Bonus实验报告：CPU参数对性能的影响分析

- 姓名：王浩雄
- 学号：3230106032

一、实验目的

通过调整CPU微结构参数，使用仿真软件分析其对不同Benchmark（qsort、matrix_mul）性能指标的影响，理解各参数对系统性能的作用机理，找出最佳参数组合。

二、实验环境与参数

2.1 实验环境

- 操作系统：Ubuntu 24.04（WSL）
- 仿真软件：Gem5 Version 24.1.0.3
- Python 版本：3.12.3
- 仿真体系结构：x86

2.2 参数控制

CPU类型	分支预测器	Cache级数	L1大小(KB)	L2大小(KB)	L1关联度	L2关联度	Cache块大小(B)	Benchmark
O3CPU（乱序执行）	TournamentBP/NoBP	1/2	8/16/32/64	64/128/256/512	2/4/8/16	2/4/8/16	64/128	qsort/matrix_mul
TimingSimpleCPU（顺序执行）	-	2	32	256	8	8	64	qsort/matrix_mul

2.3 Benchmark 设计

本实验共设计2个Benchmark，分别为快速排序（qsort）和矩阵乘法（matrix_mul）。其中，快速排序对应分支密集型场景，矩阵乘法对应计算密集型场景，具有代表性。

1. 快速排序（qsort）程序代码

```
void quicksort(vector<int>& arr, int left, int right) {
    if (left >= right) return;
    int pivot = arr[right];
    int i = left - 1;

    for (int j = left; j < right; ++j) {
        if (arr[j] < pivot) {
            ++i;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[right]);
    int pivotIndex = i + 1;

    quicksort(arr, left, pivotIndex - 1);
    quicksort(arr, pivotIndex + 1, right);
}

int main() {
    const int N = 100000;
    vector<int> data(N);

    srand(time(nullptr));
    for (int i = 0; i < N; ++i) {
        data[i] = rand();
    }

    quicksort(data, 0, N - 1);

    cout << "Quicksort done." << endl;
    return 0;
```

```
}
```

2. 矩阵乘法（matrix_mul）程序代码

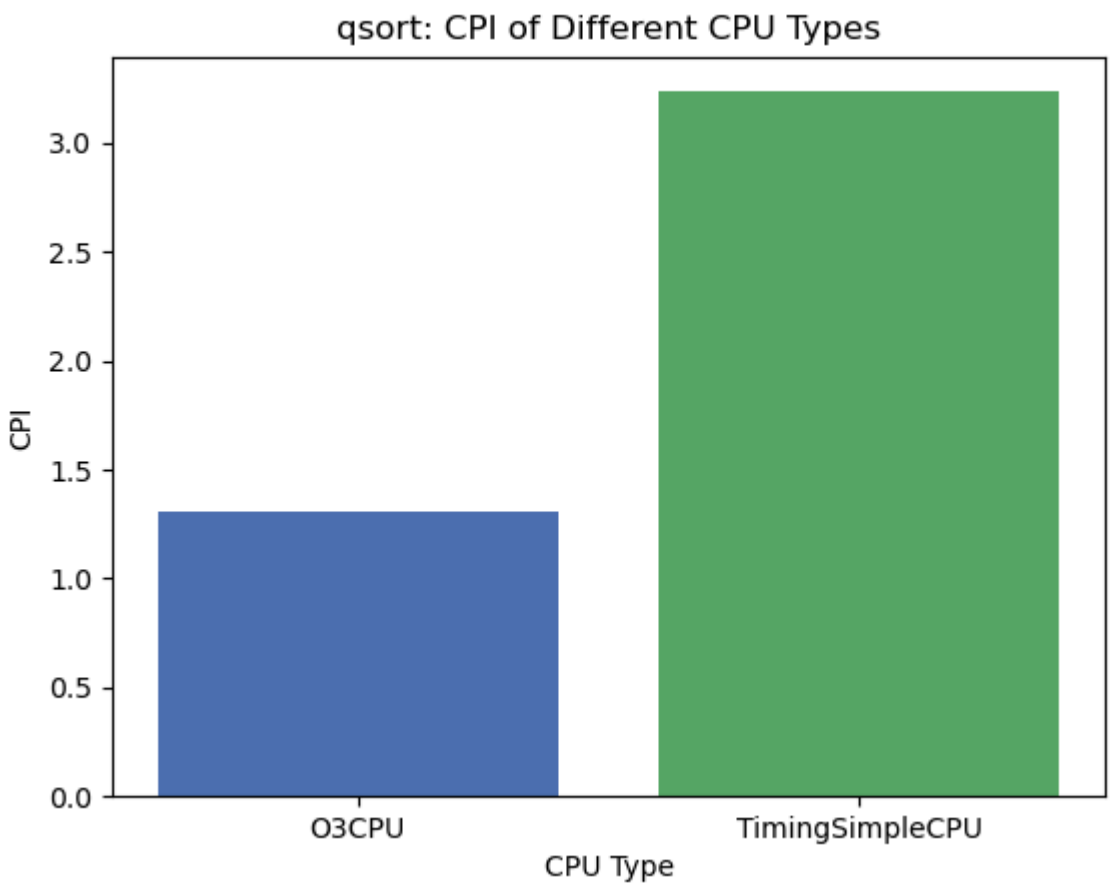
```
int main() {
    const int N = 200; // 矩阵大小
    vector<vector<int>> A(N, vector<int>(N));
    vector<vector<int>> B(N, vector<int>(N));
    vector<vector<int>> C(N, vector<int>(N, 0));

    srand(time(nullptr));

    // 初始化矩阵A和B
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j) {
            A[i][j] = rand() % 100;
            B[i][j] = rand() % 100;
        }

    // 矩阵乘法 C = A * B
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            for (int k = 0; k < N; ++k)
                C[i][j] += A[i][k] * B[k][j];

    cout << "Matrix multiplication done." << endl;
    return 0;
}
```

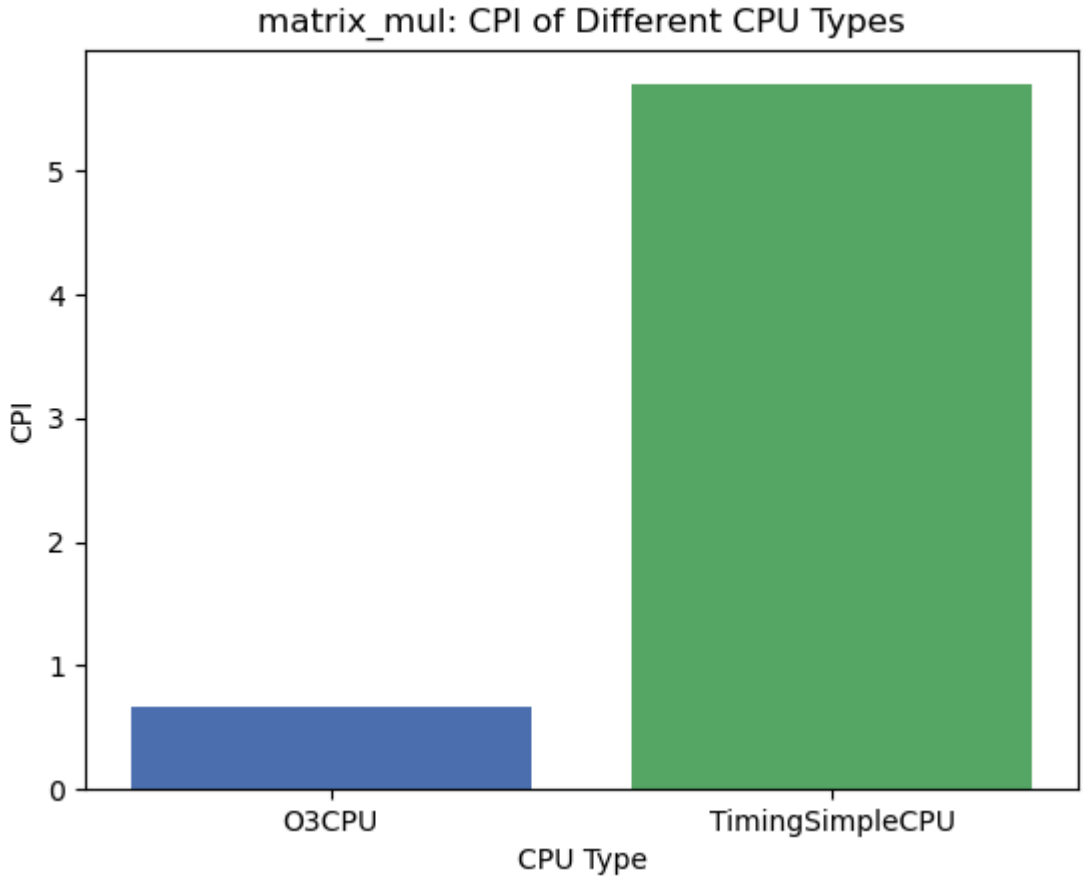
三、实验结果与分析

1. CPU类型对性能的影响

组号	Benchmark	CPU类型	CPI	IPC
1	qsort	O3CPU	1.31	0.77
19	qsort	TimingSimpleCPU	3.24	0.31

组号	Benchmark	CPU类型	CPI	IPC
2	matrix_mul	O3CPU	0.67	1.49
20	matrix_mul	TimingSimpleCPU	5.70	0.18


qsort: CPI of Different CPU Types
CPI
3.0
2.5
2.0
1.5
1.0
0.5
0.0
O3CPU
TimingSimpleCPU
CPU Type


matrix_mul: CPI of Different CPU Types
CPI
5
4
3
2
1
0
O3CPU
TimingSimpleCPU
CPU Type

分析：

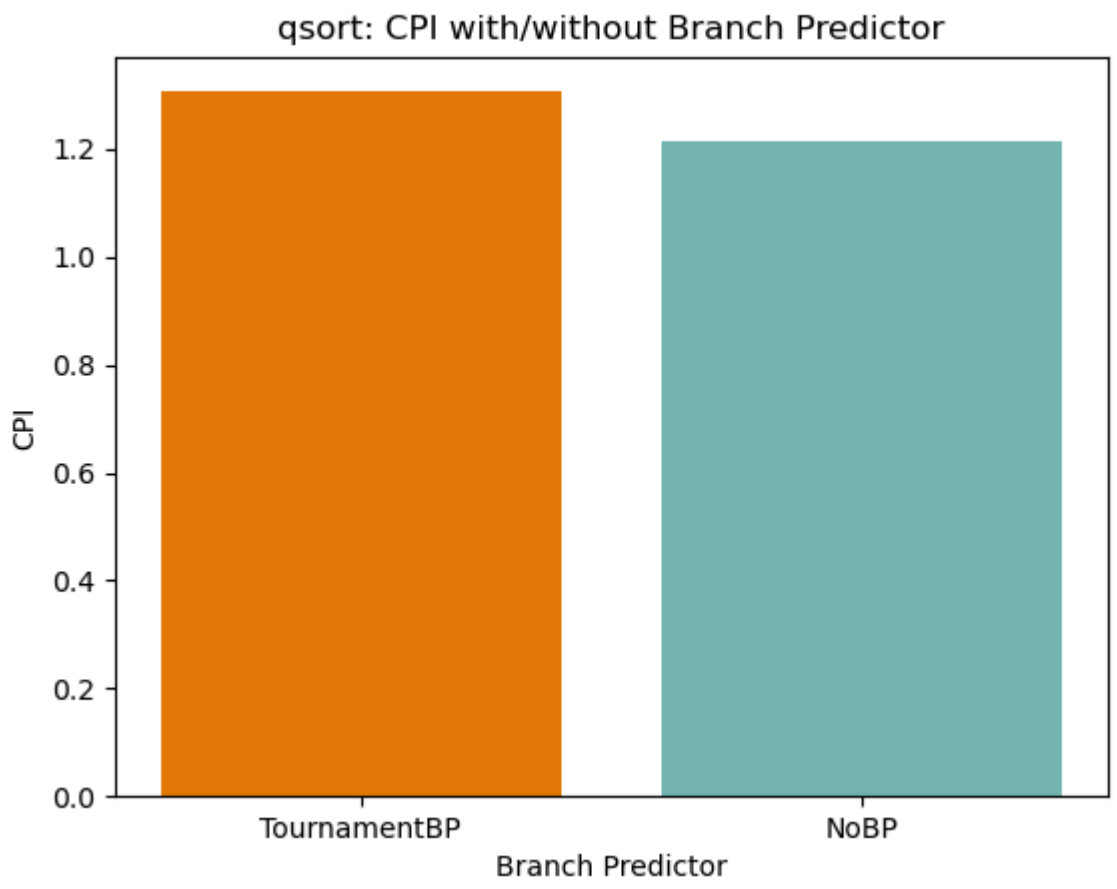
O3CPU（乱序执行）相比TimingSimpleCPU（顺序执行）在两类Benchmark下均表现出更低的CPI和更高的IPC，尤其在matrix_mul（计算密集型）下，CPI下降幅度更大。

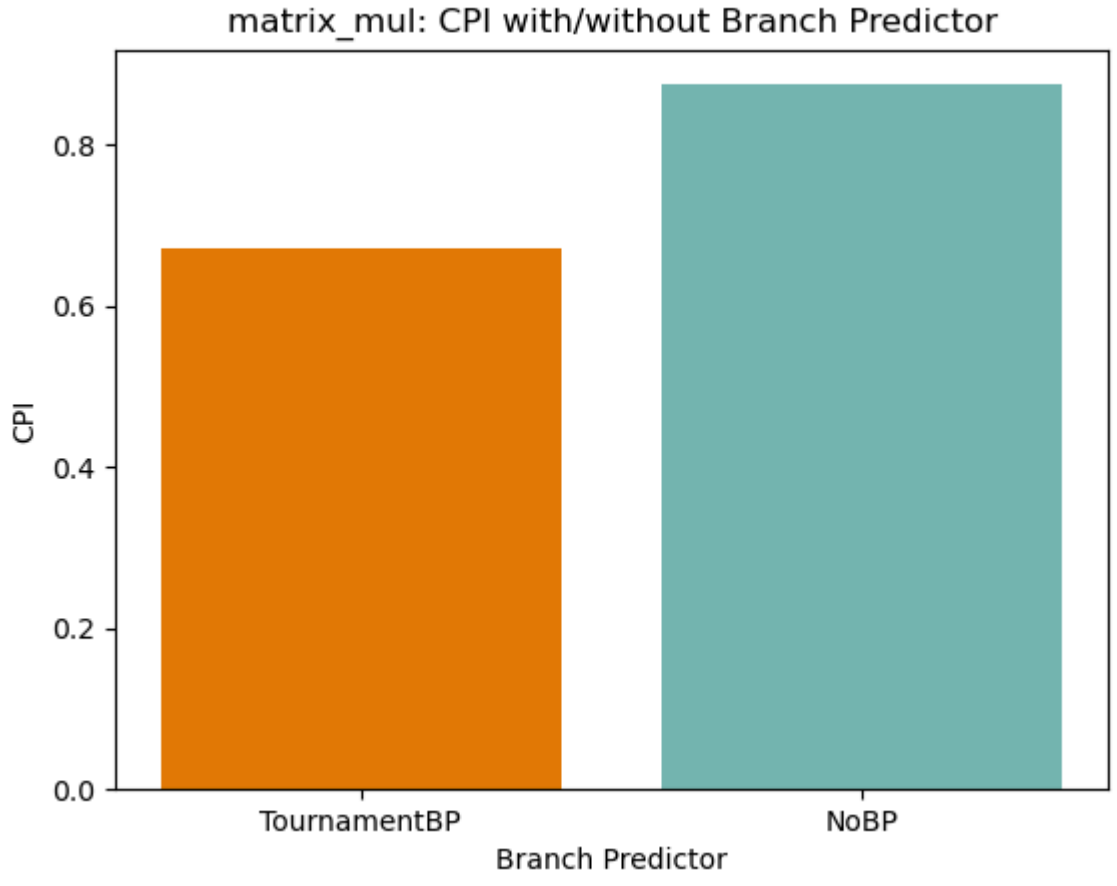
这说明乱序执行架构能够动态调度指令，充分利用指令级并行性，有效隐藏访存延迟和数据相关性带来的阻塞。对于qsort这类分支密集型程序，虽然乱序执行也有提升，但受限于分支预测准确率，提升幅度相对有限。而对于matrix_mul这类数据相关性较弱、可并行度高的程序，乱序执行能极大提升流水线利用率，显著提高性能。

---

2. 分支预测器对性能的影响

组号	Benchmark	分支预测器类型	CPI	分支预测命中率
1	qsort	TournamentBP	1.31	88.54%
21	qsort	NoBP	1.21	-
2	matrix_mul	TournamentBP	0.67	91.36%
22	matrix_mul	NoBP	0.87	-


qsort: CPI with/without Branch Predictor
CPI
0.0
0.2
0.4
0.6
0.8
1.0
1.2
TournamentBP
NoBP
Branch Predictor


matrix_mul: CPI with/without Branch Predictor
CPI
0.0
0.2
0.4
0.6
0.8
TournamentBP
NoBP
Branch Predictor

分析：

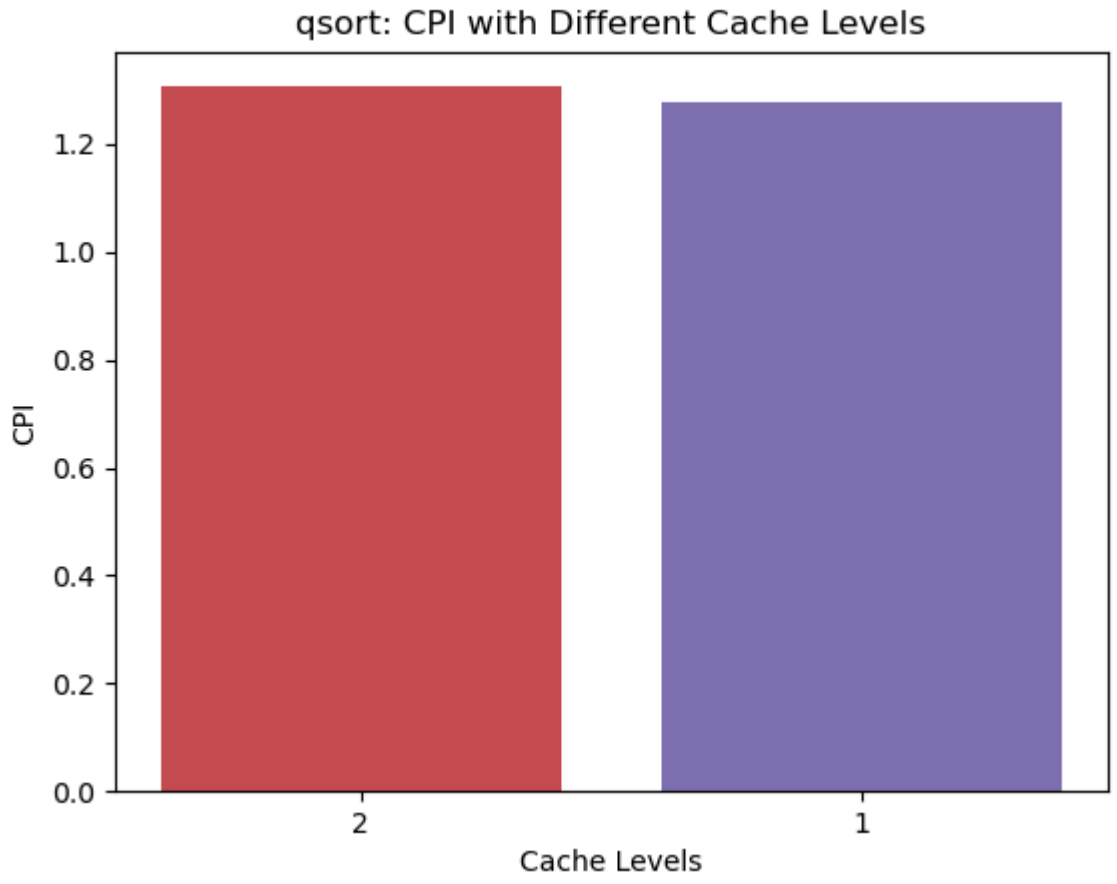
从数据来看，qsort（分支密集型）在使用TournamentBP分支预测器时，分支预测命中率为88.54%，CPI为1.31；而在不使用分支预测器（NoBP）时，CPI为1.21，反而略低于有分支预测器的情况。这一现象与常规预期不符，理论上分支预测器应能减少分支错误带来的流水线清空和性能损失。出现这种结果，可能与qsort的具体数据规模、分支分布、或仿真环境下的实现细节有关。

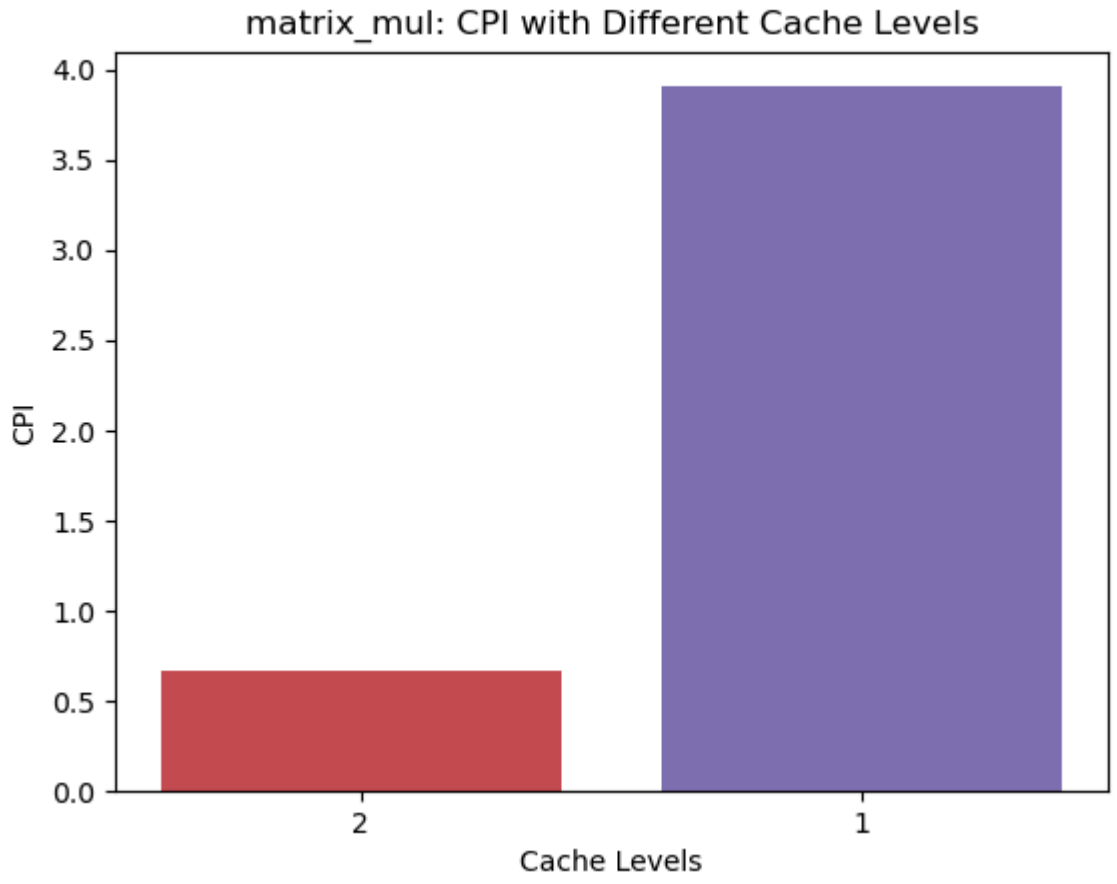
总体来看，分支预测器的作用在分支密集型程序中理论上应更为明显，但实际效果还需结合具体程序结构、数据规模和仿真环境综合分析。

---

3. Cache层级对性能的影响

组号	Benchmark	Cache级数	CPI	L1失效率	L2失效率
1	qsort	2	1.31	3.07%	54.58%
3	qsort	1	1.28	2.43%	-
2	matrix_mul	2	0.67	1.63%	4.34%
4	matrix_mul	1	3.91	27.32%	-


qsort: CPI with Different Cache Levels
CPI
0.0
0.2
0.4
0.6
0.8
1.0
1.2
2
1
Cache Levels


matrix_mul: CPI with Different Cache Levels
CPI
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
2
1
Cache Levels

分析：

对于qsort，Cache层级变化对CPI影响不大，说明其数据局部性较弱，Cache命中率提升有限。

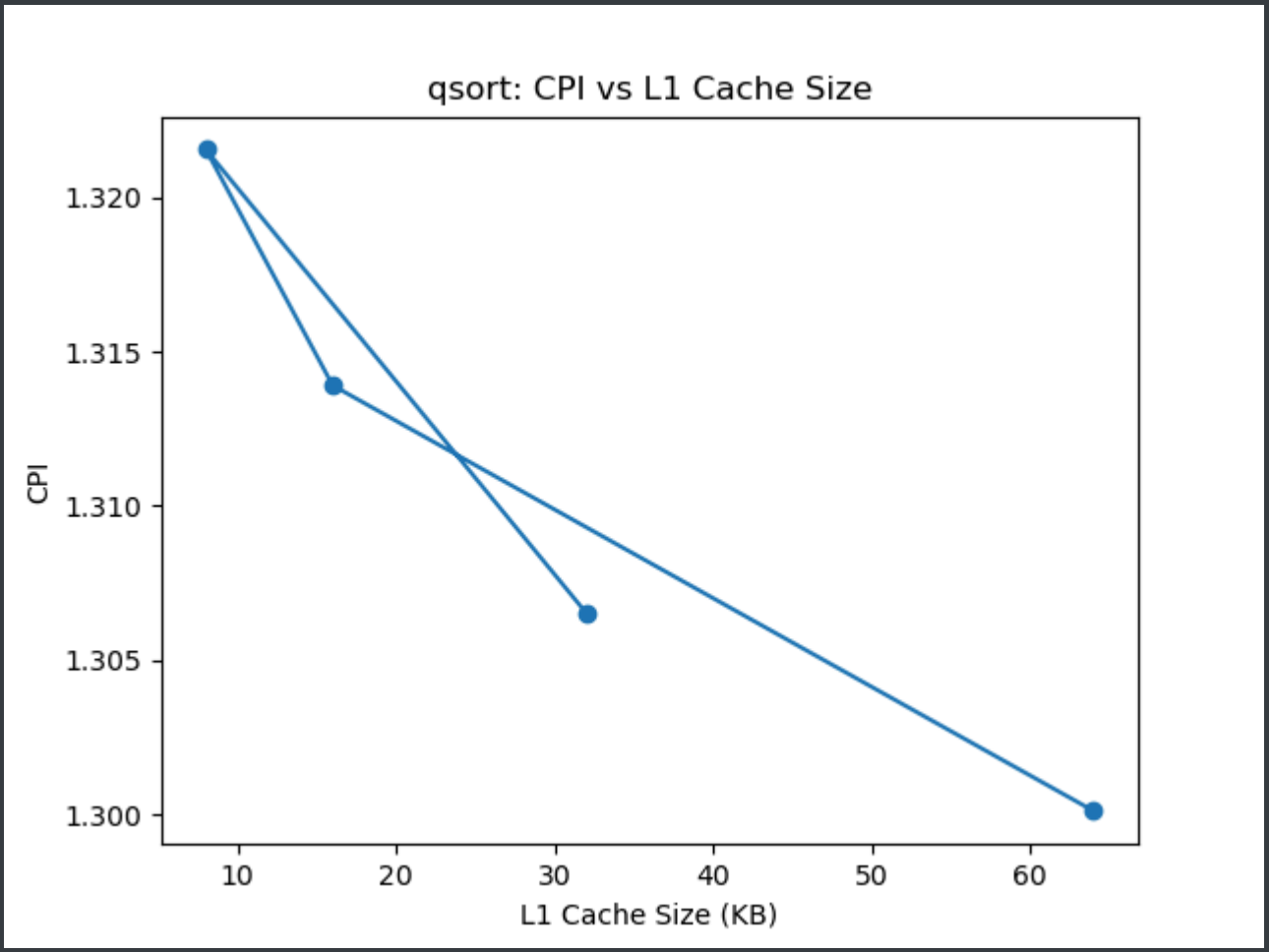
而matrix_mul在二级Cache下CPI大幅下降，L1失效率和L2失效率均显著降低，说明其数据访问呈现强烈的空间和时间局部性，二级Cache能有效缓解主存带宽压力，减少访存延迟。

结论：

Cache层级对数据密集型、局部性强的程序提升极大，对分支密集型、局部性弱的程序作用有限。

4. Cache大小对性能的影响

组号	L1大小(KB)	CPI	L1失效率
11	8	1.32	3.64%
12	16	1.31	3.32%
1	32	1.31	3.07%
13	64	1.30	2.77%



分析:

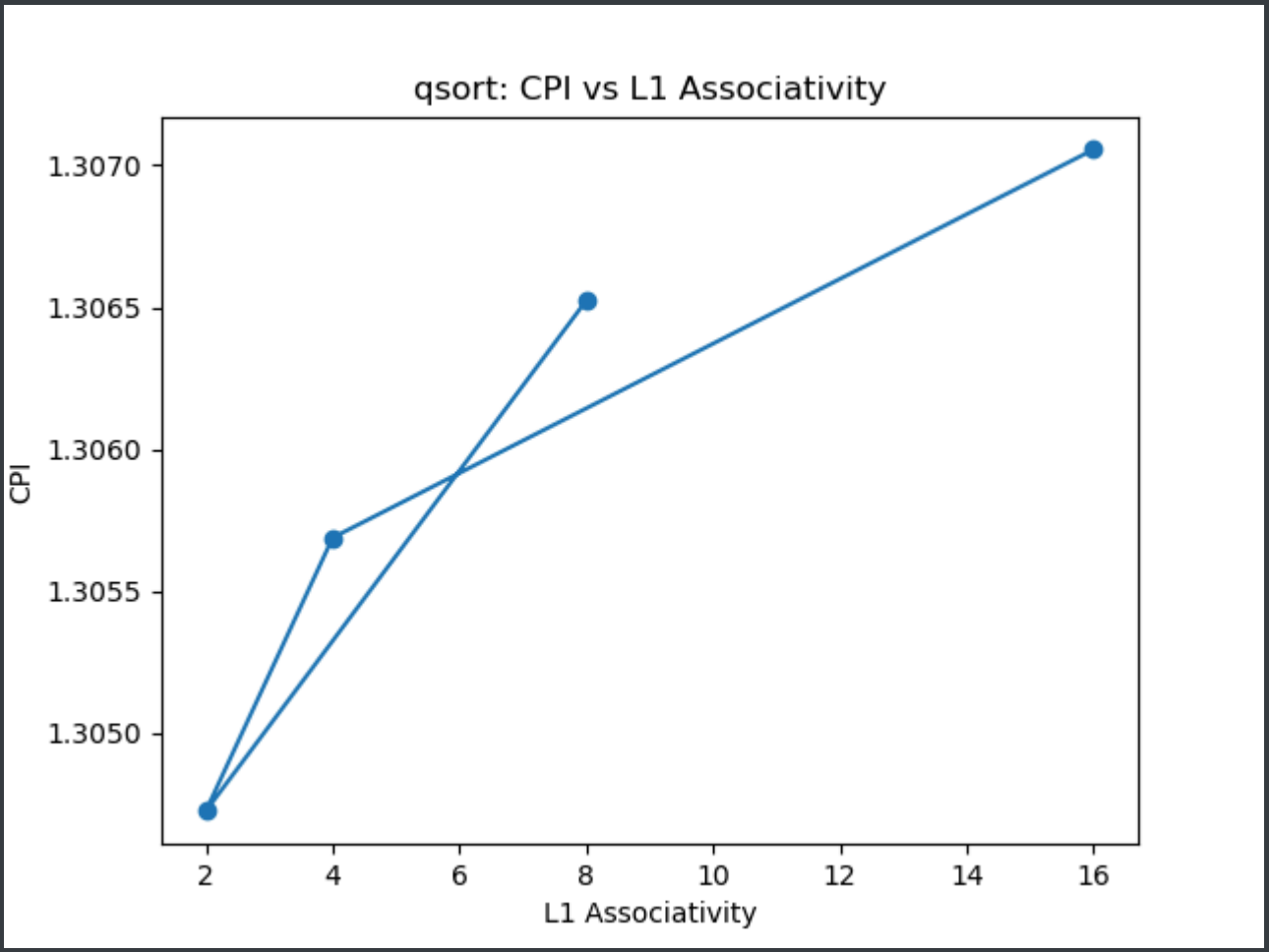
随着L1 Cache容量增大，CPI和L1失效率均逐步下降，但边际效益递减。8KB到64KB，L1失效率从3.64%降至2.77%，CPI仅从1.32降至1.30。

这说明对于本实验的工作集，L1 Cache容量已基本满足大部分数据需求，继续增大容量提升有限。

---

5. Cache关联度对性能的影响

组号	L1关联度	CPI	L1失效率
5	2	1.30	3.01%
6	4	1.31	3.04%
1	8	1.31	3.07%
7	16	1.31	3.08%



分析:

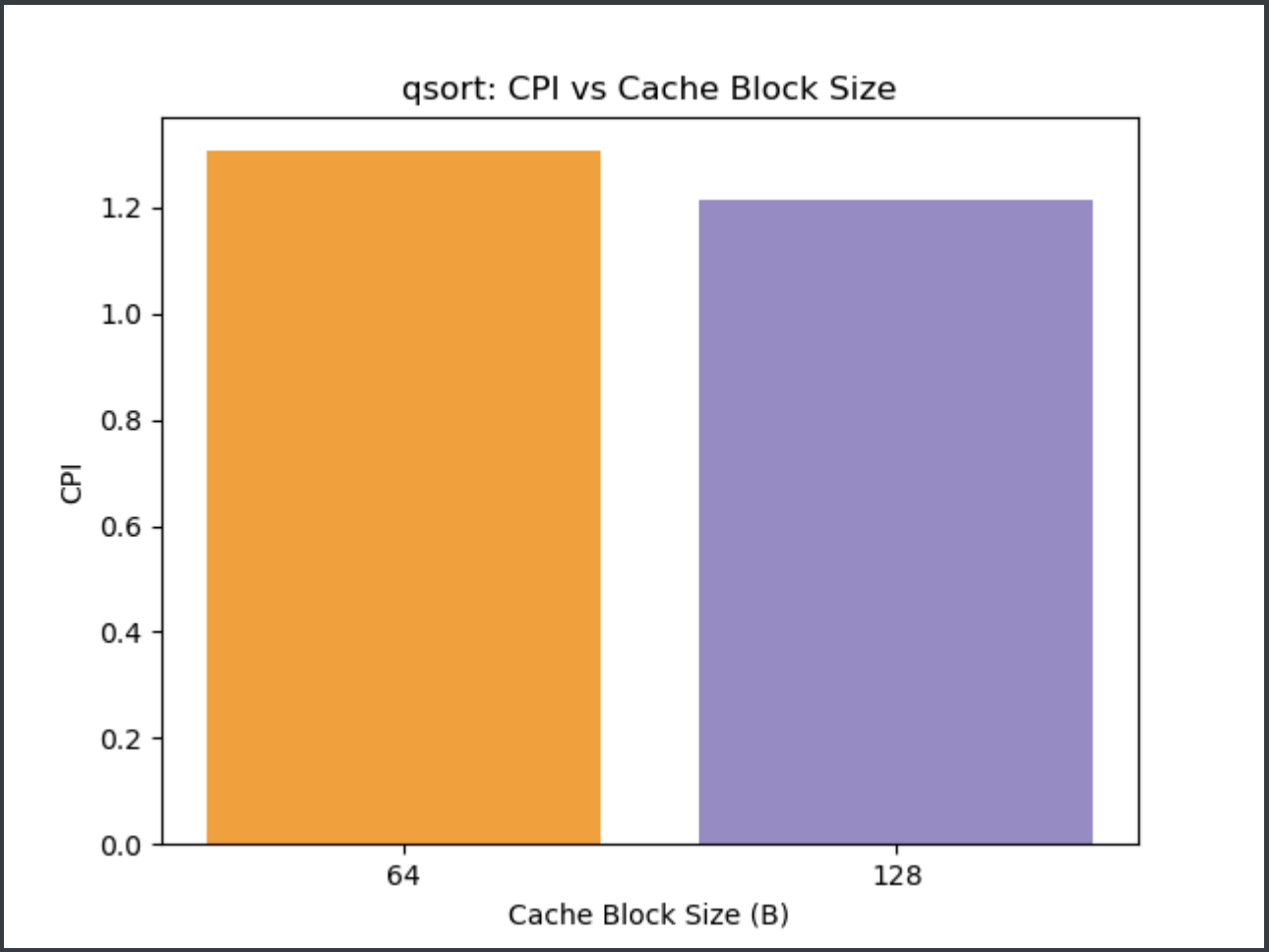
L1关联度从2提升到16，CPI和L1失效率变化极小。这表明本实验程序的访问模式较为均匀，冲突失效不明显，提升关联度带来的收益有限。

但在实际应用中，低关联度可能导致热点数据频繁冲突失效，适当提升关联度有助于提升Cache命中率，但过高会增加硬件复杂度和功耗。

---

6. Cache块大小对性能的影响

组号	Cache块大小(B)	CPI	L1失效率
1	64	1.31	3.07%
17	128	1.21	1.99%



分析:

Cache块大小从64B提升到128B，L1失效率明显下降，CPI也有所降低。这说明实验程序具有较强的空间局部性，较大块大小能一次性加载更多相关数据，减少缺失次数。

但块大小过大可能导致Cache污染和带宽浪费，不应该盲目选用更大的块。

四、实验结论

1. 乱序执行（O3CPU）通过动态调度和乱序发射，有效提升指令级并行性，尤其对计算密集型、可并行度高的程序提升最为明显。
2. 高命中率分支预测器能极大减少分支延迟，提升CPU利用率。对分支较少的程序影响有限。
3. 增加Cache层级和容量能显著降低失效率，提升性能，尤其对数据密集型程序效果明显。
4. 适当提高Cache关联度和块大小有助于提升命中率，但需权衡硬件复杂度和能耗。
5. 不同Benchmark对参数的敏感性不同，需结合应用场景优化相关参数。

AI使用声明：本Bonus的完成过程中，部分脚本（绘图、Markdown表格制作）由AI辅助完成。AI生成内容经过人工审核和修改，以确保准确性。