# سیستم‌های عامل پیشرفته – تمرین سه

استاد: دکتر حسین اسدی

نیمسال دوم 401 - 402

محمدمهدی قره‌گوزلو – 401206836

پرسش شماره یک:

یک نکته که در سوال "یک" و "دو" وجود داشت محدودیتی است که حالت DIRECT_O برای نوشتن ایجاد می‌کند. بدین صورت که بافر مورد استفاده برای نوشتن باید با اندازه بلاک مورد استفاده فایل سیستم همخوانی داشته باشد. اصطلاحا باید Block Align باشند. اندازه بلاک را با استفاده از دستور ioctl() و پرچم BLKSSZGET می‌توان به دست آورد. البته از نسخه کرنل 2.6 به بعد استفاده از سایز 512 بایت به عنوان اندازه بلاک و Align کردن نسبت به آن به کفایت می‌کند. در غیر این صورت هنگام نوشتن اخطار Invalid Argument دریافت می‌شود و عملیات نوشتن به درستی انجام نمی‌شود.

نکته دیگر درباره drop_cache: این که بین خواندن(نوشتن) اول و دوم از drop_cache استفاده کنیم یا نکنیم منطقا نباید تاثیری در بهتر یا بدتر شدن نتایج داشته باشد. زیرا خواندن(نوشتن) دوم پرچم DIRECT_O را فعال کردن و از حافظه نهان استفاده نمی‌کند. یعنی ما چه حافظه نهان را خالی کرده باشیم چه نکرده باشیم نباید در عملکرد خواندن تاثیر بگذارد.

الف) با توجه به توضیحات سوال ترتیب و شرایط اجرا بدین صورت بوده است: خواندن بدون پرچم O_DIRECT، فلاش کردن حافظه نهان، و در نهایت خواندن با استفاده از پرچم O_DIRECT.

ب) با توجه به توضیحات سوال ترتیب و شرایط اجرا بدین صورت بوده است: خواندن بدون پرچم O_DIRECT، عدم فلاش کردن حافظه نهان، و در نهایت خواندن با استفاده از پرچم O_DIRECT.

| الف و ب | 1st Read | 2nd Read \| DIRECT |
|---|---|---|
| Drop Cache | 4.358521 s | 1.711244 s |
| No – Drop Cache | 4.469709 s | 2.203422 s |

تحلیل بنده با فرض درست بودن برنامه و اعداد بالا: استفاده از پرچم DIRECT در خواندن ترتیبی( Sequential Read) فایل بزرگ ظاهرا باعث بهبود عملکرد می‌شود. شاید بتوان اینطور توجیح کرد که چون Locality نداریم و فقط یک بار از روی هر بلاک می‌خوانیم، سعی در استفاده از حافظه نهان صرفا overhead ایجاد می‌کند و از طرفی عملکرد حافظه نهان را نیز دچار اختلال می‌کند. البته شاید بتوان سیاست Read Ahead را اعمال کرد، مطمین نیستم که آیا RA در اینجا استفاده شده یا خیر.

ج) ترتیب اجرا مانند قسمت "الف" و "ب" می‌باشد با این تفاوت که عمل نوشتن را انجام می‌دهیم.

| ج | 1st Write | 2nd Write \| DIRECT |
|---|---|---|
| Drop Cache | 26.76037 s | 31.19299 s |
| No – Drop Cache | 26.610925 s | 30.87353 s |

به نظر می‌رسد که تاثیر حافظه نهان در بهبود عملکرد در نوشتن محسوس است. در خواندن که وضعیت بدتر می‌شد. در نوشتن حداقل کمی بهبود مشاهده می‌شود.

## پرسش شماره دو:

| پرسش دو | 1st Run | 2nd Run | 3rd Run | 4th Run | 5th Run |
|---|---|---|---|---|---|
| Write \|Direct | 162291 µs | 157272 µs | 155450 µs | ~2 s | 168705 µs |
| Read \|Direct | 2051 µs | 2039 µs | 2206 µs | 2045 µs | 2949 µs |
| Write | 9095 µs | 7855 µs | 9431 µs | 8984 µs | 8328 µs |
| Read | 2463 µs | 2507 µs | 7349 µs | 2513 µs | 2135 µs |

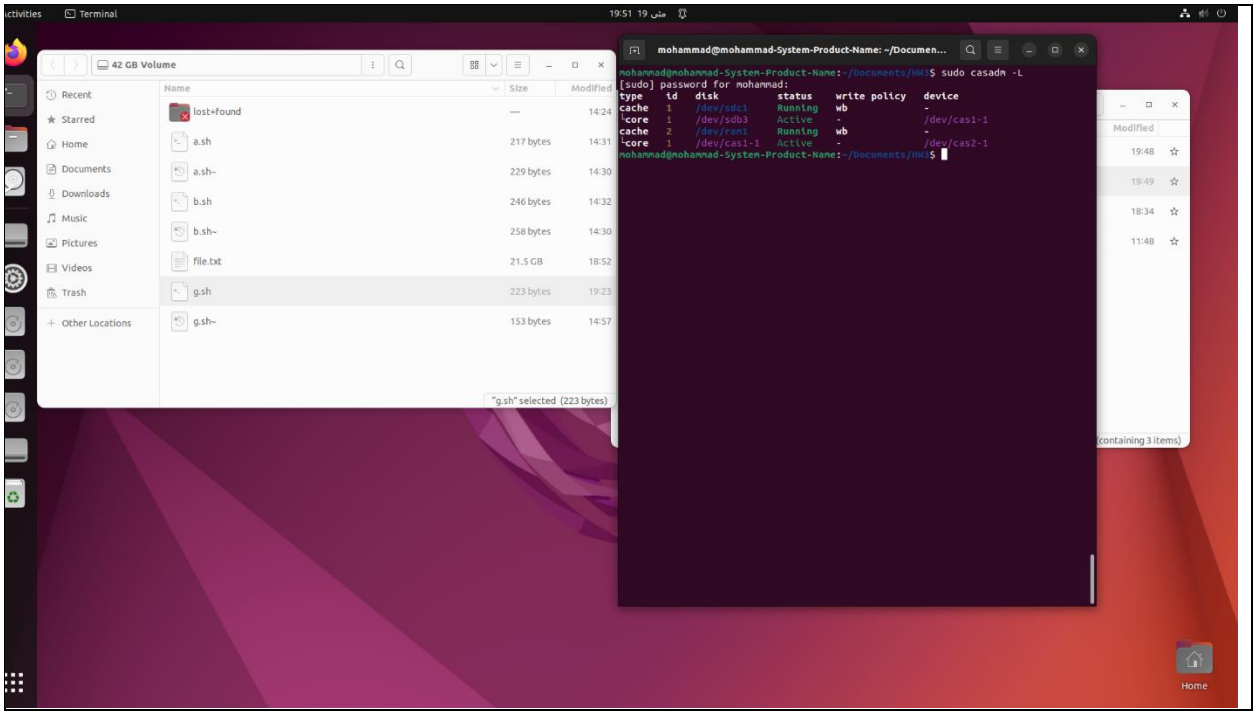
بین خواندن با استفاده از حافظه نهان و بدون حافظه نهان تفاوت چندانی مشاهد نمی‌شود، البته مانند سوال قبل در حالت DIRECT یعنی بدون حافظه نهان به نظر می‌رسد که وضعیت بهتری را تجربه می‌کنیم. کلا برای خواندن ترتیبی بهتر از حافظه نهان استفاده نشود چون وضعیت را بدتر می‌کند.

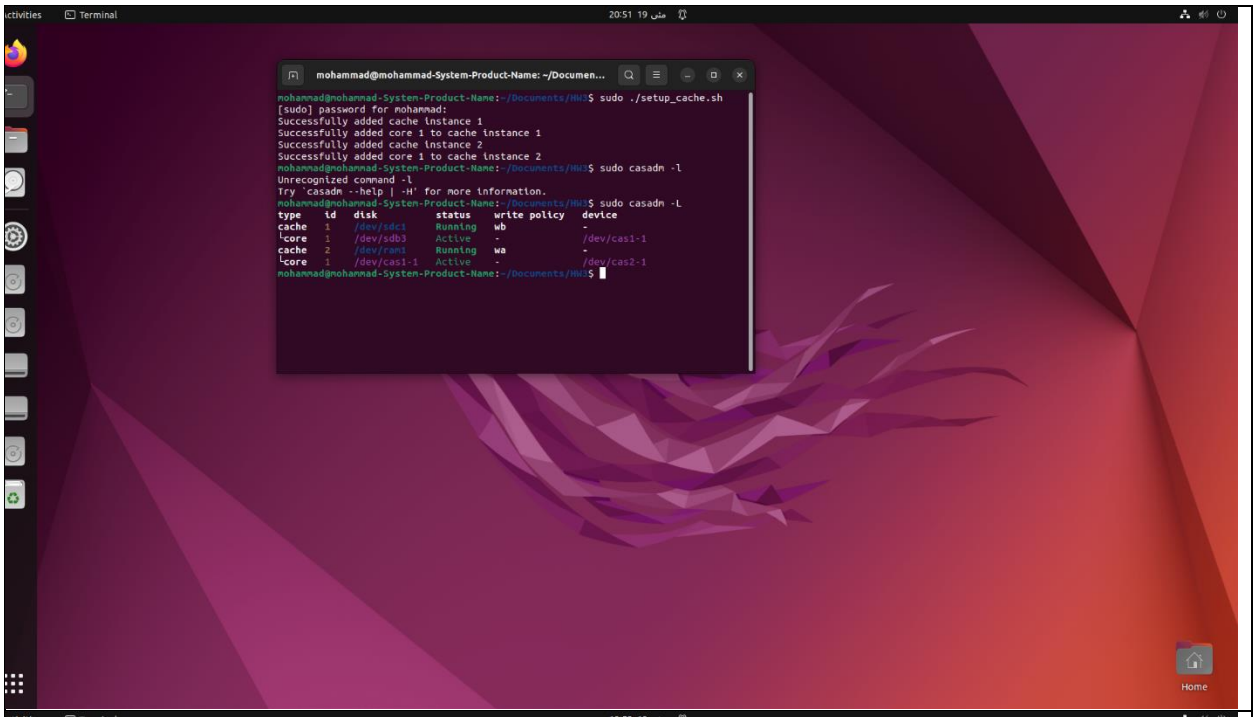
برای نوشتن اما تفاوت بسیار محسوس است. استفاده از حافظه نهان زمان نوشتن را قطعا بهبود می‌دهد.

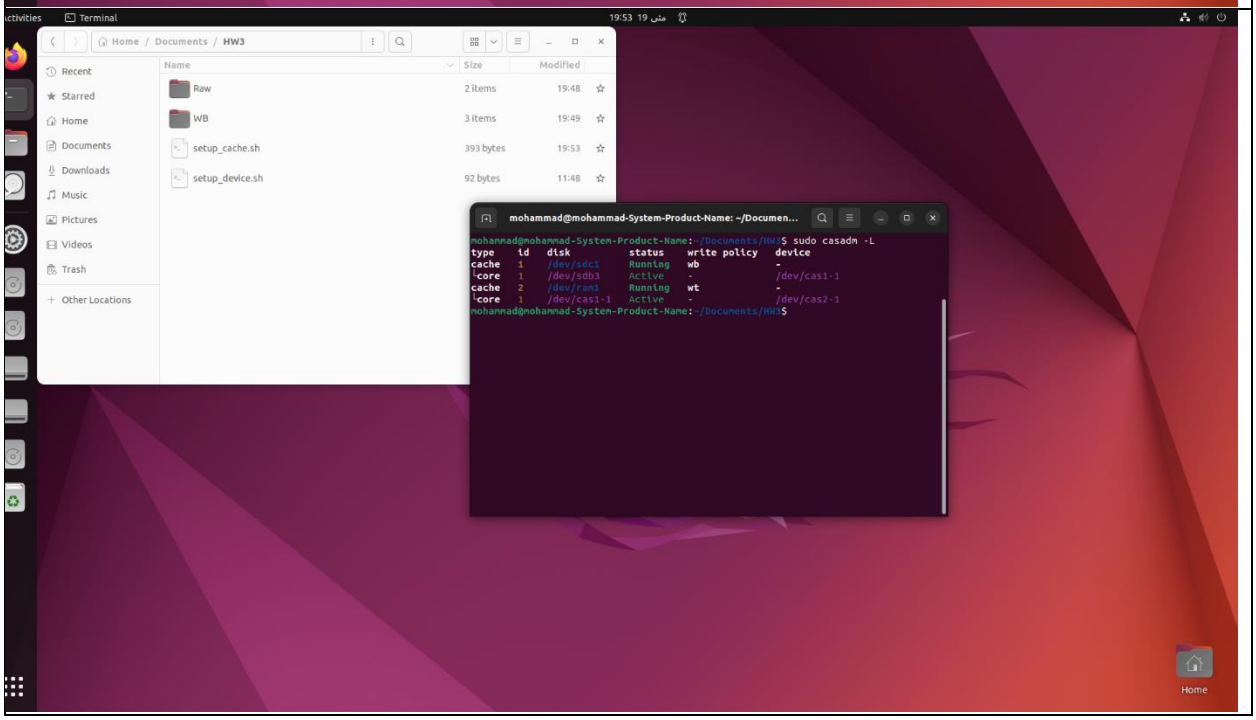
## پرسش شماره سه:

برای انجام دادن این سوال یک حافظه نهان دو سطحی ایجاد کردم که سطح اول آن حافظه DRAM با حجم یک گیگابایت و سطح دوم آن حافظه SSD با حجم 5 گیگابایت بود. برای حافظه جانبی نیز از HDD استفاده کردم.

نحوه کار بدین صورت انجام شد که سیستم عامل لینوکس را روی یک دیسک فیزیکی مستقل از سایر اجزا نصب کردم(sda) و (sdb)HDD مورد استفاده برای تست، همچنین (sdc)SSD حافظه نهان کاملا مستقل بودند. ابتدا با استفاده از SSD یک حافظه نهان ساخته سپس خود cas1-1 را به عنوان core device با حافظه Ram تبدیل به کش سطح یک کردم. اسکریپت‌های مربوطه را در دو فایل Bash پیوست شده می‌توانید مشاهده کنید. تصاویر زیر مربوط به اجرای دستور casadm –L هستند.






در هر سه حالت، سیاست سطح دو یعنی SSD را Write Back گذاشتم. برای سطح یک یعنی DRAM اما سیاست‌های Write Back و Write Through و Write Around را مورد آزمایش قرار دادم.

بارهای کاری ترتیبی Sequential همگی بدون اشکال خاصی روی تمامی سیاست‌ها اجرا شدند اما هنگام اجرای بار کاری تصادفی Random Read یکسری چالش پیش آمد که مانع از اجرای کامل و صحیح آن شد. اول اینکه بارکاری تصادفی با مشخصاتی که شما اعلام کردید روی HDD بدون هیچ گونه حافظه نهان بیش از یک هفته طول می‌کشد. بنده به مدت 210

دقیقه گذاشتم بلکه پایدار(stable) شود تا شاید زمان را کمتر تخمین بزند اما تفاوتی نکرد. یعنی انگار واقعا یک هفته به درازا می‌انجامد.

برای سیاست‌های WB و WT با تغییر فایل fio مربوطه و سبک‌تر کردن آن موفق به اجرای کامل بار تصادفی شدم. اما هنگام اجرا روی WA به مشکل برخودم. ظاهرا ساختار فایل سیستم روی درایو مربوطه کلا به هم ریخته بود و مجبور به فرمت و نصب دوباره فایل سیستم شدم. بعد از این کار بار کاری تصادفی را اجرا کردم که در کمال تعجب زیر یک دقیقه تمام شد. اما بعد از آن دوباره فایل سیستم دچار مشکل شد و حتی با فرمت و بدون کش و در حالت عادی نیر نتوانستم fio را اجرا کنم. فکر کنم کلا یک مشکلی برای دیسک مورد اشاره پیش آمد، مطمین نیستم البته.

فایل خروجی کامل اجرای fio روی سیاست‌های مختلف در پوشه‌هایی با نام‌های WB, WT, WA, RAW(Plain HDD) به پیوست ارسال می‌گردد.

| IOPS | Plain HDD | WB | WT | WA |
|---|---|---|---|---|
| Sequential Write | 11 | 10 | 7 | 12 |
| Sequential Read | 12 | 13 | 13 | 14 |
| Random Read | * | 3515 | 964 | 32k |

| Bandwidth | Plain HDD | WB | WT | WA |
|---|---|---|---|---|
| Sequential Write | 47 mb | 42 mb | 28 mb | 51 mb |
| Sequential Read | 49 mb | 55 mb | 55 mb | 56 mb |
| Random Read | * | 13 mb | 3858 kb | 126 mb |

| Exe Time | Plain HDD | WB | WT | WA |
|---|---|---|---|---|
| Sequential Write | 7m, 14s | 8m, 8s | 11m, 58s | 6m, 37s |
| Sequential Read | 6m, 59s | 6m, 20s | 6m, 11s | 6m, 7s |
| Random Read | 210m to no avail | 24m, 52s | 22m, 39s | 42s |

همانطور که در جداول بالا مشاهده می‌کنید، به صورت کلی استفاده از حافظه نهان بیشتر از آنکه روی پهنای باند تاثیرگذار باشد IOPS را بهبود می‌دهد. هم‌چنین روی داده ترتیبی معمولا بهبود حاصل نمی‌شود. اما برای خواندن و نوشتن تصادفی استفاده از حافظه نهان می‌تواند بسیار مفید باشد.

شاید جالب‌ترین بخش نتایج سیاست Write Through باشد، در این سیاست به دلیل رعایت اصول Reliability و Consistency هر تغییری که در حافظه نهان ایجاد می‌شود همزمان در لایه‌های پایین‌تر نیز باید اعمال شود. بنابراین مشاهده می‌شود که عملکرد این سیاست برای نوشتن حتی از HDD تنها نیز بدتر است.

نتایج WA کمی برای بنده عجیب است، در واقع فکر می‌کنم پس از آنکه فایل سیستم را دوباره نصب کردم fio به درستی اجرا نشده و در واقع نتیجه به دست آمده برای Write Around صحیح نیست. تفاوت سیاست WA و WT در این است که اولی کلا حافظه نهان را برای عملیات نهان Skip می‌کند. در واقع درخواست های نوشتن کش نمی‌شوند.