# 神威高性能计算机系统用户试算报告

**——惯性约束-lammps课题**

1. **项目背景及意义**
2. **程序移植与优化**

## 2.1主核程序移植

1. 某些浮点运算异常及个别依赖库无法找到

解决办法: 修改编译参数, 加入并修改对应浮点运算以数学库的链接编译参数

2. 由于所支持的浮点位数的变化(double\_v4 => double\_v8), 原本其所依赖的个别从核的向量化数学函数无法使用.

解决办法: 参考原本大概的实现方法, 单独实现这些功能(simd\_exp, simd\_pos).

3. 由于数据结构在主从核的复杂传递导致的正确性问题

解决办法: 调整数据结构的传输方式, 减小耦合

## 2.2众核并行算法

1、热点分析

|  |  |  |
| --- | --- | --- |
| 热点函数 | 运行时间 | 占比 |
| PairEffcut::compute() | 1137 | 75.8 |
| PauliElecElec() | 315 | 21.0 |
| NPairHalf() | 18 | 1.2 |
| ev\_tally\_xyz() | 10.5 | 0.7 |
| pack\_reverse() | 4.5 | 0.3 |
| 核心计算过程 | 1503s | 100% |

2、众核并行算法

1) 众核并行基本策略

根据将所有计算粒子分配到多个从核上的策略, 将不同次迭代的粒子分配到不同的page(ipage, jpage)中, 而page即每个从核进行计算的粒子数, page大小一般为从核数量或者是其整数倍

2) 混合内存更新

在计算过程中, 当进行两层粒子迭代的时候, 尽管内层粒子(j)的迭代是随机的, 但外层粒子(i) 却是串行的, 这种方式会在并行实现维护数据的连续性时会带来很大的开销.

这里我们使用RCA的方法, 在进行访存的同时进行计算, 并减小上述产生的开销.

## 2.3程序优化方法

1) 通过实现软件缓存策略来提升访存性能

一般来说, 对于邻接列表上的两层粒子迭代的内存访问都是随机的, 因此我们无法预测固定的访存顺序, 但我们可以通过增大访存块(block)的大小来尽可能重用数据, 以此来提高性能, 而这里由于LDM大小的限制, 会选择一个在LDM范围的一个合适的大小构建软件缓存块.

并且, 由于对齐与单次访存大小的原因, 我们会使用DMA操作来将数据取到LDM中来提升访存性能

2) 解决在向量化时不规则访存带来的问题

使用内存索引(j)来进行数据访问, 并使用向量化方法将粒子信息放入缓存块中, 并使用transpose来进行向量化处理.

1. **测试结果**

## 3.1众核加速比测试

1、计算模型：

使用256核组进行测试

2、测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 热点函数 | 单纯主核程序运行时间 | 众核程序运行时间 | 加速比 |
| PairEffcut::compute() | 1137 | 405 | 2.8 |
| PauliElecElec() | **315** | **315** | **1** |
| NPairHalf() | **18** | **18** | **1** |
| ev\_tally\_xyz() | **10.5** | **10.5** | **1** |
|  |  |  |  |
| 核心计算过程 | **1503** | **771** | **1.9** |

3、对比测试

|  |  |  |  |
| --- | --- | --- | --- |
|  | 新神威机器 | 太湖之光 | Intel、GPU |
|  | 771 | 1652 |  |
|  |  |  |  |

## 3.2并行可扩展性测试

1、计算模型：

依次只对64，128, 256个少量核组进行测试

2、测试结果

举例

|  |  |  |
| --- | --- | --- |
| 核组数 | 每时间步平均耗时（s） | 并行效率 |
| 64 | 15.62 | 100% |
| 128 | 13.58 | 87.2% |
| 256 | 12.96 | 83.4% |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## 3.3试算结果

主核结果对比正确, 从核结果正确性依然有偏差

1. **试算体会**

与太湖之光等机器相比的优缺点、对下一代神威高性能计算机系统的期待等。

单位名称

2019年7月31日