基于 socket 通信和存储管理优化的虚拟 SSD

陶稼辉 等

January 2024

§1 问题重述

SSD 与原来的机械硬盘不同，机械硬盘由于存在着两个机械运动，一是盘片旋转，第二个是磁头寻道，所以天生在读取数据时，有旋转延迟和磁头寻道这两个机械动作，所以性能很难得到突破。

SSD 则不同，由于没有机械运动，所以 SSD 天生具有高带宽的特点。但是 SSD 的存储部件是 Flash 芯片，所以，当芯片内部空闲的页面不太有的情况下，就会需要将一些页擦除，而擦除是比较耗时的。所以，SSD 在使用一段时间后，性能会下降。

现在我们再来一次硬件的软件模拟，使用 Python 来实现一块 SSD。这块 SSD 完全是由 Python 在内存或外存中模拟出来的，容量当然不能太小。既然要模拟真实的 SSD，就需要模拟到位。一个 SSD 是由若干 Flash 芯片作为存储，每块 Flash 划分为若干个页，一个页一般是 4K Bytes。除 Flash 芯片以外，还有一块主控，主控的一个任务就是将所有的写尽可能均匀地分散到不同的页上，这样，保证 Flash 芯片不会被写坏。

这样的一个 Python 牌 SSD，最好能够模拟成一个驱动器，然后可以被格式化，复制文件进去。最好还有一个工具，能够看这个硬盘是的 Flash 芯片的页的情况，哪些页是空的，哪些页是被占用了。

§2 我们的工作

我们的程序主要由三部分组成：

- SSD 存储与地址管理策略
- 基于 Socket 的多终端用户交互支持
- 基于 matplotlib 的 SSD 使用监视器

我们首先书写了一个 SSD 的管理内核，这个 SSD 内核管理着下属的若干个 Flash 芯片，它们被假想为一个类与文件存在。通过基于 Socket 的多终端用户交互支持，Server 程序支持同时处理多个用户对 SSD 的操作。同时，基于 matplotlib 的 SSD 使用监视器可以直观的查看 SSD 中每个 Flash 的存储占用情况，更好的展示和窥探 SSD 的存储与地址管理策略。

§2.1 SSD 存储与地址管理策略

SSD 主要由 SSD、Mapping、Flash 三个类组成，同时由 Atomic、DataStruct、Exceptions 提供一些额外的通用支持。

2.1.1 类分工与关系

SSD 是用户交互的主接口，所有 SSD 管理都应该通过 SSD 类调度。它提供了启动和初始化 SSD、列出占用、拷入、拷出、关闭 SSD 等方法。SSD 初始化时，支持用户自定义 SSD 大小、Flash 数量和页大

小。SSD 类管辖了一个 Mapping 对象和若干个 Flash 对象，这些对象用于具体管理地址和存储，由 SSD 调度。其中，Mapping 对象作为 SSD 的成员对象被持有，Flash 对象作为 SSD 的子线程被持有。

Mapping 对象用于管理 SSD 中虚拟地址到若干 Flash 中真实地址的映射关系，描述了 SSD 中的占用情况。Mapping 对象在磁盘以 Mapping 文件的形式存在，所有地址管理与规划都应该由 Mapping 完成。Mapping 对象作为 SSD 类的受保护成员存在，为了为“基于 matplotlib 的 SSD 使用监视器”提供更好的权限，监视器应该被授予直接访问 Mapping 对象的权限。

Flash 对象直接管理磁盘中的 Flash 文件，每个 Flash 对象对应管理磁盘中的一个 Flash 文件，所有来自 SSD 的底层读写都应该递交由 Flash 完成。Flash 的优化是目前 SSD 拷贝速度的关键，也是瓶颈。

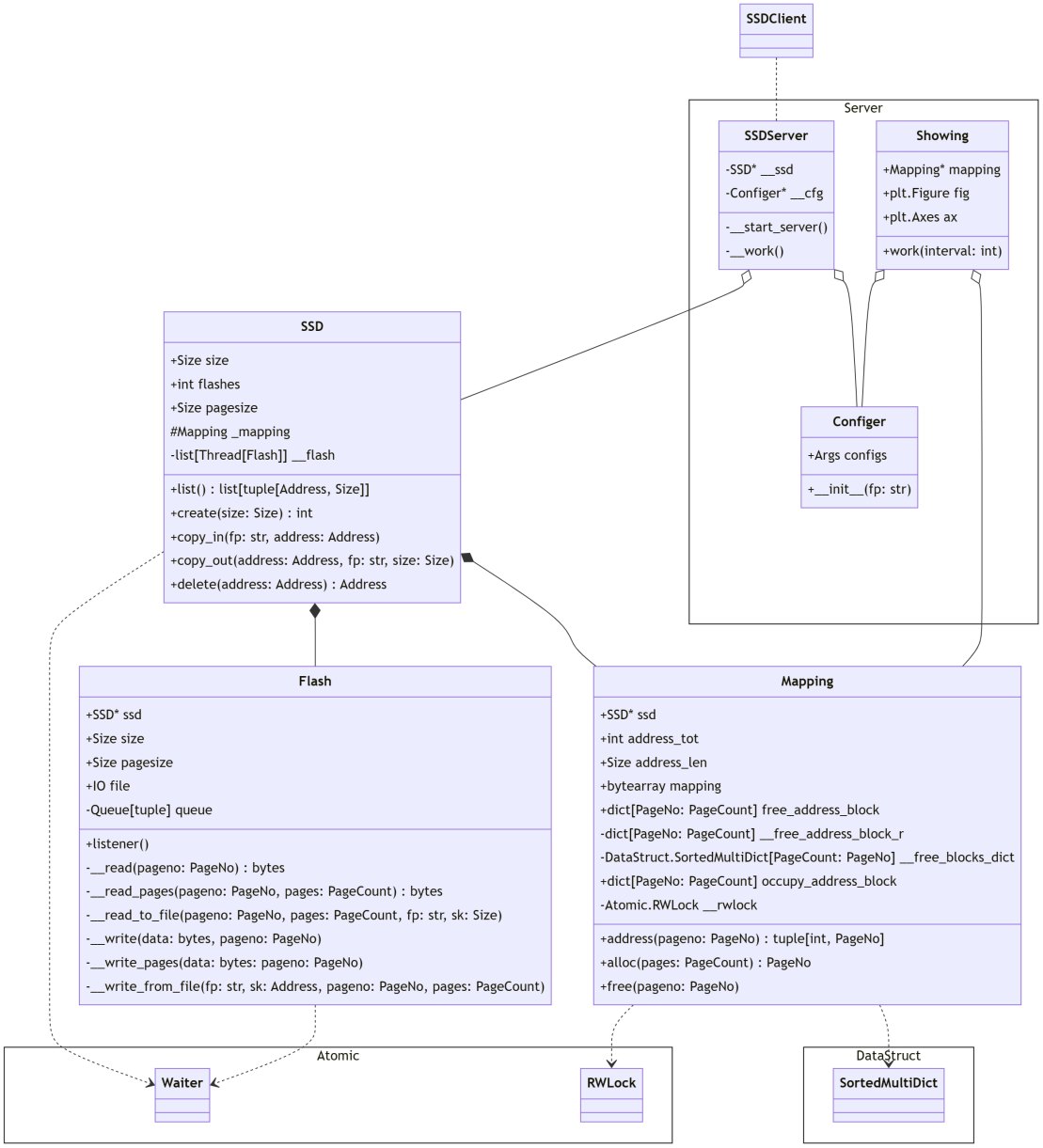

图 1: 设计类图

2.1.2 Mapping 的存储管理策略

Mapping 的存储管理采用最佳适应策略（Best Fit），即每次从剩余可供分配的碎片中，选取最小的合适碎片用于存储。在 Mapping 中，我们维护了 ___free__blocks__dict 成员，它是 DataStruct.SortedMultiDict 类型的，它是第三方库 sortedcollections 中的 SortedDict 的再封装，其底层是红黑树实现的，支持对数时

间的二分查找。当期望申请新的空间时，它都会在 Mapping 中找到最小的合适连续碎片用于存储。

值得注意的是，在我们的 SSD 系统中，所有存储分配的单元都是页，也就是说，即使您仅需存储 1KB 或者更小的文件，它将申请的存储也是一个页（默认 4KB），目前没有做页内存储多元素的支持。

这也就是说，对于一个大约 64 GB 的我们设计的 SSD（默认页大小 4KB），它将被管理的页数是 16M 个，目前我们将这些页信息全部载入内存，它产生的连续存储碎片数一定不超过 8M 个，这对于支持二分查找的数据结构来说是一个非常好的时间和可接受的空间开销。

Mapping 中保留了对待擦除页和损坏页的状态设置，但这些状态的相应操作并没有被实现。这里想表达的是，SSD 的虚拟化管理实际上并不完全，过度虚拟化甚至会降低虚拟 SSD 的整体效率。

2.1.3 Flash 与 SSD 间通信以及读写操作

在我们的 SSD 中，Flash 以子线程的形式存在，被 SSD 管理。Flash 与 SSD 之间通过队列（Queue）通信，每个 Flash 分别占有对应队列的读端，而写端由 SSD 持有。Flash 中的监听程序将接收 SSD 指令并分别执行相应读写指令，每完成一条指令将向 SSD 发送反馈数据。其中反馈接收器被支持为一个 Atomic.Waiter（我们实现的多线程等待计数器）或一个队列，它在 SSD 发送指令时被传入，而 SSD 指令发送函数将等待 Flash 响应才会返回。

Flash 的读写效率始终是一个难以解决的问题，我们一直试图通过多线程 IO 和提升数据局部性以获取尽可能高的 IO 效率，目前经过三次 Flash 读写优化，其拷贝速度基本达到理想效果。

(Gbps)	拷入	拷出
单客户测试	9.84	5.04
双客户并发	6.74 × 2	2.66 × 2
直接转发 (对比)	4.94	
系统拷贝 (对比)	5.33	

表 1: 本机测速 (8GB 拷贝测试)

（直接转发：使用 read / write 经过内存的拷贝（每次 1GB）；系统拷贝：使用 shutil.copy 实现的拷贝）

2.1.4 对多终端用户交互的支持

对多终端用户交互的支持，对我们的 SSD 提出了更高的要求，它要求我们的 SSD 必须是线程安全的。在 SSD 类内，只存在 Mapping 对象和与 Flash 通信的消息队列存在安全风险。

在 Flash 与 SSD 的通信上，我们使用 Python 内置的 queue.Queue 实现两者之间的通信，它是线程安全的。而在 Mapping 中，存在对临界值的访问，我们考虑使用读写锁实现对 Mapping 线程安全的保障。简单查阅网络后，我们没有找到一个很好的 Python 内置或第三方的读写锁，因此我们实现了 Atomic.RWLock 用于 Mapping 中。

另外，值得一提的是，我们在 SSD 中函数阻塞返回的设计，使得在用户视角，其自身将阻塞到操作结束，而用户之间是并行的。

§2.2 基于 Socket 的多终端用户交互支持

程序基于 socket 库建立了一个简单的客户端-服务器通信系统，并且通过多线程支持了多个客户端的并发需求。客户端负责与用户交互，然后将用户操作的信息序列化发送给服务端。服务端直接与 SSD 通信，实现用户操作，然后返回操作结果。

客户端的交互方式有两种：

- 输入选项数字一步步交互。

- 输入一行包含所有信息的指令。

自定义了 IO 库，专门用于判断输入数据格式的正确性，例如输入是否是整数，是否是合法路径。同时 IO 库还负责处理文件单位的转换，在代码内部，文件大小统一以 Bit 运算和存储，但为了用户输入的方便性和观察的直观性，所以支持了加上单位的输入方式，输出显示的是进行了合适单位转换后的大小。

通信信息以字节流的形式传递，采用了 pickle 库进行信息序列化与反序列化的工作。

§2.3 基于 matplotlib 的 SSD 使用监视器

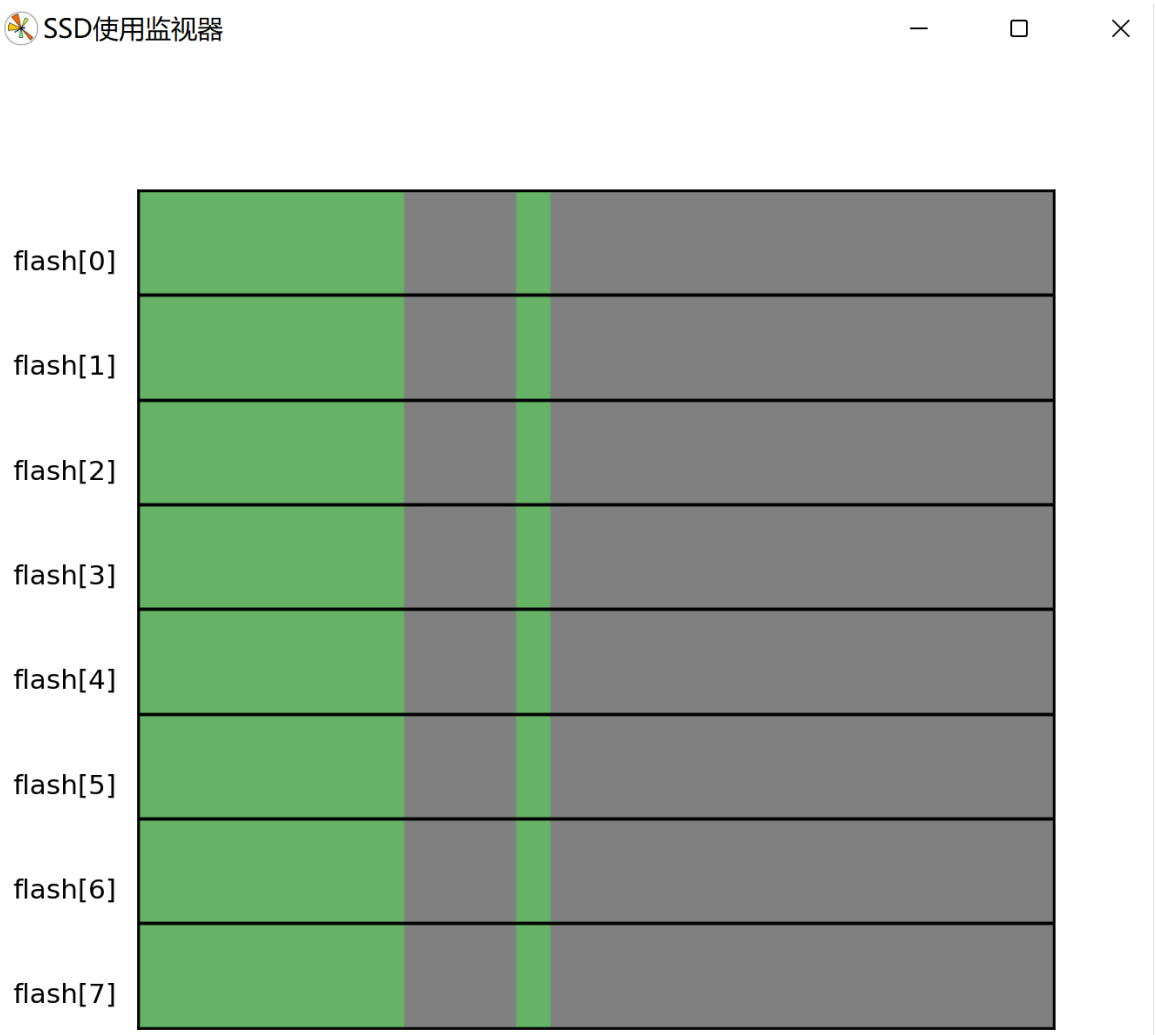

图 2: 界面展示

Showing 对象用于展示 SSD 的使用情况。它利用 matplotlib 绘制一个表示 SSD 存储占用的矩形图，并尝试绘制空闲地址块。考虑到存储占用的数值较小且数量多，而空闲块则以离散块状呈现且互不覆盖，这样可以减小绘图计算量，提高绘图速度。

因此，Showing 对象遍历了 free_address_block 字典，在 SSD 上显示空闲的地址块，并根据其值绘制矩形来表示空闲地址块。对于每个空闲区域，为了均匀地分布在每个闪存芯片上，我们对每个空闲值进行函数映射，使其均匀地分布在各个区域。

动画每 300 毫秒更新一次，动态地展示 SSD 使用情况的变化。

§3 总结

§3.1 分工与合作

3.1.1 分工与责任明细

本项目包括 11 个项目相关代码文件、1400 行代码，全部由小组成员共同协作完成，其中：

陶稼辉负责整体设计、SSD 存储与地址管理策略。负责总体设计规划，并设计包括 SSD、Flash、Mapping、Atomic、DataStruct、Exceptions、GlobalConfig 在内的类。实现 SSD 的存储与地址管理、实现 Flash 与 SSD 之间的安全并发、进行 Flash 拷贝的性能调优、对 SSD 提供对外的并发支持和线程安全、SSDServer 及其他参数的可配置化问题。

XXX 负责基于 Socket 的多终端用户交互支持。设计包括 SSDServer、SSDClient、IO 在内的类和包。其中，SSDServer 完成对于 SSD 的调度，同时实现与 SSDClient 之间的通信，它支持多客户同时与 SSDServer 建立连接。基于 Socket 通信，独立解决包括 SSDServer 与 SSDClient 之间的通信协议、SSDServer 对多用户的支持、保障 SSDServer 的安全关闭等问题。

XXX 负责基于 matplotlib 的 SSD 使用监视器。设计包括 Showing 在内的类。实现 SSD 使用监视器实时监视 SSD 使用情况，它实际监视了 SSD 对象的受保护成员 _mapping 内部变化。完成了从选择合适的 UI 库，到正常显示与定时刷新的工作。同时接触使用 git 参与到项目协作中。

3.1.2 协作过程展示

我们的项目从 1 月 8 日开始筹备，在 1 月 8 日傍晚陶稼辉同学完成了项目的整体设计。完成了 SSD、Mapping、Flush 的基本框架与尚存问题的代码，预留了对外部 SSDServer、SSDClient、Showing 预留的接口。

随后我们的工作，基本通过 git 协作完成。（见 附录）

§3.2 待提升的点

- 全系统（Mapping、Flash、SSDServer）的异常安全问题一直没有被很好的解决。当程序因为代码逻辑错误或 Server 进程被非正常退出，导致 Mapping、Flash、SSDServer 非正常关闭时，将导致 Mapping 文件异常、Flash 读写中断。这可能导致地址映射关系丢失、拷贝中数据出现错误、之后 SSDServer 无法正常启动。

- Flash 目前读取（拷出）速度或许依然存在可以优化的点。Flash 写入速度对比系统速度做出显著优化，但对比 Flash 自身读与写速度，发现读取速度基本无法突破系统速度，多用户读取速度也基本没有产生额外效益（见 本机测速）。事实上这个问题在优化之初就有体现，随着优化深入愈发凸显，猜测是文件读写持锁问题，目前未进一步排查问题。

- Mapping 的碎片空间问题。目前 Mapping 使用最佳匹配算法尽量减少碎片空间，但依然无法避免碎片空间的存在。当出现大量碎片空间时，Mapping 是否可以在 Flash 空闲时，自动管理优化底层碎片。

- 由于用户端拥有关闭服务端的功能，所以服务端在接受到用户端的请求后需要等待很小一段时间的解码，这一等待过程会暂时阻塞线程。同时，为了防止等待用户输入时也阻塞主线程，服务端与用户端的通信并非一直保持，而是每次用户端需要发送信息时再建立链接，这一过程不太符合通信规范。

- matplotlib 库只保证自己作为主线程时会正常工作，而图形化界面跑在服务端，因此服务端的主线程只能是 matplotlib，在用户端发送关闭服务端的请求时，如果此时 matplotlib 的图形界面未关闭，则会阻塞服务端的关闭行为。

- 对于 SSD 使用监视器，目前只能显示某块区域是否可以被占用，如若可以用不同的颜色来区分不同的文件类型，可以更直观的展示存储占用情况。

```
PS D:\code\VirtualSSDbyPython> git log --pretty=oneline --graph
*   3b3c1e5d81da9d70fde6afacc6ed4a42718bc84b (HEAD -> main, origin/main, origin/jamhus-tao, origin/HEAD) Merge branch 'main' into cursor
|\
| * 5ad7aa27539d1a4ae455797906ce911e935bcae3 (origin/smallC, smallC) 更改了文件单位的显示
| * 5fc7782744f43ce3170d7fa5d641580f49d9ae1c 修复了一个格式问题
| * 59ecdb6adc2c4cc485f9eab2bf1c261ff129ce54 加入了 help 内容
| *   460db59aa5cc98caa259c06c8c8f17ce6a676770 Merge branch 'smallC'
| |\
| | * 109039ae2c4e91c7099ac723c87d492fa6a4d586 添加了人性化的输出
| * | 0120eef1ae8892402e990fd3297d591f058cb52c Merge branch 'main' into jamhus-tao
| |\|
| | * 326016f29380ff5fa8535dfed90b674f7c575170 加入了人性化的大小输入
| | *   c3b9edce3e88ccbe6b022c641649c559b6fb5f7f Merge branch 'smallC'
| | |\
| | | * 7e4101a4e67c80f33ccb2d106e79926e095c8735 修复了一些交互 bug
| * | | 932343a49558eb670a95f95deaca14b3a976584b Exception 的一处代码规范问题
| |/ /
| * | 1e5d297d841f3d86b8b597c439990ff61d1b164e 删除 Flash 中不会跑到的遗留分支
* | | 25d0d323af67fc38ba0d544527edb29668b3c2d9 (origin/BornStrange) 修改窗口标题
|/ /
* | 03457d8d1f93e84813b38644803c04df7c1cf242 修复无法实时更新的问题
* | 4c641bf01c6169eb784de788086d5be8c24a3a2d 删除网格线
|/
*   67870c8faeb2f9fe4599430af90dec6f89d4de21 Merge branch 'main' into jamhus-tao
|\
| *   cb72f29a76041b537f48864cf1cc5f07c47fa6bf Merge branch 'smallC'
| |\
| | * 647c09446369ea30a81bbdee4cebc6a9e0e2f6db 加入另外一种交互方式（输入指令）
* | | b8445b6e7a91aac84caa3266ad9c6771dd058b07 Flash 的新增多用户并发支持
* | | 9a17d8e93014712a7f4363da85231c7c7180abac Showing 中一处刷新问题
|/ /
* | 8c75d64d2681123cd64ebb7ec4609d3d4f189192 将 Showing 中的刷新间隔设置为可配置的
* | 06ce1579a4082ac81f809972fa36315ae788a3d5 将 Showing 代码规范化
* | 9886fe456ba6559d8465d6ea7458eb679352f557 增加了新的 Configer 配置
* | 4baeefd1e7430644f1dbdee8f38bd5a81ea7af7d 在 IO 库中新增 parse_time_seconds 类
|/
* af539ffae93ef0b70304b0f9608fb895d93285c5 新建 IO 库
* 8551547aba57fe995c86a2d93feda169ba8db990 将 Server.yml 暂时加入 gitignore 避免贡献者之间配置混乱
* c80e555d32ac6e0d06935ea37df7cc9892da5a88 将 Configer 从 SSDServer 中抽出，作为单一的类存在
* 78a2bfcc82bfb429b07a037462526c867f9e8ebc SSDServer 调整 filesys 配置
* 6790ef5e27dbe5dcd5fa72e3a90660f5da9db050 SSDServer Merge 好像出了点问题，现在修好了
*   fdc05b7cba79b95d690968e7bfa7134f18af07f8 Merge branch 'jamhus-tao' into cursor
|\
| * 7ea550ca9be1aaecb656090ce489d30f5587207d 对 SSDServer 的启动配置化进行优化
| * 18d689fad8f5c1693db3d0739ad18e322259735c Flash 使用新的底层管理策略，速度提升约 10 倍
| * 5dd5949e4095b89445300aaa2d48d29bce54b4e7 bugfix: 修复更改通信协议后 SSD 产生的新问题
| * 876e9c84d730d7b0ae70bdbfd2ecca1921039279 bugfix: mapping_fix 好像还是存在问题，应该是修复了
| * 8fd0442285fbc1ed2713a5b446cabc5733e841b6 bugfix: 修复 Showing 中的显示条比例问题
| * 35aa592c734547b0f7baffa470978d21a6b5dfae 修改 Mapping 锁操作更精细
| * 32675b42510cd21219aeb3d633f40014d7b93da2 补充通信协议一行注释
| * ac06c4086c07ea20f760faadd3a8a3505765e6d8 修改 SSD 与 Flash 的通信协议，发送 dict 通信
| * 60d14e656e7cd13566c2759dd90e1efa51af7fbf Mapping 新增 address_interval 函数，快速计算区间地址
| * 6271c658b7a72ed0aae9ad080a89517214b2037c 别把你的 test push 上来 \生气\生气
| *   647d378a83687da7c7528d167c17b07fb753bf4c Merge remote-tracking branch 'origin/main' into jamhus-tao
| |\
| * | 0ac7fb3b617b209d85ae6ecb450853935e637be3 去除所有 os.chdir 在 SSD 内部全部使用绝对路径 && bugfix: 发现 SSD 每次创建配置而不读取
* | | 6017694584c5c01fe878a2a9b43f9ad7c7b70f9f 补充了IO类，统计了操作花费秒数，完善了 Dict
| |/
|/|
* | f9cb86b45a8300963 8db7de5adc092a5182f1cb4 修改了部分交互，增加了格式化
|/
* 48a72691394c657519b77c1d71fc66b4b66329ad 现在 mapping fix 也与 Server.yml 配置绑定，同时清理 dict.bin
* 787b5b2fa9eee441dcda6d5337f6c4b1017542b6 bugfix: SSD 存在一行错误，修复拷入小文件时的失败
* 25d6a7fecdf855e4d5a6b8336197609a99e413a4 bugfix: 修复 mapping_fix 的一个问题
* 787b5b2fa9eee441dcda6d5337f6c4b1017542b6 bugfix: SSD 存在一行错误，修复拷入小文件时的失败
* 25d6a7fecdf855e4d5a6b8336197609a99e413a4 bugfix: 修复 mapping_fix 的一个问题
* af4b131d56cbd9758922244b35a4d1f463179d9de 为 SSDServer 提供配置化
* a8dcd15be16550bf4e9cf502b7e6afa50e5ff3dc bugfix: 修改一处错误调用
*   5d9b93c2de7f75cd046cd7eb01349bb25768a9d2 Merge branch 'cursor' into jamhus-tao
|\
| * 00f7ea01566ab88f97196a78ced352bec5102776 恢复了 Mapping 中的读写锁
* | 8d52a23183bafbe61f26b3982f796cc60fbf9e35 更改了 dict.bin 的路径
|/
*   1ca8c15bf26dc216e60df4ea363023053f7cca91 Merge branch 'main' into smallC
|\
| * d47b61c94c21ea57ac099987247c9d9b9609805c 从 Showing 中删除了 Showing.ShowWrapper
| * 069b400367ae323f56aa49d5a2b6db73b6bb9feb 对 Flash 的读写速度做了一定优化：1M/s
* | de01d2ee29e285723dc7affc2c621f5c7ab33fa3 修复了多线程问题，把 SSDServer 集成成了一个类
|/
* 5522e4923fe9bc420a4edcb3c7d634b27fc8d0e8 完成了 SSDClient 和 SSDServer
* 02a92d02c93e9650d406077cf0296d54a6e40235 新增 ShowingWrapper 类，用于启动 Showing，目前只有接口
* 15b1616f2695c3a6d96cc8a879a0702423284da8 将 temp 类文件加入 ignore
* 34220116ac6fe86b9cb925e8260e0cbef958722c 添加 Showing 启动代码，目前只有一个接口
* 411f9888ba3a55560ed1bb00118c2d9ab92bbec2 新增图形界面 Showing，目前启动存在一些问题 - zlm
* ddd674d6f7211d05e29f35aa889aedbce93b2000 删除 Mapping 中的读写锁，Mapping 始终只被一个线程持有
*   3af45604c44e57bd3fa33b87b9e2ee07f18c43a4 Merge branch 'jamhus-tao'
|\
| * 6560d72506bc995745e80cfe2b88f395455d499c 修复 mapping fix 的问题
| * 5d89f721bc38b0062f7977021a90d66f69defe93 调整 Flash / Modulo 的一些权限
* | 73a48a98d2e4a1d579dd9c7d107114123d2330e0 完成了部分 SSDClient 和 SSDServer
|/
* 8d849b373d003ef81ebe0b68527b9f45ed43cbe0 bugfix: Mapping 存在严重问题 && 新增 DataStruct.SortedMultiDict 以规范化代码
* 326149e7f8e13f96ac577eb1d895a6a72875cf7f 暂时提供修复脚本 mapping fix
* 941e8f5ec78b1e76f6e9c61c811b817008371a61 bugfix: SSD create 方法返回值有问题
* 234f26e949b71ac23a08158a0515fb8d58aa1442 bugfix: SSD 未正常关闭 Mapping
* 2e8e408276981f0015da3422a741d3de6ed22981 新增 RWLock，处理 Mapping 的并发安全问题
* 9d322d683ce9c94272357204815ca26377076a64 改写 Atomic.Waiter，提供 with 语法支持
* 30cf996ea1fe317550c3d207fdc839c93eed8f55 bugfix: SSD copy out 未创建文件
* da54e9cfce6af6c7ac6310e593dfea4e10a10811 atomic 加入一个泄漏的异常检测
* ef3efbf4a88bc8c3a532e4b7b5b1061cab8dd75d init commit
* acd40a7f818edfa81a6d09b7bbb9025ba148fa1d init commit
* d22cd16812b031b0ffd1204512f4e2f7523719a0 first commit
```

图 3: 协作过程展示