Low Latency Sockets

или коротко о SO_BUSY_POLL

Рыбак Евгений, Минск

Модель OSI

Блоки	Транспортный безопасное и надёжное соединие точка-точка
Пакеты	Сетевой Определение пути и IP (логическая адресация)
Кадры	Канальный MAC и LLC (Физическая адресация)
Биты	Физический кабель, сигналы, бинарная передача данных

Методы обработки входящих пакетов

- Метод прерываний (IRQ)
- Метод опроса (NAPI polling)
- Совмещенный метод (IRQ & NAPI polling)

IRQ

Методы обработки входящих пакетов: прерывания (IRQ)

Прерывание генерируется на каждый кадр

Top halve

Обработчик прерывания

Bottom halve

«Рабочая» функция

NAPI poll

Ночному клубу "Ночной клуб" срочно требуется креативный директор ...

Методы обработки входящих пакетов: NAPI poll

- *NAPI - New Api*
- *Вместо прерываний на каждый кадр, вычитываем из буфера драйвера N кадров*

Без SO_BUSY_POLL

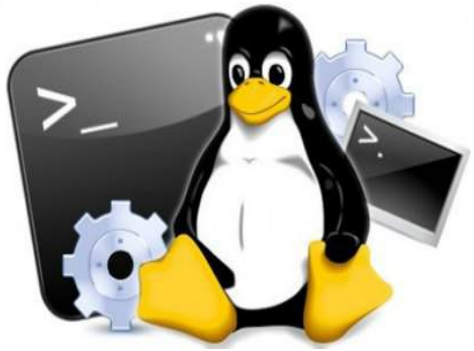
1. Приложение создает сокет и пытается из него прочитать.
2. Данные отсутствуют, ожидаем момент когда они появятся в буфере сокета
3. Данные приходят на интерфейс
4. Генерируется прерывание, данные передаются на уровень протоколов
5. Данные передаются в буфер сокета
6. Приложение может прочитать данные из сокета

SO_BUSY_POLL

Обзор опции SO_BUSY_POLL

Поддержка ядра

Поддержка драйвера сетевой карты



Настройки системы

Поддержка ядра

- Ядро версии 3.11+ должно быть скомпилированно с опцией CONFIG_NET_RX_BUSY_POLL

- *<linux\netdevide.h>*

  struct net_device_ops

  *int(*ndo_busy_poll)(struct napi_struct *dev);*

- *<linux\skbuff.h>*
- *<linux\sock.h>*
- *<net\socket.c>*

Поддержка драйвера сетевой карточки

- Драйвер сетевой карточки должен реализовать функцию с прототипом *int(*ndo_busy_poll)(struct napi_struct *dev);*
- Инициализировать указатель *ndo_busy_poll структуры net_device_ops*

Настраиваем SO_BUSY_POLL

- Глобальные настройки для всех сокетов в системе:

  sysctl net.core.busy_read=50

  sysctl net.core.busy_poll=50

- Локально для сокета:

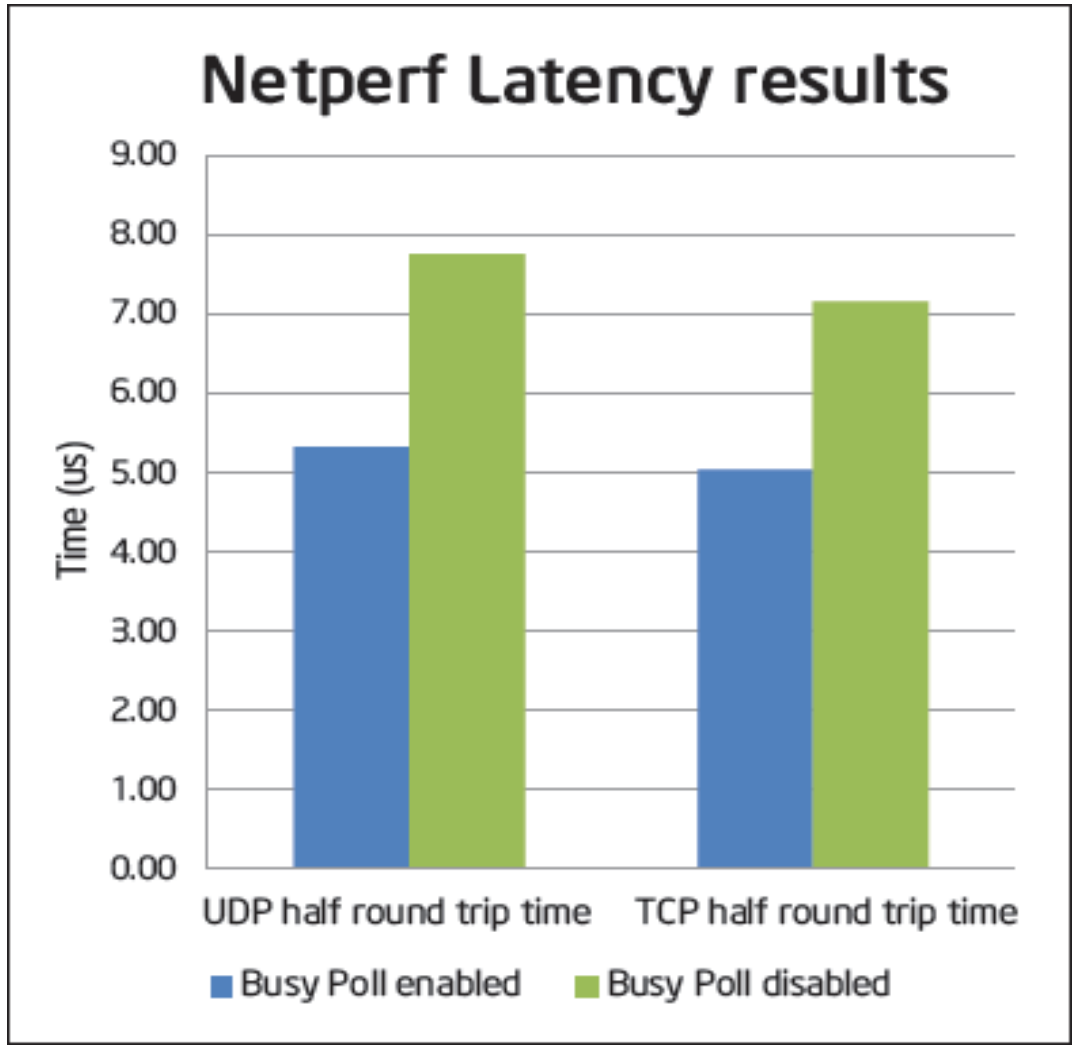
  int busy_read = 50;

  setsockopt(sockfd, SOL_SOCKET, SO_BUSY_POLL, & busy_read, sizeof(int))

SO_BUSY_POLL

1. Приложение создает сокет и пытается из него прочитать.
2. Данные отсутствуют, ожидаем момент
2. Переходим к функции драйвера busy poll
3. Данные приходят на интерфейс
4. Данные передаются в буфер сокета
5. Приложение может прочитать данные из сокета

SO_BUSY_POLL и производительность



Разница busy_poll vs NAPI poll

- Busy poll инициирует «принудительную» передачу пакетов из входящей очереди драйвера в очередь сокета, тем самым исключая «лишнее» переключение контекста и прерывание.

- обработчика NAPI poll, система передает количество пакетов, которое оно готово обработать в данный момент. Если количество пакетов в очереди драйвера больше, чем количество пакетов, которое система готова обработать в текущий момент, то обработчик NAPI poll будет еще раз поставлен в очередь вызовов; через некоторый промежуток времени обработчик будет вызван в очередной раз.

Спасибо

Вопросы?

Рыбак Евгений
Mr.Engler@yahoo.com

Источники

- A way towards Lower Latency and Jitter, pdf
- Open Source Kernel Enhancements for Low Latency Socket, pdf
- net: low latency Ethernet device polling
- Understanding Linux Network Internals
- Linux Kernel Development
- Linux Kernel Sources 3.18.14