初始代码运行结果：

```
(base) ubuntu@k8s-master:~/workspace/pythonProject2/mianshi/workshop$ sudo ./workshop
Matrix multiply iteration 1: cost 3.755 seconds
Matrix multiply iteration 2: cost 2.852 seconds
Matrix multiply iteration 3: cost 2.406 seconds
Matrix multiply iteration 4: cost 2.345 seconds
Matrix multiply iteration 5: cost 2.188 seconds
Matrix multiply iteration 6: cost 2.174 seconds
Matrix multiply iteration 7: cost 2.245 seconds
Killed
(base) ubuntu@k8s-master:~/workspace/pythonProject2/mianshi/workshop$
```

平均耗时：2.566 s

优化后代码运行结果：

```
(llm) ubuntu@k8s-master:~/workspace/pythonProject2/test/workshop$ sudo ./workshop
Matrix multiply iteration 1:  0.400 s  (42955.23 MFLOPS)
Matrix multiply iteration 2:  0.380 s  (45198.46 MFLOPS)
Matrix multiply iteration 3:  0.402 s  (42692.11 MFLOPS)
Matrix multiply iteration 4:  0.383 s  (44848.36 MFLOPS)
Matrix multiply iteration 5:  0.399 s  (43041.41 MFLOPS)
Matrix multiply iteration 6:  0.384 s  (44726.07 MFLOPS)
Matrix multiply iteration 7:  0.388 s  (44232.30 MFLOPS)
Matrix multiply iteration 8:  0.392 s  (43798.10 MFLOPS)
Matrix multiply iteration 9:  0.385 s  (44594.80 MFLOPS)
Matrix multiply iteration 10:  0.377 s  (45528.43 MFLOPS)
Matrix multiply iteration 11:  0.381 s  (45120.01 MFLOPS)
Matrix multiply iteration 12:  0.390 s  (43996.30 MFLOPS)
Matrix multiply iteration 13:  0.396 s  (43392.61 MFLOPS)
Matrix multiply iteration 14:  0.378 s  (45467.90 MFLOPS)
Matrix multiply iteration 15:  0.375 s  (45759.10 MFLOPS)
Matrix multiply iteration 16:  0.391 s  (43960.58 MFLOPS)
Matrix multiply iteration 17:  0.400 s  (42932.05 MFLOPS)
Matrix multiply iteration 18:  0.389 s  (44203.79 MFLOPS)
Matrix multiply iteration 19:  0.394 s  (43635.44 MFLOPS)
Matrix multiply iteration 20:  0.401 s  (42809.29 MFLOPS)
Matrix multiply iteration 21:  0.384 s  (44736.01 MFLOPS)
Matrix multiply iteration 22:  0.401 s  (42839.59 MFLOPS)
Matrix multiply iteration 23:  0.384 s  (44684.24 MFLOPS)
Matrix multiply iteration 24:  0.384 s  (44684.91 MFLOPS)
Matrix multiply iteration 25:  0.388 s  (44285.91 MFLOPS)
Matrix multiply iteration 26:  0.382 s  (45031.20 MFLOPS)
Matrix multiply iteration 27:  0.391 s  (43959.85 MFLOPS)
Matrix multiply iteration 28:  0.395 s  (43443.64 MFLOPS)
Matrix multiply iteration 29:  0.404 s  (42476.54 MFLOPS)
Matrix multiply iteration 30:  0.392 s  (43880.49 MFLOPS)
Matrix multiply iteration 31:  0.380 s  (45228.67 MFLOPS)
Matrix multiply iteration 32:  0.386 s  (44561.52 MFLOPS)
Matrix multiply iteration 33:  0.389 s  (44168.14 MFLOPS)
Matrix multiply iteration 34:  0.382 s  (44956.65 MFLOPS)
Matrix multiply iteration 35:  0.388 s  (44237.82 MFLOPS)
Matrix multiply iteration 36:  0.390 s  (44097.60 MFLOPS)
Matrix multiply iteration 37:  0.384 s  (44724.85 MFLOPS)
Matrix multiply iteration 38:  0.393 s  (43660.27 MFLOPS)
Matrix multiply iteration 39:  0.393 s  (43668.42 MFLOPS)
Matrix multiply iteration 40:  0.404 s  (42575.61 MFLOPS)
Matrix multiply iteration 41:  0.401 s  (42852.58 MFLOPS)
Matrix multiply iteration 42:  0.409 s  (41974.07 MFLOPS)
Matrix multiply iteration 43:  0.411 s  (41796.19 MFLOPS)
Matrix multiply iteration 44:  0.408 s  (42068.68 MFLOPS)
Matrix multiply iteration 45:  0.396 s  (43379.27 MFLOPS)
Matrix multiply iteration 46:  0.397 s  (43324.93 MFLOPS)
Matrix multiply iteration 47:  0.386 s  (44472.64 MFLOPS)
Matrix multiply iteration 48:  0.387 s  (44352.98 MFLOPS)
Matrix multiply iteration 49:  0.389 s  (44174.54 MFLOPS)
```

```
Matrix multiply iteration 50:  0.410 s  (41931.90 MFLOPS)
Matrix multiply iteration 51:  0.396 s  (43424.39 MFLOPS)
Matrix multiply iteration 52:  0.409 s  (42024.62 MFLOPS)
Matrix multiply iteration 53:  0.421 s  (40817.70 MFLOPS)
Matrix multiply iteration 54:  0.413 s  (41645.24 MFLOPS)
Matrix multiply iteration 55:  0.392 s  (43878.22 MFLOPS)
Matrix multiply iteration 56:  0.415 s  (41408.44 MFLOPS)
Matrix multiply iteration 57:  0.402 s  (42751.47 MFLOPS)
Matrix multiply iteration 58:  0.386 s  (44504.21 MFLOPS)
Matrix multiply iteration 59:  0.393 s  (43665.30 MFLOPS)
Matrix multiply iteration 60:  0.390 s  (44016.00 MFLOPS)
Matrix multiply iteration 61:  0.404 s  (42479.33 MFLOPS)
Matrix multiply iteration 62:  0.412 s  (41745.76 MFLOPS)
Matrix multiply iteration 63:  0.401 s  (42805.48 MFLOPS)
Matrix multiply iteration 64:  0.407 s  (42190.85 MFLOPS)
Matrix multiply iteration 65:  0.404 s  (42532.68 MFLOPS)
Matrix multiply iteration 66:  0.404 s  (42556.09 MFLOPS)
Matrix multiply iteration 67:  0.404 s  (42570.25 MFLOPS)
Matrix multiply iteration 68:  0.403 s  (42672.05 MFLOPS)
Matrix multiply iteration 69:  0.390 s  (44038.25 MFLOPS)
Matrix multiply iteration 70:  0.395 s  (43506.34 MFLOPS)
Matrix multiply iteration 71:  0.400 s  (42910.49 MFLOPS)
Matrix multiply iteration 72:  0.397 s  (43263.41 MFLOPS)
Matrix multiply iteration 73:  0.391 s  (43896.56 MFLOPS)
Matrix multiply iteration 74:  0.377 s  (45597.13 MFLOPS)
Matrix multiply iteration 75:  0.415 s  (41358.14 MFLOPS)
Matrix multiply iteration 76:  0.419 s  (40957.00 MFLOPS)
Matrix multiply iteration 77:  0.411 s  (41850.07 MFLOPS)
Matrix multiply iteration 78:  0.407 s  (42232.96 MFLOPS)
Matrix multiply iteration 79:  0.402 s  (42732.00 MFLOPS)
Matrix multiply iteration 80:  0.406 s  (42356.80 MFLOPS)
Matrix multiply iteration 81:  0.403 s  (42617.19 MFLOPS)
Matrix multiply iteration 82:  0.397 s  (43239.23 MFLOPS)
Matrix multiply iteration 83:  0.398 s  (43118.81 MFLOPS)
Matrix multiply iteration 84:  0.400 s  (42917.05 MFLOPS)
Matrix multiply iteration 85:  0.401 s  (42851.77 MFLOPS)
Matrix multiply iteration 86:  0.396 s  (43385.27 MFLOPS)
Matrix multiply iteration 87:  0.396 s  (43381.50 MFLOPS)
Matrix multiply iteration 88:  0.392 s  (43873.63 MFLOPS)
Matrix multiply iteration 89:  0.407 s  (42249.68 MFLOPS)
Matrix multiply iteration 90:  0.401 s  (42808.74 MFLOPS)
Matrix multiply iteration 91:  0.405 s  (42408.08 MFLOPS)
Matrix multiply iteration 92:  0.407 s  (42249.78 MFLOPS)
Matrix multiply iteration 93:  0.413 s  (41581.22 MFLOPS)
Matrix multiply iteration 94:  0.403 s  (42606.40 MFLOPS)
Matrix multiply iteration 95:  0.418 s  (41091.71 MFLOPS)
Matrix multiply iteration 96:  0.397 s  (43285.56 MFLOPS)
Matrix multiply iteration 97:  0.381 s  (45119.04 MFLOPS)
Matrix multiply iteration 98:  0.390 s  (44090.88 MFLOPS)
Matrix multiply iteration 99:  0.396 s  (43356.65 MFLOPS)
Matrix multiply iteration 100:  0.381 s  (45036.47 MFLOPS)
```

平均耗时：0.39621 s

加速比：2.566 / 0.396 ≈ 6.47

优化方案:

1、 内存分配与对齐

- **问题**：循环内反复 malloc/free，且手动对齐复杂。
- **方案**：一次性用 posix_memalign 分配 256 字节对齐内存，循环内复用。
- **收益**：避免系统调用开销、减少内存碎片、代码更简洁。
- **涉及文件**：main.c

2、 高精度稳定计时

- **问题**：gettimeofday 受系统时间调整影响。
- **方案**：封装 now_sec()，使用单调时钟 clock_gettime(CLOCK_MONOTONIC)。
- **收益**：计时结果更稳定、精度达到纳秒级。
- **涉及文件**：main.c

3、 访存局部性优化

- **问题**：原始三重循环对 B 的列访问导致频繁 cache miss。
- **方案**：预先转置 B → BT，内核按行顺序访问 BT；接口配合调整。
- **收益**：提高缓存命中率，显著提升性能。
- **涉及文件**：mul.c（新增 transpose_mat、修改 mul_kernel）、mul.h（函数声明调整）、main.c（调用转置并传 BT）

4、 线程模型精简

- **问题**：线程参数冗余、缺少 pthread_create/join 返回值检查。
- **方案**：删掉未用字段，仅保留必要矩阵指针与索引；对线程创建/回收结果进行错误检查。
- **收益**：消除内存泄漏，提升鲁棒性与可读性。
- **涉及文件**：thrmodel.c（核心修改）、mul.h（参数列表同步精简）

5、 代码健壮性

- **问题**：cgroup 配置写死且失败即退出；写文件用 sizeof(str) 可能写出多余字节。
- **方案**：改为“尝试-失败-警告”的容错策略；使用 strlen() 写入准确字节数并封装 safe_write_str()。
- **收益**：提升可移植性，避免在非 root/非 cgroup 环境直接崩溃。
- **涉及文件**: util.c

6、 编译器优化提示

- **问题**：未向编译器提供别名信息与常量性质提示。
- **方案**：为指针参数添加 restrict；将不可变量改为 static const。
- **收益**：编译器可更积极地进行矢量化与指令调度。
- **涉及文件**：mul.h、mul.c（参数加 restrict）、main.c（常量改为 static const）