南開大學
Nankai University

计算机学院

并行程序设计实验报告

体系结构实验报告

姓名：张惠程

学号：2112241

专业：计算机科学与技术

2023 年 3 月 12 日

目录

1 摘要 **2**

2 任务一：n*n 矩阵与向量内积 **2**

2.1 问题重述与分析 2
2.2 平凡算法与优化算法设计 2
2.2.1 平凡算法 2
2.2.2 优化算法 2
2.3 编程实现 2
2.4 性能测试与数据对比 2
2.5 基于 VTune 对结果进行分析 2
2.6 结果分析 3

3 任务二：n 个数求和 **3**

3.1 问题重述与分析 3
3.2 平凡算法与优化算法设计 4
3.3 编程实现 4
3.4 性能测试与数据对比 4
3.5 基于 VTune 对结果进行分析 5
3.6 结果分析 5

4 总结与反思 **5**

1 摘要

本次实验使用 X86 指令集的 Window11 系统，应用 Intel Core 11th 芯片进行实验，CPU 主频为 2.5GHZ，8 核心，14nm 制作工艺，采用 Rocket Lake S 架构，GPU 频率为 0.35GHZ，加速频率 1.3GHZ。编译器选择 VS2019，编译力度无优化。借助 VTune 分析软件，分析相同的算法时间复杂度下计算机执行不同算法的时间差异，分析平凡算法和优化算法的产生时间差异的原因。最后使用 LaTex 软件进行写作。

2 任务一：n*n 矩阵与向量内积

2.1 问题重述与分析

题目为：给定一个 n × n 矩阵，计算每一列与给定向量的内积，考虑两种算法设计思路：逐列访问元素的平凡算法和 cache 优化算法，进行实验对比。

2.2 平凡算法与优化算法设计

2.2.1 平凡算法

计算两向量内积，需要遍历向量的每个维度进行相乘，计算出结然后相加，最后得到的结果即为向量内积的结果。计算 n*n 矩阵内积需筑列访问矩阵元素，一步外层循环计算（内存循环一次完整执行）计算出一个内积结果。时间复杂度为 $O(n^2)$。

2.2.2 优化算法

计算矩阵内积的算法和向量内积的算法必定需要遍历每一个元素，所以时间复杂度必定为 $O(n^2)$。算法复杂度无法继续进行优化，我们考虑从计算机内存存储结构出发对性能进行优化。具体的 cache 优化算法为逐行访问矩阵元素，一步外层循环计算无法计算出任何一个内积，只是向每一个内积累加一个乘法结果。cache 优化算法具有很好的空间局部性，使程序的性能提高，执行时间缩短。

2.3 编程实现

并行实验一:n*n 矩阵与向量内积平凡算法和优化算法代码

2.4 性能测试与数据对比

2.5 基于 VTune 对结果进行分析

数据规模	循环次数	平凡算法总时间（ms）	优化算法总时间 (ms)
10	294	335	340
20	573	1305	1218
30	712	5480	7841
40	229	2075	1738
50	289	3198	2889
60	179	4638	3941
70	179	5283	4496
80	279	8657	8488
90	60	3621	3081
100	60	5767	5200
200	12	3885	2934
300	12	6178	4996
400	5	6364	7733

表 1: 平凡算法与 cache 优化算法时间对比

	平凡算法	优化算法
指令数	2836345085	2832772262
Elapsed time	0.329	0.256
CPU time	0.299	0.230
CPI	0.467	0.358
L1 Miss	93651642	1200483
L2 Miss	5401167	0
L3 Miss	0	0
L1 Hit	1603430892	2147004603
L2 Hit	86446806	1200432
L3 Hit	0	0
L1 命中率	0.945	0.999
L2 命中率	0.941	1
L3 命中率	none	none

表 2: 平凡算法与 cache 优化算法底层参数对比

2.6 结果分析

二维数组按行主次序进行储存，比如 a[1][1] 和 a[1][2] 的地址是连续的，而 a[1][1] 和 a[2][1] 这样属于同一列的数据地址不连续，需要更多的读取地址数据的时间。由此可见，cache 优化算法的访存模式与行主存储匹配，具有更好的空间局部性。

在底层参数比较中，cache 优化算法的 CPU 时间与 CPI 均优于平凡算法，cache 优化算法的指令数少于平凡算法，而且主要为 L1 指令，速度快，L2 指令数较少，命中率高，因此有更好的性能表现。

3 任务二：n 个数求和

3.1 问题重述与分析

关于 n 个数求和问题，有两类算法设计思路：逐个累加的平凡算法（链式）；超标量优化算法。

3.2 平凡算法与优化算法设计

平凡算法（链式）：遍历每一个元素进行累加。

优化算法（超标量）：递归算法——两两相加、中间结果再两两相加，依次类推，直至只剩下最终结果。

3.3 编程实现

并行实验二：n 个数求和平凡算法和优化算法代码

3.4 性能测试与数据对比

数据规模	循环次数	平凡算法总时间（ms）	优化算法总时间 (ms)
2	300000	1212	8439
4	150000	1527	6767
8	75000	1676	4932
16	37500	1744	3690
32	18750	1924	2624
64	9000	1144	2977
128	9000	1912	4879
256	5000	2177	4061
512	4000	3000	5433
1024	2500	3852	5832
2048	1500	4735	5803
4096	1000	6301	8186
8192	500	6041	8048
16384	250	6364	8249
32768	100	4961	6619
65536	100	10948	13132

表 3: 平凡算法与 unroll 优化算法时间对比

3.5 基于 VTune 对结果进行分析

	平凡算法	优化算法
指令数	189787972	60517836
Elapsed time	0.021	0.028
CPU time	0.016	0.019
CPI	0.384	0.356
L1 Miss	0	0
L2 Miss	0	0
L3 Miss	0	0
L1 Hit	264251832	222210432
L2 Hit	0	0
L3 Hit	0	0
L1 命中率	1	1
L2 命中率	none	none
L3 命中率	none	none

表 4: 平凡算法与 unroll 优化算法底层参数对比

3.6 结果分析

实验以数量为 8192 规模的数据进行 1000 次重复运算，得到实验结果。对比普通的链式算法，两路链式算法能更好地利用 CPU 超标量架构，两条求和的链可令两条流水线充分地并发运行指令。平凡算法的 CPI 为 0.384，优化算法 CPI 为 0.356，明显优于平凡算法。其原因在于二路链式算法通过并行式的计算，两个流水线同时进行，减少运算时间，有效地降低了 CPI。

4 总结与反思

本次实验通过 Vtune 工具对同一时间复杂度的平凡算法和优化算法进行分析，让我了解到可以通过利用计算机的体系结构和并行式计算对程序的性能进行优化。同时，了解到 CPU 时间，CPI 等用于评估性能的参数，具有较深的指导意义。

在计算 n*n 矩阵与向量内积的实验中，cache 算法采用与行主存储匹的访存模式，具有更好的空间局部性，进而实现优化。在计算 n 个数求和时，对比普通的链式算法，两路链式算法能更好地利用 CPU 超标量架构，两条求和的链可令两条流水线充分地并发运行指令，通过并行式计算提高性能。