Средство анализа объемов трафика пользователей суперкомпьютера

К.Э. Енокян

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Целью данной работы является создание программного компонента, осуществляющего подсчет входящего и исходящего трафика различных пользователей суперкомпьютера. Было получено средство для анализа объемов трафика, реализованное в виде модуля ядра Linux. Данный программный компонент осуществляет подсчет трафика, результат которого, предоставляется пользователю по запросу. Обеспечивается возможность работы на многопроцессорной системе. Более наглядное представление данных осуществляется разработанным веб-интерфейсом. Программное средство успешно проходит эксплуатацию на суперкомпьютере «Ломоносов» суперкомпьютерного комплекса МГУ.

Ключевые слова: подсчет трафика, Linux, модуль ядра, Ломоносов, веб-интерфейс.

1. Введение

В современном мире потребление вычислительных ресурсов – средств вычислительной системы, которые могут быть выделены процессу обработки данных на определенный интервал времени, огромно, и потому возникает необходимость вести их учет. Делать это нужно для нескольких целей. Во-первых, это оплата за использование вычислительной системы. Во-вторых, потребность в обеспечении распределения ресурсов между пользователями. И, в-третьих, для различного рода анализа учтённых ресурсов, к примеру, для выявления дефицита какого-то вида ресурсов или тех ресурсов, которые используются неэффективно.

Для таких ресурсов, как процессорное время, оперативная память, энергопотребление, существуют специализированные средства для подсчета и анализа их потребления. Но так как системы становятся все сложнее, а видов ресурсов становится все больше, то для аналитики важен учет всех видов ресурсов. Одним из таких ресурсов является внешний трафик суперкомпьютера. Учёт его необходимо осуществлять по нескольким причинам. Во-первых, несмотря на то, что трафик по сравнению, например, с электричеством стоит недорого, необходимо так же рассчитывать, какие средства нужны на оплату. И, во-вторых, учёт трафика даёт возможность понять, какие внешние каналы необходимы и как они используются разными пользователями. Поэтому возникла необходимость создания программного средства, осуществляющего подсчет трафика по различным пользователям компьютера.

Целью работы является создание программного средства, осуществляющего подсчет объемов трафика, создаваемого пользователями суперкомпьютера. Созданное средство должно осуществлять подсчет входящего и исходящего трафика с разделением по отдельным пользователям. Средство должно быть пригодно к применению на современных суперкомпьютерах.

2. Существующие решения рассматриваемой задачи

Чаще всего, когда стоит задача подсчета трафика, подразумевается подсчёт трафика пользователей у провайдеров доступа к Интернет. В этом случае различные пользователи используют разные IP-адреса, а, значит, разделение трафика разных пользователей не составляет труда. Для решения такой задачи можно использовать, например, Netflow, IP accounting или данные от счётчиков на интерфейсах к пользователю.

Netflow – сетевой протокол, предназначенный для учёта сетевого трафика, разработанный компанией Cisco Systems [1]. Для сбора информации о трафике по протоколу Netflow требуются сенсор, собирающий статистику по проходящему через него трафику, коллектор, который собирает получаемые от сенсора данные и помещает их в хранилище, и анализатор, – который анализирует собранные коллектором данные и формирует отчёты. Netflow использует UDP или SCTP для передачи данных о трафике коллектору. Сенсор выделяет из проходящего трафика потоки, характеризуемые такими параметрами, как адрес источника, адрес назначения, порт источника для UDP и TCP, порт назначения для UDP и TCP, номер протокола IP и другие. Когда сенсор определяет, что поток закончился, информация отправляется в коллектор. Собранная информация отправляется в виде записей, содержащих информацию о входящем и исходящем сетевом интерфейсе, адресах источника и назначения, портах источника и назначения, количестве байт и пакетов в потоке, номере протокола IP и др. То есть содержится информация необходимая для анализа трафика пользователей, использующих разные IP-адреса.

IP accounting – также средство, разработанное Cisco, которое позволяет собирать информацию о сетевом трафике [2]. Это средство считает количество байт и пакетов на основе IP-адресов отправителя и получателя. Отличие от Netflow состоит в том, что IP accounting считает трафик, покидающий маршрутизатор, тогда как Netflow считает трафик, на входящем интерфейсе маршрутизатора. Также IP accounting собирает информацию о трафике, с разделением по IP-адресам, но без разделения по портам. Данный вариант решения не подходит для учета трафика пользователей, не использующих разные IP-адреса.

Другой способ учета трафика – его подсчет на прокси-серверах. В данном случае становится возможным разделение трафика по пользователям одного сервера, так как на прокси пользователи авторизуются с именем и паролем.

Для данной цели служит, например, Squid – программный пакет, использующийся в UNIX-подобных системах, который реализует функцию кэширующего прокси-сервера для протоколов HTTP, FTP и HTTPS [3]. Информация о каждом запросе, проходящем через squid, записывается в лог-файл – файл, хранящий служебную информацию. Далее лог-файл просматривается, анализируется и составляется отчёт о трафике.

Те же функции выполняет и прокси-сервер для Microsoft Windows – WinGate [4]. Аналогично squid, информация о запросе записывается в лог-файл и впоследствии обрабатывается сторонней программой для подсчета трафика – ProxyInspector, которая анализирует лог-файл и выдаёт отчёт о трафике [5].

Однако, применительно к поставленной задаче, подсчет трафика с использованием прокси-сервера, по нескольким причинам не подходит. Во-первых, относительно данной задачи потребовались бы большие ресурсы для организации прокси-серверов из-за пропускной способности канала. А, во-вторых, такой подход не универсален, так как не все используемые протоколы могут работать через прокси-сервер. Например, SCP – протокол копирования файлов, использует в качестве транспорта сетевой протокол SSH. Потому из-за необходимости большей легковесности и универсальности, а также необходимости учета всего трафика, включая SSH, было реализовано решение, описанное в данной работе.

3. Подсистемы ядра Linux, использованные для решения задачи

Разработанное программное средство представляет собой модуль ядра Linux, реализующий цель для правил утилиты Iptables и добавляющий эту цель в библиотеку Iptables. Каждый входящий и исходящий пакет обрабатывается в соответствие с созданной целью – производится подсчет числа входящих и исходящих байт и пакетов. Доступ к информации о трафике осуществляется с помощью виртуальной файловой системы /proc, а более наглядное представление данных с помощью разработанного веб-интерфейса.

Общая организация обработки пакета при помощи разработанного средства представлена на рисунке 3.1.

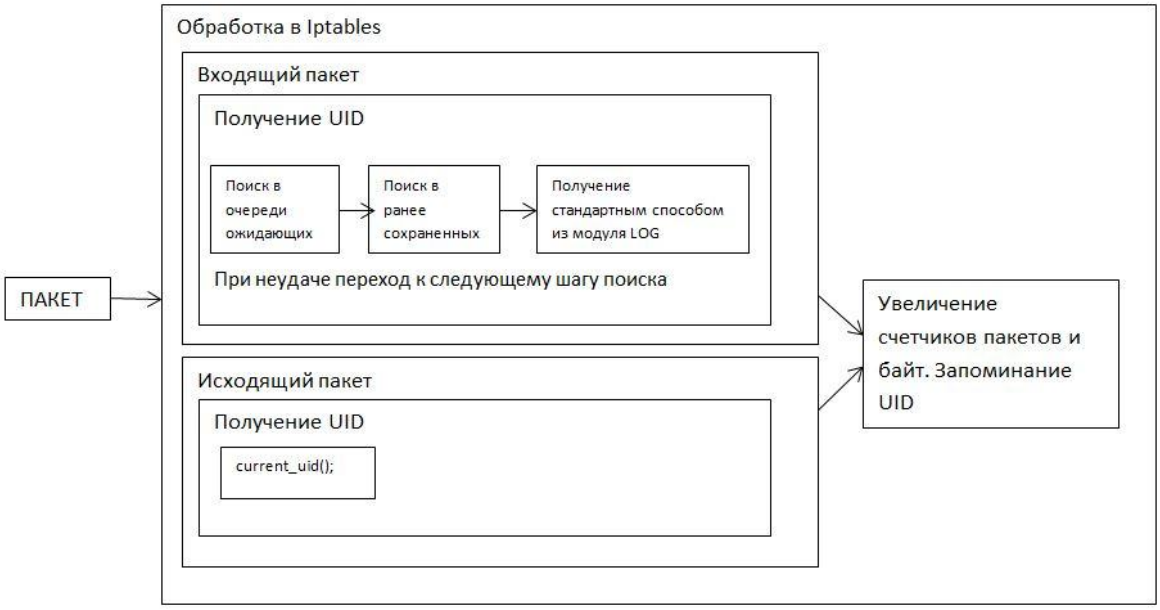

Рис. 3.1. Обработка пакета

Общая организация доступа к полученным данным представлена на рисунке 3.2.

Рис. 3.2. Доступ к полученным данным

3.1 Пакетный фильтр Iptables

Iptables – это стандартный интерфейс управления работой межсетевого экрана NETFilter для ядер Linux. Для использования утилиты Iptables требуются привилегии суперпользователя. Ключевыми понятиями являются [6]:

- Критерий — логическое выражение, анализирующее свойства пакета и/или соединения и определяющее, подпадает ли данный конкретный пакет под действие текущего правила.
- Действие — описание действия, которое нужно проделать с пакетом и/или соединением в том случае, если они подпадают под действие этого правила.

• Счетчик — компонент правила, обеспечивающий учет количества пакетов, которые попали под критерий данного правила.

• Правило — состоит из критерия, действия и счетчика. Если пакет соответствует критерию, к нему применяется действие, и он учитывается счетчиком.

• Цепочка — упорядоченная последовательность правил.

• Таблица — совокупность цепочек, объединенных общим функциональным назначением.

Все пакеты пропускаются через определенные для них последовательности цепочек. При прохождении пакетом цепочки, к нему последовательно применяются все правила этой цепочки в порядке их следования. Под применением правила понимается: во-первых, проверка пакета на соответствие критерию, и во-вторых, если пакет этому критерию соответствует, применение к нему указанного действия. Под действием может подразумеваться как элементарная операция – встроенное действие, например, ACCEPT, MARK; – так и переход в одну из пользовательских цепочек. В свою очередь, действия могут быть как терминальными, то есть прекращающими обработку пакета в рамках данной базовой цепочки, например, ACCEPT, REJECT, так и нетерминальными, то есть не прерывающими процесса обработки пакета. Если пакет прошел через всю базовую цепочку, и к нему так и не было применено ни одного терминального действия, к нему применяется действие по умолчанию для данной цепочки, обязательно терминальное.

3.2 Получение информации об UID

Одна из функций, которую обеспечивает Iptables, – это запоминание служебной информации в системный журнал. Данную функцию реализует модуль LOG, реализующий цель с тем же именем, позволяющий записывать в системный журнал информацию о пакете – длину, адрес отправителя, адрес получателя, количестве байт и пакетов и др.

Получение информации о UID пользователя, реализованное в модуле LOG, осуществляется при помощи сокета. Сокет – это программный интерфейс для обеспечения межпроцессного взаимодействия. При таком обмене процессы могут исполняться как на одном компьютере, так и на разных компьютерах, связанных сетью. В модуле LOG, реализован способ получения информации об UID пользователя, представляющий собой получение данных о текущем UID с помощью сокета. Происходит это так: для каждого приходящего или исходящего пакета, определяется запись в ядре о сокете, а далее по сокету определяются учетные данные о владельце пакета, которые запоминаются в момент создания сокета.

Однако такой способ получения информации о UID пользователя не является достаточным. Такой вариант получения UID, путем поиска записи о сокете в ядре и получения по сокету учетных данных владельца, не работает в случае SSH/SCP. SSH – сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой, который шифрует весь трафик, включая и передаваемые пароли. SCP – протокол копирования файлов, использующий в качестве транспорта SSH. Способ, реализованный в модуле LOG, в случае SSH/SCP дает неверные результаты, так как изначально серверные процессы ssh запускаются от имени суперпользователя, который имеет UID равный 0. Это необходимо для переключения выполнения на права пользователя, который подключился по SSH, а такое переключение доступно только суперпользователю. Еще одна причина изначального запуска с UID, равным 0 состоит в том, что в Linux системе открыть сокет на привилегированном порту (порту с номерами ниже 1024), может только суперпользователь, а в случае SSH необходимо открывать сокет на 22 порту. По этим причинам сокет открывается от суперпользователя, и UID равный 0 записывается в свойства сокета, как владелец. Далее серверный процесс SSH производит переключение на пользователя, от имени которого дальше будет производиться работа. Однако переключение на другого пользователя не меняет записи о владельце сокета, в ядре linux не предусмотрено такого механизма. По этой причине в способе получения UID, реализованном в модуле LOG, весь трафик по SSH/SCP считается с UID равным 0, поэтому потребовались дополнительные способы получения UID, так как на суперкомпьютеры пользователи подключаются по SSH, а данные передаются по SCP, и этот трафик тоже необходимо учитывать.

В разработанном программном средстве способ получения UID, реализованный в модуле LOG, также применяется, но наряду с другими способами. Общая организация получение UID, реализованная в разработанном средстве, представлена на рисунке 3.3. Если на одном из шагов не удалось найти UID, то происходит переход на следующий шаг поиска.

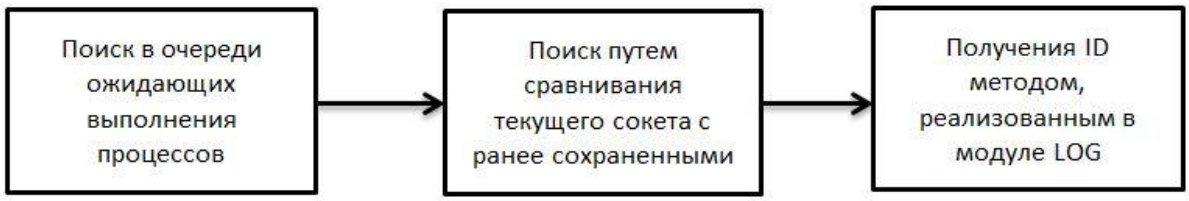

Рис. 3.3. Получение UID

Одним из таких способов является получения UID пользователя с помощью очереди ожидающих процессов. В системе Linux хранится информация о каждом активном процессе, а также список всех процессов, ожидающих выполнения. Когда пакет попадает на обработку в модуль LOG, специальная функция реализует поиск UID процесса-владельца сокета, однако такой способ поиска не единственный. С каждым сокетом связана очередь процессов, ждущих прихода данных в этом сокете. Если в момент обработки пакета очередь процессов, ожидающих данных на сокете, соответствующем пакету, не пуста, то самый первый процесс в этой очереди является процессом, который получит данный пакет. Это справедливо для входящих пакетов. Доступ к первому процессу осуществляется при помощи структуры данных ядра, описывающей сокет, в котором содержится информация об очереди записей о процессах. Отсюда осуществляется доступ к информации о первом процессе в очереди, а, значит, и к учетным данным, содержащим UID пользователя, запустившего этот процесс. Таким образом, может быть решена задача нахождения UID пользователя через очередь процессов. Пример этого способа представлен на рисунке 3.4.

```
        kuid_t my_kuid;
        struct user_namespace *cur_ns;
        uid_t my_current_uid;
        wait_queue_t *tmp;
        struct task_struct *mytask;
        const struct cred *mycred;
        struct socket_wq *wq;

        wq = rcu_dereference(sk->sk_wq);
        if (wq_has_sleeper(wq))
        {
                list_for_each_entry(tmp, &wq->wait.task_list, task_list)
                {
                        if (tmp->private &&
                                (tmp->func == autoremove_wake_function ||
                                        tmp->func == default_wake_function))
                        {
                                 mycred = __task_cred(mytask);
                                 my_kuid = task_uid(mytask);
                                 goto found;
                        }
                }
        }
found:   my_current_uid=from_kuid_munged(cur_ns, my_kuid);
```

Рис. 3.4. Поиск UID в очереди ожидающих процессов

Другим способом получения UID для входящего пакета является способ поиска соответствия текущего сокета с одним из сокетов, запомненных ранее на исходящих пакетах. Если к моменту приходу пакета, требующего обработку, как входящего, уже известны

некоторые данные о сокете и UID пользователя пакетов исходящих, то есть сохранялась пара значений: сокет, UID, то получить информацию о UID пользователя входящего пакета можно произведя поиск по сокетам по сохраненным данным. И в случае, если сокеты совпали, то нужный UID пользователя найден. Реализация данного метода представлена на рисунке 3.5.

```
uid_t my_current_uid;
int sk_users = 0;

struct SockUid
{
     struct sock *sk;
     uid_t myuid;
};

struct SockUid *skUid;

skUid[sk_users].sk = sk;
skUid[sk_users].myuid = my_current_uid;
sk_users++;

for (i=0; i<sk_users; i++)
{
     if (sk == skUid[i].sk)
          my_current_uid = skUid[i].myuid;
}
```

Рис. 3.5. Поиск UID путем сравнения с одним из ранее сохраненных

В разработанном модуле TRAF_ACCT используется следующий алгоритм нахождения UID для входящих пакетов. По каждому входящему пакету сначала определяется его сокет, далее определяется, пуста ли очередь ждущих процессов на этом сокете, если нет, то производится попытка получить UID пользователя, используя способ с использованием очереди процессов, упомянутый выше. Если данный способ не дал результатов, либо же очередь ждущих процессов оказалась пуста, производится попытка получения UID при помощи поиска соответствия сокета одному из ранее запомненных на исходящих пакетов. Если и такой метод не дал результатов, то есть ранее не было пакета с таким же сокетом, то UID получается тем способом, который используется в модуле LOG (данные о владельце процесса, открывшего этот сокет). Такой алгоритм нахождения UID дает результаты, похожие на ожидания.

Для исходящих же пакетов в момент обработки пакета известна информация об учетных данных текущего процесса. Тем самым известна информация о процессе, которому предназначается пакет, а, значит, о UID пользователя запустившего этот процесс. Получение UID исходящего пакета показано на рисунке 3.6.

```
my_uid = current_uid();
```

Рис. 3.6. Получение UID исходящего пакета

3.3 Модуль ядра, реализующий подсчет трафика

Существующий модуль LOG лишь даёт возможность записи служебной информации в системный журнал. Изучение его исходного кода позволяет понять, каким образом происходит получение ID пользователя. Реализованное программное средство также представляет собой модуль ядра Linux, но со всем необходимыми для данной задачи функциями.

Модуль подсчёта трафика, в дальнейшем модуль TRAF_ACCT, реализует цель, которую можно использовать в правилах, и добавляет эту цель в библиотеку Iptables. Вызов подобного правила, позволяющего подсчитывать трафик, осуществляется добавлением к команде цели, представленной на рисунке 3.7.

```
-j TRAF_ACCT
```

Рис. 3.7. Разработанная цель

Разработанное программное средство производит подсчет входящего и исходящего трафика. Выбрать направление можно при помощи указание INPUT или OUTPUT в цепочке правила.

Модуль TRAF_ACCT предоставляет функцию подсчёта числа входящих пакетов и байт, исходящих пакетов и байт по каждому конкретному пользователю, идентифицируемому UID. Хранение информации о UID пользователя, количестве входящих байт, количестве входящих пакетов, количестве исходящих байт и исходящих пакетов осуществляется в соответствующих полях структуры.

Также модуль обеспечивает функции обнуления счётчиков входящих и исходящих байт и пакетов и запоминания информации о данных по запросу пользователя. При сбросе по запросу пользователя накопленной информации в хранилище, реализуется копирование данных в структуру аналогичную существующей для сохранения всех полученных данных.

В данном программном средстве предусмотрен интерфейс для взаимодействия с пользователем, а также сторонний веб-интерфейс, не включенный в модуль ядра, для представления полученных данных.

3.4 Интерфейс для взаимодействия пользователя с модулем ядра

В Linux используется виртуальная файловая система /proc. Изначально /proc разрабатывалась как средство предоставления информации о выполняющихся в системе процессах. Но из-за ее удобства многие подсистемы ядра стали использовать эту файловую систему как средство предоставления информации и динамического конфигурирования. Файловая система /proc содержит каталоги и виртуальные файлы. Виртуальный файл, может предоставлять пользователю информацию, полученную из ядра и, кроме того, служить средством передачи в ядро пользовательской информации [7].

Данная виртуальная файловая система используется в разработанном модуле для организации интерфейса для взаимодействия с пользователем. Для создания виртуального файла в /proc используется функция proc_create, принимающая в качестве параметров имя создаваемого файла, режим доступа к файлу, один из подкаталогов /proc для его размещения и операции, необходимые для выполнения. В разработанном программном средстве создание виртуального файла происходит при инициализации модуля.

Разработанный модуль собирает и хранит большие объемы информации, но реализация больших файлов в /proc затруднительна. Для решения данной проблемы был использован интерфейс seq_file. Этот интерфейс предоставляет простой набор функций для реализации больших виртуальных файлов ядра. Без использования seq_file было бы необходимо хранит в памяти строку со всей информацией, которая бы выводилась по запросу пользователя. С использованием же данного интерфейса достаточно сохранять информацию в структурах, как это и реализовано в модуле, а по запросу, при помощи функций из seq_file, информация из структур будет выведена пользователю.

В реализованном программном средстве доступ к полученным данным осуществляется путем запроса, представленного на рисунке 3.8.

```
cat /proc/Traffic
```

Рис. 3.8. Запрос на получение информации о текущем трафике

Аналогично, доступ к информации из хранилища осуществляется по запросу, представленному на рисунке 3.9. Подробнее о хранилище в п.3.5.

```
cat /proc/Traffic_storage
```

Рис. 3.9. Запрос на получение информации о трафике из хранилища

Команда cat в Linux – это утилита, которая позволяет сцеплять, связывать файлы, а также выводить содержимое файла.

По запросу на отображение информация выдается в виде последовательных строк, содержащих данные о UID пользователя и соответствующему ему количеству входящих байт и пакетов и исходящих байт и пакетов, как это показано на рисунке 3.10.

```
TRAFFIC
UID 1000  bytes_in: 16411 packets_in: 40 bytes_out: 4828 packets_out: 29
UID 65534 bytes_in: 0     packets_in: 0  bytes_out: 1448 packets_out: 16
UID 0     bytes_in: 200   packets_in: 5  bytes_out: 1207 packets_out: 17
```

Рис. 3.10. Общий вид получаемых данных

3.5 Управление

Разработанный модуль предоставляет функцию обнуления накопленных данных по запросу пользователя. Вид данного запроса представлен на рисунке 3.11.

```
1) echo 0 > /proc/Traffic
2) echo 0 > /proc/Traffic_storage
```

Рис. 3.11. Запрос на обнуление текущего трафика и хранилища

В строке 1 рисунка приведен запрос для обнуления структуры данных, в строке 2 запрос для обнуления структуры, являющейся хранилищем собранной информации.

Echo – это команда Linux, предназначенная для отображения строки текста. А при запросах указанных на рисунке 3.5 эта команда записывает то, что стоит после нее в указанный файл, в данном случае, /proc/Traffic или /proc/Traffic_storage.

При запросе в модуль поступает информация о необходимости обнулить накопленные в структуре данные о входящих пакетах и байтах и исходящих пакетах и байтах. После чего происходит непосредственное обнуление структуры данных.

В разработанном программном средстве также предусмотрена функция запоминания всех накопленных данных. Данная функция реализуется через пользовательский запрос, представленный на рисунке 3.12.

```
echo 1 > /proc/Traffic
```

Рис. 3.12. Запрос на сохранение текущего трафика в хранилище

Данный запрос поступает в модуль, и далее обрабатывается по следующему алгоритму: все данные из структуры сбора информации копируются в аналогичную структуру для хранения информации. Запоминается UID пользователя, количество входящих байт и пакетов и исходящих байт и пакетов соответствующих этому пользователю. Далее все поля структуры сбора информации, кроме UID пользователей, обнуляются. Тем самым последние сохраненные данные соответствуют данным за промежуток времени от предыдущего запоминания до текущего момента времени.

Информацию, сохраненную в хранилище, можно просмотреть через запрос, который был представлен на рисунке 3.9. в п.3.4.

3.6 Работа на многопроцессорной системе

Для обеспечения возможности работы на многопроцессорной системе, потребовалось избавиться от борьбы за ресурсы – то есть синхронизировать разные процессы.

Были добавлены атомарные операции для работы со структурами, которые хранят данные о трафике. Атомарные операции – это операции, выполняющиеся как единое целое, либо не выполняющиеся вовсе. Атомарность операций имеет особое значение в многопроцессорных системах, так как доступ к разделяемым ресурсам должен быть обязательно атомарным. Атомарная операция открыта влиянию только одного потока. Добавление атомарных операций избавило от возможности одновременного изменения счетчиков конкурирующими процессами.

Также в разработанный модуль были добавлены спин-блокировки. Спин-блокировка – это низкоуровневый примитив синхронизации, применяемый в многопроцессорных системах для реализации взаимного исключения [8]. Спин-блокировка является взаимным исключением устройства, которое может иметь только два значения: заблокировано и разблокировано. Физически спин-блокировка представляет собой переменную в памяти и реализуется на атомарных операциях. Каждый процессор, желающий получить доступ к разделяемому ресурсу, атомарно записывает условное значение “занято” в эту переменную. Если предыдущее значение переменной было “свободно” то считается, что данный процессор получил доступ к ресурсу, в противном случае, процессор находится в цикле, ожидая, пока разделяемый ресурс будет освобождён. После работы с разделяемым ресурсом процессор, использовавший этот ресурс, должен записать в переменную памяти, идентифицирующую спин-блокировку, условное значение “свободно”.

Добавление спин-блокировок позволило избавиться от возможности чтения и записи данных разными процессами в одно и то же время.

В итоге, была обеспечена возможность работы модуля на многопроцессорной системе.

3.7 Веб-интерфейс для представления данных

Для представления полученных данных был разработан веб-интерфейс, осуществляющий анализ и предоставление данных пользователю. Разработанный интерфейс состоит из программы для сбора данных и программы для анализа собранных данных и их представления в виде таблиц и графиков.

Программа сбора данных с некоторых выбранным интервалом по времени осуществляет запоминание полученных модулем TRAF_ACCT данных в хранилище, используя функцию запоминания данных, описанную в п.3.5. Далее информация из хранилища считывается, форматируется и записывается в файл с расширением .csv. Имя файла соответствует дате сбора информации, то есть при непрерывной работе данной программы информация о потребленном трафике будет разделена по файлам, соответствующим датам потребления. В результате работы программы сбора данных создаются файлы в формате csv, пригодные для просмотра в Excel и LibreOffice.

Однако для более наглядного представления данных была разработана программа, являющаяся веб-интерфейсом, которая анализирует полученные файлы формата csv. При открытии html страницы пользователю предоставляется возможность выбрать даты, за время которых требуется увидеть потребленный трафик. После выбора в программе осуществляется загрузка файлов, соответствующих этим датам. Выбрав даты, пользователь далее выбирает время, за которое необходимо получить объемы потребленного трафика. После выбора промежутка времени, пользователю по умолчанию предоставляется информация в виде таблицы о топ10 пользователях, с наибольшим объемом входящего трафика, а так же предоставляется график суммарного потребления всеми пользователями входящего, исходящего и совместного трафика за выбранный период времени.

Пользователю предоставляется возможность изменения числа, по которому будет построен топ пользователей в таблице. Также есть возможность выбора, отображать топ пользователей по входящему трафику, как по умолчанию, либо же по исходящему или суммарному трафику. Пример данной таблицы представлен на рисунке 3.13.

Выбранный промежуток времени также возможно изменять в процессе работы веб-интерфейса.

Date (from): 18 04 2016 **(to)** 24 04 2016

Time (from): 08:00:00 **(to)** 08:00:00

Input Output Both

Top 20 Users

№	User ID	Input Bytes	Input Packets	Output Bytes	Output Packets	Bytes sum	Packets sum
1	0	3632056482	18322461	40721123942	13225163	44353180424	31547624
2	10061	1133003517	2682612	15103729848	2312915	16236733365	4995527
3	10026	167434778	2532884	11615458476	5026205	11782893254	7559089
4	10046	122101750	898399	7425861603	3135448	7547963353	4033847
5	10016	141297940	225609	3729497436	1379465	3870795376	1605074
6	10035	221115353	3059218	2091538220	749213	2312653573	3808431
7	10007	1399815034	1135680	507224376	357569	1907039410	1493249
8	10080	1342508684	904678	29525220	172808	1372033904	1077486
9	10079	706397584	485511	26535696	262428	732933280	747939
10	10019	43124784	410728	547899057	580134	591023841	990862
11	10017	14636794	51152	365824211	221717	380461005	272869
12	10055	10220	166	353848980	90632	353859200	90798
13	10066	2941592	25801	346259288	626038	349200880	651839
14	10037	155264592	267301	135515684	63738	290780276	331039
15	10020	33454128	192135	247372031	61676	280826159	253811
16	10069	4220981	57138	255608729	63782	259829710	120920
17	93	258615308	204531	533257	4916	259148565	209447
18	10068	334708	4982	115785808	25341	116120516	30323
19	10042	33326160	42367	63017480	37621	96343640	79988
20	74	3880571	47990	64490284	136403	68370855	184393

Рис. 3.13. Пример работы веб-интерфейса: таблица топ пользователей

График также имеет разные виды представления. Помимо суммарного представления трафика за выбранный период, установленного по умолчанию, есть опция для отображения топ5 или топ10 пользователей по суммарному объему потребления. Пример графика, предоставляемого разработанным веб-интерфейсом, показан на рисунке 3.14.

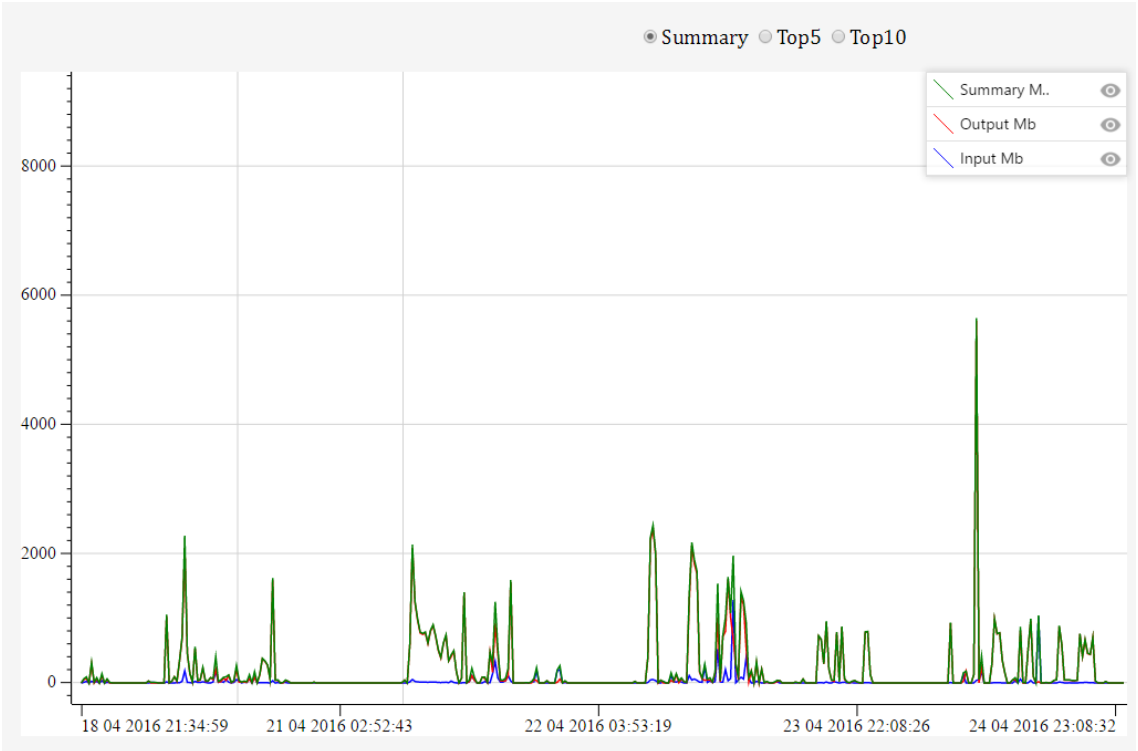

Рис. 3.14. Пример работы веб-интерфейса: график суммарного потребления трафика

3.8 Тестовая эксплуатация

Разработанное программное средство в данный момент проходит успешную тестовую эксплуатацию на суперкомпьютере Ломоносов. Получаемые данные могут предоставляться как с помощью низкоуровневого интерфейса для взаимодействия пользователя с модулем – по запросу, представленному на рисунке 3.5, так и с помощью разработанного веб-интерфейса, предоставляющего более наглядные данные, а также позволяющего выбирать промежуток времени, за который эти данные необходимо отобразить.

Для определения накладных расходов, вносимых разработанным программным средством было проведено тестирование производительности. На двух серверах с интерфейсами Ethernet 1 Гбит/с, соединенных общим коммутатором, была запущена утилита iperf. На сервере, где проводились измерения, установлены 2 десятиядерных процессора Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz. по 20 ядер HyperThreading на каждом процессоре. Был произведен запуск iperf сначала в режиме клиента на 40 потоков, затем в режиме сервера. На другой машине соответственно запускался сервер или клиент на 40 потоков. При помощи утилиты top была измерена загрузка процессора в режиме ядра ОС. При работе iperf в режиме клиента загрузка составляла 0.1-0.2%, при работе в режиме сервера – 0.3-0.4%. Значимых отличий в загрузке при работе без подсчета трафика, то есть без загруженного модуля, и с подсчетом трафика, то есть с загруженным модулем и установленными правилами Iptables для подсчета, обнаружить не удалось. Таким образом, дополнительная нагрузка, создаваемая подсчетом трафика, лежит в пределах 0.1%. Пропускная способность, измеренная iperf, по всех случаях равнялась 947Мбит/с, т.е. на пропускную способность подсчет трафика тоже не влияет. Из этого можно сделать вывод, что разработанное средство не вносит существенных накладных расходов в работу системы.

4. Заключение

В результате данной работы был разработан модуль ядра Linux, инициализирующий правило для подсчета трафика отдельных пользователей. Для удобного и наглядного представления полученных данных был разработан веб-интерфейс, предоставляющий информацию о трафике за выбранный интервал времени. Разработанное программное средство, в виде модуля ядра Linux для подсчета трафика, в данный момент успешно проходит тестовую эксплуатацию на суперкомпьютере Ломоносов.

Литература

1. Чубин И. Netflow. URL: http://xgu.ru/wiki/NetFlow.
2. Клайз Б., Волтер В. Подсчет Ip через ip sctp аутентификацию. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp/iap-cr-book/iap-i1.html.
3. Вессель Д. Squid: Оптимизация веб-сайтов. URL: http://www.squid-cache.org/.
4. Кьюбик Д. Wingate 8. URL: http://www.wingate.ru/products.php?todo=view&id=1.
5. Кьюбик Д. ProxyInspector. URL: http://www.wingate.ru/products.php?todo=view&id=7.
6. Андреассен О. Руководство по iptables. URL: https://www.opennet.ru/docs/RUS/iptables.
7. Корбет Д., Рубини А., Кроан-Хартман Г. Драйверы устройств Linux, третье издание. URL: http://dmilvdv.narod.ru/Translate/LDD3/ldd_debugging_querying.html.
8. Корбет Д., Рубини А., Кроан-Хартман Г. Драйверы устройств Linux, третье издание. URL: http://dmilvdv.narod.ru/Translate/LDD3/ldd_spinlocks.html.

Tool for the accounting of the supercomputer's users' traffic

K. Enokyan

Moscow State University

Faculty of computational mathematics and Cybernetics

The aim of this work is to develop a software component which counts incoming and outgoing traffic of different users. A tool was created for the traffic accounting. The tool is implemented as Linux kernel module. This software component performs the traffic accounting, the result of which is provided to the user upon request. A visual representation of the data is provided by the web interface. The developed tool is being evaluated on Lomonosov supercomputer.

Keywords: traffic accounting, Linux, kernel module, Lomonosov, web-interface.

References

1. Chubin I. Netflow. URL: http://xgu.ru/wiki/NetFlow.
2. Claise B., Wolter R. Podschet Ip cherez ip sctp autentifikatsiyu. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp/iap-cr-book/iap-i1.html.
3. Wessels D. Squid: Optimizatsiya veb-saytov. URL: http://www.squid-cache.org/.
4. Qbik D. Wingate 8. URL: http://www.wingate.ru/products.php?todo=view&id=1.
5. Qbik D. ProxyInspector. URL: http://www.wingate.ru/products.php?todo=view&id=7.
6. Andreasson O. Rukovodstvo po iptables. URL: https://www.opennet.ru/docs/RUS/iptables.
7. Corbet D., Rubini A., Chroan-Hartman G. Drayvery ustroystv Linux, tret'e izdanie. URL: http://dmilvdv.narod.ru/Translate/LDD3/ldd_debugging_querying.html.
8. Corbet D., Rubini A., Chroan-Hartman G. Drayvery ustroystv Linux, tret'e izdanie. URL: http://dmilvdv.narod.ru/Translate/LDD3/ldd_spinlocks.html.