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	recur-eliminate	recursion	unroll-2	unroll-4	unroll-8
8	2097152	0.000000270620	0.000000278244	0.000000278079	0.000000263833	0.000000260891	0.000000259854
16	1048576	0.000000295314	0.000000305355	0.000000310217	0.000000277043	0.000000274083	0.000000269215
32	524288	0.000000349405	0.000000358519	0.000000363010	0.000000309193	0.000000294029	0.000000290393
64	262144	0.000000463133	0.000000469887	0.000000472769	0.000000370563	0.000000338805	0.000000331039
128	131072	0.000000668509	0.000000704339	0.000000712042	0.000000496915	0.000000427200	0.000000408503
256	65536	0.000001083509	0.000001136477	0.000001148860	0.000000744988	0.000000622462	0.000000561734
512	32768	0.000001888257	0.000001966816	0.000002027215	0.000001223779	0.000000980544	0.000000883752
1024	16384	0.000003534999	0.000003641069	0.000003764553	0.000002193301	0.000001704996	0.000001482479
2048	8192	0.000006815780	0.000007043776	0.000007056105	0.000004141162	0.000003168578	0.000002716559
4096	4096	0.000013358777	0.000013714426	0.000013933376	0.000008030291	0.000006078574	0.000005191174
8192	2048	0.000026422456	0.000027398315	0.000027336978	0.000015754536	0.000011894580	0.000010126094
16384	1024	0.000052639121	0.000054308604	0.000054191201	0.000031740869	0.000024069111	0.000020588945
32768	512	0.000104782637	0.000108488281	0.000108154551	0.000063199297	0.000047876621	0.000040873711
65536	256	0.000209327500	0.000216811406	0.000216802344	0.000126463828	0.000095503984	0.000081281680
131072	128	0.000420328906	0.000445883125	0.000443657344	0.000262364766	0.000190380313	0.000161993125
262144	64	0.000837718437	0.000920696250	0.000921833906	0.000523046875	0.000381293437	0.000323660625
524288	32	0.001674049062	0.001906608125	0.001905493125	0.001047207187	0.000762858438	0.000648234062
1048576	16	0.003365957500	0.003893178750	0.003889631250	0.002101994375	0.001528597500	0.001292880000
2097152	8	0.006787066250	0.007895242500	0.007866430000	0.004262626250	0.003070501250	0.002625528750
4194304	4	0.014094087500	0.016277475000	0.016350417500	0.009050127500	0.006688947500	0.005699150000
8388608	2	0.029942435000	0.033396810000	0.033163900000	0.018380430000	0.013721520000	0.011907530000

表 3: 鲲鹏 arm 平台 O0 优化等级下的测试结果（时间单位：s）

5 实验结果及分析

5.1 鲲鹏 arm 平台

对于 O0、O1 低优化等级，横向比较表 3、4 可以发现，6 种算法速度排序为 unroll-8 > unroll-4 > unroll-2 > recursion > recur-eliminate > ordinary 。

对于 O2、O3 高优化等级，横向比较表 5、6 可以发现，前 3 种算法速度排序为 recursion recur-eliminate > ordinary 。而 3 种不同程度的循环展开执行速度与数据规模有关。

对于小数据量，3 种循环展开执行速度大致相同，优于递归算法，与平凡算法相近。可能的原因是小数据量的循环展开带来的并行收益不大，而递归算法的额外开销会造成性能损失。

对于中等数据量，3 种算法速度排序为 unroll-8 > unroll-4 > unroll-2，可以发现循环展开程度越大，加速效果越好。

对于大数据量，3 种算法速度排序为 unroll-8 > unroll-2 > unroll-4，可能的原因是根据反汇编代码，unroll-2 和 unroll-4 使用 SIMD 加速，而 unroll-8 使用多发射并行加速。4 路链式算法的循环展开代码过长，包含大量数据搬运，容易造成 cache 失效，抵消了 SIMD 带来的性能增长。

5.2 本地 AMD x86 平台

对于小数据量，各种算法执行速度大致相同。

对于中等数据量，纵向比较表 7、8 可以发现从 n = 1024 是一个反常数据，各种优化算法的执行时间大幅增加，而其他数据均表现一致：递归和递归消去算法表现最优，循环展开算法次之，随着展开次数增加效率增加，平凡算法最劣。

对于大数据量，递归和递归消去算法和平凡算法逐渐表现一致，循环展开算法最佳，且随着展开次数增加效率增加。可能的原因是递归算法的数组访问跨度较大，递归额外开销增加导致效率下降，而循环展开对于大数据量优化效果好。

对于 O0 表 7 和 O3 表 8 的对比，可以看到差异不大，和 arm 架构 O3 优化下运行结果（表 6）对比优势巨大，可能是由于运算简单，x86 架构 CPU 在 O0 下也可以自动进行多发射优化。

n	repeat	ordinary	recur-eliminate	recursion	unroll-2	unroll-4	unroll-8
8	2097152	0.000000265840	0.000000264825	0.000000266694	0.000000254363	0.000000262140	0.000000251627
16	1048576	0.000000287456	0.000000272133	0.000000272251	0.000000255643	0.000000255102	0.000000263581
32	524288	0.000000331122	0.000000293938	0.000000291477	0.000000261275	0.000000258614	0.000000257641
64	262144	0.000000417455	0.000000326943	0.000000320137	0.000000273224	0.000000265933	0.000000265507
128	131072	0.000000590308	0.000000396055	0.000000370941	0.000000299027	0.000000284289	0.000000280737
256	65536	0.000000934326	0.000000503469	0.000000473253	0.000000357254	0.000000321671	0.000000311306
512	32768	0.000001623738	0.000000731355	0.000000649185	0.000000466294	0.000000395081	0.000000372781
1024	16384	0.000003003076	0.000001158761	0.000001012789	0.000000676620	0.000000553022	0.000000495921
2048	8192	0.000005759386	0.000001976619	0.000001707999	0.000001090792	0.000000856852	0.000000751340
4096	4096	0.000011279597	0.000003631853	0.000003097927	0.000001916912	0.000001446880	0.000001249727
8192	2048	0.000022342192	0.000006944038	0.000005867354	0.000003551831	0.000002626938	0.000002243110
16384	1024	0.000044733008	0.000013924844	0.000011684805	0.000007219424	0.000005260586	0.000004594883
32768	512	0.000089905898	0.000027815742	0.000023253906	0.000014117891	0.000010254258	0.000008930156
65536	256	0.000180491016	0.000056294180	0.000046795938	0.000027967695	0.000020298984	0.000017689922
131072	128	0.000359985703	0.000134806797	0.000111738281	0.000056413672	0.000041203594	0.000038057578
262144	64	0.000721862656	0.000336019062	0.000272587969	0.000115204844	0.000082747344	0.000076590938
524288	32	0.001443754062	0.000770574063	0.000615920312	0.000231297812	0.000166070312	0.000170740313
1048576	16	0.002895880625	0.001596490000	0.001341835000	0.000483440000	0.000352754375	0.000336547500
2097152	8	0.005800995000	0.003343596250	0.002805205000	0.001013960000	0.000753855000	0.000732407500
4194304	4	0.011799570000	0.007125665000	0.006181937500	0.002468862500	0.001916330000	0.001845620000
8388608	2	0.023867920000	0.014761550000	0.013078895000	0.008575625000	0.003883035000	0.003769850000

表 4: 鲲鹏 arm 平台 O1 优化等级下的测试结果（时间单位：s）

n	repeat	ordinary	recur-eliminate	recursion	unroll-2	unroll-4	unroll-8
8	2097152	0.000000254309	0.000000261299	0.000000261180	0.000000254903	0.000000253641	0.000000251628
16	1048576	0.000000259057	0.000000265752	0.000000265881	0.000000253966	0.000000255097	0.000000254381
32	524288	0.000000270031	0.000000270545	0.000000270654	0.000000262146	0.000000259697	0.000000257101
64	262144	0.000000292773	0.000000276334	0.000000276268	0.000000260110	0.000000265090	0.000000264718
128	131072	0.000000339501	0.000000288277	0.000000288259	0.000000268537	0.000000278410	0.000000279362
256	65536	0.000000450166	0.000000316273	0.000000315068	0.000000285473	0.000000303419	0.000000309877
512	32768	0.000000652488	0.000000361510	0.000000361765	0.000000319982	0.000000354151	0.000000371698
1024	16384	0.000001049482	0.000000441749	0.000000447415	0.000000387505	0.000000451382	0.000000494955
2048	8192	0.000001837921	0.000000614285	0.000000606001	0.000000523361	0.000000659222	0.000000749255
4096	4096	0.000003413464	0.000000938687	0.000000928477	0.000000798372	0.000001061875	0.000001250059
8192	2048	0.000006567896	0.000001573535	0.000001560811	0.000001363940	0.000001892109	0.000002233203
16384	1024	0.000012660498	0.000003462510	0.000003523105	0.000003575088	0.000005096055	0.000004598066
32768	512	0.000025521543	0.000007426660	0.000007359219	0.000006829004	0.000009904824	0.000008944941
65536	256	0.000052112344	0.000016920352	0.000016834102	0.000014338555	0.000019949609	0.000017656602
131072	128	0.000105935000	0.000054291719	0.000054178516	0.000039727891	0.000046394141	0.000040220234
262144	64	0.000212726562	0.000168271719	0.000169906562	0.000077883750	0.000090778125	0.000076358906
524288	32	0.000425641875	0.000416400625	0.000417097500	0.000166255000	0.000181347813	0.000153590000
1048576	16	0.000888264375	0.000919945625	0.000917731250	0.000412812500	0.000424858750	0.000325724375
2097152	8	0.001837380000	0.001898206250	0.001905310000	0.000795367500	0.000863787500	0.000728592500
4194304	4	0.004162402500	0.003773457500	0.003782610000	0.002016400000	0.002293040000	0.001852327500
8388608	2	0.008626170000	0.007506790000	0.007512580000	0.004182875000	0.004425445000	0.003795245000

表 5: 鲲鹏 arm 平台 O2 优化等级下的测试结果（时间单位：s）

n	repeat	ordinary	recur-eliminate	recursion	unroll-2	unroll-4	unroll-8
8	2097152	0.000000254783	0.000000261253	0.000000261205	0.000000253527	0.000000253509	0.000000251606
16	1048576	0.000000258977	0.000000265849	0.000000265908	0.000000254280	0.000000255101	0.000000254374
32	524288	0.000000270111	0.000000270900	0.000000270659	0.000000256630	0.000000259701	0.000000257083
64	262144	0.000000292845	0.000000276298	0.000000276290	0.000000260494	0.000000265297	0.000000265380
128	131072	0.000000339422	0.000000288206	0.000000288100	0.000000269420	0.000000278426	0.000000279309
256	65536	0.000000450366	0.000000316289	0.000000315840	0.000000287087	0.000000302735	0.000000309457
512	32768	0.000000654426	0.000000362279	0.000000361216	0.000000321051	0.000000354207	0.000000371566
1024	16384	0.000001048832	0.000000437697	0.000000437982	0.000000391018	0.000000451593	0.000000494967
2048	8192	0.000001837910	0.000000614800	0.000000605914	0.000000530441	0.000000659811	0.000000749819
4096	4096	0.000003412610	0.000000950820	0.000000943455	0.000000813982	0.000001058877	0.000001249260
8192	2048	0.000006677456	0.000001586455	0.000001564897	0.000001382603	0.000001878115	0.000002235005
16384	1024	0.000013140117	0.000003513877	0.000003521260	0.000003597324	0.000005114834	0.000004565156
32768	512	0.000026309629	0.000007358457	0.000007434199	0.000006922559	0.000009862598	0.000008917520
65536	256	0.000052115547	0.000016909102	0.000016840508	0.000014416094	0.000020021914	0.000017638906
131072	128	0.000105733516	0.000054351953	0.000054478359	0.000039387031	0.000046551172	0.000038307734
262144	64	0.000210343594	0.000167667500	0.000170268906	0.000078432656	0.000089466719	0.000077230156
524288	32	0.000422880625	0.000417447813	0.000418255625	0.000163295313	0.000180316563	0.000155742500
1048576	16	0.000911424375	0.000921091875	0.000917970000	0.000366436875	0.000374958750	0.000330123750
2097152	8	0.001807893750	0.001889492500	0.001897137500	0.000792485000	0.000846782500	0.000734950000
4194304	4	0.004306067500	0.003791750000	0.003773520000	0.002059920000	0.002243800000	0.001852612500
8388608	2	0.008603645000	0.007513080000	0.007486435000	0.003914820000	0.004451030000	0.003765995000

表 6: 鲲鹏 arm 平台 O3 优化等级下的测试结果（时间单位：s）

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

表 7: 本地 AMD x86 平台 O0 优化下的测试结果（时间单位：s）

n	repeat	ordinary	recur-eliminate	recursion	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 8: 本地 AMD x86 平台 O3 优化下的测试结果（时间单位：s）

	ordinary	recur-eliminate	recursion	unroll-2	unroll-4	unroll-8
O0	2.51	2.48	2.52	2.58	2.58	2.59
O3	2.55	2.43	2.45	2.58	2.56	2.59

表 9: perf 测试本地 AMD x86 平台 O0 和 O3 优化下的 IPC

由表 9 可以发现，平凡算法的 IPC 并没有很低，可以认为是 CPU 执行期间的优化，而递归算法的 IPC 较低，可能是由于控制跳转语句过多造成的影响。