MQSim: A Simulator for Modern NVMe and SATA SSDs

MQSim is a simulator that accurately captures the behavior of both modern multi-queue SSDs and conventional SATA-based SSDs. MQSim faithfully models a number of critical features absent in existing state-of-the-art simulators, including (1) modern multi-queue-based host–interface protocols (e.g., NVMe), (2) the steady-state behavior of SSDs, and (3) the end-to-end latency of I/O requests. MQSim can be run as a standalone tool, or integrated with a full-system simulator.

The full paper is published in FAST 2018 and is available online at
https://people.inf.ethz.ch/omutlu/pub/MQSim-SSD-simulation-framework_fast18.pdf

Citation

Please cite our full FAST 2018 paper if you find this repository useful.

> Arash Tavakkol, Juan Gomez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu, "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices" Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST), Oakland, CA, USA, February 2018.

```
@inproceedings{tavakkol2018mqsim,
  title={{MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices}},
  author={Tavakkol, Arash and G{\'o}mez-Luna, Juan and Sadrosadati, Mohammad and Ghose, Saugata
  booktitle={FAST},
  year={2018}
}
```

Additional Resources

To learn more about MQSim, please refer to the slides and talk below:

- Slides: `(pptx)` `(pdf)`
- Talk: `Introduction to MQSim` from the `Understanding and Designing Modern NAND Flash-Based Solid-State Drives (SSDs)` course

MQSim：一个用于现代 NVMe 和 SATA SSD 的模拟器

MQSim 是一个模拟器，能够精确地捕捉现代多队列 SSD 和传统 SATA 基础 SSD 的行为。MQSim 忠实地模拟了现有最先进模拟器中缺失的许多关键特性，包括（1）基于现代多队列的主机 - 接口协议（例如，NVMe）、（2） SSD 的稳态行为，以及（3） I/O 请求的端到端延迟。MQSim 可以作为独立工具运行，或与完整系统模拟器集成。

全文发表在 FAST 2018 上，并可在以下网址在线获取
MQSim：一个用于现代 NVMe 和 SATA SSD 的模拟器引用

引用

如果您觉得这个仓库有用，请引用我们的全文 FAST 2018 论文。

> Arash Tavakkol、 Juan Gomez-Luna、 Mohammad Sadrosadati、 Saugata Ghose 和 Onur Mutlu， “MQSim：一个用于实现现代多队列 SSD 设备真实研究的框架 ” USENIX 文件和存储技术会议（ FAST ）第 16 届会议论文集，Oakland，CA，美国，2018 年 2 月。

@inproceedings{tavakkol2018mqsim, title={{MQSim：一个用于实现现代多队列 SSD 设备真实研究的框架 }}, author={Tavakkol, Arash and G{\'o}mez-Luna, Juan and Sadrosadati, Mohammad and Ghose, Saugata booktitle={FAST}, year={2018} }

其他资源

要了解更多关于 MQSim 的信息，请参考以下幻灯片和演讲：

- 幻灯片： `(pptx)` `(pdf)` 演讲： MQSim 简介来自理解和设计基于 NAND 闪存的固态驱动器（ SSD ）
- 课程的演讲

Usage in Linux

Run following commands:

```
$ make
$ ./MQSim -i <SSD Configuration File> -w <Workload Definition File>
```

Usage in Windows

1. Open the MQSim.sln solution file in MS Visual Studio 2017 or later.
2. Set the Solution Configuration to Release (it is set to Debug by default).
3. Compile the solution.
4. Run the generated executable file (e.g., MQSim.exe) either in command line mode or by clicking the MS Visual Studio run button. Please specify the paths to the files containing the 1) SSD configurations, and 2) workload definitions.

Example command line execution:

```
$ MQSim.exe -i <SSD Configuration File> -w <Workload Definition File>
```

MQSim Execution Configurations

You can specify your preferred SSD configuration in the XML format. If the SSD configuration file specified in the command line does not exist, MQSim will create a sample XML file in the specified path. Here are the definitions of configuration parameters available in the XML file:

Host

1. **PCIe_Lane_Bandwidth:** the PCIe bandwidth per lane in GB/s. Range = {all positive double precision values}.
2. **PCIe_Lane_Count:** the number of PCIe lanes. Range = {all positive integer values}.
3. **SATA_Processing_Delay:** defines the aggregate hardware and software processing delay to send/receive a SATA message to the SSD device in nanoseconds. Range = {all positive integer values}.
4. **Enable_ResponseTime_Logging:** the toggle to enable response time logging. If enabled, response time is calculated for each running I/O flow over simulation epochs and is reported in a log file at the end of each epoch. Range = {true, false}.

在 Linux 中的使用

运行以下命令：

```
$ make
$ ./MQSim -i <SSD 配置文件 > -w < 工作负载定义文件 >
```

Usage in Windows

1. 在 MS Visual Studio 2017 或更高版本中打开 MQSim.sln 解决方案文件。
2. 将解决方案配置设置为 Release （默认设置为 Debug ）。
3. 编译解决方案。
4. 运行生成的可执行文件（例如，MQSim.exe ），可以在命令行模式下运行，或通过点击 MS Visual Studio 运行按钮运行。请指定包含 1) SSD 配置和 2) 工作负载定义的文件路径。

示例命令行执行：

```
$ MQSim.exe -i <SSD 配置文件 > -w < 工作负载定义文件 >
```

MQSim 执行配置

您可以在 XML 格式中指定首选 SSD 配置。如果命令行指定的 SSD 配置文件不存在，MQSim 将在指定路径创建一个示例 XML 文件。以下是 XML 文件中可用的配置参数定义：

Host

1. **PCIe_Lane_Bandwidth:** 每个 PCIe 通道的带宽，单位为 GB/s。范围 = {all positive double 精度值 }。
2. **PCIe_Lane_Count:** PCIe 通道数。范围 = { 所有正整数值 }。 3. **SATA_Processing_Delay:** 定义了向 SSD 设备发送 / 接收 SATA 消息的硬件和软件总处理延迟，以 纳秒为单位。范围 = { 所有正整数值 }。

4. **Enable_ResponseTime_Logging:** 响应时间记录的开关。如果启用，将在每个模拟周期内为每个运行中的 I/O 流计算响应时间，并在每个周期结束时在日志文件中报告。范围 = {true, false}。

5. **ResponseTime_Logging_Period_Length:** defines the epoch length for response time logging in nanoseconds. Range = {all positive integer values}.

SSD Device

1. **Seed:** the seed value that is used for random number generation. Range = {all positive integer values}.
2. **Enabled_Preconditioning:** the toggle to enable preconditioning. Range = {true, false}.
3. **Memory_Type:** the type of the non-volatile memory used for data storage. Range = {FLASH}.
4. **HostInterface_Type:** the type of host interface. Range = {NVME, SATA}.
5. **IO_Queue_Depth:** the length of the host-side I/O queue. If the host interface is set to NVME, then **IO_Queue_Depth** defines the capacity of the I/O Submission and I/O Completion Queues. If the host interface is set to SATA, then **IO_Queue_Depth** defines the capacity of the Native Command Queue (NCQ). Range = {all positive integer values}
6. **Queue_Fetch_Size:** the value of the QueueFetchSize parameter as described in the FAST 2018 paper [1]. Range = {all positive integer values}
7. **Caching_Mechanism:** the data caching mechanism used on the device. Range = {SIMPLE: implements a simple data destaging buffer, ADVANCED: implements an advanced data caching mechanism with different sharing options among the concurrent flows}.
8. **Data_Cache_Sharing_Mode:** the sharing mode of the DRAM data cache (buffer) among the concurrently running I/O flows when an NVMe host interface is used. Range = {SHARED, EQUAL_PARTITIONING}.
9. **Data_Cache_Capacity:** the size of the DRAM data cache in bytes. Range = {all positive integers}
10. **Data_Cache_DRAM_Row_Size:** the size of the DRAM rows in bytes. Range = {all positive power of two numbers}.
11. **Data_Cache_DRAM_Data_Rate:** the DRAM data transfer rate in MT/s. Range = {all positive integer values}.
12. **Data_Cache_DRAM_Data_Burst_Size:** the number of bytes that are transferred in one DRAM burst (depends on the number of DRAM chips). Range = {all positive integer values}.
13. **Data_Cache_DRAM_tRCD:** the value of the timing parameter tRCD in nanoseconds used to access DRAM in the data cache. Range = {all positive integer values}.
14. **Data_Cache_DRAM_tCL:** the value of the timing parameter tCL in nanoseconds used to access DRAM in the data cache. Range = {all positive integer values}.
15. **Data_Cache_DRAM_tRP:** the value of the timing parameter tRP in nanoseconds used to access DRAM in the data cache. Range = {all positive integer values}.
16. **Address_Mapping:** the logical-to-physical address mapping policy implemented in the Flash Translation Layer (FTL). Range = {PAGE_LEVEL, HYBRID}.

5. **响应时间日志周期长度：**定义了用于响应时间日志记录的周期长度，单位为纳秒。范围 = { 所有正整数值 }。

SSD 设备

1. **种子值：**用于随机数生成的种子值。范围 = { 所有正整数值 }。

2. **启用预条件化：**启用预条件化的开关。范围 = {true, false}。
3. Me **内存类型：**用于数据存储的非易失性内存类型。范围 = { 闪存 } .
4. **主机接口类型：**主机接口的类型。范围 = {NVME, SATA}。
5. **I/O 队列深度：**主机端 I/O 队列的长度。如果主机接口设置为 NVME，则 **I/O 队列深度**定义了 I/O 提交和 I/O 完成队列的容量。如果主机接口设置为 SATA，则 **I/O 队列深度**定义了原生命令队列（ NCQ ）的容量。范围 = { 所有正整数值 }

6. **队列获取大小：**FAST 2018 论文中描述的 QueueFetchSize 参数的值 [1]。范围 = { 所有正整数值 }

7. **缓存机制：**设备上使用的数据缓存机制。范围 = { 简单：实现简单的数据下缓冲区，高级：实现具有不同共享选项的高级数据缓存机制，用于并发流 }。

8. **数据缓存共享模式：**当使用 NVMe 主机接口时，并发运行的 I/O 流之间 DRAM 数据缓存（缓冲区）的共享模式。范围 = { 共享，等分 }。

9. **数据缓存容量：** DRAM 数据缓存的大小，以字节为单位。范围 = { 所有正整数 }
10. **数据缓存 DRAM 行大小：** DRAM 行的大小，以字节为单位。范围 = { 所有 2 的正整数幂 }。

11. **数据缓存 DRAM 数据速率：** DRAM 数据传输速率，单位为 MT/s。范围 = { 所有正整数值 }。

12. **数据缓存 DRAM 数据突发大小：** 在一次 DRAM 突发中传输的字节数（取决于 DRAM 芯片的数量）。范围 = { 所有正整数值 }。
13. **数据缓存 DRAMtRCD：** 用于访问数据缓存中 DRAM 的时序参数 tRCD 的值，单位为纳秒。范围 = { 所有正整数值 }。
14. **数据缓存 DRAMtCL：** 用于访问数据缓存中 DRAM 的时序参数 tCL 的值，单位为纳秒。范围 = { 所有正整数值 }。
15. **数据缓存 DRAMtRP:** 用来访问数据缓存中 DRAM 的时序参数 tRP 的值（单位：纳秒）。范围 = { 所有正整数值 }。
16. **地址映射 :** 闪存翻译层（ FTL ）中实现的逻辑地址到物理地址的映射策略。范围 = { 页级别 , 混合 }。

17. **Ideal_Mapping_Table:** if mapping is ideal, table is enabled in which all address translations entries are always in CMT (i.e., CMT is infinite in size) and thus all adddress translation requests are always successful (i.e., all the mapping entries are found in the DRAM and there is no need to read mapping entries from flash)
18. **CMT_Capacity:** the size of the SRAM/DRAM space in bytes used to cache the address mapping table (Cached Mapping Table). Range = {all positive integer values}.
19. **CMT_Sharing_Mode:** the mode that determines how the entire CMT (Cached Mapping Table) space is shared among concurrently running flows when an NVMe host interface is used. Range = {SHARED, EQUAL_PARTITIONING}.
20. **Plane_Allocation_Scheme:** the scheme for plane allocation as defined in Tavakkol et al. [3]. Range = {CWDP, CWPD, CDWP, CDPW, CPWD, CPDW, WCDP, WCPD, WDCP, WDPC, WPCD, WPDC, DCWP, DCPW, DWCP, DWPC, DPCW, DPWC, PCWD, PCDW, PWCD, PWDC, PDCW, PDWC}
21. **Transaction_Scheduling_Policy:** the transaction scheduling policy that is used in the SSD back end. Range = {OUT_OF_ORDER as defined in the Sprinkler paper [2], PRIORITY_OUT_OF_ORDER which implements OUT_OF_ORDER and NVMe priorities}.
22. **Overprovisioning_Ratio:** the ratio of reserved storage space with respect to the available flash storage capacity. Range = {all positive double precision values}.
23. **GC_Exect_Threshold:** the threshold for starting Garbage Collection (GC). When the ratio of the free physical pages for a plane drops below this threshold, GC execution begins. Range = {all positive double precision values}.
24. **GC_Block_Selection_Policy:** the GC block selection policy. Range {GREEDY, RGA *(described in [4] and [5])*, RANDOM *(described in [4])*, RANDOM_P *(described in [4])*, RANDOM_PP *(described in [4])*, FIFO *(described in [6])*}.
25. **Use_Copyback_for_GC:** used in GC_and_WL_Unit_Page_Level to determine block_manager→Is_page_valid gc_write transaction
26. **Preemptible_GC_Enabled:** the toggle to enable pre-emptible GC (described in [7]). Range = {true, false}.
27. **GC_Hard_Threshold:** the threshold to stop pre-emptible GC execution (described in [7]). Range = {all possible positive double precision values less than GC_Exect_Threshold}.
28. **Dynamic_Wearleveling_Enabled:** the toggle to enable dynamic wear-leveling (described in [9]). Range = {true, false}.
29. **Static_Wearleveling_Enabled:** the toggle to enable static wear-leveling (described in [9]). Range = {all positive integer values}.
30. **Static_Wearleveling_Threshold:** the threshold for starting static wear-leveling (described in [9]). When the difference between the minimum and maximum erase count within a memory unit (e.g., plane in flash memory) drops below this threshold, static wear-leveling begins. Range = {true, false}.

17. **理想映射表：**如果映射是理想的，则启用映射表，其中所有地址转换条目始终在 CMT （即，CMT 的大小是无限的），因此所有地址转换请求始终成功（即，所有映射条目都在 DRAM 中，并且不需要从闪存读取映射条目)

18. **CMT 容量：**用于缓存地址映射表（缓存映射表）的 SRAM/DRAM 空间的字节数。范围 = { 所有正整数值 }。
19. **CMT 共享模式：**确定当使用 NVMe 主机接口时，整个 CMT （缓存映射表）空间如何在并发运行的流之间共享的模式。范围 = { 共享 , 等分 }。

20. **平面分配方案：**Tavakkol 等人定义的平面分配方案 [3]。范围 = {CWDP, CWPD, CDWP, CDPW, CPWD, CPDW, WCDP, WCPD, WDCP, WDPC, WPCD,WPDC, DCWP, DCPW, DWCP, DWPC, DPCW, DPWC, PCWD, PCDW, PWCD, PWDC, PDCW,PDWC}

21. **事务调度策略：** SSD 后端使用的事务调度策略。范围 = {OUT_OF_ORDER as defined in the Sprinkler paper [2],PRIORITY_OUT_OF_ORDER which implements OUT_OF_ORDER and NVMe priorities}。
22. **过预留比率：** 预留存储空间与可用闪存存储容量的比率。范围 = {all positive double precision values}。
23. **GC 执行阈值：** 开始垃圾回收（ GC ）的阈值。当某个平面的空闲物理页面的比率低于此阈值时，GC 执行开始。范围 = {all positive double precision values}。

24. **GC 块选择策略：** GC 块选择策略。范围 { 贪婪 , RGA *(describedin [4] and [5])*, 随机 *(described in [4])*, 随机 P *(described in [4])*, 随机 PP*(described in [4])*, FIFO *(described in [6])*}。

25. **GC_and_WL_Unit_Page_Level 使用拷回：** 在 GC_and_WL_Unit_Page_Level 中使用，以确定 block_manager→ 是否为页有效 gc 写入事务
26. **可抢占 GC 启用：** 启用抢占式 GC 的开关（在 [7] 中描述）。范围 ={true, false}。

27. **GC 硬阈值：** 停止可抢占 GC 执行（在 [7] 中描述）。范围 = { 所有小于 GC_Exect_Threshold 的双精度正数值 }。
28. **动态磨损均衡启用：** 启用动态磨损均衡的开关（在 [9] 中描述）。范围 = {true, false}。

29. **静态磨损均衡启用：** 启用静态磨损均衡的开关（在 [9] 中描述）。范围 = { 所有正整数值 }。

30. **静态磨损均衡阈值：** 静态磨损均衡的启动阈值（在 [9] 中描述）。当内存单元（例如闪存中的平面）内的最小和最大擦除计数之间的差值低于此阈值时，静态磨损均衡开始。范围 = {true, false}。

31. **Preferred_suspend_erase_time_for_read:** the reasonable time to suspend an ongoing flash erase operation in favor of a recently-queued read operation. Range = {all positive integer values}.
32. **Preferred_suspend_erase_time_for_write:** the reasonable time to suspend an ongoing flash erase operation in favor of a recently-queued read operation. Range = {all positive integer values}.
33. **Preferred_suspend_write_time_for_read:** the reasonable time to suspend an ongoing flash erase operation in favor of a recently-queued program operation. Range = {all positive integer values}.
34. **Flash_Channel_Count:** the number of flash channels in the SSD back end. Range = {all positive integer values}.
35. **Flash_Channel_Width:** the width of each flash channel in byte. Range = {all positive integer values}.
36. **Channel_Transfer_Rate:** the transfer rate of flash channels in the SSD back end in MT/s. Range = {all positive integer values}.
37. **Chip_No_Per_Channel:** the number of flash chips attached to each channel in the SSD back end. Range = {all positive integer values}.
38. **Flash_Comm_Protocol:** the Open NAND Flash Interface (ONFI) protocol used for data transfer over flash channels in the SSD back end. Range = {NVDDR2}.

NAND Flash

1. **Flash_Technology:** Range = {SLC, MLC, TLC}.
2. **CMD_Suspension_Support:** the type of suspend command support by flash chips. Range = {NONE, PROGRAM, PROGRAM_ERASE, ERASE}.
3. **Page_Read_Latency_LSB:** the latency of reading LSB bits of flash memory cells in nanoseconds. Range = {all positive integer values}.
4. **Page_Read_Latency_CSB:** the latency of reading CSB bits of flash memory cells in nanoseconds. Range = {all positive integer values}.
5. **Page_Read_Latency_MSB:** the latency of reading MSB bits of flash memory cells in nanoseconds. Range = {all positive integer values}.
6. **Page_Program_Latency_LSB:** the latency of programming LSB bits of flash memory cells in nanoseconds. Range = {all positive integer values}.
7. **Page_Program_Latency_CSB:** the latency of programming CSB bits of flash memory cells in nanoseconds. Range = {all positive integer values}.
8. **Page_Program_Latency_MSB:** the latency of programming MSB bits of flash memory cells in nanoseconds. Range = {all positive integer values}.

31. **优先暂停擦除时间用于读取：**在闪存擦除操作中暂停以优先处理最近排队读取操作的合理时间。范围 = { 所有正整数值 }。

32. **优先暂停擦除时间用于写入：**在闪存擦除操作中暂停以优先处理最近排队读取操作（写入）的合理时间。范围 = { 所有正整数值 }。

33. **优先暂停写入时间用于读取：**在闪存擦除操作中暂停以优先处理最近排队编程操作（读取）的合理时间。范围 = { 所有正整数值 }。

34. **闪存通道数量：**SSD 后端中的闪存通道数量。范围 = { 所有正整数值 }。

35. **闪存通道宽度：**每个闪存通道的字节宽度。范围 = { 所有正整数值 }。

36. **通道传输速率：**SSD 后端中闪存通道的传输速率，单位为 MT/s。范围 = { 所有正整数值 }。

37. **每个通道的闪存芯片数量：** SSD 后端每个通道连接的闪存芯片数量。范围 = { 所有正整数值 }。

38. **闪存通信协议：** SSD 后端闪存通道上用于数据传输的开源 NAND 闪存接口（ ONFI ）协议。范围 = {NVDDR2}。

NAND Flash

1. **闪存技术：** 范围 = {SLC, MLC, TLC}。
2. **命令暂停支持：** 闪存芯片支持的暂停命令类型。范围 ={NONE, PROGRAM, PROGRAM_ERASE, ERASE}。
3. **页读取 LSB 位延迟：** 闪存单元读取 LSB 位的延迟，单位为纳秒。范围 = { 所有正整数值 }。
4. **页读取 CSB 位延迟：**闪存单元读取 CSB 位的延迟，单位为纳秒。范围 = { 所有正整数值 }。
5. **页读取延迟 _MSB：**闪存单元读取 MSB 位的延迟，单位为纳秒。范围 = { 所有正整数值 }。
6. **页编程延迟 _LSB：**闪存单元编程 LSB 位的延迟，单位为纳秒。范围 = { 所有正整数值 }。

7. **页编程延迟 _CSB：**闪存单元编程 CSB 位的延迟，单位为纳秒。范围 = { 所有正整数值 }。

8. **页编程延迟 _MSB：**闪存单元编程 MSB 位的延迟，单位为纳秒。范围 = { 所有正整数值 }。

9. **Block_Erase_Latency:** the latency of erasing a flash block in nanoseconds. Range = {all positive integer values}.
10. **Block_PE_Cycles_Limit:** the PE limit of each flash block. Range = {all positive integer values}.
11. **Suspend_Erase_Time:** the time taken to suspend an ongoing erase operation in nanoseconds. Range = {all positive integer values}.
12. **Suspend_Program_Time:** the time taken to suspend an ongoing program operation in nanoseconds. Range = {all positive integer values}.
13. **Die_No_Per_Chip:** the number of dies in each flash chip. Range = {all positive integer values}.
14. **Plane_No_Per_Die:** the number of planes in each die. Range = {all positive integer values}.
15. **Block_No_Per_Plane:** the number of flash blocks in each plane. Range = {all positive integer values}.
16. **Page_No_Per_Block:** the number of physical pages in each flash block. Range = {all positive integer values}.
17. **Page_Capacity:** the size of each physical flash page in bytes. Range = {all positive integer values}.
18. **Page_Metadat_Capacity:** the size of the metadata area of each physical flash page in bytes. Range = {all positive integer values}.

MQSim Workload Definition

You can define your preferred set of workloads in the XML format. If the specified workload definition file does not exist, MQSim will create a sample workload definition file in XML format for you (i.e., workload.xml). Here is the explanation of the XML attributes and tags for the workload definition file:

1. The entire workload definitions should be embedded within <MQSim_IO_Scenarios> </MQSim_IO_Scenarios> tags. You can define different sets of *I/O scenarios* within these tags. MQSim simulates each I/O scenario separately.
2. We call a set of workloads that should be executed together, an *I/O scenario*. An I/O scenario is defined within the <IO_Scenario></IO_Scenario> tags. For example, two different I/O scenarios are defined in the workload definition file in the following way:

9. **块擦除延迟：**闪存块擦除的延迟，单位为纳秒。范围 = { 所有正值 }。
10. **块 PE 周期限制：**每个闪存块的 PE 限制。范围 = { 所有正 lues}.
11. **暂停擦除时间：**暂停正在进行的擦除操作所需的时间，单位为纳秒。范围 = { 所有正整数值 }。
12. **Suspend_Program_Time:** the time taken to suspend an ongoing program operation in nanoseconds. Range = {all positive integer values}.
13. **每芯片 die 数量：**每个闪存芯片中的 die 数量。范围 = { 所有正整数值 } ues}。
14. **每 die 平面数量：**每个 die 中的平面数量。范围 = { 所有正整数值 } ues}.
15. **每平面块数量：**每个平面中的闪存块数量。范围 = { 所有正整数值 }。 ger
16. **每块页数量：**每个闪存块中的物理页数量。范围 = { 所有正整数值 }。 积极
17. 页面容量：每个物理闪存页的大小（以字节为单位）。范围 = { 所有正整数值 }。
18. 页面元数据容量：每个物理闪存页面的元数据区域大小（以字节为单位）。范围 = { 所有正整数值 }。

MQSim 工作负载定义

您可以在 XML 格式中定义您所需的工作负载集。如果指定的工 作负载定义文件不存在，MQSim 将为您创建一个示例工作负载定义文件（即 workload.xml ）。以下是工作负载定义文件的 XML 属性和标签说明：

1. 整个工作负载定义应该嵌入在 <MQSim_IO_Scenarios></MQSim_IO_Scenarios> 标签中。您可以在这些标签中定义不同的 I/O 场景 集合。MQSim 会分别模拟每个 I/O 场景。
2. 我们将一组应一起执行的工作负载称为 I/O 场景。I/O 场景是在 <IO_Scenario></IO_Scenario> 标签内定义。例如，工作负载定义文件中定义了两个不同的 I/O 场景，如下所示：

```
<MQSim_IO_Scenarios>
        <IO_Scenario>
        .............
        </IO_Scenario>
        <IO_Scenario>
        .............
        </IO_Scenario>
</MQSim_IO_Scenarios>
```

For each I/O scenario, MQSim 1) rebuilds the Host and SSD Drive model and executes the scenario to completion, and 2) creates an output file and writes the simulation results to it. For the example mentioned above, MQSim builds the Host and SSD Drive models twice, executes the first and second I/O scenarios, and finally writes the execution results into the workload_scenario_1.xml and workload_scenario_2.xml files, respectively.

You can define up to 8 different workloads within each IO_Scenario tag. Each workload could either be a disk trace file that has already been collected on a real system or a synthetic stream of I/O requests that are generated by MQSim's request generator.

Defining a Trace-based Workload

You can define a trace-based workload for MQSim, using the <IO_Flow_Parameter_Set_Trace_Based> XML tag. Currently, MQSim can execute ASCII disk traces define in [8] in which each line of the trace file has the following format:
1.Request_Arrival_Time 2.Device_Number 3.Starting_Logical_Sector_Address
4.Request_Size_In_Sectors 5.Type_of_Requests[0 for write, 1 for read]

The following parameters are used to define a trace-based workload:

1. **Priority_Class:** the priority class of the I/O queue associated with this I/O request. Range = {URGENT, HIGH, MEDIUM, LOW}.
2. **Device_Level_Data_Caching_Mode:** the type of on-device data caching for this flow. Range= {WRITE_CACHE, READ_CACHE, WRITE_READ_CACHE, TURNED_OFF}. If the caching mechanism mentioned above is set to SIMPLE, then only WRITE_CACHE and TURNED_OFF modes could be used.
3. **Channel_IDs:** a comma-separated list of channel IDs that are allocated to this workload. This list is used for resource partitioning. If there are C channels in the SSD (defined in the SSD configuration file), then the channel ID list should include values in the range 0 to C-1. If no resource partitioning is required, then all workloads should have channel IDs 0 to C-1.

```
<MQSim_IO_Scenarios><
IO_Scenario>.............</
IO_Scenario><
IO_Scenario>.............</
IO_Scenario></
MQSim_IO_Scenarios>
```

对于每个 I/O 场景，MQSim 1) 重建主机和 SSD 驱动器模型并执行场景直至完成，2) 创建输出文件并将仿真结果写入其中。对于上述示例，MQSim 两次构建主机和 SSD 驱动器模型，执行第一个和第二个 I/O 场景，最后将执行结果分别写入 workload_scenario_1.xml 和 workload_scenario_2.xml 文件。

您可以在每个 IO_Scenario 标签内定义最多 8 个不同的工作负载。每个工作负载要么是已经在一个真实系统上收集的磁盘跟踪文件，要么是由 MQSim 的请求生成器生成的合成 I/O 请求流。

定义基于跟踪的工作负载

您可以使用基于跟踪的工作负载来定义 MQSim。
<IO_Flow_Parameter_Set_Trace_Based> XML 标签。目前，MQSim 可以执行 ASCII 磁盘跟踪，这些跟踪定义在 [8] 中，其中每个跟踪文件的行具有以下格式：1. 请求到达时间 2. 设备编号 3. 起始逻辑扇区地址
4. 请求大小（扇区） 5. 请求类型 [0 对于写入，为 1 ；对于读取，为]

以下参数用于定义基于跟踪的工作负载：

1. **优先级类别：**与该 I/O 请求关联的 I/O 队列的优先级类别。范围 ={ 紧急、高、中、低 }。

2. **设备级数据缓存模式：**此流设备的类型数据缓存。范围 ={ 写入缓存、读取缓存、写入读取缓存、已关闭 }。如果上述缓存机制设置为 SIMPLE，则只能使用 WRITE_CACHE 和 TURNED_OFF 模式。

3. **通道 ID：**分配给此工作负载的通道 ID 的逗号分隔列表。此列表用于资源分区。如果在 SSD 中有 C 个通道（在 SSD 配置文件中定义），则通道 ID 列表应包括 0 到 C-1 范围内的值。如果不需要资源分区，则所有工作负载都应具有 0 到 C-1 的通道 ID。

4. **Chip_IDs:** a comma-separated list of chip IDs that are allocated to this workload. This list is used for resource partitioning. If there are W chips in each channel (defined in the SSD configuration file), then the chip ID list should include values in the range 0 to W-1. If no resource partitioning is required, then all workloads should have chip IDs 0 to W-1.
5. **Die_IDs:** a comma-separated list of chip IDs that are allocated to this workload. This list is used for resource partitioning. If there are D dies in each flash chip (defined in the SSD configuration file), then the die ID list should include values in the range 0 to D-1. If no resource partitioning is required, then all workloads should have die IDs 0 to D-1.
6. **Plane_IDs:** a comma-separated list of plane IDs that are allocated to this workload. This list is used for resource partitioning. If there are P planes in each die (defined in the SSD configuration file), then the plane ID list should include values in the range 0 to P-1. If no resource partitioning is required, then all workloads should have plane IDs 0 to P-1.
7. **Initial_Occupancy_Percentage:** the percentage of the storage space (i.e., logical pages) that is filled during preconditioning. Range = {all integer values in the range 1 to 100}.
8. **File_Path:** the relative/absolute path to the input trace file.
9. **Percentage_To_Be_Executed:** the percentage of requests in the input trace file that should be executed. Range = {all integer values in the range 1 to 100}.
10. **Relay_Count:** the number of times that the trace execution should be repeated. Range = {all positive integer values}.
11. **Time_Unit:** the unit of arrival times in the input trace file. Range = {PICOSECOND, NANOSECOND, MICROSECOND}

Defining a Synthetic Workload

You can define a synthetic workload for MQSim, using the <IO_Flow_Parameter_Set_Synthetic> XML tag.

The following parameters are used to define a trace-based workload:

1. **Priority_Class:** same as trace-based parameters mentioned above.
2. **Device_Level_Data_Caching_Mode:** same as trace-based parameters mentioned above.
3. **Channel_IDs:** same as trace-based parameters mentioned above.
4. **Chip_IDs:** same as trace-based parameters mentioned above.
5. **Die_IDs:** same as trace-based parameters mentioned above.
6. **Plane_IDs:** same as trace-based parameters mentioned above.
7. **Initial_Occupancy_Percentage:** same as trace-based parameters mentioned above.
8. **Working_Set_Percentage:** the percentage of available logical storage space that is accessed by generated requests. Range = {all integer values in the range 1 to 100}.

4. **芯片 ID：** a comma-separated list of 芯片 ID that are allocated to this workload. This list is used for resource partitioning. If there are W 芯片 in each channel (defined in the SSD 配置文件), then the 芯片 ID list should include values in the range 0 to W-1. If no resource partitioning is required, then all workloads should have 芯片 ID 0 to W-1.
5. **Die_IDs：** a comma-separated list of 芯片 ID that are allocated to this workload. This list i s used for resource partitioning. If there are D dies in each 闪存 chip (defined in the SSD 配置文件), then the die ID list should include values in the range 0 to D-1. If no resource partitioning is required, then all workloads should have die IDs 0 to D-1.
6. **Plane_IDs：** a comma-separated list of 平面 IDs that are allocated to this workload. This list is used for resource partitioning. If there are P planes in each die (defined in the SSD 配置文件), then the plane ID list should include values in the range 0 to P-1. If no resource partitioning is required, then all workloads should have 平面 IDs 0 to P-1.
7. **初始占用百分比：**在预条件期间填充的存储空间（即逻辑页）的百分比。范围 = {1 到 100 范围内的所有整数值 }。
8. **文件路径：**输入跟踪文件的相对 / 绝对路径。
9. **执行百分比：**输入跟踪文件中应执行的请求的百分比。范围 = {1 到 100 范围内的所有整数值 }。
10. **重播次数：**跟踪执行应重复的次数。范围 = { 所有正整数值 }。
11. **时间单位：**输入跟踪文件中到达时间的单位。范围 = {PICOSECOND, NANOSECOND, MICROSECOND}。

定义合成工作负载

您可以使用 <IO_Flow_Parameter_Set_Synthetic> XML 标签为 MQSim 定义一个合成工作负载。

以下参数用于定义基于跟踪的工作负载：

1. **优先级类别：**与上述提到的基于跟踪的参数相同。 2. **设备级数据缓存模式：**与上述提到的基于跟踪的参数相同。 3. **通道 ID：**与上述提到的基于跟踪的参数相同。

4. **芯片 ID：**与上述提到的基于跟踪的参数相同。
5. **Die_IDs：**与上述提到的基于跟踪的参数相同。
6. **平面 ID：**与上述提到的基于跟踪的参数相同。
7. **初始占用百分比：**与上述提到的基于跟踪的参数相同的百分比

   生成的请求。范围 = {1 到 100 范围内的所有整数值 }。

9. **Synthetic_Generator_Type:** determines the way that the stream of requests is generated. Currently, there are two modes for generating consecutive requests, 1) based on the average bandwidth of I/O requests, or 2) based on the average depth of the I/O queue. Range = {BANDWIDTH, QUEUE_DEPTH}.
10. **Read_Percentage:** the ratio of read requests in the generated flow of I/O requests. Range = {all integer values in the range 1 to 100}.
11. **Address_Distribution:** the distribution pattern of addresses in the generated flow of I/O requests. Range = {STREAMING, RANDOM_UNIFORM, RANDOM_HOTCOLD, MIXED_STREAMING_RANDOM}.
12. **Percentage_of_Hot_Region:** if RANDOM_HOTCOLD is set for address distribution, then this parameter determines the ratio of the hot region with respect to the entire logical address space. Range = {all integer values in the range 1 to 100}.
13. **Generated_Aligned_Addresses:** the toggle to enable aligned address generation. Range = {true, false}.
14. **Address_Alignment_Unit:** the unit that all generated addresses must be aligned to in sectors (i.e. 512 bytes). Range = {all positive integer values}.
15. **Request_Size_Distribution:** the distribution pattern of request sizes in the generated flow of I/O requests. Range = {FIXED, NORMAL}.
16. **Average_Request_Size:** average size of generated I/O requests in sectors (i.e. 512 bytes). Range = {all positive integer values}.
17. **Variance_Request_Size:** if the request size distribution is set to NORMAL, then this parameter determines the variance of I/O request sizes in sectors. Range = {all non-negative integer values}.
18. **Seed:** the seed value that is used for random number generation. Range = {all positive integer values}.
19. **Average_No_of_Reqs_in_Queue:** average number of I/O requests enqueued in the host-side I/O queue (i.e., the intensity of the generated flow). This parameter is used in QUEUE_DEPTH mode of request generation. Range = {all positive integer values}.
20. **Bandwidth:** the average bandwidth of I/O requests (i.e., the intensity of the generated flow) in bytes per seconds. MQSim uses this parameter in BANDWIDTH mode of request generation.
21. **Stop_Time:** defines when to stop generating I/O requests in nanoseconds.
22. **Total_Requests_To_Generate:** if Stop_Time is set to zero, then MQSim's request generator considers Total_Requests_To_Generate to decide when to stop generating I/O requests.

Analyze MQSim's XML Output

You can use an XML processor to easily read and analyze an MQSim output file. For example, you can open an MQSim output file in MS Excel. Then, MS Excel shows a set of options and you should choose "Use the XML Source task pane". The XML file is processed in MS Excel and a task pane is

9. **合成生成器类型：**决定了请求流生成的方式。目前，生成连续请求有两种模式，1) 基于 I/O 请求的平均带宽，或 2) 基于 I/O 队列的平均深度。范围 ={ 带宽 , 队列深度 }。
10. **读取百分比：**生成的 I/O 请求流中读取请求的比例。范围 = {1 到 100 范围内的所有整数值 }。
11. **地址分布：**生成的 I/O 请求流中地址的分布模式。范围 = { 流式 , 随机均匀 , 随机冷热 , 混合流式随机 }。
12. **热区域百分比：**如果地址分布设置为随机冷热，则此参数决定了热区域在整个逻辑地址空间中的比例。范围 = {1 到 100 范围内的所有整数值 }。
13. **生成对齐地址：**启用对齐地址生成的开关。范围 ={true, false}。
14. **地址对齐单位：**所有生成的地址必须对齐到的单位，以扇区（即 512 字节）为单位。范围 = { 所有正整数值 }。
15. **请求大小分布：**生成的 I/O 请求流中请求大小的分布模式。范围 = {FIXED, NORMAL}。
16. **平均请求大小：**生成的 I/O 请求的平均大小，以扇区（即 512 字节）为单位。范围 = { 所有正整数值 }。
17. **请求大小方差：**如果请求大小分布设置为 NORMAL，则此参数决定 I/O 请求大小的方差，以扇区为单位。范围 = { 所有非负整数值 }。
18. **种子值：**用于随机数生成的种子值。范围 = { 所有正整数值 }。
19. **队列中平均请求数：**主机端 I/O 队列中平均排队 I/O 请求数（即生成的流强度）。此参数用于请求生成的 QUEUE_DEPTH 模式。范围 = { 所有正整数值 }。
20. **带宽：** I/O 请求的平均带宽（即生成的流量强度），单位为每秒字节数。MQSim 在请求生成的 BANDWIDTH 模式下使用此参数。
21. **停止时间：** 定义在纳秒级别停止生成 I/O 请求的时间。
22. **生成总请求量：** 如果停止时间设置为零，则 MQSim 的请求生成器会考虑生成总请求量来决定何时停止生成 I/O 请求。

Analyze MQSim's XML Output

您可以使用 XML 处理器轻松读取和分析 MQSim 输出文件。例如，您可以在 MS Excel 中打开一个 MQSim 输出文件。然后，MS Excel 会显示一组选项，您应该选择 “ 使用 XML 源任务窗格 ”。XML 文件在 MS Excel 中被处理，并显示一个任务窗格

shown with all output parameters listed in it. In the task pane on the right, you see different types of statistics available in the MQSim's output file. To read the value of a parameter, you should:

1. Drag and drop that parameter from the task source pane to the Excel sheet.,
2. Right click on the cell that you have dropped the parameter and select *XML* > *Refresh XML Data* from the drop-down menue.

The parameters used to define the output file of the simulator are divided into categories:

Host

For each defined IO_Flow, the following parameters are shown:

1. **Name:** The name of the IO flow, e.g. Host.IO_Flow.Synth.No_0
2. **Request_Count:** The total number of requests from this IO_flow.
3. **Read_Request_Count:** The total number of read requests from this IO_flow.
4. **Write_Request_Count:** The total number of write requests from this IO_flow.
5. **IOPS:** The number of IO operations per second, i.e. how many requests are served per second.
6. **IOPS_Read:** The number of read IO operations per second.
7. **IOPS_Write:** The number of write IO operations per second.
8. **Bytes_Transferred:** The total number of data bytes transferred across the interface.
9. **Bytes_Transferred_Read:** The total number of data bytes read from the SSD Device.
10. **Bytes_Transferred_write:** The total number of data bytes written to the SSD Device.
11. **Bandwidth:** The total bandwidth delivered by the SSD Device in bytes per second.
12. **Bandwidth_Read:** The total read bandwidth delivered by the SSD Device in bytes per second.
13. **Bandwidth_Write:** The total write bandwidth delivered by the SSD Device in bytes per second.
14. **Device_Response_Time:** The average SSD device response time for a request, in nanoseconds. This is defined as the time between enqueueing the request in the I/O submission queue, and removing it from the I/O completion queue.
15. **Min_Device_Response_Time:** The minimum SSD device response time for a request, in nanoseconds.
16. **Max_Device_Response_Time:** The maximum SSD device response time for a request, in nanoseconds.
17. **End_to_End_Request_Delay:** The average delay between generating an I/O request and receiving a corresponding answer. This is defined as the difference between the request arrival time, and its removal time from the I/O completion queue. Note that the request arrival_time is the same as the request enqueue_time, when using the multi-queue properties of NVMe drives.
18. **Min_End_to_End_Request_Delay:** The minimum end-to-end request delay.
19. **Max_End_to_End_Request_Delay:** The maximum end-to-end request delay.

其中列出了所有输出参数。在右侧的任务窗格中，您可以看到 MQSim 输出文件中可用的不同类型的统计数据。要读取参数的值，您应该：

1. 将该参数从任务源窗格拖放到 Excel 工作表。, 2. 右键单击您放置参数的单元格，并从下拉菜单中选择 <code>XML</code> > <code> 刷新 XML 数据 </code>。

用于定义模拟器输出文件的参数按类别划分：

Host

对于每个定义的 IO 流，显示以下参数：

<code>1. </code><code>Name:</code><code> IO 流的名称，例如 Host.IO_Flow.Synth.No_0</code>

2. **Request_Count:** The total number of requests from this IO_flow.

<code>3. </code><code>Read_Request_Count:</code><code> 该 IO 流的总读取请求数。</code>

4. **Write_Request_Count:** The total number of write requests from this IO_flow.
5. **IOPS:** The number of IO operations per second, i.e. how many requests are served per second.
6. **读取 IOPS：** 每秒读取的 IO 操作次数。
7. **写入 IOPS：** 每秒写入的 IO 操作次数。
8. **Bytes_Transferred:** The total number of data bytes transferred across the interface.
9. **读取传输字节数：** 从 SSD D 读取的总数据字节数 evice.
10. **写入传输字节数：** 写入到 SSD Dev 的总数据字节数 ice.
11. **带宽：** SSD 设备每秒提供的总带宽（字节） .
12. **读取带宽：** SSD 设备每秒提供的总读取带宽（字节） ond.
13. **写入带宽：** SSD 设备每秒提供的总写入带宽（字节） ond.
14. **设备响应时间：** SSD 设备请求的平均响应时间，单位为 na纳秒。这定义为在 I/O 提交队列中入队请求和从 I/O 完成队列中移除请求之间的时间。 eue，和从 I/O 完成队列中移除它。
15. **最小设备响应时间：** 请求的最小 SSD 设备响应时间，以纳秒为单位。
16. **最大设备响应时间：** 请求的最大 SSD 设备响应时间，以纳秒为单位。
17. **端到端请求延迟：** 生成 I/O 请求和接收相应答复之间的平均延迟。这定义为请求到达时间与其从 I/O 完成队列中移除时间之间的差值。请注意，在使用 NVMe 驱动器的多队列特性时，请求到达时间与请求入队时间相同。
18. **最小端到端请求延迟：** 最小端到端请求延迟。
19. **最大端到端请求延迟：** 最大端到端请求延迟。

SSDDevice

The output parameters in the SSDDevice category contain values for:

1. Average transaction times at a lower abstraction level (SSDDevice.IO_Stream)
2. Statistics for the flash transaction layer (FTL)
3. Statistics for each queue in the SSD's internal flash Transaction Scheduling Unit (TSU): In the TSU exists a User_Read_TR_Queue, a User_Write_TR_Queue, a Mapping_Read_TR_Queue, a Mapping_Write_TR_Queue, a GC_Read_TR_Queue, a GC_Write_TR_queue, a GC_Erase_TR_Queue for each combination of channel and package.
4. For each package: the fraction of time in the exclusive memory command execution, exclusive data transfer, overlapped memory command execution and data transfer, and idle mode.

SSD 设备

SSD 设备类别中的输出参数包含以下值的：

1. 较低抽象级别的事务时间（SSDDevice.IO_Stream）2. 闪存事务层（FTL）的统计数据 3. SSD 内部闪存事务调度单元（TSU）中每个队列的统计数据：TSU 中存在一个 User_Read_TR_Queue 、一个 User_Write_TR_Queue 、一个 Mapping_Read_TR_Queue 、一个 Mapping_Write_TR_Queue 、一个 GC_Read_TR_Queue 、一个 GC_Write_TR_queue 以及一个 GC_Erase_TR_Queue，每个通道和封装组合对应一个。 4. 对于每个封装：独占内存命令执行、独占数据传输、重叠内存命令执行和数据传输以及空闲模式的时间比例。

References

[1] A. Tavakkol et al., "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices," FAST, pp. 49 - 66, 2018.

[2] M. Jung and M. T. Kandemir, "Sprinkler: Maximizing Resource Utilization in Many-chip Solid State Disks," HPCA, pp. 524-535, 2014.

[3] A. Tavakkol et al., "Performance Evaluation of Dynamic Page Allocation Strategies in SSDs," ACM TOMPECS, pp. 7:1--7:33, 2016.

[4] B. Van Houdt, "A Mean Field Model for a Class of Garbage Collection Algorithms in Flash-based Solid State Drives," SIGMETRICS, pp. 191-202, 2013.

[5] Y. Li et al., "Stochastic Modeling of Large-Scale Solid-State Storage Systems: Analysis, Design Tradeoffs and Optimization," SIGMETRICS, pp. 179-190, 2013.

[6] P. Desnoyers, "Analytic Modeling of SSD Write Performance", SYSTOR, pp. 12:1-12:10, 2012.

[7] J. Lee et al., "Preemptible I/O Scheduling of Garbage Collection for Solid State Drives," Vol. 32, No. 2, pp. 247-260, 2013.

[8] J. S. Bucy et al., "The DiskSim Simulation Environment Version 4.0 Reference Manual", CMU Tech Rep. CMU-PDL-08-101, 2008.

[9] Micron Technology, Inc., "Wear Leveling in NAND Flash Memory", Application Note AN1822, 2010.

参考文献

[1] A. Tavakkol 等人，“MQSim: 一个用于实现现代多队列 SSD 设备现实研究的框架 ,” FAST, 第 49-66 页 , 2018 年。

[2] M. Jung 和 M. T. Kandemir, "Sprinkler: 在多芯片固态盘中最大化资源利用率 ," HPCA, pp. 524-535, 2014.

[3] A. Tavakkol 等人 , "SSD 中动态页面分配策略的性能评估 ," ACM TOMPECS, pp. 7:1--7:33, 2016.

[4] B. Van Houdt, " 闪存固态驱动器中垃圾回收算法的均值场模型 ," SIGMETRICS, pp. 191-202, 2013.

[5] Y. Li 等人 , " 大规模固态存储系统的随机建模：分析、设计权衡和优化 ," SIGMETRICS, pp. 179-190, 2013.

[6] P. Desnoyers, "SSD 写入性能的解析建模 ", SYSTOR, pp. 12:1-12:10, 2012.

[7] J. Lee 等人，“ 固态硬盘的垃圾回收可抢占式 I/O 调度 ”，第 32 卷，第 2 期，第 247-260 页，2013 年。

[8] J. S. Bucy 等人，“DiskSim 仿真环境版本 4.0 参考手册 ”，CMU 技术报告 CMU-PDL-08-101，2008。

美光科技公司，“NAND 闪存磨损均衡 ”，应用笔记 AN1822，2010 年。