Perf

محمد مهدی رسول امینی	4001830235	آزمایشگاه سیستم عامل	تمرین 3	1403/10/11

خلاصه اهداف، روشها و نتایج:

گزارش کار 8:

این گزارش به نصب و استفاده از ابزار **Perf**برای نظارت و سنجش کارایی سیستم لینوکسی می‌پردازد. مراحل نصب شامل به‌روزرسانی مخزن‌ها و هماهنگی با نسخه هسته است. همچنین روش تنظیم دسترسی کاربران عادی و استفاده از دستوراتی مانند recordبرای پروفایل‌گیری فرآیندها بدون وقفه توضیح داده شده است.

گزارش کار 9:

این گزارش بر کاربردهای پیشرفته ابزار **Perf**متمرکز است، از جمله رهگیری فراخوانی‌های سیستمی با دستور trace، حاشیه‌نویسی کدها با annotate، و تولید گزارش‌های شخصی‌سازی‌شده با .scriptتفسیر نتایج کارایی و به‌کارگیری داده‌ها برای تحلیل و تصمیم‌گیری نیز بررسی شده است.

اطلاعات پیش‌زمینه‌ای و معرفی موضوع آزمایش:

ابزار **Perf**یکی از قدرتمندترین ابزارهای لینوکس برای نظارت و تحلیل عملکرد سیستم و برنامه‌هاست. این ابزار امکان پروفایل‌گیری از فرآیندها، اندازه‌گیری رویدادهای عملکردی، و بررسی رفتار برنامه‌ها را با استفاده از داده‌های دقیق فراهم می‌کند **Perf** .با هسته لینوکس ادغام شده و قابلیت‌های گسترده‌ای مانند رهگیری فراخوانی‌های سیستمی، ثبت رفتار پردازنده و حافظه، و تولید گزارش‌های شخصی‌سازی‌شده را ارائه می‌دهد. هدف این آزمایش، یادگیری نصب، تنظیم، و به‌کارگیری این ابزار برای تحلیل کارایی سیستم و برنامه‌هاست.

توضیح مراحل انجام کار با جزئیات کافی:

توضیح مراحل انجام کار (گزارش کار 8):

نصب و استفاده از ابزار Perf برای تحلیل کارایی یک اپلیکیشن ساده

مراحل انجام کار:

1. **تنظیم مجوزهای کاربر معمولی :**برای اینکه کاربران معمولی نیز بتوانند از ابزار Perf استفاده کنند، تغییراتی در تنظیمات سیستم اعمال شد. این تغییرات به صورت زیر انجام شدند:
    - به کاربر روت سوئیچ کرده و دستور زیر برای فعال‌سازی استفاده از Perf توسط کاربران معمولی اجرا شد:

```
sudo su -
echo 0 > /proc/sys/kernel/perf_event_paranoid
exit
```

برای دائمی کردن این تغییرات، فایل sysctl.conf ویرایش و خط زیر به آن اضافه شد:

```
sudo nano /etc/sysctl.conf
kernel.perf_event_paranoid = 0
```

2. **تحلیل کارایی یک اپلیکیشن ساده :**پس از تنظیمات لازم، اقدام به تحلیل کارایی یک اپلیکیشن ساده با استفاده از ابزار Perf شد. برای این منظور، از دستور perf recordبرای ثبت پروفایل کارایی فرآیند مورد نظر استفاده گردید:
    - ابتدا با استفاده از دستور pidofشناسه فرآیند اپلیکیشن مورد نظر به دست آمد.
    - سپس پروفایل کارایی این فرآیند ثبت شد:

```
sudo perf record -p <pid>
```

    - در نهایت، با استفاده از دستور perf reportگزارش تحلیل کارایی اپلیکیشن به نمایش درآمد:

```
sudo perf report
```

1. **هدف:**

بررسی عملکرد پردازش‌ها و تحلیل رفتار آن‌ها با استفاده از ابزار perfبرای سنجش و جمع‌آوری اطلاعات آماری.

2. **مراحل انجام:**

1. **شناخت پردازش:**

ابتدا پردازشی که نیاز به تحلیل دارد شناسایی می‌شود. در اینجا پردازش 38918 انتخاب شده است.

2. **اجرای دستور perf stat:**

دستور sudo perf stat -p 38918برای جمع‌آوری اطلاعات کلی مانند تعداد تغییر

کانتکست‌ها، خطاهای صفحه، و مهاجرت CPU اجرا شد. این مرحله دید کلی از عملکرد پردازش ارائه می‌دهد.

3. **ضبط داده‌ها:**

با استفاده از دستور sudo perf record -p 38918 داده‌های نمونه برای تحلیل دقیق‌تر ذخیره شدند. خروجی نشان داد که 206 نمونه داده در فایل perf.data ذخیره شده است.

4. **تحلیل داده‌ها:**

داده‌های ذخیره‌شده با دستور sudo perf report تجزیه و تحلیل شدند. خروجی نشان داد که توابعی مثل snap_account و strlen بیشترین سهم در مصرف منابع را داشته‌اند.

5. **بررسی نتایج:**

با مشاهده سهم هر بخش، نقاط پرمصرف شناسایی شدند که می‌توانند هدف بهینه‌سازی باشند.

3. **ابزارها و نرم‌افزارهای استفاده‌شده:**
   1. perf: برای مانیتور و تحلیل عملکرد پردازش.
   2. محیط ترمینال و دسترسی root.

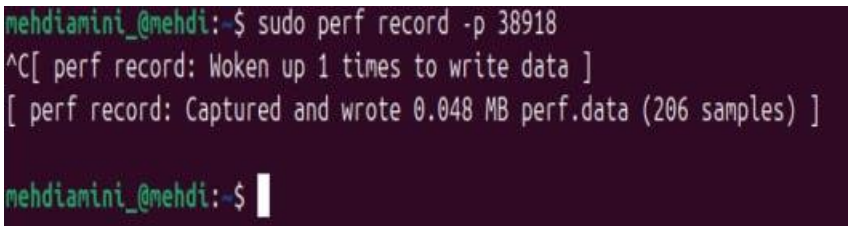
```
event syntax error: 'topdown-retiring/metric-id=topdown!1retiring/,TOPDOWN.SLOTS/metric-id=TOPDOWN.SLOTS/,topdown-fe-..'
                    \___ Cannot find PMU `topdown-retiring'. Missing kernel support?
^C
 Performance counter stats for process id '38918':

         1,084.67 msec task-clock                       #    0.106 CPUs utilized
           17,295      context-switches                 #   15.945 K/sec
              565      cpu-migrations                   #  520.894 /sec
           17,720      page-faults                      #   16.337 K/sec
   <not supported>      cycles
   <not supported>      instructions
   <not supported>      branches
   <not supported>      branch-misses

       10.269924074 seconds time elapsed

mehdiamini_@mehdi:~$
```

دستور:

دستور sudo perf stat -p 38918 برای مانیتور کردن عملکرد پردازش با آی‌دی 38918 استفاده شده است.

خروجی:

- task-clock:زمان کلی مصرف شده توسط پردازش در میلی‌ثانیه.
- context-switches:تعداد سوییچ‌های بین کانتکست‌ها در هر ثانیه.
- cpu-migrations:تعداد مهاجرت‌های پردازشی بین هسته‌های CPU.
- page-faults:تعداد خطاهای صفحه.
- فیلدهایی مانند cycles, instructions, branches و branch-misses پشتیبانی نشده‌اند. این ممکن است به عدم سازگاری سخت‌افزار یا تنظیمات کرنل اشاره داشته باشد.
- استفاده از CPU در این پردازش 0.106 بوده است.

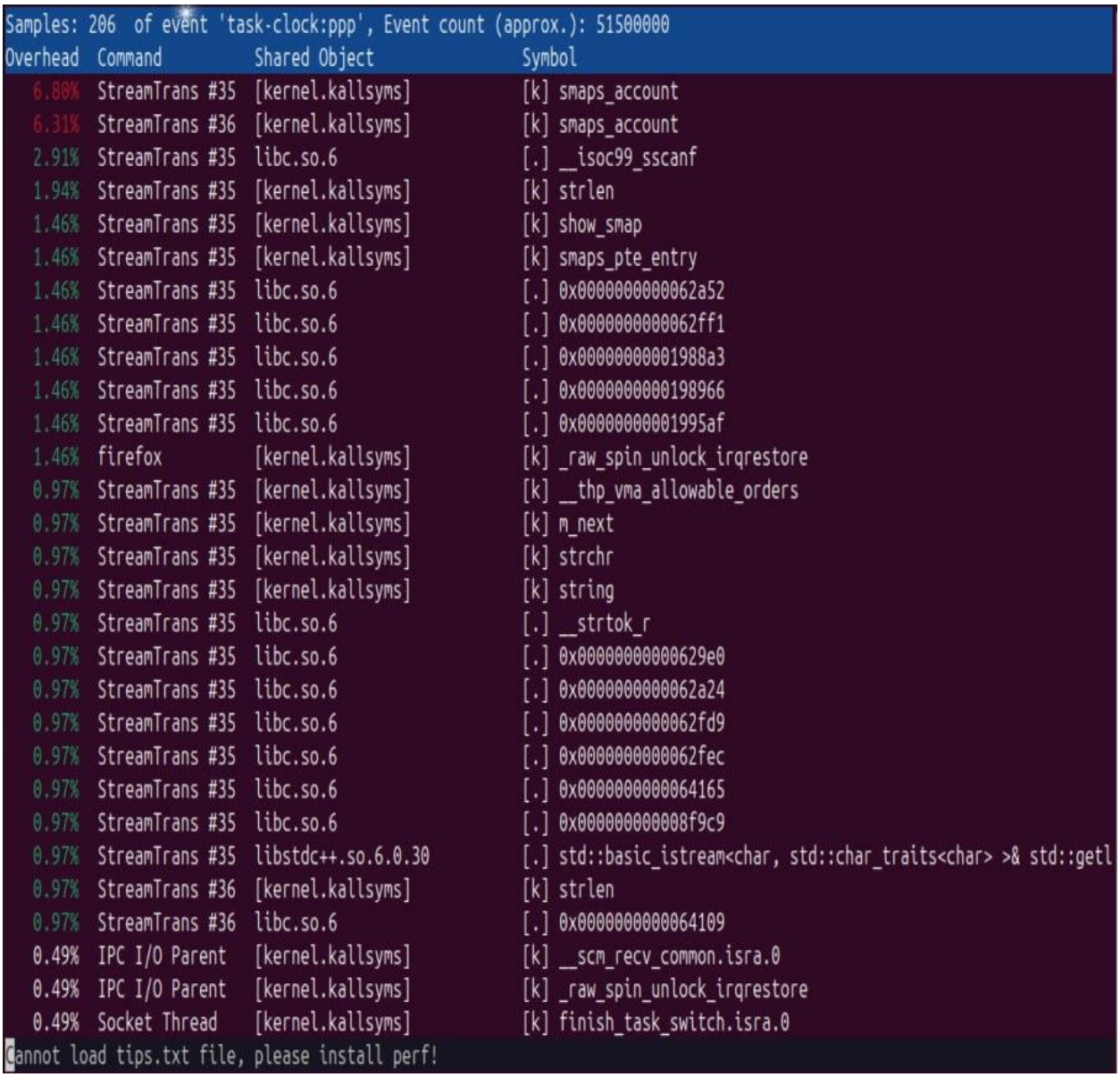


دستور:

با دستور sudo perf record -p 38918 داده‌های نمونه‌ای برای آنالیز پردازش ذخیره شده است.

خروجی:

- پیغام Captured and wrote 0.048 MB perf.data نشان می‌دهد که 206 نمونه داده ذخیره شده‌اند.



```
Samples: 206  of event 'task-clock:ppp', Event count (approx.): 51500000
Overhead  Command          Shared Object              Symbol
   6.80%  StreamTrans #35  [kernel.kallsyms]         [k] smaps_account
   6.31%  StreamTrans #36  [kernel.kallsyms]         [k] smaps_account
   2.91%  StreamTrans #35  libc.so.6                 [.] __isoc99_sscanf
   1.94%  StreamTrans #35  [kernel.kallsyms]         [k] strlen
   1.46%  StreamTrans #35  [kernel.kallsyms]         [k] show_smap
   1.46%  StreamTrans #35  [kernel.kallsyms]         [k] smaps_pte_entry
   1.46%  StreamTrans #35  libc.so.6                 [.] 0x0000000000062a52
   1.46%  StreamTrans #35  libc.so.6                 [.] 0x0000000000062ff1
   1.46%  StreamTrans #35  libc.so.6                 [.] 0x00000000001988a3
   1.46%  StreamTrans #35  libc.so.6                 [.] 0x0000000000198966
   1.46%  StreamTrans #35  libc.so.6                 [.] 0x00000000001995af
   1.46%  firefox          [kernel.kallsyms]         [k] _raw_spin_unlock_irqrestore
   0.97%  StreamTrans #35  [kernel.kallsyms]         [k] __thp_vma_allowable_orders
   0.97%  StreamTrans #35  [kernel.kallsyms]         [k] m_next
   0.97%  StreamTrans #35  [kernel.kallsyms]         [k] strchr
   0.97%  StreamTrans #35  [kernel.kallsyms]         [k] string
   0.97%  StreamTrans #35  libc.so.6                 [.] __strtok_r
   0.97%  StreamTrans #35  libc.so.6                 [.] 0x00000000000629e0
   0.97%  StreamTrans #35  libc.so.6                 [.] 0x0000000000062a24
   0.97%  StreamTrans #35  libc.so.6                 [.] 0x0000000000062fd9
   0.97%  StreamTrans #35  libc.so.6                 [.] 0x0000000000062fec
   0.97%  StreamTrans #35  libc.so.6                 [.] 0x0000000000064165
   0.97%  StreamTrans #35  libc.so.6                 [.] 0x000000000008f9c9
   0.97%  StreamTrans #35  libstdc++.so.6.0.30       [.] std::basic_istream<char, std::char_traits<char> >& std::getl
   0.97%  StreamTrans #36  [kernel.kallsyms]         [k] strlen
   0.97%  StreamTrans #36  libc.so.6                 [.] 0x0000000000064109
   0.49%  IPC I/O Parent   [kernel.kallsyms]         [k] __scm_recv_common.isra.0
   0.49%  IPC I/O Parent   [kernel.kallsyms]         [k] _raw_spin_unlock_irqrestore
   0.49%  Socket Thread    [kernel.kallsyms]         [k] finish_task_switch.isra.0
Cannot load tips.txt file, please install perf!
```

دستور:

دستور sudo perf reportبرای تحلیل داده‌های ذخیره شده استفاده شده است.

خروجی:

- نمونه‌های گرفته‌شده با جزئیات نمایش داده شده‌اند.
- ستون "Overhead" نشان می‌دهد که هر تابع یا پردازش چه سهمی از زمان کلی را مصرف کرده است.
- توابعی مثل snap_accountو strlenبا بیشترین سهم در بالای لیست آمده‌اند.

- این تحلیل می‌تواند به شناسایی نقاط پرمصرف پردازش کمک کند.

توضیح مراحل انجام کار (گزارش کار 9):

مراحل انجام کار:

1. **دستورهای پیشرفته:Perf**
   - **دستور فرعی :trace**این دستور برای رهگیری فراخوانی‌های سیستمی و رویدادهای مختلف از فایل data.perfاستفاده می‌شود. به عنوان مثال:

```
sudo perf script
```

   - **نمایش هدر :trace** برای نمایش تمام رویدادها از data.perfهمراه با اطلاعات اضافی هدر، دستور زیر اجرا شد:

```
sudo perf script --header
```

   - **تخلیه داده‌های خام :**برای مشاهده داده‌های خام به صورت هگز از فایل data.perf، از گزینه -Dاستفاده شد:

```
sudo perf script -D
```

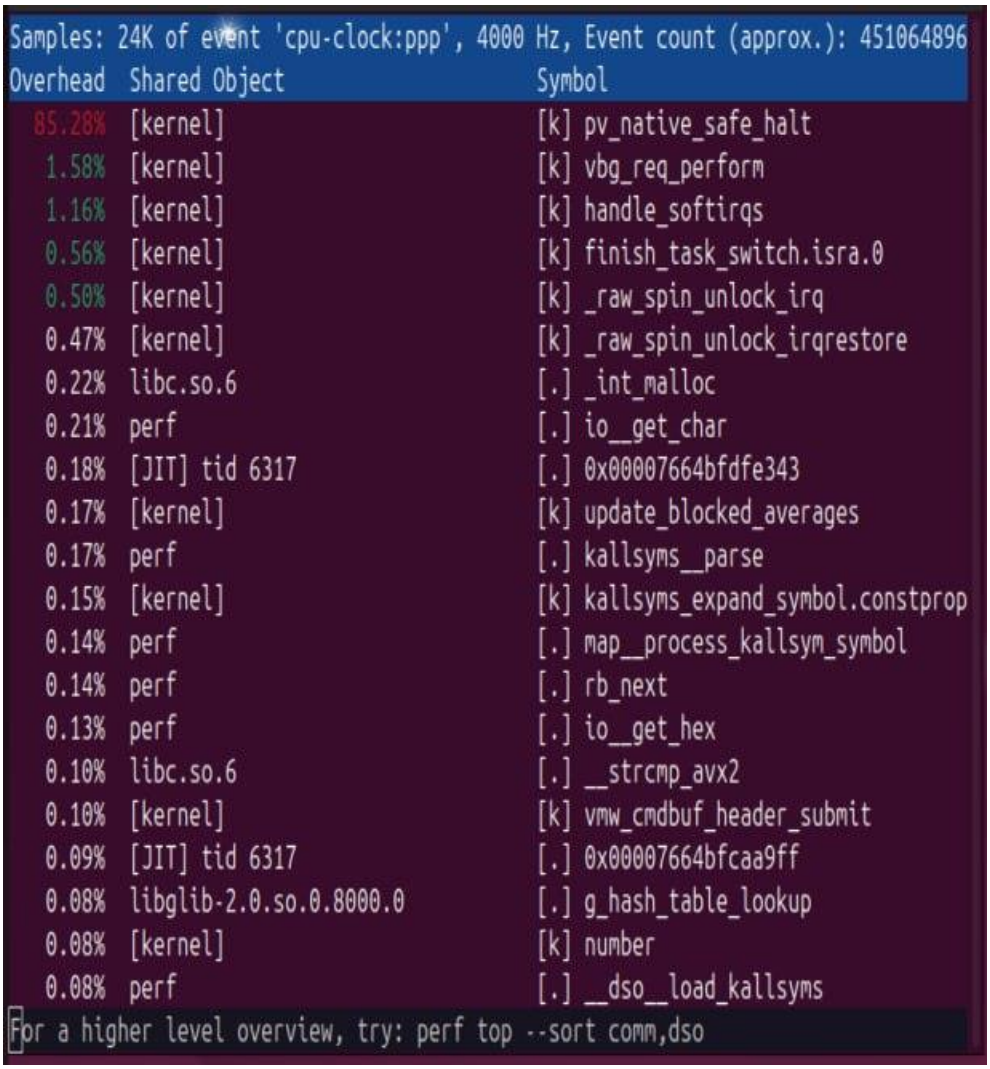
   - **حاشیه‌نویسی داده‌ها با دستور :annotate**این دستور برای حاشیه‌نویسی کد منبع و مشاهده جزئیات بیشتر رویدادها استفاده می‌شود:

```
sudo perf annotate --stdio -v
```

2. **تحلیل کارایی :**پس از اجرای دستورات پیشرفته، اقدام به تحلیل کارایی اپلیکیشن‌ها بر اساس داده‌های annotateو reportشد. این داده‌ها کمک کرد تا به درک بهتری از کارایی و رفتار برنامه‌ها دست یابیم.
3. **شناسایی اپلیکیشن با بیشترین استفاده از :CPU** برای شناسایی اپلیکیشنی که بیشترین میزان استفاده از CPU را دارد، از دستور perf topاستفاده شد:

```
sudo perf top
```

با اجرای این دستور، لیستی از اپلیکیشن‌ها و فرآیندهایی که بیشترین بار پردازشی را بر روی CPU ایجاد می‌کنند به نمایش درآمد. این ابزار به صورت زنده وضعیت استفاده از منابع را نشان می‌دهد و به تحلیل کارایی کمک می‌کند.



خروجی دستور sudo perf top برای پیدا کردن اینکه سی پی یو بیشتر دست کیه

pv_native_safe_halt هم یعنی اینکه سی پی یو بیکاره (رو این مد سوییچ میکند)

نتایج:

در این دو گزارش کار، به بررسی و استفاده از ابزار **Perf** پرداخته شد که به طور کلی برای تحلیل و پروفایل‌گیری عملکرد سیستم‌ها و اپلیکیشن‌ها در لینوکس مورد استفاده قرار می‌گیرد.

1. **گزارش کار 8:**

در این گزارش، با استفاده از ابزار Perf و تنظیمات مختلف آن، تلاش شد تا عملکرد یک اپلیکیشن ساده در سیستم لینوکسی بررسی و تحلیل شود. از جمله دستورات مورد استفاده، می‌توان به perf recordبرای ضبط پروفایل‌ها، perf reportبرای مشاهده نتایج، و همچنین تنظیمات دسترسی به ابزار توسط کاربران معمولی اشاره کرد. نتایج تحلیل‌ها نشان داد که Perf می‌تواند بدون وقفه در عملکرد سیستم، اطلاعات جامع و دقیقی از پروفایل‌های کارایی فرآیندها ارائه دهد.

2. **گزارش کار 9:**

در این گزارش، دستورات پیشرفته‌تر Perf شامل trace, top, annotate وscript به کار گرفته شدند. این دستورات به شناسایی و تحلیل جزئیات بیشتری از کارایی سیستم کمک کردند. از طریق دستور perf top,اپلیکیشن‌هایی که بیشترین بار پردازشی روی CPU دارند شناسایی شدند، و از دستور perf annotateبرای حاشیه‌نویسی داده‌ها و تحلیل دقیق‌تر استفاده شد. این مراحل کمک کرد تا بتوان اپلیکیشنی که بیشترین میزان استفاده از منابع سیستم را دارد شناسایی کرده و نتایج را برای بهینه‌سازی در نظر گرفت.

نتیجه نهایی :ابزار Perf با قابلیت‌های پیشرفته‌اش ابزاری مؤثر برای تحلیل و بهینه‌سازی عملکرد سیستم‌ها و اپلیکیشن‌ها است. با استفاده از دستورات پایه و پیشرفته، می‌توان به جزئیات دقیق کارایی پرداخته و مشکلات و نقاط ضعف عملکردی را شناسایی کرد. در این گزارش‌ها، توانستیم ابزار Perf را برای تحلیل کارایی یک اپلیکیشن ساده و شناسایی اپلیکیشن‌های پر مصرف CPU به کار بگیریم و به نتایج مفیدی دست یابیم.

منابع:

1. **ChatGPT, OpenAI**

برای دریافت راهنمایی‌ها و توضیحات مفید در مورد استفاده از ابزار Perf و رفع اشکالات در دستورات و تحلیل‌ها.

2. **Perf Tool Documentation**

Linux Performance Analysis with Perf: https://perf.wiki.kernel.org

3. **Linux Manual Pages**

Perf Command and its Subcommands: https://man7.org/linux/man-pages/man1/perf.1.html