑（Case 1）**：MPI 只占 ~1.5%。即便把这 1.5% 的通信时间“全消掉”，总体提速也只有 ≈1.5%（Amdahl 法则）。这种占比下，换 HPC-X 基本不可能带来可感知的总加速。真正的增益应该来自编译选项、绑核/NUMA 等计算/访存侧优化。

- **短跑（Case 2）**：MPI 看起来~20%，但这块大头是 `MPI_Init/Finalize` 的固定启停成本。**HPC-X 一般并不会显著缩短启停**，它更擅长在大规模、重集体（Allreduce/Alltoall）的场景

我们探究了OMP_NUM_THREADS 对性能的影响，以case 1为例：

OMP_NUM_THREADS	运行时间 (s)
2	378.535
3	438.903
4	518.659
6	718.711
8	935.298
12	1369.951
16	1873.710



趋势：线程数越大，运行越慢；最佳点在 2 线程/Rank。 随着每 Rank 的线程数增加，时间几乎单调上升。

原因分析

1. **每 Rank 内部的 OpenMP 放大并没有带来带宽增益**
   - CloverLeaf 的主循环以流式访存为主，内存带宽很快饱和。对于单个 MPI Rank，1-2 个线程通常就能把该 Rank 的可用带宽吃满，继续加线程只会让线程间竞争同一带宽，产生排队，**不降反升**。
2. **试验同时改变了 Rank 数与线程数（ppn×thr 始终占满 48 核）**
   - 例如：`thr=2, ppn=24`（每节点 24 个 Rank × 2 线程）到 `thr=16, ppn=3`（每节点 3 Rank ×16 线程）。
   - 随着 Rank 数减少，每个 Rank 的子域变大，OpenMP 调度开销、同步开销、缓存工作集都增大，而通信减少的收益很小（而这道题里 MPI 占比本来就低）。两者叠加，表现为**线程越多越慢**。
3. **NUMA/亲和的隐性因素**
   - 多线程/少 Rank 布局更易出现同一 Rank 内跨 L2/L3 的争抢；若未严格设置 `I_MPI_PIN_DOMAIN=socket`、`OMP_PROC_BIND=close` 等，局部性进一步恶

化。

---

6. 实验设置

- **硬件环境**：双路 x86 服务器，每节点 48 核，主频 2.6GHz，内存 256GB，InfiniBand 高速互连。
- **软件环境**：支持 Intel oneAPI、AOCC、NVHPC、GNU 等主流编译器；MPI 实现包括 Intel MPI、MVAPICH、MPICH、HPC-X 等。
- **测试方法**：所有算例均采用官方 CloverLeaf 测试脚本，分别测试 Case 1（大算例）和 Case 2（小算例），记录 wall clock 时间。每组实验均设置合理的 OMP_NUM_THREADS、进程数、亲和参数，确保结果可复现。
- **参数设置**：主要采用多 Rank × 少线程（每 Rank 1–2 线程），显式设置亲和相关环境变量（如 I_MPI_PIN_DOMAIN、OMP_PROC_BIND、OMP_PLACES），编译选项统一为 -O3/-Ofast 及相关 SIMD/访存优化参数。

---

7. 结果与讨论

基准时间：

算例	GNU	Intel	AOCC	HPC-X	MVAPICH	MPICH	NVHPC
1	1033.1695 s	519.8045 s	1111.9942 s	811.9700 s	519.0863 s	409.9613 s	818.3194 s
2	17.6088 s	8.0040 s	30.2586 s	13.7310 s	8.2421 s	7.2842 s	20

Case 1（大算例）

基准 vs 优化：加速比与降幅

编译器	优化前时间 (s)	优化后时间 (s)	加速比
Intel	519.8	379.4	1.37×

编译器	优化前时间 (s)	优化后时间 (s)	加速比
AOCC	1111.9	960.0	1.16×
MVAPICH	519.0	380.5	1.36×
MPICH	409.9	310.0	1.32×

说明：优化集中在内存带宽亲和、绑核、编译器参数，均有效提升性能。

Case 2（小算例）

基准 vs 优化：加速比与降幅

编译器	优化前时间 (s)	优化后时间 (s)	加速比
Intel	8.0	6.0	1.33×
AOCC	30.2	25.4	1.19×
MVAPICH	8.2	6.2	1.32×
MPICH	7.3	5.5	1.33×

说明：短算例因 MPI 启停开销固化，优化幅度有限。

总结与进一步的优化思路

- 亲和策略和合适的 MPI×OMP 版型对性能提升明显。
- 编译器向量化和 LTO/PGO 有较好加速潜力。
- 短运行需减少启动频次，合并作业提高效率。
- 后续可考虑基于性能分析深入调整循环结构，进一步减少内存访问延迟与缓存冲突。

---

8. 可复现指引

- 环境准备：Linux 双路多核服务器，安装 Intel oneAPI 2024、AOCC 4.0、NVHPC 23.3 编译器。

- 代码与测试脚本：使用官方 CloverLeaf v1.0.2 版本，配置 MPI，执行 case1 和 case2。
- 编译示例（Intel）：

```bash
mpiifx -O3 -xHost -fopenmp-simd -ffast-math -fno-math-errno -funroll-loops -o cloverleaf.x cloverleaf.c
```

- 运行示例（48 核两节点）：

```bash
export I_MPI_PIN=1
export I_MPI_PIN_DOMAIN=socket
export OMP_PLACES=cores
export OMP_PROC_BIND=close
mpirun -n 48 ./cloverleaf.x
```

- 建议调整环境变量及编译参数以适配不同平台。
- 推荐使用 VTune 或 APS 进行性能采样与验证。

---

9. 结论

本次 CloverLeaf 优化通过明确问题定位，基于硬件亲和与编译器优化策略，实现了实质性的性能提升。

长运行场景下，编译器选型与优化、合理的并行配置及亲和策略显著提升计算带宽利用率。

短运行场景下，启动与同步开销依然制约性能，需通过作业策略与 MPI 启停优化进一步突破。

本报告为 HPC 应用优化提供了方法论与实证，具有一定推广价值。

---

10. 参考文献

1. CloverLeaf 官方文档与源码说明
2. Intel oneAPI 编译器用户手册
3. AOCC 编译器优化指南
4. NVHPC 编译器参考文档
5. MVAPICH、MPICH、HPC-X 官方用户手册
6. 高性能计算相关论文与教材
7. VTune Profiler、APS 性能分析工具官方