УДК 004.457

СИСТЕМА КОНТРОЛЯ АППАРАТНЫХ МЕТРИК. АРХИТЕКТУРА, РЕАЛИЗАЦИЯ

А. А. Богданов, Н. Н. Залялов, А. А. Ярулина
(РФЯЦ-ВНИИЭФ, г. Саров)

Дается описание программной системы контроля аппаратных метрик СКАМ, предназначенной для выявления сбоев или отклонений в работе оборудования больших вычислительных комплексов. Приводится классификация архитектур аналогичных систем. Описывается модульная децентрализованная архитектура СКАМ и особенности ее реализации, приведены избранные результаты использования СКАМ.

Ключевые слова: система контроля аппаратных метрик, протокол IPMI, система мониторинга, вычислительный комплекс, модульная децентрализованная архитектура.

Введение

Современные вычислительные комплексы (ВК) содержат значительное число единиц оборудования. Время от времени на устройствах ВК происходят сбои, и тем чаще, чем больше устройств. Чтобы поддерживать функционирование ВК на уровне, достаточном для эксплуатации, необходимо эти сбои своевременно выявлять.

ВК обычно включает в себя устройства, предоставляющие информацию о работе оборудования с помощью различных открытых или проприетарных протоколов — SNMP (Simple Network Management Protocol), IPMI и др. Современные ВК генерируют большой поток данных, которые нужно обработать, придать им вид, удобный для пользователя.

Средствами для сбора данных служат встроенные контроллеры устройств и соответствующее программное обеспечение (ПО) — система мониторинга (СМ). Примером встроенных контроллеров являются BMC (Baseboard Management Controller) серверов, разного рода контроллеры присутствуют также практически во всех инфраструктурных компонентах ВК.

Управление функционированием ВК в простых случаях заключается либо в замене устройства (в случае выхода из строя), либо в перезагрузке операционной системы сервера (в случае программного сбоя). Управление осуществляется на основе данных, собранных СМ.

В большинстве случаев количество контролируемых параметров в ВК растет с увеличением его производительности. Следовательно, необходимо, чтобы СМ ВК выполняла свои функции независимо от размеров ВК.

Существующие СМ ВК построены на основе архитектуры, имеющей ограничения в масштабируемости из-за централизованных компонентов: СУБД, сбора данных, пользовательского интерфейса. Архитектура описываемой системы контроля аппаратных метрик СКАМ свободна от подобных компонентов.

Архитектура

Основной задачей СМ является предоставление данных для принятия решений относительно состояния оборудования ВК. Масштабируемость СМ — это ее свойство выполнять свою задачу независимо от количества мониторируемого оборудования.

Способность системы выполняться на нескольких процессорах одного сервера назовем SMP-расширяемостью, на нескольких серверах — MPP-расширяемостью. СМ, обладающая обеими этими способностями, может одинаково хорошо функционировать как на нескольких серверах, так и на одном.

SMP-расширяемость достигается за счет распределения приложения по нескольким процессорам одного сервера. Назовем такое прило-

жение сервисом. Сервис СМ — это компонент СМ, функционирующий на одном сервере. Если несколько сервисов могут выполнять одинаковые функции для разного оборудования и предоставлять единообразный доступ к результатам выполненных функций, то такая система сервисов является MPP-расширяемой, или, другими словами, распределенной СМ. Если, в такой системе отсутствует центральный элемент, то ее можно назвать децентрализованной.

Архитектура СМ ВК. ПО, предназначенное для мониторинга ВК, многообразно. Существует большое количество подобных систем — одни узкоспециализированы, другие объединяют несколько функций. Однако почти всегда в структуре СМ можно выделить подсистемы сбора, обработки, хранения, визуализации данных.

По наличию тех или иных подсистем, способам их взаимодействия архитектуры СМ можно отнести к пяти типам. Номера типов архитектур соответствуют номерам иллюстраций на рис. 1, где овалом обозначается приложение мониторинга, прямоугольником без надписи — мониторируемое устройство, прямоугольником с надписью — пользовательское приложение, кругом, разделенным на секторы, — сервис, состоящий из нескольких модулей. Для иерархий архитектуры третьего типа приложение СМ запускается на мониторируемом сервере, поэтому совпадает с мониторируемым устройством.

Итак, на рис. 1 представлены следующие типы СМ:

1. *Клиент-сервер.* Простейшее ПО для получения данных. Приложение мониторинга обращается к одному серверу с запросом на данные. К этому типу относятся Ping, Ipmitool [1], NetSNMP [2] и т. п.
2. *Мультиклиент-сервер.* Приложение мониторинга обращается с запросами одновременно к нескольким серверам. Представители этого типа — OpenHPI [3], Freeipmi [4], Sysman [5] и т. п.
3. *Однородная иерархия.* В этом случае мультиклиент сам становится сервером и предоставляет данные мультиклиенту, находящемуся на другом уровне иерархии. Иерархия строится из копий одной и той же программы. Представители этого типа — Ganglia (без поддержки Gmetad) [6], Cerebro [7], Supermon [8] и т. п.
4. *Двухуровневая иерархия.* На первом (верхнем) уровне системы такого типа имеют средство визуализации, хранения данных или и то, и другое. ПО первого уровня является мультиклиентом по отношению к

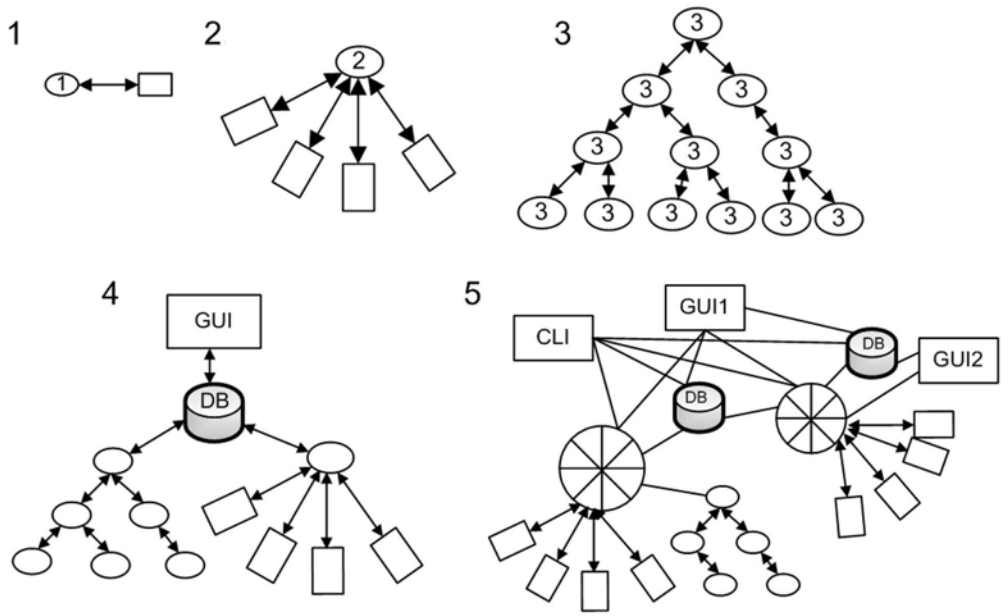

Рис. 1. Типы архитектур СМ (GUI — графический пользовательский интерфейс; CLI — интерфейс командной строки; DB — база данных)

ПО второго уровня. Приложения второго уровня получают данные от сборщиков произвольного типа. Несколько верхних уровней могут выступать в роли второго уровня в случае объединения данных от нескольких ВК. Отличие такой иерархии от однородной в том, что в *листьях* и узлах *дерева* функционирует разное ПО. Представители этого типа — Gmetad (Ganglia), Nagios [9], Zabbix [10], Zenoss [11], Lemon [12], Cpanel [13], OpenOVIS [14] и т. п.

5. *Модульные системы.* У этих систем нет привязки к какому-либо графу связей. Разнообразие функций реализуется модулями. В различных поставках таких систем неизменными остаются только средства межмодульного взаимодействия, остальные свойства могут варьироваться. Сервис может объединять в себе один или несколько модулей, реализующих сбор, передачу, хранение, обработку или визуализацию данных. К модульным системам относятся Collectd [15], OpenScada [16].

Первые три типа СМ обычно не подразумевают наличия подсистемы долговременного хранения и предназначены для доступа к текущим значениям мониторируемых метрик, т. е. они не позволяют анализировать историю поведения ВК.

Системы четвертого типа имеют жестко заданные пользовательский интерфейс и подсистемы обработки. Их масштабируемость и расширяемость полностью зависят от масштабируемости и расширяемости этих компонентов.

Пятый тип СМ лишен жестких рамок и может быть настроен на функционирование в режиме любого из остальных типов. Если модули СМ, а также средства их объединения реализованы с поддержкой многопоточности, то такая СМ будет SMP-расширяемой. Наличие модуля, обеспечивающего взаимодействие между несколькими сервисами либо между сервисами и пользовательскими приложениями, обеспечит MPP-расширяемость. Если один из сервисов выполняет особую роль, обеспечивающую функционирование всех элементов СМ, то такая СМ является распределенной, но централизованной. Если же все сервисы равноправны, то система распределенная и децентрализованная. Масштабируемость децентрализованной системы не зависит от масштабируемости какого-либо сервиса в отдельности.

Следовательно, наиболее подходящей для масштабируемой СМ является архитектура, совмещающая в себе модульность и децентрализованность. СКАМ построен на основе такой архитектуры. Чтобы отличать от остальных, назовем ее модульной децентрализованной. В рамках приведенной классификации она относится к пятому типу.

Рассмотрим свойства расширяемости применительно к архитектурам, приведенным в классификации. SMP-расширяемость можно не рассматривать, так как ею обладают большинство СМ независимо от архитектуры. MPP-расширяемость распространена не настолько.

Первые два типа исключаются из рассмотрения, так как у них всегда используется только одна утилита.

Третий тип по определению является MPP-расширяемым, так как подразумевает запуск одинаковых взаимодействующих приложений на разных серверах. Однако каждое из этих приложений получает данные только от сервера, на котором работает. Такие приложения не предназначены для обработки данных, собираемых централизованно. Поэтому MPP-расширяемость систем с однородной иерархией ограниченна.

Двухуровневые СМ лишены этого недостатка, так как в рамках единой базы данных могут объединяться данные как от нескольких однородных иерархий, так и от централизованных сборщиков. В этом случае также можно сказать, что MPP-расширяемость присутствует, однако она имеет место только тогда, когда ее поддерживает центральный элемент двухуровневой СМ. То есть архитектура двухуровневой иерархии обладает свойством MPP-расширяемости лишь условно.

В полной мере MPP-расширяемостью обладают модульные системы, аналогичные Scada. В частности, это OpenScada и Collectd, однако в силу особенностей реализации они слабо подходят для целостного мониторинга ВК. OpenScada предназначена для получения данных от промышленных контроллеров, а Collectd работает только с событиями и не предоставляет явных инструментов для обработки текущего состояния системы.

Модульная децентрализованная архитектура СКАМ принадлежит к пятому типу, поэтому является MPP-расширяемой. SMP-расширяемость обеспечивается реализацией модулей СКАМ. В архитектуре СКАМ предусмотрены механизмы работы как с событиями, так и с текущими

состояниями устройств. Поэтому архитектура СКАМ является лучшим выбором для реализации децентрализованной системы мониторинга.

Архитектура СКАМ. Модульная децентрализованная архитектура не накладывает каких-либо ограничений на объем обрабатываемого потока данных. Модульность позволяет использовать различные сборщики данных, а децентрализованность — выполнять сбор и обработку параллельно. Ограничения снимаются благодаря возможности добавлять модули либо распределять нагрузку между несколькими серверами мониторинга.

Модули СКАМ оперируют двумя типами данных — метриками и событиями. Метрика — это числовое значение, характеризующее состояние некой части мониторируемой системы. Метрикой может быть показание датчика, значение счетчика и т. п. Событие — это описание изменения в состоянии системы. Каждое событие привязано к моменту времени.

Каждый сервис СКАМ хранит массив метрик. Совокупность таких массивов образует массив состояний мониторируемой системы. Сервисы СКАМ всегда хранят последние состояния метрик и предоставляют их для считывания. События не хранятся в рамках сервисов. Обнаруженное событие обрабатывается модулями сервиса и либо удаляется, либо сохраняется для дальнейшей обработки. Каждый модуль может генерировать событие и поставить его в очередь (FIFO) на обработку.

Таким образом, объединение модулей осуществляется на основе общего доступа к массиву состояний и механизма обмена событиями. В качестве прототипа для массива состояний использовалось понятие фазового пространства состояний системы, описанное в [17].

Модули реализуют функции обновления и обработки массива метрик, обработки событий, т. е. решают задачи СМ. Объектные коды модулей локализуются в динамических библиотеках — по библиотеке на модуль. Вспомогательные функции, такие как конфигурирование или журналирование, вынесены в исполняемый файл (Skamd). Во время выполнения этот файл динамически присоединяет библиотеки (модули), указанные в настройках, и становится сервисом СКАМ.

Достоинство такой организации — в возможности использования разными сервисами разных комбинаций модулей. Благодаря этому разные сервисы могут выполнять различные комбинации функций СМ (сбор данных, сохранение или обработку). Если ресурсов одного сервера мониторинга достаточно для функционирования СМ, то используется один сервис. Если же ресурсов одного сервера мониторинга недостаточно или некоторые данные не могут поступать на один сервер мониторинга, то модули могут быть распределены по нескольким сервисам.

Код Skamd выполняется при запуске процесса сервиса, его остановке, а также при обработке сигналов операционной системы. В остальное время выполняется код модулей несколькими *нитями* внутри процесса сервиса.

На рис. 2 показана структура отдельного сервиса. Прямоугольники с горизонтальными надписями символизируют части исполняемого файла.

Реализация

На текущий момент полностью реализованы четыре модуля:

1) Sclient (libsclient.so) — предоставляет интерфейс к собираемым данным JSON (JavaScript Object Notation);
2) Sipmi (libsipmi.so) — собирает данные по протоколу IPMI (Intelligent Platform Management Interface);
3) Sdb (libsdb.so) — сохраняет детектируемые события в базе данных;
4) Sib (libsib.so) — собирает данные от InfiniBand-устройств с помощью пакетов MAD (Management Datagram).

На рис. 3 проиллюстрирована одна из возможных конфигураций СКАМ. Три сервиса (S1, S2,

Рис. 2. Структура сервиса СКАМ

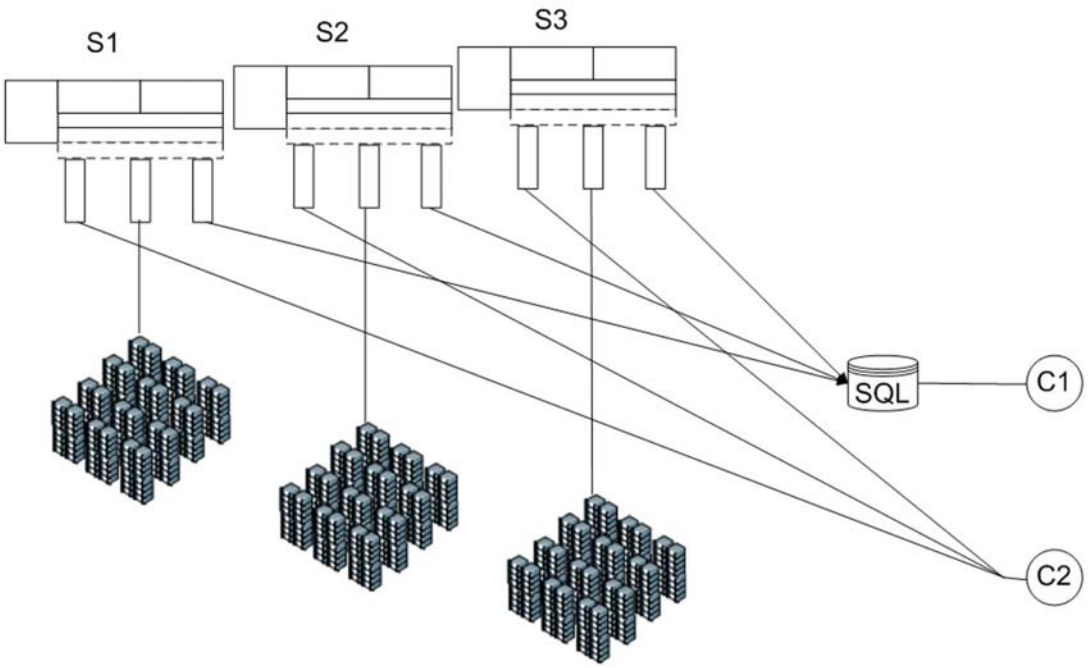

Рис. 3. Используемая конфигурация СКАМ (S1, S2, S3 — сервисы; C1, C2 — пользовательские приложения)

S3) обрабатывают данные, получаемые с трех разных частей ВК. В каждом сервисе функционируют три модуля: сбор данных, сохранение данных, облуживание пользовательского приложения.

Собранные показания датчиков передаются пользователю по его запросу. При необходимости пользовательское приложение объединяет данные от нескольких сервисов. Аппаратные события и журналы операционной системы записываются в базу данных и также предоставляются пользователю по запросу. Набор модулей сервиса, а также настройки каждого из них определяются в конфигурационных файлах. Сервисы могут функционировать на различных серверах мониторинга.

Децентрализация позволяет мониторировать, с одной стороны, ВК произвольного размера при недостаточной производительности одного сервера мониторинга, а с другой стороны — несколько разделенных ВК как единое целое.

Исполняемый файл сервиса СКАМ. Исполняемый файл Skamd выполняет процедуры запуска и останова модулей. Запуск — это создание и заполнение структур данных, загрузка и регистрация каждого из модулей, запуск соответствующих нитей. Останов — вызов для каждого из модулей функции завершения и освобождение занятой модулем памяти.

Кроме запуска модулей, Skamd создает и инициализирует структуры данных, предназначенные для конфигурирования, журналирования и обмена данными между модулями (структуры для хранения массива метрик и обработки событий).

Доступ к данным. Доступ пользователя к собираемым данным организован с помощью модуля доступа Sclient и пользовательского приложения. Модуль доступа создает серверную нить, "слушающую" сетевой порт. Пользовательское приложение посылает на этот порт запрос и соответственно получает ответ. В качестве протокола выбран JSON-RPC 2.0 [18] как приемлемый компромисс между избыточностью XML и негибкостью бинарных протоколов. Пользовательское приложение имеет интерфейс командной строки.

На рис. 4, 5 представлены два примера визуализации данных. На рис. 4 показаны группы устройств в формате *списка отклонений*, аналогичном используемому в Pdsh [19] и Lemon [12]. Здесь каждому уровню возможных состояний (хорошее, подкритическое, критическое и т. п.) ставится в соответствие группа устройств. На рис. 5 данные представлены в виде *карты по-*

```
Status of sensors
not inited(00000):
timeout(00000):
disabled(00000):
not set(00000):
unknown(00000):
ok(00129): ab[2008-2011],fb[2001],vb[7000-7123]
noncritical(00000):
critical(00000):
nonrecoverable(00000):
[bogdanov@ae2010 ~]$
```

Рис. 4. Визуализация в виде списка отклонений

ля — таблицы, где каждому устройству ставится в соответствие один символ, характеризующий состояние устройства. Строки, не содержащие ошибок, не выводятся — это сокращает выдачу и повышает наглядность.

Время отклика на запрос пользовательского приложения. На текущий момент компонентами СМ, ограничивающими масштабируемость СКАМ, являются пользовательское приложение и модуль доступа к данным. На рис. 6 приведена зависимость времени ответа сервиса СКАМ на запрос одной метрики для визуализации списка отклонений (см. рис. 4) и карты поля (см. рис. 5) от количества устройств, на рис. 7 — от количества метрик для 50 тыс. устройств.

Из рисунков видно, что запрос списка отклонений масштабируется лучше, чем запрос карты поля, и время отклика на запрос списка отклонений для 100 тыс. устройств не превышает 10 с.

Как видно, зависимость времени отклика СКАМ не сильно отличается от линейной. Для ВК с числом мониторируемых устройств $\sim 10^4$ и числом метрик $\sim 10^2$ время отклика составляет ~ 1 с. Для ВК большего размера пользовательское приложение, возможно, потребует переработки.

```
Status of sensors
      0         1         2         3         4         5         6         7         8         9
      0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789
-------------------------------------------------------------------------------------------------------------------
ab 20 .........++ ++......... .......... .......... .......... .......... .......... .......... .......... .......... 20
fb 20 .+......... .......... .......... .......... .......... .......... .......... .......... .......... .......... 20
vb 70 ++++++++++ ++++++++++ ++++++++++ ++++++++++ ++++++++++ ++++++++++ ++++++++++ ++++++++++ ++++++++++ ++++++++++ 70
   71 ++++++++++ ++++++++++ ++++...... .......... .......... .......... .......... .......... .......... .......... 71
-------------------------------------------------------------------------------------------------------------------
      0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789 0123456789
      0         1         2         3         4         5         6         7         8         9
not inited(d):00000 timeout(t):00000 disabled(i):00000 not set(s):00000 unknown(u):00000 ok(+):00129 noncritical(*):00000
critical(#):00000 nonrecoverable(X):00000
[bogdanov@ae2010 ~]$
```

Рис. 5. Визуализация в виде карты поля

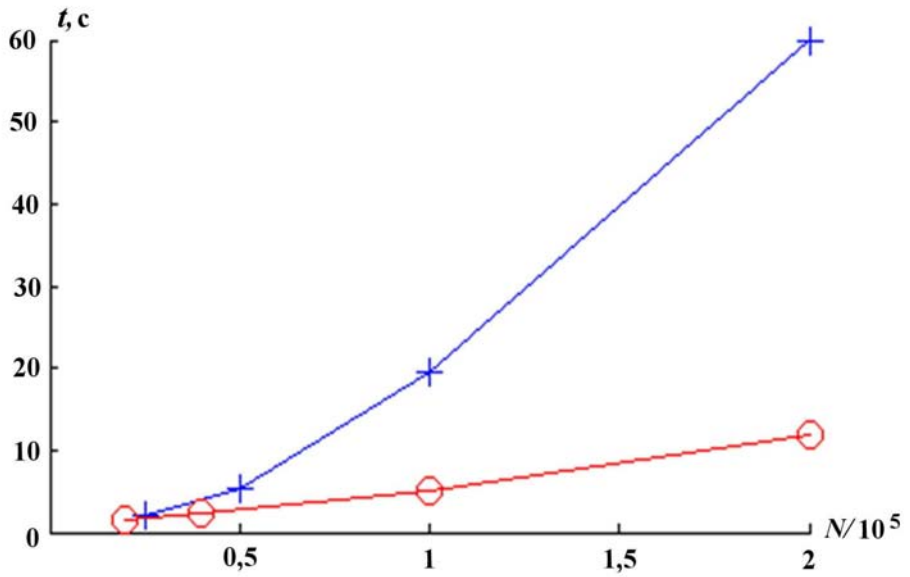

Рис. 6. Зависимость времени отклика на запрос от количества устройств N: —+— — для карты поля; —○— — для списка отклонений

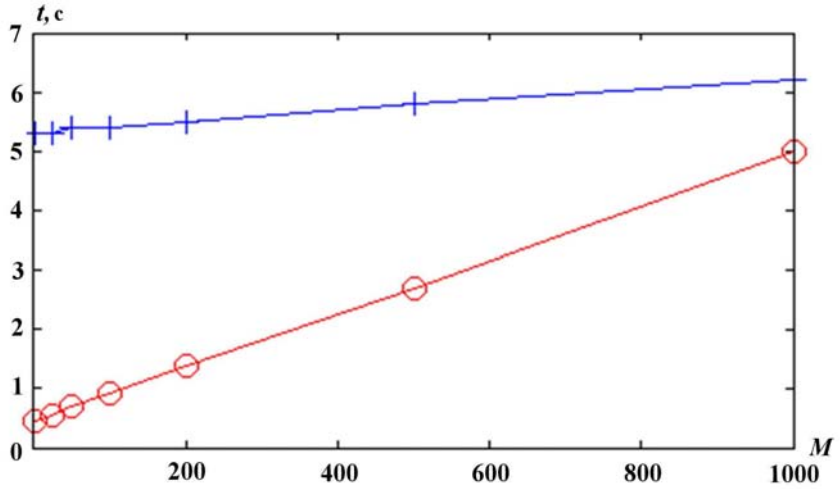

Рис. 7. Зависимость времени отклика на запрос от числа метрик для $5 \cdot 10^4$ устройств: —+— — для карты поля; —○— — для списка отклонений

Модуль сбора IPMI-метрик и консольных журналов. Модуль Sipmi выполняет одновременно две функции: сканирует состояние датчиков и собирает содержимое консольных журналов мониторируемых серверов.

Существующие аналоги Sipmi (программы, собирающие данные по протоколу IPMI [20]) — Ipmitool [1], OpenIPMI [21], FreeIPMI [4], IpmiUtil [22], OpenHPI [3] — реализуют эти возможности отдельными функциями или даже приложениями. Поэтому для них каждый запрос датчиков требует инициализации новой IPMI-сессии. Это значительно увеличивает количество передаваемых сетевых пакетов в каждом запросе на получение показаний датчиков.

В случае сбора консольных журналов IPMI-сессия должна подтверждаться, как минимум, каждую минуту [20], т. е. должен совершаться по крайней мере один обмен между пользовательским приложением и BMC сервера. Обычно для такого обмена используется пустой IPMI-запрос. Sipmi совмещает опрос датчиков и получение консольных журналов, благодаря чему количество передаваемых сетевых пакетов значительно уменьшается.

Частично Sipmi может быть заменен модулем Rsyslog [23]. Rsyslog предназначен для передачи файлов системных журналов разных серверов в единое хранилище. Он не использует ни протокола IPMI, ни ресурсов BMC. Rsyslog выполняется непосредственно в операционной системе контролируемого сервера, поэтому может влиять на процессы, исполняемые на серверах. Sipmi не имеет компонентов, функционирующих в операционной системе мониторируемого сервера, и соответственно не вносит помех в работу пользовательских приложений.

Sipmi предназначен для организации и длительного поддержания большого числа IPMI-сессий, которое может достигать десятков тысяч. Поэтому подход, когда на каждую сессию либо каждый пакет приходится отдельная нить, не эффективен из-за затрат на переключение контекстов. Последовательная обработка приходящих пакетов не масштабируется на SMP-системах.

Наиболее приемлемый вариант — использовать пул нитей, обрабатывающих пакеты выделенного набора BMC. Одна нить принимает пакеты (*получатель*) и распределяет их между *обработчиками* (рис. 8).

Skamd запускает основную нить Sipmi, которая создает дополнительные нити получателей и обработчиков IPMI-пакетов. Количество получателей и приходящихся на каждый из них обработчиков определяется в конфигурационном файле.

Получатель ожидает прихода пакетов с помощью мультиплексирующей функции epoll. По получении очередного IPMI-пакета исходя из номера сокета источника определяется номер нити обработчика, далее указатель на пакет передается выбранному потоку. После того как поток-обработчик становится активным, пакеты выбираются из очереди, пока она не опустеет.

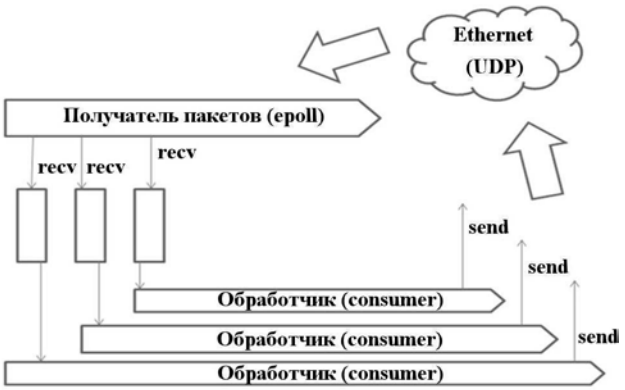

Рис. 8. Цикл обработки пакетов

Благодаря такому подходу организована SMP-расширяемость данного модуля. В зависимости от объема мониторируемого оборудования и конфигурации модуля можно задействовать различное количество процессоров.

Модуль передачи событий в базу данных. Регистрация событий позволяет осуществлять анализ их последовательностей во времени с привязкой к конкретным устройствам. Для событий в отличие от числовых последовательностей можно использовать реляционную СУБД. Модуль Sdb не "привязан" к конкретной СУБД, но на текущий момент реализована поддержка только PostgreSQL.

Код Sdb выполняется одной нитью, обрабатывающей очередь событий. Период обработки очереди задается в конфигурационном файле. Пополнение очереди выполняется нитью обработки сообщений.

База данных. Таблицы, а также другие элементы базы данных (индексы, серверные функции) генерируются автоматически при первом доступе модуля к СУБД, что упрощает администрирование. Таблица для каждого типа событий генерируется при первой регистрации события соответствующего типа. Тип события — это тип его источника (журнал операционной системы, BMC, InfiniBand-устройства и т. п.).

События представляются в виде строки, привязанной ко времени и устройству. Каждое устройство имеет два идентификатора — позиционный (идентификатор позиции, куда устройство установлено) и уникальный (идентификатор самого устройства). Источники идентификаторов определяются модулями-сборщиками. Так, например, для Sipmi позиционное имя — это доменное имя (hostname) BMC, а уникальный идентификатор — MAC-адрес BMC. Для Sib позиционный идентификатор — это имя InfiniBand-устройства, а уникальный идентификатор — GUID (Globally Unique Identifier) этого устройства. Соответственно события от BMC будут храниться в таблице с префиксом MAC, а события от InfiniBand — в таблице с префиксом GUID. Также для каждого префикса можно создавать несколько таблиц, по таблице на каждый тип событий.

Выбор идентификатора обеспечивает модулю универсальность, автоматическое создание таблиц — удобство администрирования, а схема с двойными идентификаторами позволяет автоматически отслеживать миграцию оборудования внутри вычислительного поля, что может быть важно в случае возврата устройств после ремонта.

Средства доступа к содержимому базы данных. Для доступа к информации, записанной в базе данных, реализованы консольные утилиты. Sdb предоставляет обобщенный доступ к содержимому базы и имеет достаточно сложный интерфейс, а Console-, Sib- и Sevents-сценарии упрощают работу с Sdb. Специальная утилита Maps позволяет получать карты поля, аналогичные приведенным на рис. 4, 5. Console предназначен для упрощенной работы с консольными журналами. Sevents — средство для упрощенной работы с событиями, генерируемыми BMC.

Модуль сбора InfiniBand-метрик. Создание собственного сборщика метрик InfiniBand обусловлено прекращением поддержки использовавшегося ранее средства Ibmon, входящего в состав IbAdm [24], и отсутствием некоммерческих аналогов.

Модуль сбора InfiniBand-метрик (Sib) периодически опрашивает устройства коммуникационной системы и считывает счетчики портов. Каждый InfiniBand-порт имеет несколько счетчиков ошибок и объема передаваемых/принимаемых данных.

Опрос устройства InfiniBand осуществляется путем обмена пакетами SMP (Subnet Management Packets), которые являются разновидностью MAD-пакета. Тип маршрутизации, используемый при опросе, — маршрутизация SMP при помощи локального идентификатора LID (Local Identifier) [25].

В рамках модуля каждое устройство InfiniBand однозначно определяется глобальным уникальным идентификатором GUID. В основной структуре данных предусмотрена возможность хранения различной информации как об устройствах (имя, тип, количество портов, LID, описание устройства, имя устройства и т. д.), так и о портах устройств (номер, показания счетчиков). Формирование структуры данных осуществляется в два этапа: на первом формируется список доступных устройств путем опроса менеджера подсети, на втором этапе опрашиваются сами устройства для получения значений метрик.

Модуль Sib позволяет отслеживать текущие состояния InfiniBand-адаптеров и коммутаторов, а также собирать, хранить изменения счетчиков ошибок, происходящих на портах устройств. Сохранение выполняется путем передачи регистрируемых событий модулю Sdb для их записи в базу данных. Благодаря использованию реляционной базы данных анализ собранной информации упрощается в отличие от Ibmon, который хранил данные в файлах.

Избранные результаты использования

Эксперименты показали, что СКАМ способен обслуживать не менее 300 тыс. метрик ВК со скоростью более 5 тыс. метрик в секунду. При этом максимальный поток поступающих текстовых событий составляет несколько тысяч в секунду.

В таблице приведены удельные частоты событий (количество событий N на один узел в час), которые могут привести к потере функциональной способности вычислительного сервера. Наиболее интересным из этого списка можно считать количество перезагрузок. Оно характеризует общее количество критичных сбоев, так как их устранение обычно не обходится без перезагрузки сервера.

Заключение

Благодаря тому, что СКАМ является децентрализованной модульной СМ, она не имеет архитектурных ограничений по расширяемости. Максимальное количество мониторируемого оборудования определяется константами компиляции и вычислительными мощностями, выделенными под СМ.

Имеющиеся модули СКАМ решают задачи сбора и обработки данных о состоянии оборудования ВК.

Модуль Sipmi, собирающий показания датчиков, аппаратных журналов, журналов операционной системы серверов, генерирует меньше трафика IPMI-пакетов, чем существующие аналоги. Модуль Sib, собирающий данные об устройствах коммуникационной среды InfiniBand, не имеет прямых аналогов среди ПО с открытым кодом. Оба модуля не взаимодействуют с операционной системой вычислительных узлов и поэтому не влияют на работу пользовательских приложений.

Модуль сохранения Sdb записывает получаемые события в реляционную базу данных, благодаря чему собранная информация доступна

Некоторые типы регистрируемых сбоев

Тип ошибки	Удельная частота ($10^6 N$)
Корректируемые ошибки памяти	45 000
Некорректируемые ошибки памяти	1
Превышения температур	879
Перезагрузки	434
События *Machine Check Event*	17 500
Сбои блоков питания	154
События *Kernel panic*	5
Потеря связи по Ethernet	24
Потеря избыточности	8 576
События журналов ОС	$3{,}942 \cdot 10^6$
Остальные события	12 427
Всего	$4{,}0 \cdot 10^6$

для анализа. Модуль Sclient позволяет оценить текущее состояние оборудования ВК.

Информация, собираемая СКАМ, используется при наладке и сопровождении ВК.

Дальнейшее развитие СКАМ видится в создании средств работы с иерархиями мониторируемых устройств, модулей автоматической обработки данных, визуализации, а также увеличении числа модулей сбора данных с целью увеличить количество контролируемого оборудования.

Список литературы

1. Ipmitool Project Web Hosting. http://ipmitool.sourceforge.net/.
2. NetSNMP. http://www.net-snmp.org/.
3. OpenHPI/ http://openhpi.org/.
4. GNU FreeIPMI. http://www.gnu.org/software/freeipmi/.
5. *Agarwala S., Poellabauer C., Kong J. et al.* Resource-aware stream management with the customizable dproc distributed monitoring mechanisms // Proc. 12th IEEE Int. Symp. on High Performance Distributed Computing. Washington: IEEE Computer Society, 2003. P. 250—259.
6. *Sacerdoti F. D., Katz M. J., Massie M. L. et al.* Wide areacluster monitoring with Ganglia // Proc. 2003 IEEE Int. Conf. on Cluster Computing. Washington: IEEE Computer Society, 2003. P. 289—298.
7. Linux @ Livermor. Cerebro. https://computing.llnl.gov/linux/cerebro.html.
8. *Sottile M. J., Minnich R. G.* Supermon: A high-speed cluster monitoring system // Proc. 2002 IEEE Int. Conf. on Cluster Computing. Washington: IEEE Computer Society, 2002. P. 39—46.
9. Nagios Core Version 3.x Documentation. http://www.nagios.org/.
10. *Olups R.* Zabbix 1.8 Network Monitoring. Olton Birmingham: Packt Publishing, 2010.
11. Zenoss Transforming IT Operations. http://community.zenoss.org/index.jspa.
12. Lemon Monitoring System. http://lemon.web.cern.ch.
13. *Дмитриев Н. А., Лашманов В. Н., Стрюков В. Н. и др.* Сервисная подсистема универсальной компактной суперЭВМ // Сб. трудов XII Межд. семинара "Супервычисления и математическое моделирование". 11—15 октября 2010 г. Саров: РФЯЦ-ВНИИЭФ, 2011. С. 156—163.
14. OpenOVIS. https://ovis.ca.sandia.gov.
15. Collectd. The System Statistics Collection Daemon. http://collectd.org/.
16. OpenScada. http://oscada.org/.
17. *Острейковский В. А.* Теория надежности: Учеб. для вузов. М.: Высшая школа, 2003.
18. JSON-RPC 2.0. http://www.jsonrpc.org/specification.
19. Parallel Distributed Shell. http://code.google.com/p/pdsh.
20. Intelligent Platform Management Interface. http://www.intel.com/design/servers/ipmi.
21. OpenIPMI. http://openipmi.sourceforge.net/.
22. Ipmi Management Utility. http://ipmiutil.sourceforge.net/.
23. Rsyslog. http://rsyslog.com.
24. IbAdm. http://melanox.com.
25. InfiniBand Architecture Release 1.2. Volume 1. General Specification. http://www.infinibandata.org.

Статья поступила в редакцию 27.12.12.
Исправленный вариант — 28.04.14.