张景超 2023200401

任务3: 缓冲区对齐的cat

1. **为什么将缓冲区对齐到系统的内存可能提高性能？你的实验结果支持这个猜想吗？为什么？**

   将缓冲区对齐到系统的内存页边界（通常为 4096 字节）可能提高性能，因为它能减少内存拷贝和缓存失效，尤其在直接 I/O 或内存映射场景下，能更好地与硬件和操作系统交互，提升数据传输效率。然而，我的实验结果显示，mycat3（对齐缓冲区，运行时间 219.7 ms）与 mycat2（未对齐缓冲区，219.9 ms）性能几乎相同，未见显著提升。这可能是因为实验使用的是缓冲 I/O 模式，数据经过内核缓存，对齐的优势被掩盖，因此不支持该猜想。

2. **为什么我们直接使用`malloc`函数分配的内存不能对齐到内存页，即使我们分配的内存大小已经是内存页大小的整数倍了。**

   使用 malloc 分配的内存通常只对齐到较小的边界（如 8 或 16 字节），以满足基本数据类型的需求，但不保证对齐到内存页边界（通常为 4096 字节）。这是因为 malloc 在堆中管理内存时，会在任意位置分配内存块，并添加内部元数据（如块头信息），导致返回的指针不一定位于页的起始地址，即使分配的大小是页大小的整数倍。

3. **你是怎么在不知道原始的malloc返回的指针的情况下正确释放内存的?**

   我没有直接使用 malloc，而是使用了 posix_memalign 来分配对齐的内存，并将其返回的指针直接传递给 free 释放。posix_memalign 设计的指针与 free 兼容，POSIX 标准保证了这种方式的正确性，因此无需知道内部原始指针即可安全释放内存。

任务4: 设置缓冲区大小为文件系统块大小的整数倍的cat

1. **为什么在设置缓冲区大小的时候需要考虑到文件系统块的大小的问题?**

   文件系统以块为单位执行 I/O 操作，若缓冲区大小为文件系统块大小的整数倍，可以减少数据在块之间的分割和重组，从而提升 I/O 效率。特别是在直接 I/O 模式下，这种设置能减少不必要的磁盘访问，优化性能。

2. **对于上面提到的两个注意事项你是怎么解决的?**

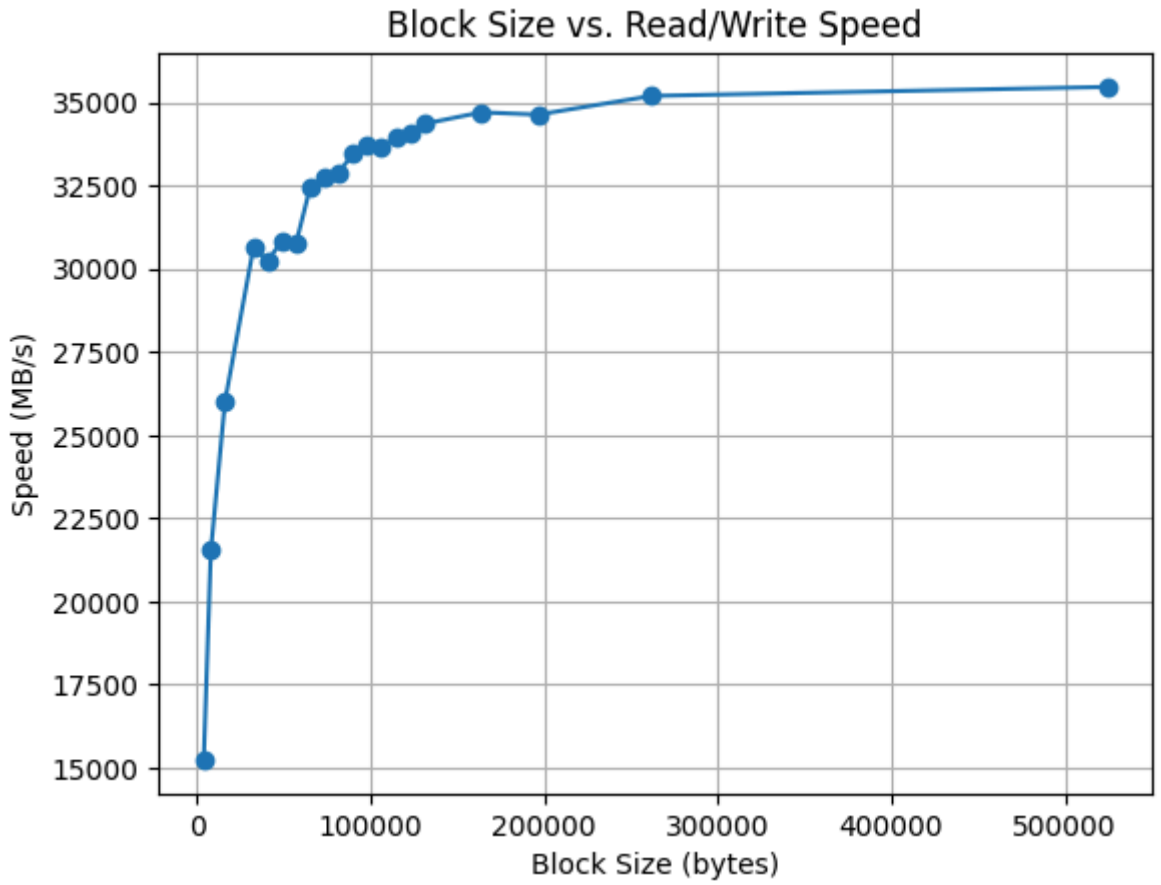
   对于第一个注意事项（文件块大小可能因文件系统而异），我通过 stat 系统调用获取每个文件的建议 I/O 块大小（st_blksize），动态调整缓冲区大小以适应具体文件；对于第二个注意事项（文件系统可能报告虚假块大小），我选择直接信任 st_blksize 作为缓冲区大小的基础，未进行额外验证，以保持实现的简洁性和对文件系统的依赖。

任务5: 考虑系统调用开销情况下的cat

1. **解释一下你的实验脚本是怎么设计的。你应该尝试了多种倍率，请将它们的读写速率画成图表包含在文档中。**

   我的实验脚本使用 dd 命令从 /dev/zero 读取数据并写入 /dev/null，以排除磁盘 I/O 的影响，专注于测量系统调用开销。脚本测试了缓冲区大小为基本单位 buf_size（4096 字节）的多个倍数（1 2 4 8 10 12 14 16 18 20 22 24 26 28 30 32 40 48 64 128），记录每个大小下的读写速度，并分析性能随块大小变化的趋势，最终找到性能提升趋于平缓的最优倍数（A = 24）。

   图表如下



任务6: 使用了系统调用`fdadvice`的cat

1. **你是如何设置`fadvise`的参数的?**

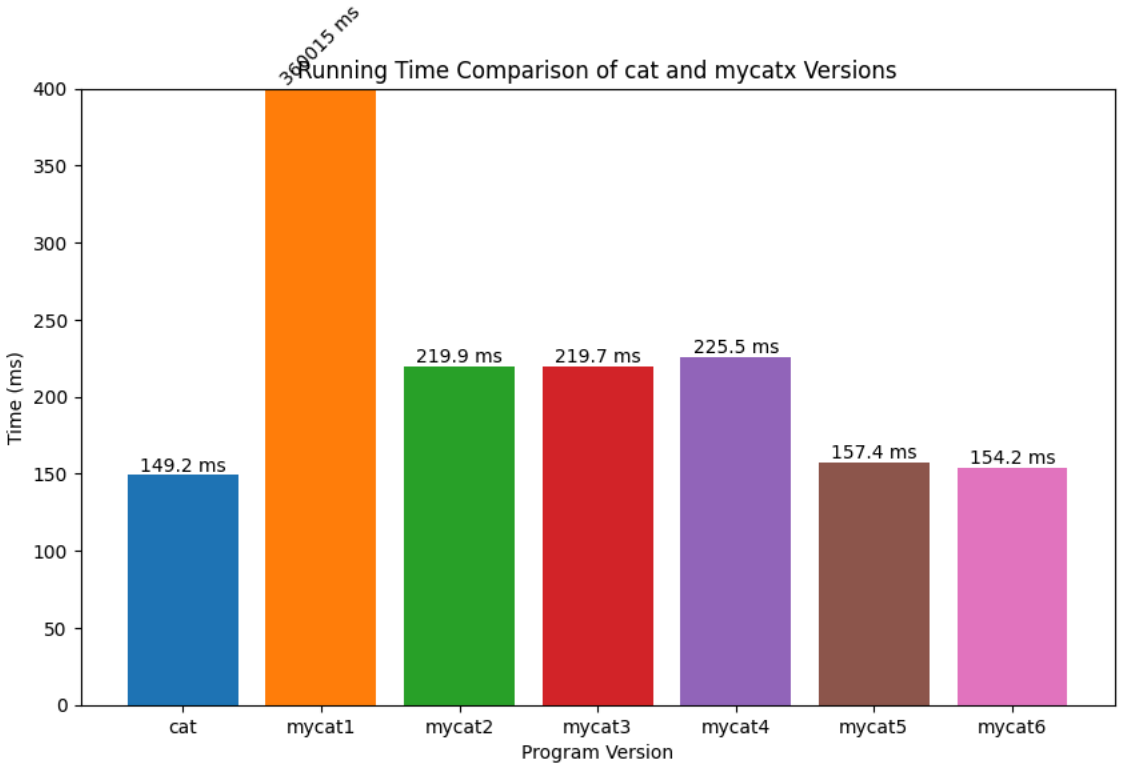
   我使用了 posix_fadvise(fd, 0, 0, POSIX_FADV_SEQUENTIAL) 来设置参数，其中 fd 是文件描述符，offset = 0 和 len = 0 表示应用于整个文件，advice = POSIX_FADV_SEQUENTIAL 提示文件系统文件将按顺序访问，从而优化预读策略。

2. **对于顺序读写的情况，文件系统可以如何调整readahead？对于随机读写的情况呢?**

   对于顺序读写，文件系统会增大预读窗口（readahead），提前将更多连续数据加载到内存，减少 I/O 等待时间，提升吞吐量；对于随机读写，文件系统会减小或禁用预读，以避免加载不必要的数据，从而优化内存使用效率并减少浪费。

任务7: 总结

1. **你的全部实验结果的柱状图。**



2. **你对上述实验结果的分析。**

实验结果表明，mycat1（无缓冲区，360015 ms）性能极差，引入缓冲区的 mycat2（219.9 ms）显著提升性能，而 mycat3（对齐缓冲区，219.7 ms）与 mycat2 接近，显示对齐在缓冲 I/O 中影响有限。mycat4（缓冲区为文件系统块大小的 LCM，225.5 ms）未见明显优化，mycat5（实验优化缓冲区大小，157.4 ms）和 mycat6（使用 fadvise，154.2 ms）进一步接近系统 cat（149.2 ms）的性能。看来，缓冲区大小和使用方式对I/O性能有显著影响。在I/O密集型任务中，减少系统调用是优化的核心，缓冲区和文件访问提示（如posix_fadvise）是有效手段。单纯依赖理论（如内存对齐或LCM）可能不足以优化性能，实验验证是找到最佳方案的关键。