Strace: примеры использования на реальных задачах

Введение

Рано или поздно любому системному администратору, и уж тем более devops инженеру, приходится сталкиваться с необходимостью более глубокой отладки какого-либо приложения. Например, приложение перестало запускаться или при запуске выдает мало информации или не выдает вообще, дебаг режим и лог файлы тоже не помогают. В таких случаях часто может помочь strace.

strace — это утилита, отслеживающая системные вызовы, которые представляют собой механизм трансляции, обеспечивающий интерфейс между процессом и операционной системой (ядром). Эти вызовы могут быть перехвачены и прочитаны. Это позволяет лучше понять, что процесс пытается сделать в данное время. Для своей работы strace использует механизм ptrace.

ptrace (от process trace) — системный вызов в некоторых unix-подобных системах (в том числе в Linux, FreeBSD, Max OS X), который позволяет трассировать или отлаживать выбранный процесс.

На днях на одном из проектов столкнулся с интересной проблемой - не возможностью запуска google chrome внутри docker контейнера, после небольшой оптимизации Dockerfile. В итоге, я смог найти и исправить причину проблемы именно с помощью strace.

Оптимизируем Dockerfile

Итак, у нас есть legacy приложение, которое написано на ruby-1.8 и rails 2.x. Переписать его на хотя бы ruby 1.9 и более современную версию рельс - просто не реально. Так что приходится страдать. Приложение собирается в docker и затем устанавливается на aws, но в контексте данной статьи это не принципиально.

Ниже привожу часть Dockerfile, которая собственно нас и интересует.

```
FROM ubuntu:12.04.5

RUN locale-gen en_US.UTF-8

ENV OPENSSL_VERSION=1.0.2g \
    CHROME_DRIVER_VERSION=2.20 \
    LANG=en_US.UTF-8 \
    LANGUAGE=en_US:UTF-8 \
    LC_ALL=en_US.UTF-8 \
    TERM=xterm \
```

```
    DEBIAN_FRONTEND=noninteractive

COPY google-chrome-stable_49.0.2623.112-1_amd64.deb /opt/

# install packages
RUN dpkg-reconfigure locales \
    && apt-get update \
    && apt-get install -y --no-install-recommends \
    ca-certificates libcurl3 librtmp0 libidn11 \
    libxss1 libappindicator1 libindicator7 wget unzip \
    gconf2-common fonts-liberation libcap2 libgconf-2-4 \
    libasound2 libnspr4 libnss3 libxtst6 openssl libexif12 \
    xdg-utils gconf-service-backend gconf-service openjdk-6-jre \
    && apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

RUN cd /opt/ && \
    wget -q
https://chromedriver.storage.googleapis.com/${CHROME_DRIVER_VERSION}/chromed
river_linux64.zip \
    && unzip chromedriver_linux64.zip \
    && mv chromedriver /usr/local/bin/ \
    && chmod +x /usr/local/bin/chromedriver \
    && rm -f chromedriver_linux64.zip \
    && dpkg -i google-chrome-stable_49.0.2623.112-1_amd64.deb

WORKDIR /opt
```

В таком виде образ собирается и google chrome запускается без проблем.

```
# docker build -t strace .
Sending build context to Docker daemon  48.48MB
Step 1/7 : FROM ubuntu:12.04.5
 ---> 5b117edd0b76
Step 2/7 : RUN locale-gen en_US.UTF-8
...
...
...
Step 7/7 : WORKDIR /opt
 ---> f83c44c16286
Removing intermediate container dc67a9763968
Successfully built f83c44c16286
Successfully tagged chrome:strace

# docker run -it --rm chrome:strace google-chrome -version
Google Chrome 49.0.2623.112
```

На первый взгляд все нормально, но я заметил один пакет - openjdk-6-jre, которого здесь быть не должно. Так как я уже говорил ранее - проект написан на ruby и java у нас не используется, от слова совсем. Причем что меня удивило, что использовался java 6, а это мамонт еще тот. И был установлен полный пакет jre, а не openjdk-6-jre-headless, например. Сам openjdk-6-jre тянет

за собой много зависимостей, поэтому удаляем его и пересобираем наш образ.

```
# docker build -t chrome:strace .
Sending build context to Docker daemon  48.48MB
Step 1/7 : FROM ubuntu:12.04.5
 ---> 5b117edd0b76
Step 2/7 : RUN locale-gen en_US.UTF-8
 ---> Using cache
 ---> 8bda6da6b60d
...
...
...
 ---> Running in 2e5d0dedabbb
Archive:  chromedriver_linux64.zip
  inflating: chromedriver
Selecting previously unselected package google-chrome-stable.
(Reading database ... 9178 files and directories currently installed.)
Unpacking google-chrome-stable (from google-chrome-
stable_49.0.2623.112-1_amd64.deb) ...
Setting up google-chrome-stable (49.0.2623.112-1) ...
xdg-icon-resource: No writable system icon directory found.
dpkg: error processing google-chrome-stable (--install):
 subprocess installed post-installation script returned error exit status 3
Errors were encountered while processing:
 google-chrome-stable
The command '/bin/sh -c cd /opt/ && wget -q
https://chromedriver.storage.googleapis.com/${CHROME_DRIVER_VERSION}/chromed
river_linux64.zip && unzip chromedriver_linux64.zip && mv chromedriver
/usr/local/bin/ && chmod +x /usr/local/bin/chromedriver && rm -f
chromedriver_linux64.zip && dpkg -i google-chrome-
stable_49.0.2623.112-1_amd64.deb' returned a non-zero code: 1
```

Какого же было мое удивление, но google chroome перестал устанавливаться с какой то невнятной ошибкой - «**xdg-icon-resource: No writable system icon directory found.**». Но я то был уверен, что для работы google chrome точно не требует java. Начал искать описание ошибки в интернете. Но явного и четкого решения не смог найти. Было 100500 подобных вопросов и в каждом случае решение было каким то уникальным, типа перегрузил компьютер и ошибка исчезла. Но так как мы не сдаемся просто так, то в бой пошел strace.

Запуск strace внутри docker

Комментируем строку с установкой google chrome, добавляем установку пакета strace, пересобираем образ и подключаемся к нему для отладки.

И здесь меня ждало тоже неожиданность - нельзя просто так взять и запустить strace внутри docker контейнера. При попытке запустить strace - я получал подобную ошибку

```
# docker run -it --rm chrome:strace bash
# strace dpkg -i google-chrome-stable_49.0.2623.112-1_amd64.deb
```

```
strace: ptrace(PTRACE_TRACEME, ...): Operation not permitted
```

После небольшого поиска находим информацию о том, что для возможности использовать strace нам необходимо запускать контейнер с флагом –cap-add SYS_PTRACE. Более подробно можно прочитать тут. Запускаем с нужным флагом и получаем ОЧЕНЬ много отладочной информации. Как правило нужная информация связанная с ошибкой располагается в конце вывода, но вы можете перенаправить весь вывод в файл с помощью ключа -o debug.log

```
# docker run --cap-add SYS_PTRACE -it --rm chrome:strace bash
# strace dpkg -i google-chrome-stable_49.0.2623.112-1_amd64.deb
execve("/usr/bin/dpkg", ["dpkg", "-i", "google-chrome-
stable_49.0.2623.1"...], [/* 14 vars */]) = 0
brk(0)                                  = 0x167b000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f972b3de000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=15544, ...}) = 0
mmap(NULL, 15544, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f972b3da000
close(3)                                = 0
...
...
...
openat(AT_FDCWD, "/var/lib/dpkg/updates/",
O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 9
fsync(9)                                = 0
close(9)                                = 0
close(4)                                = 0
munmap(0x7fe43e175000, 4096)            = 0
unlink("/var/lib/dpkg/updates/tmp.i")   = 0
fcntl(3, F_SETLK, {type=F_UNLCK, whence=SEEK_SET, start=0, len=0}) = 0
write(2, "Errors were encountered while pr"..., 42Errors were encountered
while processing:
) = 42
write(2, " google-chrome-stable\n", 22 google-chrome-stable
) = 22
exit_group(1)                           = ?
```

К сожалению данный трейс тоже не дал нам никакой полезной информации. Как мне подсказал позже один мой знакомый - необходимо запускать с флагом -f (follow forks). Пробуем еще раз запустить и теперь ближе к концу вывода мы увидим следующие строки

```
# docker run --cap-add SYS_PTRACE -it --rm chrome:strace bash
# strace -f dpkg -i google-chrome-stable_49.0.2623.112-1_amd64.deb
...
...
...
[pid    76] faccessat(AT_FDCWD, "/usr/share/icons/hicolor", W_OK) = -1
```

```
ENOENT (No such file or directory)
[pid    76] fcntl(1, F_DUPFD, 10)       = 11
[pid    76] close(1)                    = 0
[pid    76] fcntl(11, F_SETFD, FD_CLOEXEC) = 0
[pid    76] dup2(2, 1)                  = 1
[pid    76] write(1, "xdg-icon-resource: No writable s"..., 60xdg-icon-
resource: No writable system icon directory found.
) = 60
[pid    76] dup2(11, 1)                 = 1
[pid    76] close(11)                   = 0
[pid    76] exit_group(3)               = ?
Process 76 detached
[pid    73] <... wait4 resumed> [{WIFEXITED(s) && WEXITSTATUS(s) == 3}], 0,
NULL) = 76
[pid    73] --- SIGCHLD (Child exited) @ 0 (0) ---
[pid    73] rt_sigreturn(0x11)          = 76
[pid    73] exit_group(3)               = ?
Process 60 resumed
Process 73 detached
<... wait4 resumed> [{WIFEXITED(s) && WEXITSTATUS(s) == 3}], 0, NULL) = 73
--- SIGCHLD (Child exited) @ 0 (0) ---
write(2, "dpkg: error processing google-ch"..., 133dpkg: error processing
google-chrome-stable (--install):
 subprocess installed post-installation script returned error exit status 3
```

Думаю, что теперь из вывода стало понятно, в чем заключается проблема - для успешной установки директория /usr/share/icons/hicolor должна быть создана. Но почему же тогда работало при установке openjdk-6-jre? Ответ нам дает утилита apt-file

```
# apt-file search /usr/share/icons/hicolor | grep openjdk-6-jre
openjdk-6-jre: /usr/share/icons/hicolor/16x16/apps/openjdk-6.png
openjdk-6-jre: /usr/share/icons/hicolor/24x24/apps/openjdk-6.png
openjdk-6-jre: /usr/share/icons/hicolor/32x32/apps/openjdk-6.png
openjdk-6-jre: /usr/share/icons/hicolor/48x48/apps/openjdk-6.png
```

Т.е. получается, что люди, которые изначально создавали Dockerfile устанавливали java 6 только ради директории /usr/share/icons/hicolor/!

Исправляем запуск google chrome

Самое тривиальное решение - просто создать данную папку в самом Dockerfile

```
RUN cd /opt/ && \
    wget -q
https://chromedriver.storage.googleapis.com/${CHROME_DRIVER_VERSION}/chromed
river_linux64.zip \
    && unzip chromedriver_linux64.zip \
    && mv chromedriver /usr/local/bin/ \
```

```
    && chmod +x /usr/local/bin/chromedriver \
    && rm -f chromedriver_linux64.zip \
    && mkdir -p /usr/share/icons/hicolor/ \
    && dpkg -i google-chrome-stable_49.0.2623.112-1_amd64.deb

WORKDIR /opt
```

Теперь сборка контейнера завершается успешно

```
Step 6/7 : RUN cd /opt/ && wget -q
https://chromedriver.storage.googleapis.com/${CHROME_DRIVER_VERSION}/chromed
river_linux64.zip && unzip chromedriver_linux64.zip && mv chromedriver
/usr/local/bin/ && chmod +x /usr/local/bin/chromedriver && rm -f
chromedriver_linux64.zip && mkdir -p /usr/share/icons/hicolor/ && dpkg -i
google-chrome-stable_49.0.2623.112-1_amd64.deb
 ---> Running in 95c9c0310b53
Archive:  chromedriver_linux64.zip
  inflating: chromedriver
Selecting previously unselected package google-chrome-stable.
(Reading database ... 9187 files and directories currently installed.)
Unpacking google-chrome-stable (from google-chrome-
stable_49.0.2623.112-1_amd64.deb) ...
Setting up google-chrome-stable (49.0.2623.112-1) ...
update-alternatives: using /usr/bin/google-chrome-stable to provide
/usr/bin/x-www-browser (x-www-browser) in auto mode.
update-alternatives: using /usr/bin/google-chrome-stable to provide
/usr/bin/gnome-www-browser (gnome-www-browser) in auto mode.
update-alternatives: using /usr/bin/google-chrome-stable to provide
/usr/bin/google-chrome (google-chrome) in auto mode.
 ---> 3e7f756b91cc
Removing intermediate container 95c9c0310b53
Step 7/7 : WORKDIR /opt
 ---> e851af942013
Removing intermediate container 05c19f5c0028
Successfully built e851af942013
Successfully tagged chrome:strace
```

Если мы посмотрим содержимое данного каталога, то увидим следующую структуру

```
# docker run -it --rm chrome:strace ls -la /usr/share/icons/hicolor/
total 40
drwxr-xr-x 10 root root 4096 Aug 27 13:43 .
drwxr-xr-x  3 root root 4096 Aug 27 13:43 ..
drwxr-xr-x  3 root root 4096 Aug 27 13:43 128x128
drwxr-xr-x  3 root root 4096 Aug 27 13:43 16x16
drwxr-xr-x  3 root root 4096 Aug 27 13:43 22x22
drwxr-xr-x  3 root root 4096 Aug 27 13:43 24x24
drwxr-xr-x  3 root root 4096 Aug 27 13:43 256x256
drwxr-xr-x  3 root root 4096 Aug 27 13:43 32x32
drwxr-xr-x  3 root root 4096 Aug 27 13:43 48x48
```

```
drwxr-xr-x  3 root root 4096 Aug 27 13:43 64x64
```

Но думаю, что более правильный метод - найти легковесный пакет, который копирует файлы при установке. В этом нам опять таки поможет apt-file

```
# apt-file search /usr/share/icons/hicolor | awk '{print $1}' | sort | uniq
| wc -l
938
```

Вам осталось только выбрать ;) Я остановился на пакете hicolor-icon-theme