南開大學
Nankai University

计算机学院

并行程序设计第 **2.2** 次作业

数组求和性能优化

姓名：丁屹

学号：2013280

专业：计算机科学与技术

2022 年 3 月 14 日

目录

1 问题

计算 n 个数的和，考虑两种算法设计思路：

1. 逐个累加的平凡算法（链式）
2. 超标量优化算法（指令级并行），如最简单的两路链式累加；再如递归算法——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

2 程序实现

源码链接：https://github.com/ArcanusNEO/Parallel-Programming/tree/master/1/1
头文件位于 inc/，源文件位于 src/

链式算法

```
for (int i = 0; i < n; ++i) ans += arr[i];
```

2 路链式算法

```
int sum0 = 0;
int sum1 = 0;
for (int i = 0; i < n; i += 2) {
  sum0 += arr[i];
  sum1 += arr[i | 1];  // i + 1
}
ans = sum0 + sum1;
```

4 路链式算法

```
int sum0 = 0;
int sum1 = 0;
int sum2 = 0;
int sum3 = 0;
for (int i = 0; i < n; i += 4) {
  sum0 += arr[i];
  sum1 += arr[i | 1];  // i + 1
  sum2 += arr[i | 2];  // i + 2
  sum3 += arr[i | 3];  // i + 3
}
ans = sum0 + sum1 + sum2 + sum3;
```

8 路链式算法

```
int sum0 = 0;
int sum1 = 0;
int sum2 = 0;
int sum3 = 0;
int sum4 = 0;
```

```
  int sum5 = 0;
  int sum6 = 0;
  int sum7 = 0;
  for (int i = 0; i < n; i += 8) {
    sum0 += arr[i];
    sum1 += arr[i | 1];   // i + 1
    sum2 += arr[i | 2];   // i + 2
    sum3 += arr[i | 3];   // i + 3
    sum4 += arr[i | 4];   // i + 4
    sum5 += arr[i | 5];   // i + 5
    sum6 += arr[i | 6];   // i + 6
    sum7 += arr[i | 7];   // i + 7
  }
  ans = sum0 + sum1 + sum2 + sum3 + sum4 + sum5 + sum6 + sum7;
```

递归算法

```
  if (n <= 1) return;
  int halfn = n >> 1;
  for (int i = 0; i < halfn; ++i) arr[i] += arr[halfn + i];
  recur(arr, halfn);
```

手动消去尾递归算法

```
  BEGIN:
    if (n <= 1) return;
    int halfn = n >> 1;
    for (int i = 0; i < halfn; ++i) arr[i] += arr[halfn + i];
    n = halfn;
    goto BEGIN;
```

- 使用 C++11 的 chrono::high_resolution_clock 高精度计时函数测量运行时间
- ordinary 采用链式求和的平凡算法
- recursion 采用递归求和的算法
- recur-eliminate 采用消除尾递归求和的算法
- unroll-* 采用循环展开的算法
- 使用 cmake 构建

3 实验平台配置

华为鲲鹏 arm 平台部分硬件参数如表 1 所示。arm 服务器系统环境为 4.14.0 内核的 openEuler，本次实验使用基于 clang 的华为 bisheng 编译器构建。

本地 x86 平台部分硬件参数如表 2 所示。本地 x86 系统环境为 5.16.14 内核的 Arch Linux，本次实验使用 GNU GCC 编译，并且使用 perf 进行性能测试。

CPU Maximum Frequency	2600 MHz
CPU Minimum Frequency	200 MHz
L1d 缓存	64 KiB
L1i 缓存	64 KiB
L2 缓存	512 KiB
L3 缓存	49152 KiB
内存大小	191.3 GiB

表 1: 鲲鹏 arm 平台硬件配置信息

CPU 型号	AMD Ryzen 7 4800HS with Radeon Graphics
CPU Maximum Frequency	2900 MHz
CPU Minimum Frequency	1400 MHz
L1d 缓存	256 KiB
L1i 缓存	256 KiB
L2 缓存	4 MiB
L3 缓存	8 MiB
内存大小	16 GiB

表 2: 本地 x86 平台硬件配置信息

4 实验方案设计

4.1 测试脚本

测试脚本位于 bin/，”run” 是 x86 架构下的脚本，”run-pbs” 是鲲鹏服务器平台的脚本

4.2 测试数据

使用 gen-data 生成数据，规模 n 作为第一个参数传入，使用 mt19937 生成随机数。
生成了一组测试文件位于 res/，文件名形如 n.in
使用 conf/in.conf 配置输入数据路径和重复测试次数，其路径作为待测程序的第一个参数传入。

4.3 测试方法

分别测试 ordinary、recursion、recur-eliminate、unroll-2、unroll-4、unroll-8 共 6 个程序。读入定义测试输入文件路径和重复测试遍数的配置文件以便自动完成测试。程序会向标准输出打印测试结果。

重复测试代码

```
for (int _counter = 0; _counter < T; ++_counter) {
  ans = 0;
  memcpy(arr, bak, sizeof(int) * n);
  auto t1 = chrono::high_resolution_clock::now();
  func(ans, arr, n);
  auto t2  = chrono::high_resolution_clock::now();
  auto sec = chrono::duration_cast<chrono::duration<double>>(t2 - t1);
  ret += sec.count() / T;
}
```

n	repeat	ordinary	recursion	recur-eliminate	unroll-2	unroll-4	unroll-8
8	2097152	0.000000023203	0.000000024038	0.000000023632	0.000000023842	0.000000023241	0.000000023776
16	1048576	0.000000025321	0.000000025727	0.000000025150	0.000000023961	0.000000023960	0.000000024965
32	524288	0.000000027710	0.000000026110	0.000000027176	0.000000025583	0.000000025803	0.000000026539
64	262144	0.000000040858	0.000000030310	0.000000031009	0.000000033375	0.000000032347	0.000000033787
128	131072	0.000000066930	0.000000038465	0.000000037498	0.000000048353	0.000000044849	0.000000046245
256	65536	0.000000116553	0.000000057116	0.000000057574	0.000000083876	0.000000072410	0.000000073422
512	32768	0.000000216177	0.000000090888	0.000000088537	0.000000143944	0.000000134368	0.000000121745
1024	16384	0.000000451668	0.000000536241	0.000000519907	0.000000809376	0.000000720560	0.000000694832
2048	8192	0.000000800165	0.000000293033	0.000000293687	0.000000507614	0.000000455317	0.000000430666
4096	4096	0.000001593842	0.000000546658	0.000000543492	0.000000996421	0.000000890181	0.000000842910
8192	2048	0.000003276083	0.000001030757	0.000001034868	0.000001977255	0.000001768750	0.000001660928
16384	1024	0.000006360916	0.000002001890	0.000002005845	0.000003900191	0.000003504228	0.000003277901
32768	512	0.000012731436	0.000003966664	0.000003953602	0.000007784436	0.000006963982	0.000006516787
65536	256	0.000025295840	0.000007917043	0.000007855773	0.000015749094	0.000013886539	0.000013085258
131072	128	0.000050629422	0.000015817750	0.000015699625	0.000031249844	0.000027902617	0.000026089438
262144	64	0.000100877500	0.000031783891	0.000031616375	0.000062528750	0.000055733437	0.000052715687
524288	32	0.000205755031	0.000070265656	0.000071288844	0.000128424812	0.000119083875	0.000111108094
1048576	16	0.000457643875	0.000226808937	0.000232229250	0.000295474188	0.000286918312	0.000262107875
2097152	8	0.000846610500	0.000629506500	0.000588463500	0.000586497750	0.000526005500	0.000545697875
4194304	4	0.001698986750	0.001544501750	0.001504483000	0.001160096750	0.001137527500	0.001018835250
8388608	2	0.003397763500	0.003478081000	0.003308445000	0.002301293500	0.002107703500	0.002073916500

表 3: 本地 AMD x86 平台 O0 优化下的测试结果（时间单位：s）

5 实验结果及分析

n	repeat	ordinary	recursion	recur-eliminate	unroll-2	unroll-4	unroll-8
8	2097152	0.000000023060	0.000000023589	0.000000023952	0.000000022981	0.000000023389	0.000000023779
16	1048576	0.000000025031	0.000000025671	0.000000025462	0.000000023858	0.000000024064	0.000000024913
32	524288	0.000000027844	0.000000026916	0.000000026595	0.000000025560	0.000000026229	0.000000026514
64	262144	0.000000040243	0.000000030448	0.000000031106	0.000000033196	0.000000032635	0.000000033262
128	131072	0.000000068986	0.000000041315	0.000000040217	0.000000048495	0.000000046384	0.000000045532
256	65536	0.000000116935	0.000000059526	0.000000057604	0.000000084178	0.000000074792	0.000000072473
512	32768	0.000000217292	0.000000088840	0.000000087590	0.000000144055	0.000000134890	0.000000121575
1024	16384	0.000001205263	0.000000168469	0.000000448261	0.000000812190	0.000000685154	0.000000583180
2048	8192	0.000000906302	0.000000291144	0.000000301987	0.000000511434	0.000000461590	0.000000426308
4096	4096	0.000001594951	0.000000542890	0.000000541629	0.000001003731	0.000000909334	0.000000837217
8192	2048	0.000003226248	0.000001083998	0.000001041156	0.000001970377	0.000001807268	0.000001671630
16384	1024	0.000007385639	0.000002373620	0.000002142777	0.000003914742	0.000003534012	0.000003357462
32768	512	0.000012693604	0.000003933451	0.000003998584	0.000007832316	0.000007284232	0.000006504643
65536	256	0.000025440746	0.000007928711	0.000007989645	0.000015540238	0.000014240168	0.000012991934
131072	128	0.000050787945	0.000016715883	0.000017459242	0.000031223977	0.000028407477	0.000026033148
262144	64	0.000101305078	0.000031801609	0.000035963234	0.000062746453	0.000056302141	0.000052519016
524288	32	0.000207764281	0.000075900500	0.000073962438	0.000128740094	0.000126357531	0.000110466687
1048576	16	0.000447761812	0.000248268812	0.000290246875	0.000300411750	0.000282492250	0.000252569375
2097152	8	0.000856738000	0.000617823750	0.000766449750	0.000570727625	0.000610684500	0.000525231750
4194304	4	0.001724473500	0.001431223750	0.001823229750	0.001133297500	0.001132052500	0.001026378750
8388608	2	0.003445181000	0.003752353500	0.003759211500	0.002242090000	0.002054912000	0.002136882000

表 4: 本地 AMD x86 平台 O3 优化下的测试结果（时间单位：s）

6 参考文献

[1][3][2][5][6][4]

参考文献

[1] Alexis Zhang. Apple m1 wikipedia. https://zh.wikipedia.org/wiki/Apple_M1, 2020.

[2] Andrei Frumusanu. The 2020 mac mini unleashed: Putting apple silicon m1 to the test. https://www.anandtech.com/print/16252/mac-mini-apple-m1-tested, 2020.

[3] Andrei Frumusanu. Apple announces the apple silicon m1: Ditching x86 - what to expect, based on a14. https://www.anandtech.com/show/16226/apple-silicon-m1-a14-deep-dive, 2020.

[4] Erik Engheim. Why is apple' s m1 chip so fast? https://debugger.medium.com/why-is-apples-m1-chip-so-fast-3262b158cba2, 2020.

[5] Veedrac. Measures microarchitectural details. https://github.com/Veedrac/microarchitecturometer, 2020.

[6] 木头龙. 如何看待苹果 m1 芯片跑分超过 i9？. https://www.zhihu.com/question/429951450, 2020.