سیستم‌های عامل پیشرفته – تمرین یک

استاد: دکتر حسین اسدی

محمدمهدی قره‌گوزلو – 401206836

پرسش یکم: با توجه به اینکه هر ترد 1 میلیون بار 1024 کیلوبایت حافظه درخواست می‌دهد یعنی 1 ترابایت فقط توسط یک ترد درخواست داده می‌شود، با توجه به تعداد تردها در مجموع 256 ترابایت حافظه می‌شود(کلا 256 گیگابات حافظه اصلی داریم).

فرض کنیم کلا 3 نوع shootdown داشته باشیم: local با دانه بندی ریز، ipi یا همان remote با دانه بندی ریز و Flush که تنها زمان تعویض متن رخ دهد. در این مورد بیشترین shootdown ها احتمالا از نوع remote ipi خواهند بود.

اگر 160 ترد روی هسته‌های مختلف در حال اجرا باشند **برای هر صفحه** که تخصیص می‌دهند یا آزاد می‌کنند باید یک ipi صادر شده و به هسته‌های دیگر این تغییر اعلام شود. با توجه به اندازه صفحه سیستم مورد نظر این تعداد ipi ها می‌تواند تغییر کند. مثلا چیزی در حدود 250 هزار ipi به ازای هر ترد هنگام تخصیص حافظه و همین مقدار برای آزادسازی(1TB / 4KB). واقعا عدد مشخصی نمی‌توان تعیین کرد، حتی در عمل نیز این اعداد نوسان دارند و صرفا در حد تخمین می‌توان نظر داد.

حالا اگر در این میان یکی از این تردها با تعویض متن از هسته خارج شود و اگر پردازنده مورد نظر ما شماره پردازه‌ها را در TLB در کنار رکوردها نگه داری نکرده باشد یک flush کامل را نیز مجبوریم بپردازیم.

بهتر است برای انجام عملیات‌هایی مانند تخصیص حافظه با تکرار بالا که ساختار جدول صفحه را با نوسان زیاد تغییر می‌دهند وقفه را غیر فعال کنیم به 2 دلیل: اول اینکه چون صرفا در حال تخصیص و آزاد کردن هستیم و نوشتن یا خواندن صورت نمی‌گیرید نیازی به shootdown فوری نداریم چون خطری از جهت خواندن یا نوشتن غیرمجاز یا نا معتبر وجود ندارد. توجه کنید فقط در صورتی کل عملیات‌ها تخصیص و آزاد کردن باشند. و دوم اینکه مشکل flush کردن نیز اگر وقفه غیرفعال شود و تعوض متن صورت نگیرد برطرف می‌شود. برای حالت دوم البته شاید بهتر باشد شماره pid ها را در TLB نگه داریم.

توضیحاتی مختصر درباره دلایل Shootdown: آنچه در ادامه درباره دلایل shootdown نوشتم را در نمودارهای pie می‌توانید مشاهده کنید، برای هر سه سوال این نمودارها را رسم شده‌اند. مراجعه کنید به:
https://github.com/torvalds/linux/blob/4dd58158254c8a027f2bf5060b72ef64cafa3b9d/include/linux/mm_types.h#L740

متاسفانه بنده خیلی اطلاعات دقیقی درباره تعریف عبارات پیش رو نیافتم، بنابراین با چاشنی عدم اطمینان مطالعه کنید.

TLB Remote Shootdown: زمانی رخ می‌دهد که یک هسته‌ی دیگر پردازنده درخواست shootdown را داده باشد.

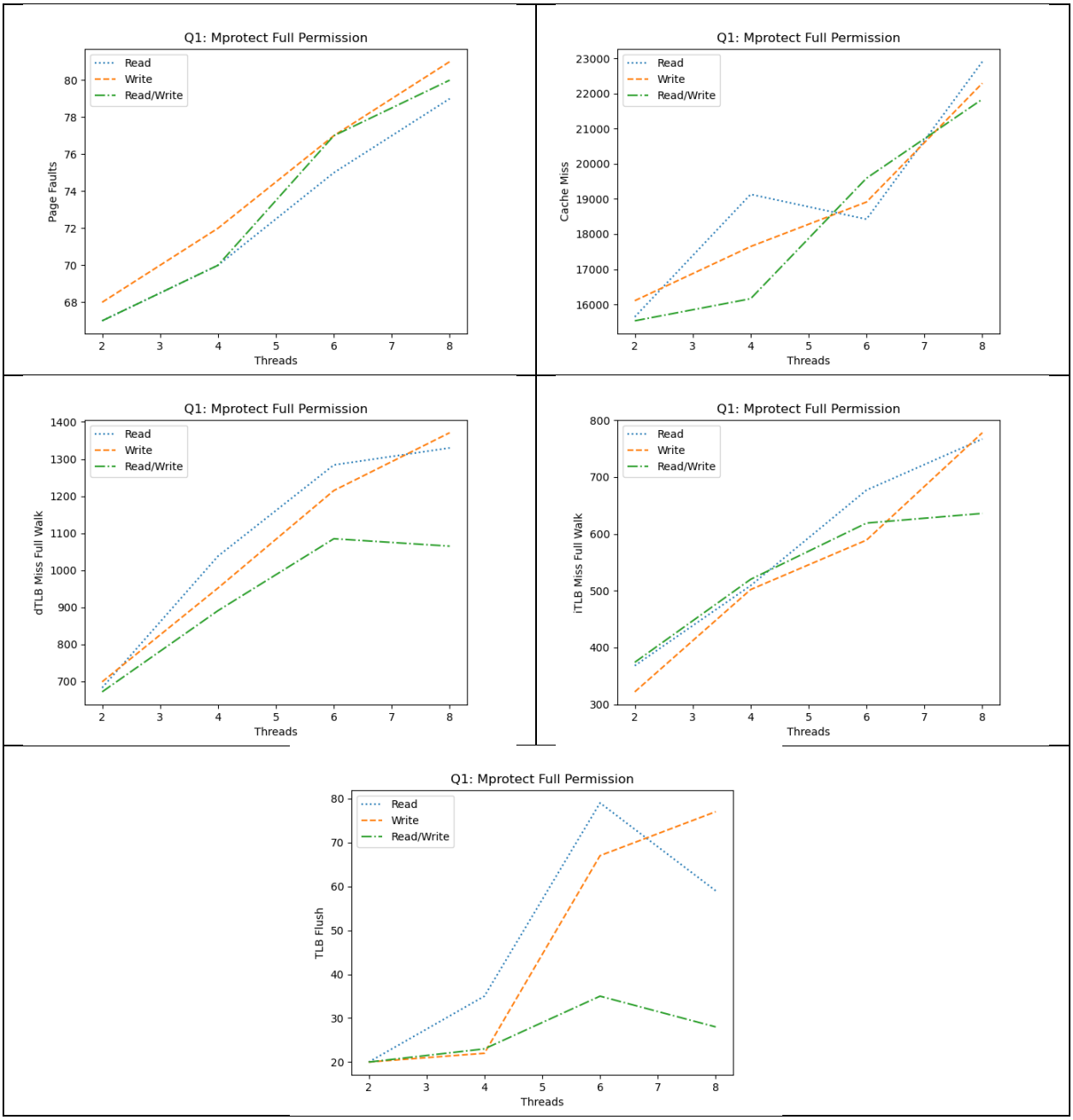
TLB Remote Send IPI: فکر می‌کنم این سیگنال و بالایی باید علت و معلول یک دیگر باید باشند، یعنی پس از ارسال ipi باید Remote shootdown صورت بگیرد. اما جلوتر خواهیم دید که تعداد آنها برابر نیستند. بنابراین مطمئن نیستم تفاوت کجاست.

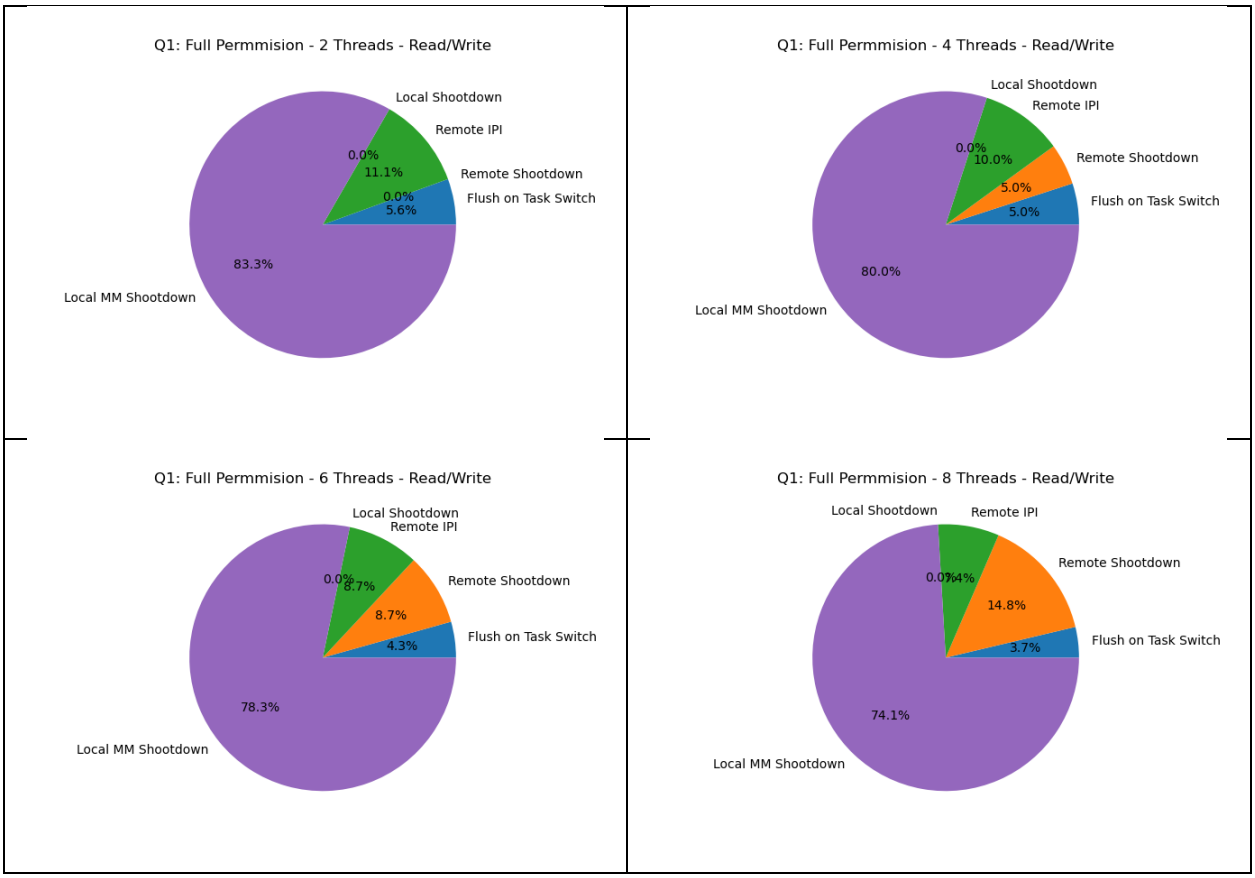
TLB Local MM Shootdown: اگر ساختار صفحه توسط هسته فعلی تغییر کند آنگاه این نوع shootdown رخ می‌دهد.

TLB Local Shootdown: هنگامی که صفحات نگاشت معکوس شوند یعنی از حافظه اصلی خارج شوند.

TLB Flush On Task Switch: از اسم مشخص است که هنگام تعویض متن رخ می‌دهد، جلوتر میبینیم که همه‌ی برنامه‌ها حتما این نوع shootdown را تجربه می‌کنند.

پرسش عملی یکم – با دسترسی PROT_READ, PROT_WRITE, PROT_EXEC

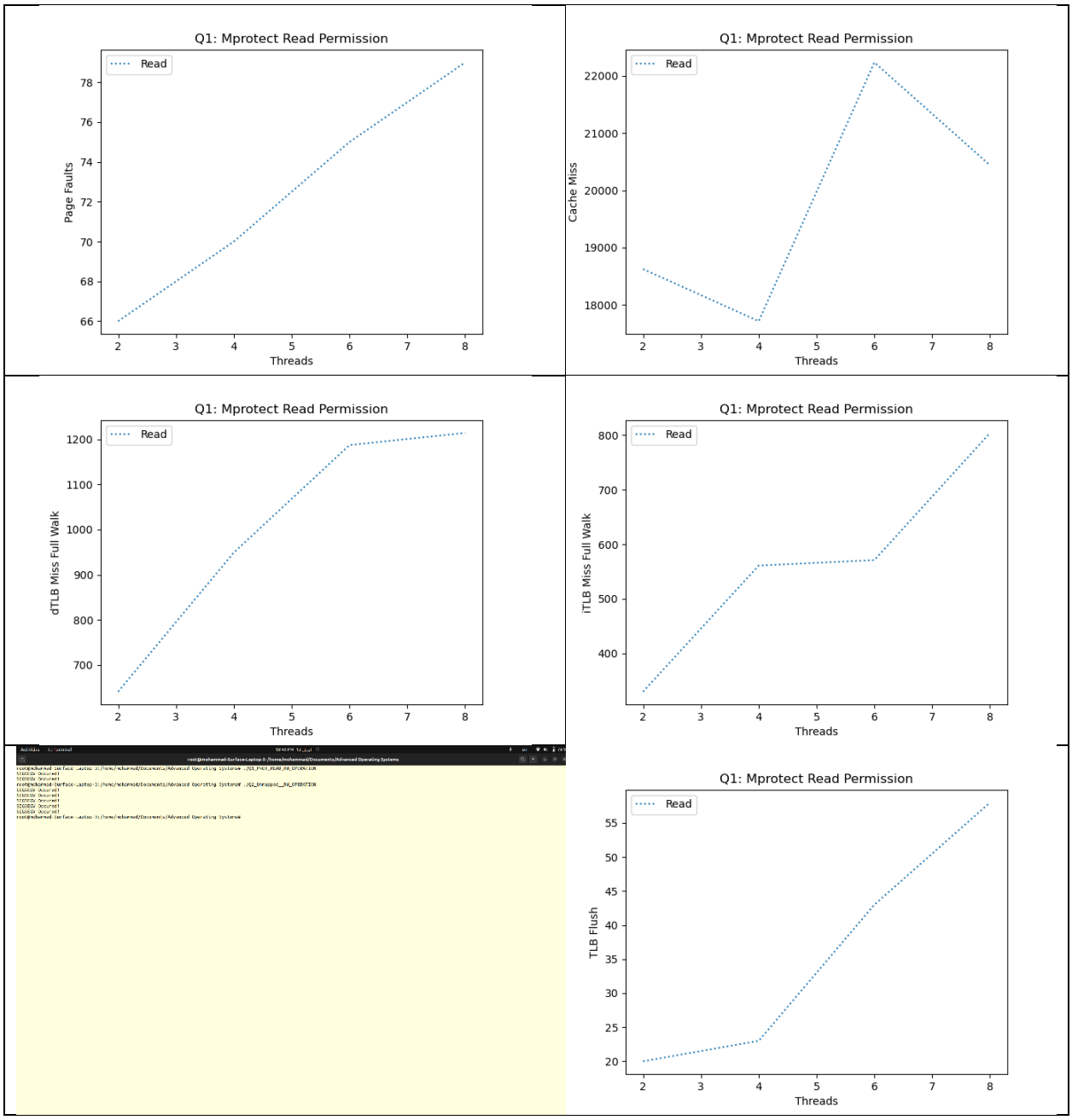


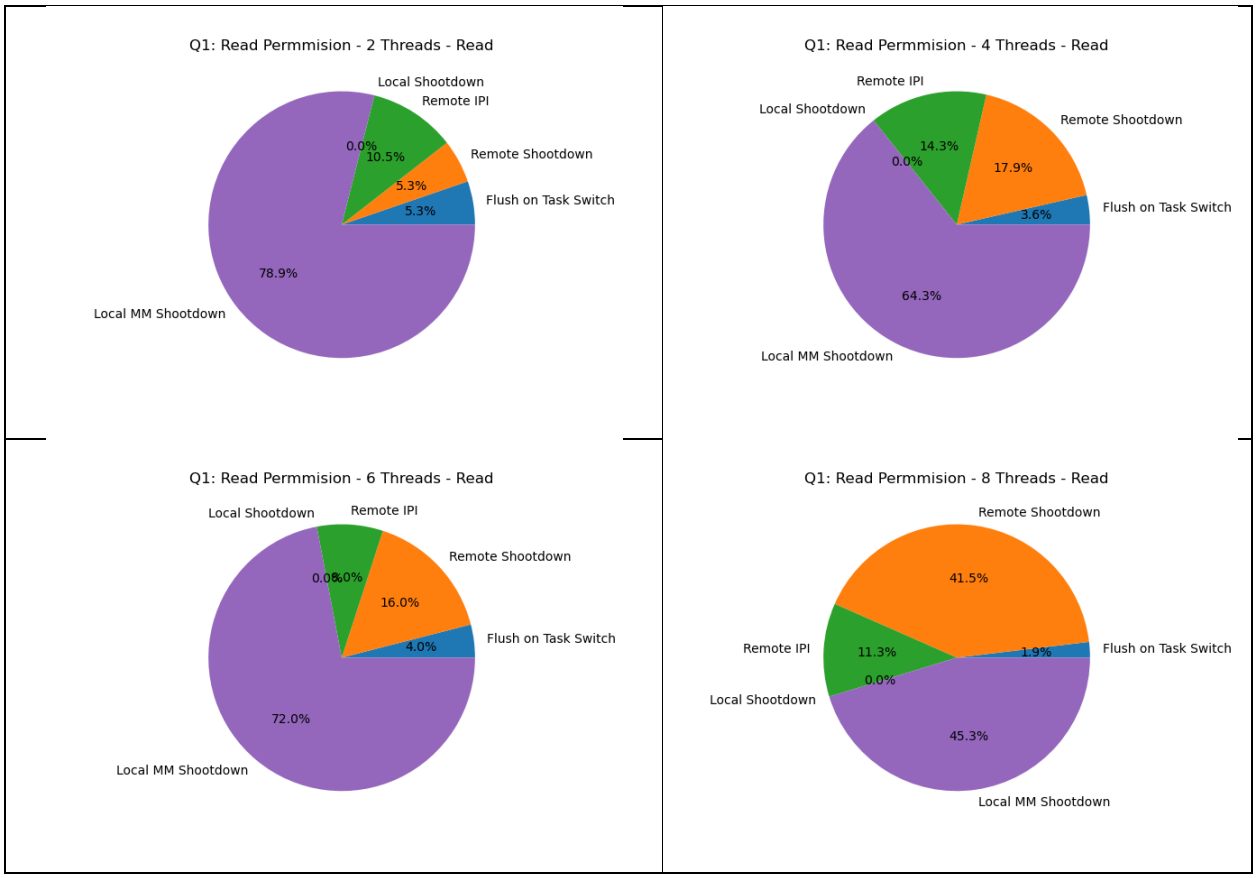


درنمودارهای خطی بالا تعداد تعداد ترد 2 در حالت Read/Write یعنی یک ترد میخواند و دیگری می‌نویسد، 8 ترد یعنی 4 – 4 و بقیه نیز به همین ترتیب. در حالت فقط Read و فقط Write تمام تردها یک عملیات را انجام می‌دهند. شیوی اجرا بدین صورت است یک بار تمام تردها در حافظه مورد نظر عملیات خود را انجام می‌دهد و پشت barrier منتظر می‌مانند تا همه عملیات را به پایان رسانیده باشند. سپس با استفاده از دستور mprotect سطح دسترسی تغییر می‌کند و دوباره تمام تردها عملیات خود را انجام می‌دهند. در این حالت من کامل دسترسی‌ها را دادم تا بتوانم تمام عملیات‌ها را انجام دهم. اگر روند کلی نمودارهای خطی را نگاه کنیم بنظر می‌رسد که با افزایش تردها میزان page fault ها و miss ها و shootdown افزایش پیدا می‌کند، البته گاها ممکن است کاهش نیز رخ دهد، من چندین بار مثلا برای تعداد 6 ترد را اجرا کردم اما واقعا الگوی مشخصی وجود ندارد. یعنی نمی‌توان نتیجه گیری کرد که مثلا با فلان تعداد ترد ممکن است میزان miss وshootdown کاهش پیدا کند. تمامی حالات دقیقا دارای یک عددTLB flush on task switch هستند، که خب طبیعی به نظر می‌رسد. تمامی page fault ها از نوع minor page fault هستند. یعنی اینکه داده‌ی مورد نظر در حافظه اصلی قرار دارد اما رکورد ترجمه آن نیست، البته دلیل دیگر نیزمی‌تواند داشته باشد مانند سطح دسترسی[Architectural and Operating System Support for Virtual memory – page 29].

اگر برنامه بالا را بدون چندنخی و مثلا فقط با یک نخ انجام دهیم و آنگاه میزان shootdown را اندازی گیری کنیم متوجه می‌شویم که تعداد ipi ها و remote ها به صفر می‌رسد که البته منطقی است. من این را با record تست کردم ولی در نمودارها نیاوردم.

پرسش عملی یکم – با دسترسی PROT_READ



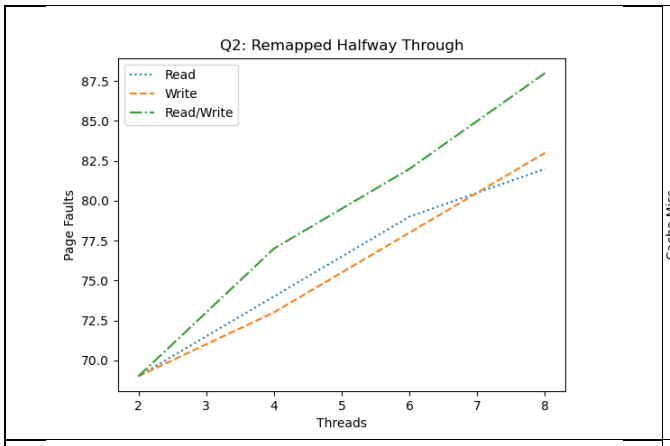


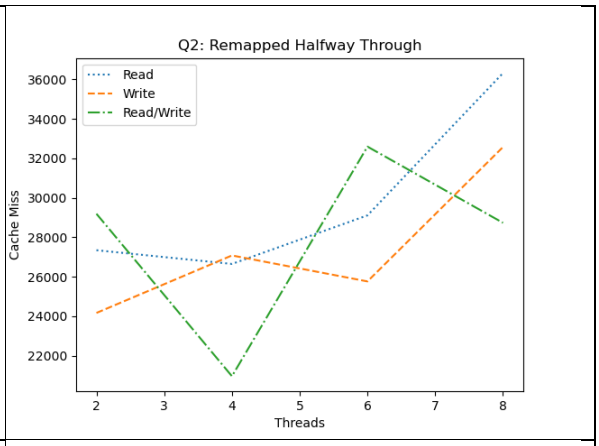
در این حالت ترتیب اجرا مانند قبل است اما سطح دسترسی را PROT_READ تعیین کردم، بنابراین فقط امکان خواندن وجود دارد. اگر پس از تغییر دسترسی بخواهیم عملیات نوشتن انجام دهیم سیگنال SIGSEGV صادر شده و خطای Segmentation fault را در میان برنامه و هنگام تلاش برای نوشتن دریافت خواهیم کرد. با استفاده از یک signal handler می‌توان آنرا دریافت کرده و عبارت SIGSEGV OCCURED را در صورت وقوع در خروجی خطای استاندارد نمایش داد. اسکرین‌شات مربوطه در جدول بالا قابل مشاهده است.

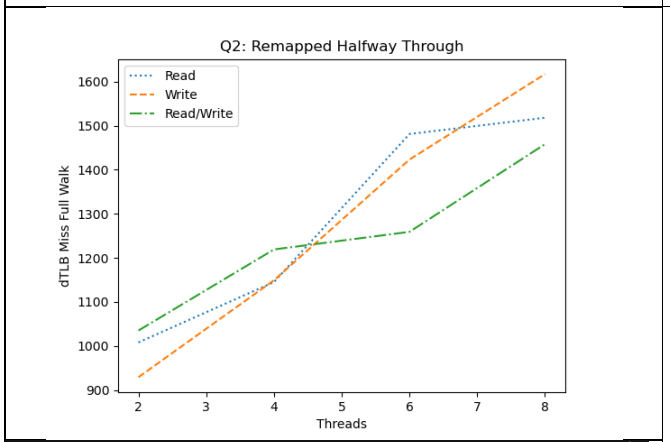
محسوس‌ترین تغییر را می‌توان در نمودار pie مشاهده کرد، اگر توجه کنید تعداد Remote Shootdown نسبت به حالت قبلی که عملا سطح دسترسی تغییری نمی‌کرد به شدت افزایش یافته، دلیلش محدود شدن سطح دسترسی توسط تردی به نام thread_protector (مراجعه شود به فایل Q2.c) که درخواست Shootdown را به سایر هسته‌ها ارسال می‌کند.(شاید هم تصادفی بوده).

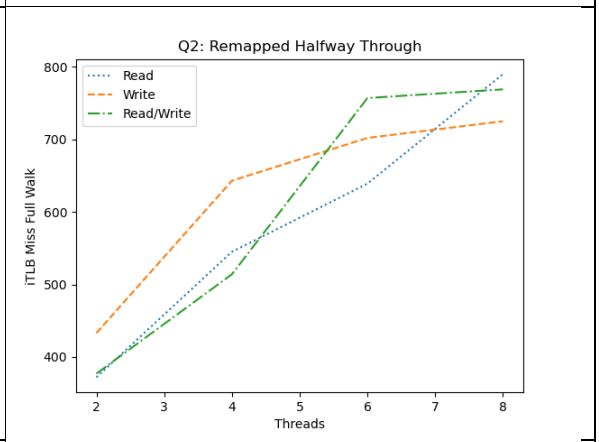
دراین حالت نیز تمامی fault page ها minor هستند. همانطورکه پیشتر گفتم این نوع نقص به دلیل عدم دسترسی به آدرس است ولی خود داده در حافظه اصلی قرار دارد.

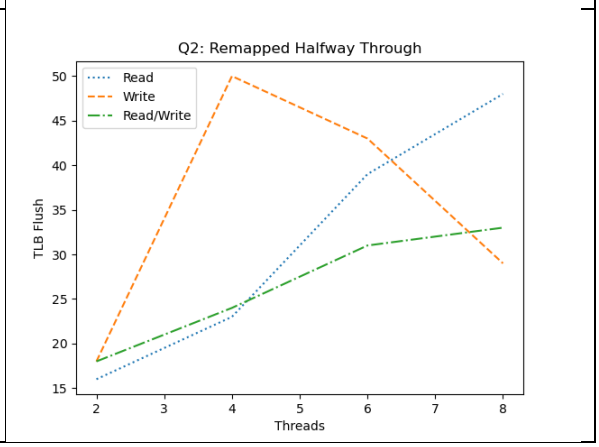
پرسش عملی دوم

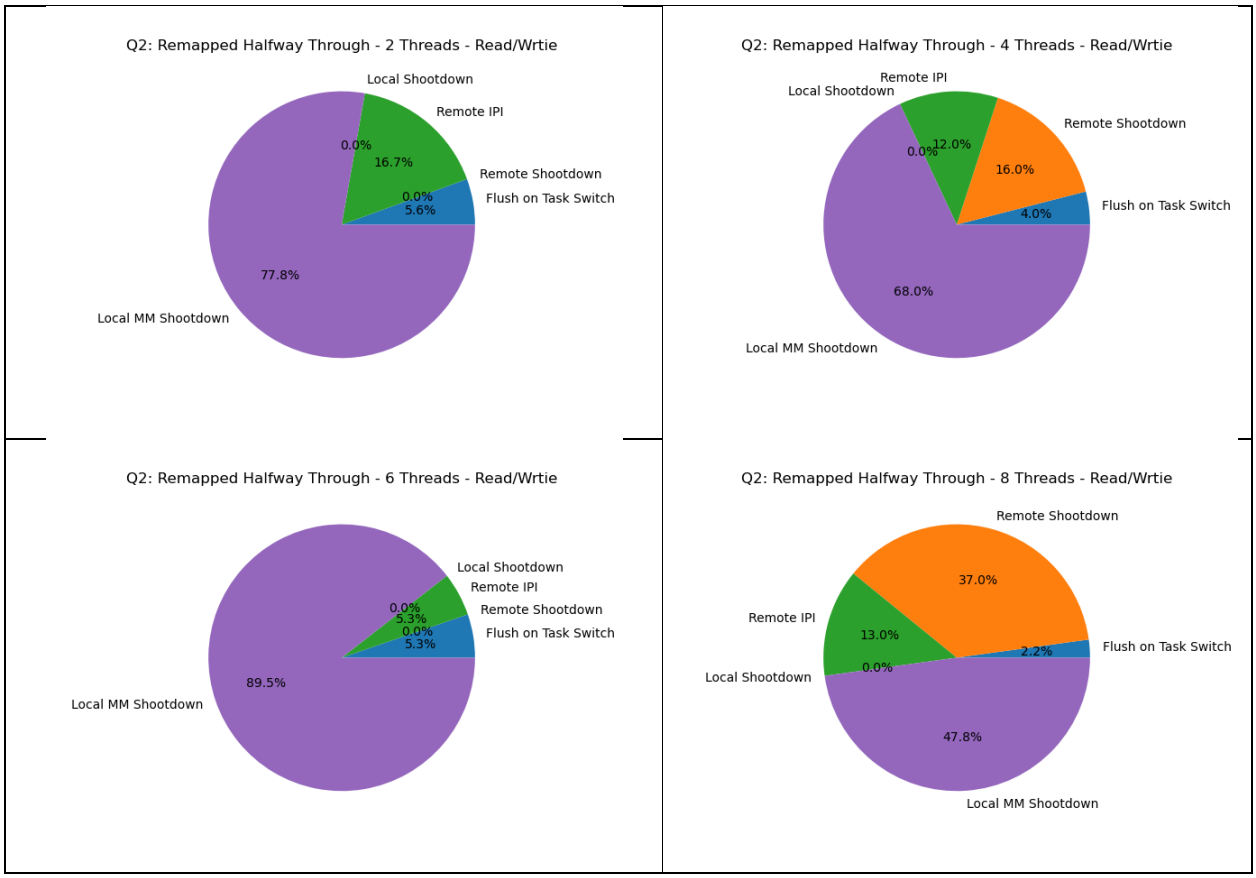












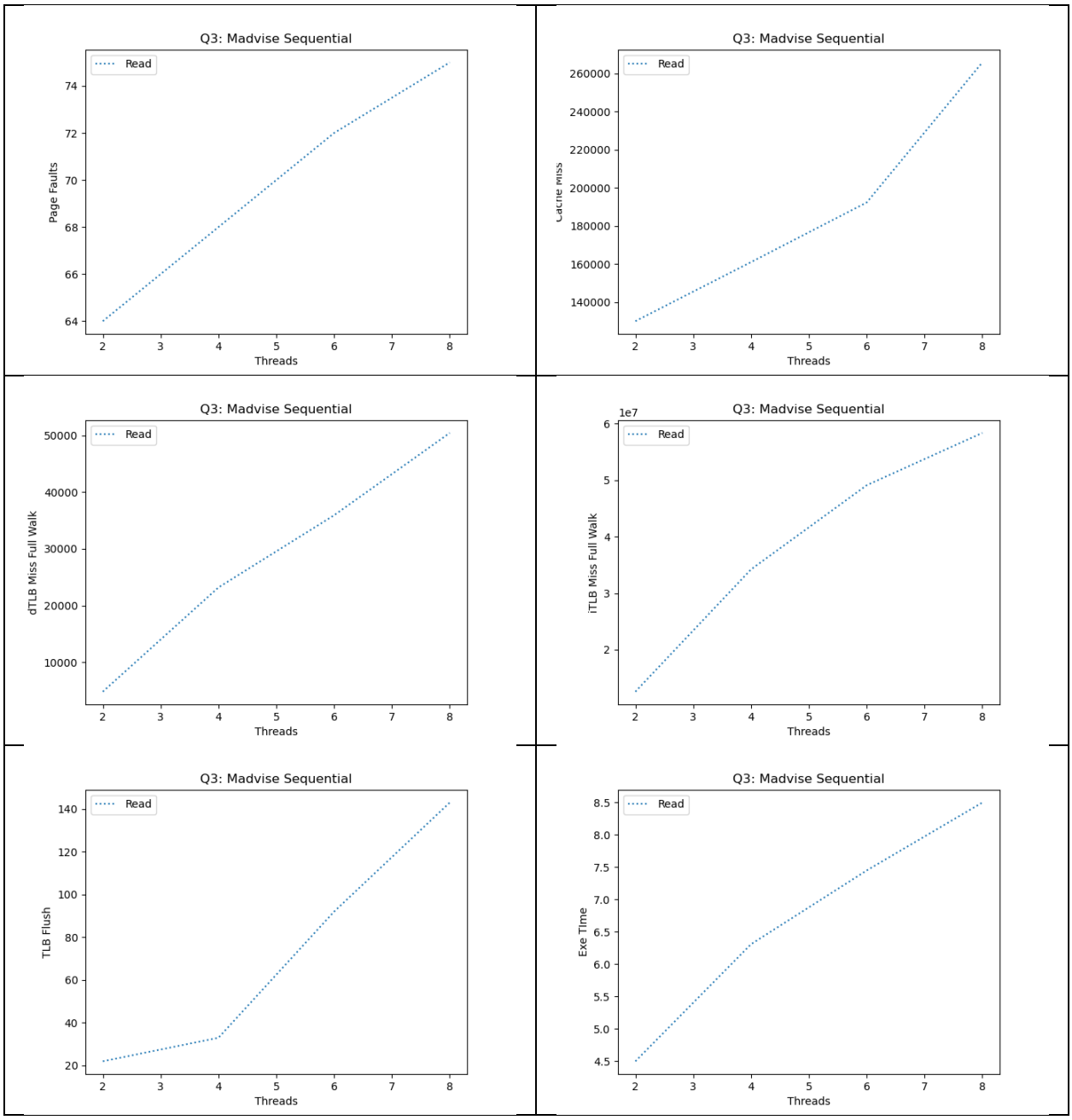
ترتیب اجرا مانند حالات پیشین و با استفاده از barrier صورت گرفته. یعنی یک بار تمامی تردها در نگاشت اول و بار دیگر همگی در نگاشت ثانویه عملیت انجام می‌دهند. اگر در این میان عمل نگاشت معکوس صورت بگیرد خطای Segmentation دریافت خواهیم کرد، اسکرین‌شات در جدول فوق این را نشان می‌دهد. دلیلش نیز بدیهی است زیرا به عملا به قسمتی تلاش دسترسی داریم که دیگر در اختیار ما نیست.

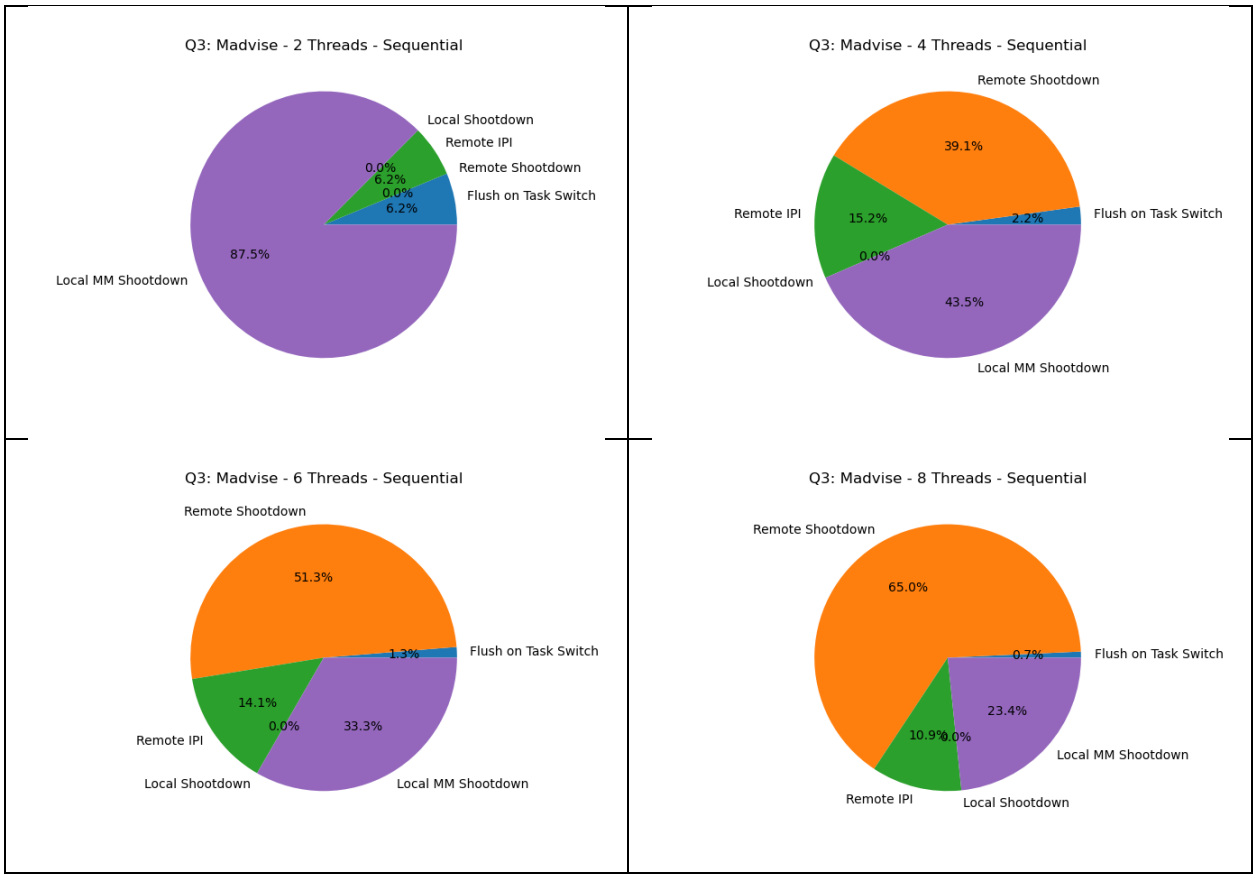
بنظرم مهم‌ترین نکته در این سوال در Page fault هاست، این تنها حالتی است Major Page Fault دریافت می‌کنیم. تعداد دقیق بدین شرح است:

Major PF	2 Threads	4 Threads	6 Threads	8 Threads
Read	3	4	4	5
Write	3	4	4	6
R / W	4	7	7	9

Major Page Fault اینطور تعریف می‌شود: تلاش برای دسترسی به داده‌ای که در حافظه اصلی وجود ندارد و باید از حافظه جانبی به حافظه اصلی آورده شود. ازآنجایی که در این سوال حافظه جانبی را نگاشت می‌کنیم به نقص‌ها بنظر کاملا منطقی می‌رسند. افزایش تعداد نقص‌ها احتمالا به این دلیل است که پس از میزان مشخصی نوشتن سیستم‌عامل محتویات نگاشت شده حافظه اصلی را به جانبی منتقل کرده و دوباره به اصلی بر می‌گرداند.

پرسش عملی سوم





شاید بهتر بود به صورت دیگر از madvise استفاده می‌شد، مثلا آنرا با آپشنSequential اجرا کرد سپس به صورت Scan و Sequential خواند یا نوشت تا تفاوت آن با حالت عادی مشخص شود.

من این دستور را با آپشن Sequential اجرا کردم، این آپشن باعث می‌شود که پس از یک بار خواندن یا نوشتن داده‌ها به سرعت از Cache و یا TLB اخراج شوند و آنهایی که جلوتر هستند وارد شوند. دلیلش نیز واضح است، چون یک بار از اول تا آخر یک چیزی را می‌خواهیم خوانیم یا بنویسیم.

این امر در نمودار pie مشخص است، به تعداد زیاد Remote Shootdown توجه کنید که با بالا رفتن تعداد تردها که هر کدام دارند madvise را فراخوانی می‌کنند به شدت افزایش پیدا می‌کند.

توضیحاتی درباره تمرینات عملی: برای دقیق‌تر بودن داده‌های استخراجی پیش از اجرای هر دستور perf و اجرای برنامه‌ها، حافظه‌ی نهان صفحه و داده را خالی کردم. همچنین دستورات stat را با فراوانی 90 بار اجرا کردم تا در حد امکان از واریناس بالا و ناهنجاری در داده‌ها جلوگیری شود. فقط پرسش سوم عملی را به دلیل طولانی بودن زمان اجرا مجبور شدم با فراوانی 1 اجرا کنم.

برای بدست آوردن دلایل tlb shootdown از آپشن record بهره بردم سپس با استفاده از script آن را به فایل قابل خواندن تبدیل کردم. دستورات مورد استفاده به شرح زیر می‌باشند:

```
echo 3 > /proc/sys/vm/drop_caches; perf stat -r 90 -e faults,major-faults,minor-faults,tlb:tlb_flush,dtlb_load_misses.walk_completed,itlb_misses.walk_completed,cache-misses,dTLB-load-misses,iTLB-load-misses

echo 3 > /proc/sys/vm/drop_caches; perf record -e tlb:tlb_flush

perf script -i perf.data > foolan.txt
```

اکثر اسکرین‌شات‌های گرفته شده در فایل پیوست موجود هستند و می‌توانید مشاهده کنید اما از آنجا که تعداد آنها بسیار زیاد بوده از آوردن آنها به داخل گزارش خودداری نمودم، توجه کنید که فقط برای نمودارهای یک قسمت چندین بار برنامه با تنظیمات مختلف کامپیال و با دو حالت مختلف stat و record هرکدام از آنها اجرا شده‌اند. این کار برای تمامی قسمت‌ها انجام پذیرفته است. در مجموع حالات شاید بیشتر از 80 بار. بنابراین تعداد محدودی اسکرین شات احتمالا داخل پیوست موجود نباشد.