上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

实验四 StratoVirt 构建及性能对比

522031910557　冯海桐

2025 年 1 月 12 日

1　调研部分

1. 计算机启动时，BIOS 会进行硬件初始化和自检，然后会查找并加载引导加载程序。BIOS 执行完成后就到了跳转至内核入口点，此时内核将会执行相关函数，使得 CPU 进入保护模式。之后将会跳转到 32 位内核入口点，正式进行 Linux 内核的启动流程，引导完成。
   在物理机上,E820 是一个可以探测硬件内存分布的硬件。E820 表中的每一项都是一个 E820Entry 数据结构，表示一段内存空间，包含了起始地址、结束地址和类型。
2. epoll_create 用于创建一个 epoll 实例，在内核中开辟管理事件的空间并返回一个文件描述符（epfd）；epoll_ctl 则通过操作内核中维护的红黑树，实现向 epoll 实例中添加、修改或删除感兴趣的文件描述符及其对应事件；epoll_wait 用于等待事件发生，内核在有事件发生时会将对应文件描述符加入维护的就绪链表，epoll_wait 返回该链表，以便应用程序获取发生事件的文件描述符并处理。Epoll 有水平触发（LT）和边缘触发（ET）两种工作模式，LT 模式下，被监控文件描述符有可读写事件发生时，epoll_wait 会不断通知直至事件处理；ET 模式下，文件描述符状态变化时 epoll_wait 触发一次通知，后续除非状态再次变化否则不再通知。Epoll 通过红黑树管理文件描述符、就绪链表通知应用事件，适用于大量并发连接场景，能提升服务器性能与响应能力。
3. 构建一个 StratoVirt 平台的虚拟机，首先利用 PIO 的 0x3F8 端口捕捉串口输出。然后，利用 mmap 给客户机分配大型连续虚拟内存，在指令级别捕获敏感指令并由管理程序在一个称为“entrance-emulation” 的循环中进行模拟。为了设置多个内存映射，虚拟机构建多个具有不同槽位的 region，并使用 set_user_memory_region 接口更新内存映射。当 StratoVirt 捕捉到 mmio 指令时，会得到目标客户机的地址，通过比较客户地址区域找到对应的主机地址，并模拟 mmio 读/写指令。虚拟机应该保存虚拟 CPU 的寄存器和特殊寄存器的信息。为了引导 linux 内核，我们需要引导协议，其中 zero page 非常重要，它包含 kernel headers 和 E820 表。当我们捕捉 pio 指令时，指令将被传递到串行，串行程序将模拟寄存器的变化。为了防止粗粒度导致的控制权争夺，虚拟机利用 epoll 在另一个线程中使用事件监听器来检查 stdin 事件。
   与 QA.sh 对话的部分记录见附录。

2　实验目的

1. 理解 StratoVirt 平台的基本结构；

2. 理解 StratoVirt 平台的基本原理；
3. 对比 StratoVirt 与 QEMU 的虚拟化性能，并加以分析。

3 实验步骤

1. 启动并登陆到 L1 虚拟机中。
2. 进入 L1 虚拟机的 compare 目录下：`cd ~/labs/compare`。其中 stratovirt 文件夹对应 StratoVirt 方案，qemu 文件夹对应 QEMU 方案。对应文件夹下的 start.sh 中存放启动脚本，使用 sudo 权限可以直接启动对应文件夹下的对应镜像。
3. 在对应的文件夹下，添加 sysbench_src 安装包。
4. 启动 StratoVirt 方案虚拟机。
5. 记录内核启动时间，启动 10 次记录结果。
6. 在主机中使用 1G 内存时，使用 pmap 记录占用的内存数量，启动 10 次记录结果。
7. 安装 sysbench。
8. 使用 sysbench cpu 完成以下测试：线程数依次设置为 1，4，16，32，64，测试不同线程情况下的结果，每种类型的测试至少跑两次。
9. 使用 sysbench memory 完成以下测试：线程数为 4，内存块大小分别设置为 1k，2k，4k，8k，测试不同内存块大小的影响，分别测试顺序访问和随机访问两种情况，每组测试至少执行两次，每次至少 60 秒钟。
10. 使用 sysbench fileio 完成以下测试：线程数为 1，bs 设置为 4k，测试模拟单个队列读写的延迟；线程数为 32，bs 设置为 128k，测试吞吐量，跑满整个磁盘带宽；线程数为 32，bs 设置为 4k，测试 IOPS。每组测试至少要测试随机读、随机写、顺序读、顺序写四种情况，每种类型的测试至少跑两次，每次至少 60 秒钟。
11. 退出 L2 虚拟机。
12. 启动 QEMU 方案虚拟机。
13. 重复步骤 5~11。
14. 将以上实验的结果以表格形式记录在报告中，并分析 StratoVirt 相比 QEMU 的轻量化方面效果如何，结合设计思想、编程语言、实现方案等分析为何会有这样的效果。

4 实验结果

本次实验所使用的 CPU 类型为 Intel。

4.1 启动时间

StratoVirt 启动时间情况如表 1 所示，QEMU 启动时间情况如表 2所示。
StratoVirt 的 BootLoader 模块模块负责操作系统内核的启动引导，主要需要实现三件事：

- 按照 Linux Boot Protocol 设计 1MB 低地址虚拟机物理内存布局，根据启动所需的信息配置填充相应的数据结构；
- 将内核镜像文件和 initrd 写入虚拟机内存；

表 1 StratoVirt 启动时间情况

实验次数	内核启动时间	用户空间启动时间	总启动时间
1	1.036s	2min 2.438s	2min 3.474s
2	1.136s	2min 3.005s	2min 4.142s
3	1.031s	2min 2.352s	2min 3.383s
4	1.102s	2min 2.137s	2min 3.240s
5	0.983s	2min 2.358s	2min 3.342s
6	1.477s	2min 2.471s	2min 3.949s
7	1.056s	2min 2.346s	2min 3.402s
8	1.006s	2min 2.523s	2min 3.530s
9	1.033s	2min 2.465s	2min 3.498s
10	1.269s	2min 2.347s	2min 3.616s
平均值	1.113s	2min 2.444s	2min 3.558s

表 2 QEMU 启动时间情况

实验次数	内核启动时间	初始内存盘加载时间	用户空间启动时间	总启动时间
1	742ms	1.245s	5.428s	7.416s
2	742ms	1.137s	4.497s	6.377s
3	740ms	1.132s	4.472s	6.345s
4	733ms	1.142s	4.470s	6.346s
5	742ms	1.131s	4.474s	6.348s
6	748ms	1.140s	4.469s	6.358s
7	746ms	1.169s	5.432s	7.347s
8	759ms	1.143s	4.457s	6.360s
9	751ms	1.146s	4.479s	6.378s
10	745ms	1.154s	4.489s	6.389s
平均值	745ms	1.154s	4.667s	6.566s

表 3 StratoVirt 内存占用情况

次数	内存段总大小 (KB)	RSS (KB)	Dirty (KB)
1	1128036	126988	121076
2	1128036	127172	121276
3	1128036	127200	121268
4	1128036	126836	120956
5	1128036	127056	121100
6	1128036	126964	121056
7	1128036	127344	121408
8	1128036	127288	121480
9	1128036	125876	119964
10	1128036	127400	121444
平均值	1128036	127012	121103

表 4 QEMU 内存占用情况

次数	内存段总大小 (KB)	RSS (KB)	Dirty (KB)
1	1583704	517768	496676
2	1580620	517716	496724
3	1583704	513588	492664
4	1586788	515772	494656
5	1585760	513728	492664
6	1583704	513592	492648
7	1583704	497368	476192
8	1657436	497192	476228
9	1585760	513612	492660
10	1583704	513492	492612
平均值	1591488	511383	490372

- 将写入内存的数据信息传递给 CPU 模块用于寄存器设置。

Stratovirt 采用 BootLoader 模块，以跳过实模式直接进入保护模式的形式直接启动 Linux 内核，这样内核就能直接从保护模式的入口开始运行。但是实验结果表明，与传统的 QEMU 相比，StratoVirt 的内核启动时间要更慢一些。

4.2 内存占用

StratoVirt 内存占用情况如表 3 所示，QEMU 内存占用情况如表 4 所示。

实验结果表明：StratoVirt 的内存占用相对稳定且较低，RSS 和 Dirty 内存的值远低于 QEMU。QEMU 由于需要模拟复杂的硬件环境和保证较高的硬件兼容性，导致其内存占用明显高于 StratoVirt。

表 5 StratoVirt CPU 性能情况

线程数	CPU 速度 (事件数/s)	平均延迟 (ms)	最大延迟 (ms)	延迟第 95 百分位数 (ms)
1	1258.20	0.79	12.82	0.86
	1254.58	0.80	4.60	0.86
4	1235.02	3.24	29.77	12.75
	1253.86	3.19	28.85	12.75
16	1248.91	12.80	136.93	61.08
	1200.74	13.31	101.02	61.08
32	1210.12	26.40	277.35	125.52
	1247.04	25.61	225.85	125.52
64	1256.95	50.77	468.25	253.35
	1231.84	51.82	428.78	253.35

表 6 QEMU CPU 性能情况

线程数	CPU 速度 (事件数/s)	平均延迟 (ms)	最大延迟 (ms)	延迟第 95 百分位数 (ms)
1	1245.16	0.80	6.03	0.86
	1234.29	0.81	2.83	0.87
4	1231.22	3.25	12.45	6.91
	1227.97	3.25	31.2	6.79
16	1230.06	12.99	48.10	16.71
	1236.13	12.93	51.49	16.71
32	1241.69	25.74	95.29	33.12
	1236.82	25.83	104.57	33.12
64	1227.23	52.05	187.80	65.65
	1231.25	51.89	187.83	65.65

4.3 CPU 性能

为测试虚拟机 CPU 性能，在终端键入命令：

```
sysbench cpu --time=60 --threads=N run
```

其中 N 项为 1，4，16，32 或 64。

StratoVirt CPU 性能情况如表 5 所示，QEMU CPU 性能情况如表 6 所示。

实验结果表明：由于头歌平台的 CPU 是单核的，所以线程数对 CPU 的速度影响不大。StratoVirt 和 QEMU 的 CPU 速度和平均延迟相差不大，但是 StratoVirt 的最大延迟和延迟第 95 百分位数明显大于 QEMU，这可能是因为二者使用了不同的线程调度算法。QEMU 的算法更注重平均，而 StratoVirt 的算法更注重线程优先级。

4.4 内存性能

为测试虚拟机内存性能，在终端键入命令：

表 7 StratoVirt 内存性能情况

内存块大小 (KB)	传输速率 (MiB/s)			
	顺序访问		随机访问	
1	2051.73	2057.26	1169.54	1171.86
2	3660.01	3831.17	1569.86	1484.15
4	6373.66	6384.70	1909.92	1898.80
8	9701.11	9762.81	2087.73	2036.55

表 8 QEMU 内存性能情况

内存块大小 (KB)	传输速率 (MiB/s)			
	顺序访问		随机访问	
1	1786.34	1795.41	1092.63	1090.44
2	3312.75	3330.06	1511.13	1471.88
4	5713.65	5673.49	1844.80	1714.41
8	8798.78	9010.34	2114.52	1954.29

```
sysbench memory --time=60 --threads=4 --memory-total-size=1000G --memory-block-size=SIZE
--memory-access-mode=STRING run
```

其中 SIZE 项为 1k，2k，4k 或 8k，STRING 项为 seq 或 rnd。

StratoVirt 内存性能情况如表 7 所示，QEMU 内存性能情况如表 8 所示。

实验结果表明：StratoVirt 在顺序访问模式下的内存性能明显优于 QEMU，但是随机访问模式下二者相差不大。同时，StratoVirt 和 QEMU 都是顺序访问速度高于随机访问。

4.5 I/O 速度

为测试虚拟机 I/O 速度，首先创建一个新文件夹，并在终端键入命令：

```
sysbench fileio --file-total-size=20M prepare
```

用于生成测试文件。

然后，在终端键入命令：

```
sysbench fileio --time=60 --threads=N --file-block-size=SIZE --file-total-size=20M
--file-test-mode=STRING run
```

其中 N 项为 1 或 32，SIZE 项为 4k 或 128k，STRING 项为 rndrd，rndwr，seqrd 或 seqwr。

最后，在终端键入命令：

```
sysbench fileio cleanup
```

用于清除测试文件。

设置线程数为 1，块大小为 4k，测试单个队列读写的延迟。StratoVirt 的情况如表 9 所示，QEMU 的情况如表 10 所示。

设置线程数为 32，块大小为 128k，测试吞吐量。StratoVirt 和 QEMU 的情况如表 11 所示。

表 9 StratoVirt 单个队列读写的延迟

测试方式	平均延迟 (ms)	最大延迟 (ms)	延迟第 95 百分位数 (ms)
随机读	0.00	3.81	0.00
	0.00	4.11	0.00
随机写	0.22	8.74	0.63
	0.22	7.12	0.63
顺序读	0.00	4.07	0.00
	0.00	3.89	0.00
顺序写	0.10	7.93	0.16
	0.10	102.16	0.17

表 10 QEMU 单个队列读写的延迟

测试方式	平均延迟 (ms)	最大延迟 (ms)	延迟第 95 百分位数 (ms)
随机读	0.00	6.27	0.00
	0.00	1.70	0.00
随机写	0.19	13.74	0.61
	0.20	30.31	0.61
顺序读	0.00	2.09	0.00
	0.00	1.58	0.00
顺序写	0.06	25.19	0.08
	0.06	16.67	0.08

表 11 StratoVirt 与 QEMU 吞吐量

测试方式	吞吐量 (MiB/s)			
	StratoVirt		QEMU	
随机读	13656.75	13690.03	12037.26	11899.84
随机写	686.95	730.32	484.44	479.57
顺序读	12535.40	12550.58	10632.70	10618.56
顺序写	540.29	559.77	390.91	389.50

表 12 StratoVirt IOPS

测试方式	读取速率 (次/s)	写入速率 (次/s)	文件同步速率 (次/s)
随机读	982908.71	-	-
	1011316.83	-	-
随机写	-	9660.41	12431.76
	-	10348.89	13312.97
顺序读	1086956.68	-	-
	1102415.97	-	-
顺序写	-	10409.14	13390.13
	-	10366.80	13336.37

表 13 QEMU IOPS

测试方式	读取速率 (次/s)	写入速率 (次/s)	文件同步速率 (次/s)
随机读	847807.92	-	-
	856456.66	-	-
随机写	-	6324.14	8162.23
	-	7520.72	9693.21
顺序读	925970.18	-	-
	921396.72	-	-
顺序写	-	11929.15	15336.57
	-	12116.22	15575.85

设置线程数为 32，块大小为 4k，测试 IOPS。StratoVirt 的情况如表 12 所示，QEMU 的情况如表 13 所示。

实验结果表明：对于队列读写的延迟，QEMU 略优于 StratoVirt。而吞吐率上，StratoVirt 明显优于 QEMU。值得注意的是，这里出现了顺序读写慢于随机读写的情况，原因可能是硬盘本身支持并行读写，而顺序读写无法完全发挥该功能。对于 IOPS，StratoVirt 在随机读写和顺序读上具有优势，而 QEMU 在顺序写上做的更好。

5 遇到的问题及解决方案

在测试内存性能的时候，使用默认的数据总大小，即 100G，发现在内存块大小设置为 2k 的时候就跑不满 60s。所以我将数据总大小提升到 1000G，问题解决。

在测试 I/O 速度的时候，生成测试文件的时候磁盘空间不足。当时有两个办法：其一是扩大磁盘容量，其二是减少文件总大小。我选择了第二种，将总文件大小设置为 20M。但是怕会影响吞吐量，所以又提升到 40M，发现吞吐量相差无几，于是便采用了 20M 的总文件大小。

A 附录

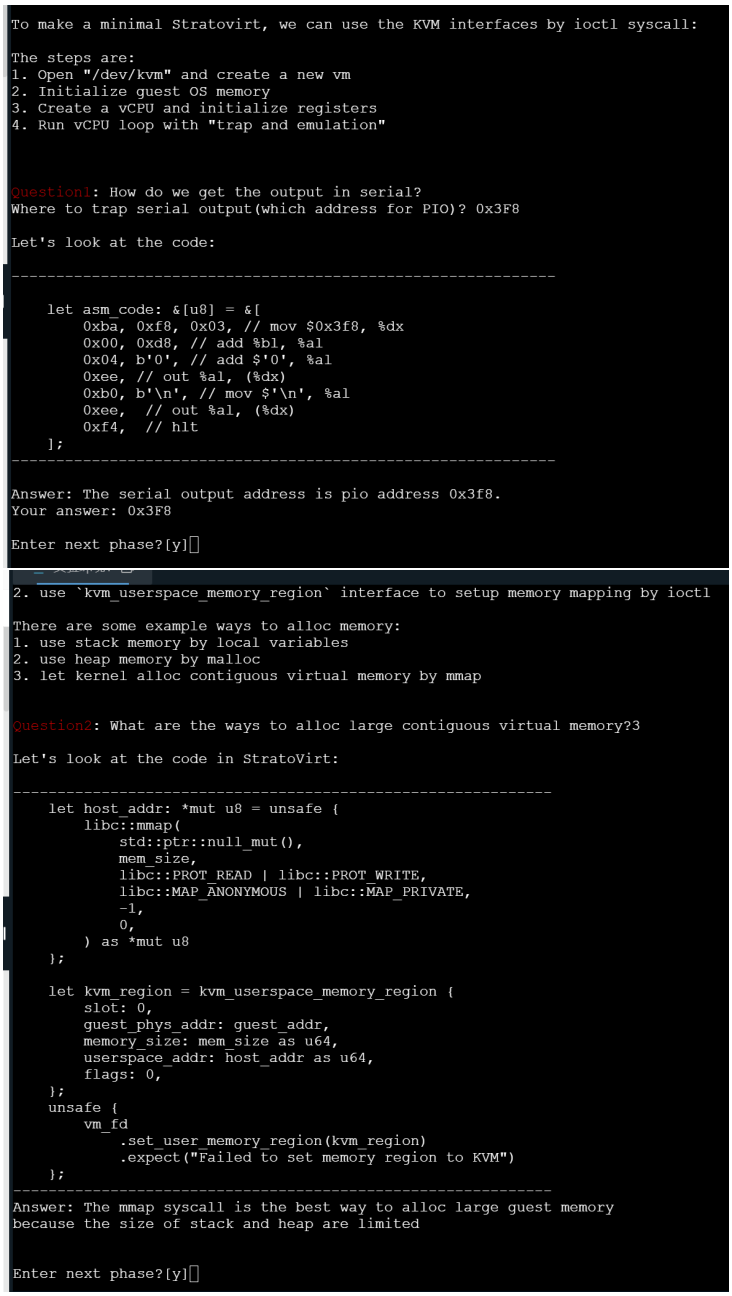

图 1 与 QA.sh 对话部分截图