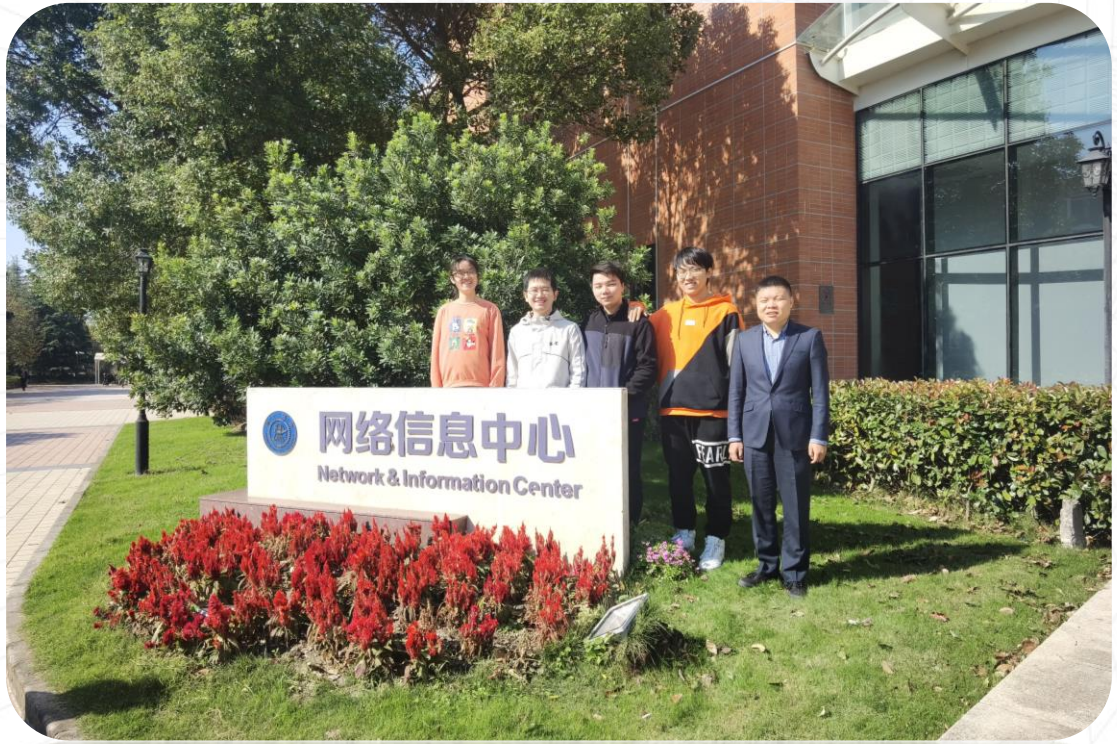
ACM中国一国际并行计算挑战赛

ACM-China International Parallel Computing Challenge

目录 CONTENTS

01 参赛队简介

- 队伍名称：SJTU-CHPC
- 队伍编号：IPCC20216032
- 参赛单位：上海交通大学
- 参赛队员：左思成，王鎏振，张洪健，揭春燕
- 指导老师：文敏华



02 应用程序运行的硬件环境和软件环境

- 硬件设备：

CPU：AMD EPYC 7452 32-Core Processor

双节点，每节点双socket，每socket 32核心

- 软件环境：

OS：CentOS Linux release 7.9.2009

GCC compiler：GCC-4.8.5

Intel compiler：ICC-20.4.3

Intel MPI: 20.4.3

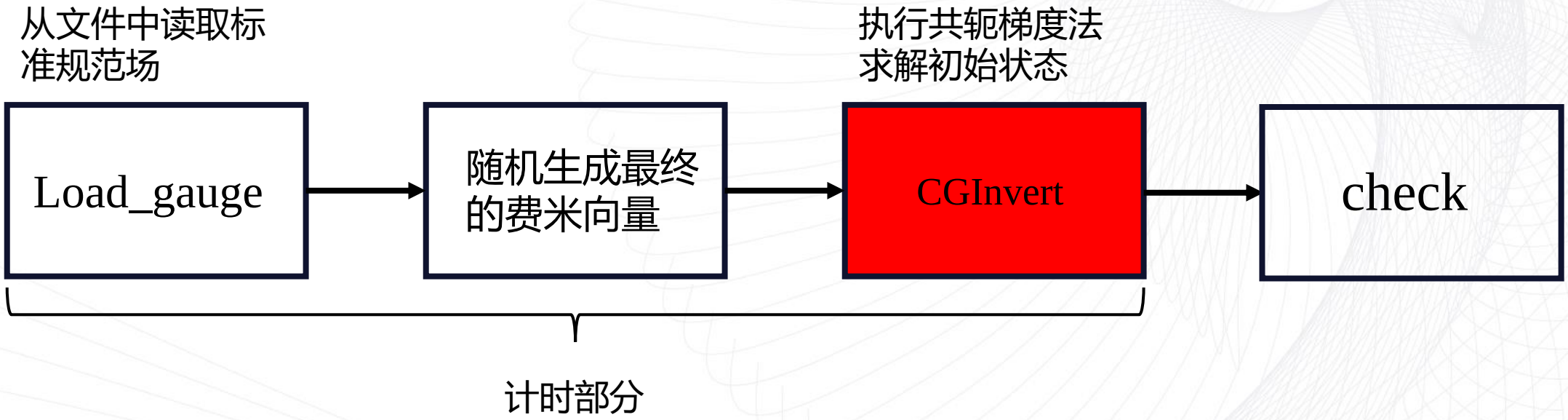
- 优化结果概要

算例1：46.99s -> 0.31s（~**150**倍加速)

算例2：103.60s -> 0.78s (~**133**倍加速)

算例3：550.49s -> 6.94s (~**79**倍加速)

03 应用程序的代码结构（1/2）



代码的热点高度集中在CGInvert函数中，而热点中的热点是Dslash操作，即8个方向的矩阵乘向量

03 应用程序的代码结构（2/2）

```
5  Rsd := 目标精度
6  M := Dslash(U_μ)
   /* 方程两侧同乘以 M† 保证正定
7  M' := M†M
8  b⃗' := M†b⃗
   /* 求解 M'x⃗ = b⃗' ，CG
9  x⃗ := 猜测解
10 for n_cg < MaxCG do
11     计算x⃗' = M'x⃗
12     计算残差 residual
13     if residual < Rsd then
14         Save x⃗
15     else
16         Update x⃗
17 Save x⃗
```

CGInvert流程

$$M = (4+m)\delta_{x,y} - \frac{1}{2}\sum_{\mu=0}^{4}[(1-\gamma_\mu)U_\mu(x)\delta_{x+\mu,y} + (1+\gamma_\mu)U_\mu(x)\delta_{x-\mu,y}]$$

Dslash算符对应矩阵

DslashEE、DslashOO： $(4+m)\delta_{x,y}$

Dslashoffd： $-\frac{1}{2}\sum_{\mu=0}^{4}[(1-\gamma_\mu)U_\mu(x)\delta_{x+\mu,y} + (1+\gamma_\mu)U_\mu(x)\delta_{x-\mu,y}]$

Dslashoffd是整个Dslash运算的热点， 以y+1方向为例，其计算内容如下：

$$\begin{bmatrix} U & 0 & 0 & U \\ 0 & U & -U & 0 \\ 0 & -U & U & 0 \\ U & 0 & 0 & U \end{bmatrix} * \begin{bmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \\ \chi_3 \end{bmatrix}$$

其中U为3*3的标准规范场，χ_i为3*1的费米向量

04 优化方法

1. 替换编译器及编译选项
2. 拆分复数类并消除冗余运算
3. 使用CheckerBoarding预处理减小条件数
4. 使用BiCGSTAB算法替代CG算法
5. 混合精度优化
6. 访存优化
7. 调整运行参数

4.1 替换编译器及编译选项

- 默认: mpicxx –std=c++11
- 替换后：mpiicpc –std=c++11 –Ofast –DNO_MPI_IO

	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	4.751	13.832	93.423
加速比	**9.89**	**7.49**	**5.89**

4.2 拆分复数类并消除冗余运算

- 将complex类分为实部和虚部两个数组存储
- 消除Dslash函数循环中的冗余运算
- 用memset代替for循环初始化
- 用乘倒数替换除法，用位运算代替模2

	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	4.245	12.548	82.094
加速比	**11.07**	**8.26**	**6.71**

4.3 使用CheckerBoarding预处理减小问题条件数（1/2）

通过奇偶预处理，可以将费米子矩阵M拆分为：

$$M = \begin{bmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{bmatrix}$$

通过舒尔补拆分分块矩阵M，得到：

$$M = \begin{bmatrix} 1 & 0 \\ M_{oe}M_{ee}^{-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} M_{ee} & 0 \\ 0 & M_{oo} - M_{oe}M_{ee}^{-1}M_{eo} \end{bmatrix} \cdot \begin{bmatrix} 1 & M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{bmatrix}$$
$$= L\tilde{M}U$$

问题转化为$\tilde{M}\tilde{X} = L^{-1}b$
其中$\tilde{X} = UX$
故求解出$\tilde{X}$后通过左乘U^{-1}可以得到X

而L与U的逆分别为：

$$L^{-1} = \begin{bmatrix} 1 & 0 \\ -M_{oe}M_{ee}^{-1} & 1 \end{bmatrix}$$

$$U^{-1} = \begin{bmatrix} 1 & -M_{ee}^{-1}M_{eo} \\ 0 & 1 \end{bmatrix}$$ （M_{ee}为对角阵）

对$\tilde{M}$，只需求解$M_{oo} - M_{oe}M_{ee}^{-1}M_{eo}$对应的线性方程组→待求解的线性方程组维度减半

4.3 使用CheckerBoarding预处理减小问题条件数（2/2）

优化效果：

迭代步数↓/Case→	Case-1	Case-2	Case-3
预处理前/step	125	132	135
预处理后/step	51	51	53

	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	1.341	3.737	28.017
加速比	**35.04**	**27.73**	**19.65**

4.4 使用BiCGSTAB算法替换CG算法

M本身不保证正定，使用CG算法必须先在方程两侧左乘M的共轭转置，而左乘M共轭转置也即Dslash运算是非常耗时的。

迭代步数↓/Case→	Case-1	Case-2	Case-3
CG/step	51	51	53
BiCGSTAB/step √	19	20	22

BiCGSTAB方法**略去了额外的左乘**M共轭转置的运算，不仅减小了计算开销，还结合了最小残差法，使得**收敛更为平滑**。

	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	0.598	1.624	12.463
加速比	**78.64**	**63.81**	**44.17**

4.5 混合精度优化

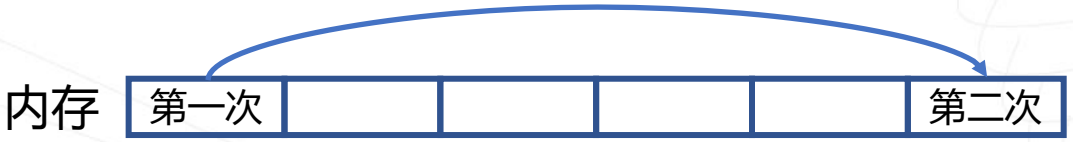
先用单精度迭代求解到指定精度，再改用双精度求得最终解

迭代步数↓/Case→	Case-1	Case-2	Case-3
单精度迭代步数	11	12	12
双精度迭代步数	9	10	10
总迭代步数	20	22	22

	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	0.523	1.409	11.224
加速比	**89.93**	**73.54**	**49.05**

4.6 访存优化

将Dslash函数中的外层循环顺序由x->y->z->t改成t->z->y->x，减少指针每次的偏移量，改善数据访问的局部性。



原始访问顺序

优化访问顺序

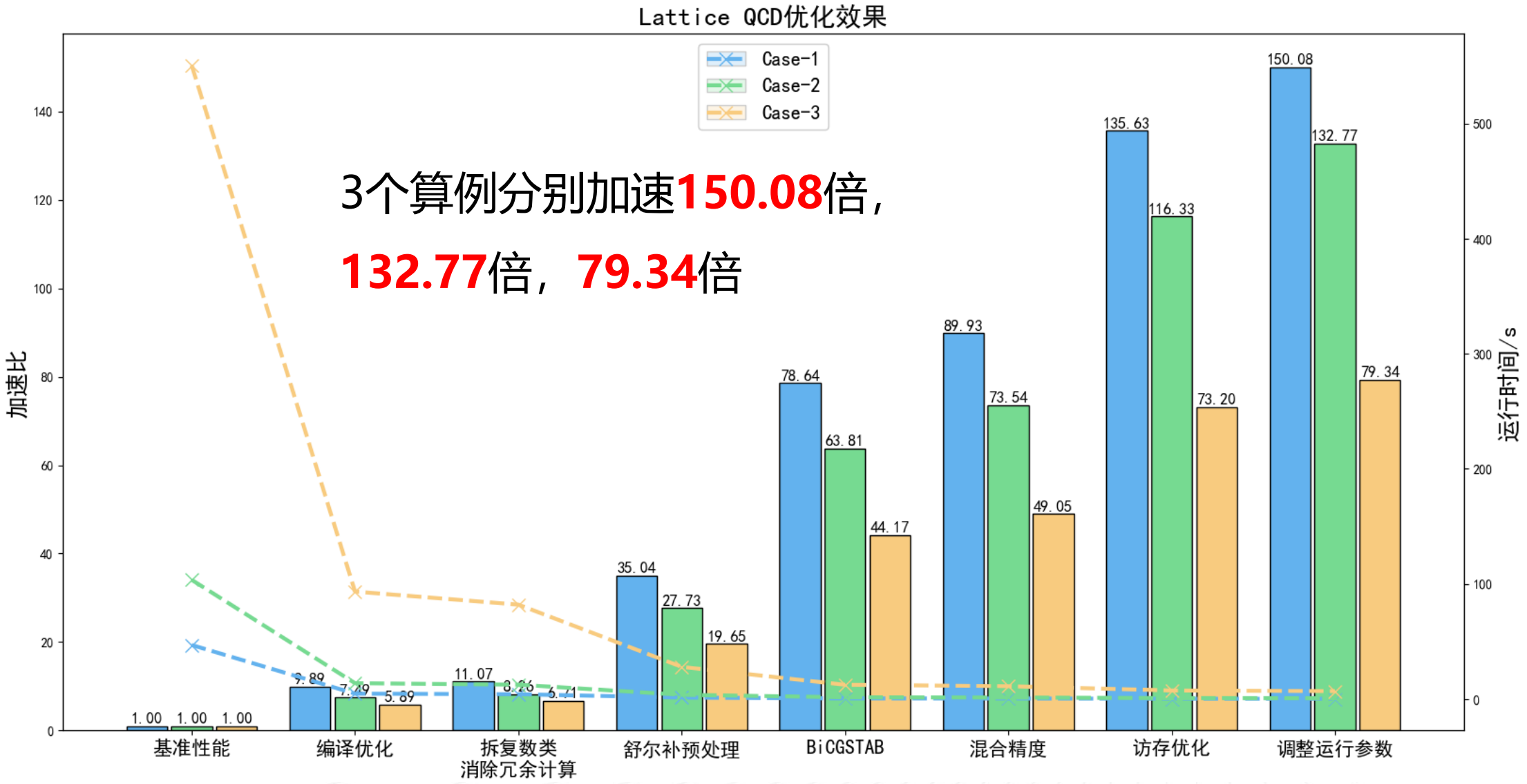
	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	0.346	0.891	7.521
加速比	**135.63**	**116.33**	**73.20**

4.7 调整运行参数

- 将进程网格由 4 x 2 x 2 x 8 改成 2 x 4 x 4 x 4
- 调整进程排布，进行进程绑定(sbatch –m block)
- 在满足精度的要求下调整迭代步数

	Case-1	Case-2	Case-3
基准性能/s	46.995	103.603	550.487
优化后耗时/s	0.313	0.780	6.938
加速比	**150.08**	**132.77**	**79.34**

05 程序运行结果


Lattice QCD优化效果
Case-1
Case-2
Case-3
加速比
运行时间/s
3个算例分别加速150.08倍，
132.77倍，79.34倍
1.00
1.00
1.00
7.89
7.49
5.89
11.07
8.26
6.71
35.04
27.73
19.65
78.64
63.81
44.17
89.93
73.54
49.05
135.63
116.33
73.20
150.08
132.77
79.34
基准性能
编译优化
拆复数类
消除冗余计算
舒尔补预处理
BiCGSTAB
混合精度
访存优化
调整运行参数

06 后续优化

- 合并循环
- 改进的混合精度
- 更优的进程网格分配
- 向量化
- 多重网格算法

感谢指导

THANKS FOR WATCHING