第一次作业报告

杨昊光 2014011619

* 问题描述

实现串行程序的优化，以Jacobi迭代为例。

算法描述（伪代码）：

for (t = 0; t < t\_steps; t++){

for (z = 0; z < z\_dim; z++){

for (y = 0; y < y\_dim; y++){

for (x = 0; x < x\_dim; x++){

grid[x][y][z] = /\*avg of neighbors in other\_grid\*/

}

}

}

Exchange grid and other\_grid;

}

* 原始代码见HW1v0.cpp，实现给定格点规模、初始条件和边界条件的100次Jacobi迭代，重复上述过程10次，求得平均耗时作为算法效率的度量。在课程所给远程工作站上运行300\*300\*300规模的运算，平均耗时6.410692s。

Test of Method 0

Step: 0, Time:6.449239

Step: 1, Time:6.407754

Step: 2, Time:6.407785

Step: 3, Time:6.406679

Step: 4, Time:6.405634

Step: 5, Time:6.404629

Step: 6, Time:6.404456

Step: 7, Time:6.404923

Step: 8, Time:6.409395

Step: 9, Time:6.406430

Average Time: 6.410692

* 优化方案及结果分析
  + 程序内部的代码优化

代码1：循环分割为3个循环，分别对同一x, y, z上的两个数加权求和再求总和；

代码2：同代码1，但循环次序改变为z, y, x；

代码3：改变代码0（原始代码）的循环次序为z, y, x；

代码4：同代码3，但是使用指导语句 #pragma ivdep；

代码5：同代码4，但是使用指导语句 #pragma vector always；

* + 针对编译选项的优化

分别针对代码3，修改编译选项为-O1， O2， -O3，-O3 –fast, -O3 -fast –ip，检测运行时间。

* 结论
  + 对代码的优化，结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | 1 | 2 | 3 | 4 | 5 |
| 1 | 25.261971 | 18.230544 | 3.825050 | 3.831109 | 12.521231 |
| 2 | 25.258562 | 18.212516 | 3.832313 | 3.830628 | 12.522496 |
| 3 | 25.260247 | 18.217334 | 3.827613 | 3.830540 | 12.528757 |
| 4 | 25.274053 | 18.208228 | 3.832165 | 3.830383 | 12.554964 |
| 5 | 25.266489 | 18.203414 | 3.833408 | 3.830112 | 12.562571 |
| 6 | 25.311937 | 18.205201 | 3.834200 | 3.830446 | 12.553884 |
| 7 | 25.281014 | 18.206074 | 3.834578 | 3.830300 | 12.555814 |
| 8 | 25.282241 | 18.201918 | 3.828110 | 3.830695 | 12.555389 |
| 9 | 25.293757 | 18.207904 | 3.825236 | 3.830830 | 12.555575 |
| 10 | 25.309940 | 18.206676 | 3.834732 | 3.831117 | 12.555778 |
| Average | 25.280021 | 18.209981 | 3.830297 | 3.830616 | 12.546646 |
| 性能 | 25.4% | 35.2% | 167.4% | 167.4% | 51.1% |

可见在这种算法设计中#pragma ivdep语句对程序性能没有提升，因为程序中本来就没有涉及到数据相互依赖的问题。调整内存读入顺序对程序性能提升明显，无条件的向量化反而可能降低程序性能。

* + 针对编译选项优化，结果如下：

（此处测试和之前的测试并非同一时间段内完成，机器负载情况可能有所变化。）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 选项 | -O1 | -O2 | -O3 | -O3 –fast | -O3 –fast -ip |
| 平均时间 | 10.605289 | 2.698404 | 2.628517 | 2.662209 | 2.695897 |
| 性能 | 100.0% | 393.0% | 403.5% | 398.4% | 393.4% |

由此可见，采用-O3可以令编译器最大程度地优化程序性能，采用更多附加选项再此问题中对提升性能没有显著影响。

* + 不同运算规模的测试

对算法3，针对边长为10，20，30，50，100，150，200，250，300，400，500进行了测试，结果如下，结论为运算时间和运算规模呈线性关系。