УДК 004.42

М.М. Заславский, К.В. Кринкин, Э.М. Рябиков, В.В. Черкалова, А.Б. Цамутали

ИНСТРУМЕНТАРИЙ НАДЕЖНОГО ХРАНЕНИЯ ДАННЫХ

M.M. Zaslavskiy, K.V. Krinkin, E.M. Ryabikov, V.V. Cherkalova, A.B. Tsamutali

TOOLSET FOR SECURE DATA STORAGE

Рассмотрены подходы к созданию программного обеспечения для функционирования и отладки систем надежного хранения данных. Изучены подходы к неинвазивному профилированию динамических библиотек в Linux. Представлен алгоритм обновления ресурсов в кластере, позволяющий обновлять ресурсы без нарушения целостности кластера (принудительной остановки узлов). Рассмотрена методика автоматизации просмотра файлов журнала, позволяющая оптимизировать просмотр и обработку данных тестирования.

КЛАСТЕР. ОБНОВЛЕНИЕ РЕСУРСОВ. ПРОФИЛИРОВАНИЕ. РАЗДЕЛЯЕМЫЕ БИБЛИОТЕКИ. АНАЛИЗ ЖУРНАЛОВ ТЕСТИРОВАНИЯ.

Different approaches to data storage systems software work and debugging tools creation are presented in this work. Approaches to non-invasive profiling of Linux dynamic libraries. Also, the algorithm of cluster resources update which allows to update the cluster resources without cluster consistency disruption (i.e. forced nodes stop) is considered. Moreover, the methods for log-file view automation, allowing the optimization of test results analysis.

CLUSTER. RESOURCE UPDATE. PROFILING. SHARED LIBRARIES. TEST LOG ANALYSIS.

Разработка инструмента неинвазивного профилирования разделяемых библиотек в ОС linux

Ограничения стандартных методов профилирования разделяемых библиотек в Linux. Одним из специальных случаев профилирования является выявление временных и частотных характеристик вызовов подпрограмм из динамически компонуемых библиотек (shared libraries). Стандартные методы, основанные на инструментации исходного кода изучаемых программ, имеют широкий диапазон применения, однако не позволяют работать при отсутствии возможности компиляции библиотек, что особенно *актуально* при отладке встраиваемых приложений и систем.

Разработанный инструмент позволяет выполнять профилирование при отсутствии исходного кода и привилегий суперпользователя в операционной системе Linux [2]. В его основе лежит методика неинвазивного профилирования.

Принципы неинвазивного профилирования разделяемых библиотек. Методика неинвазивного (noninvasive) профилирования обеспечивает решение описанных выше проблем. Перечислим ее основные положения.

1. Инструмент профилирования не должен внедряться в программный код, а профилирование не должно зависеть от прав на использование системных средств.

2. Профилирование проводится над выбранным множеством функций в конкретные моменты их вызова.

3. Действия по профилированию не должны затрагивать алгоритм работы анализируемого приложения.

4. Профилировщик должен потреблять минимальное количество ресурсов процессора и выполнять измерение с максимальной точностью.

Реализация данных идей основана на внедрении в процесс релокации символов при работе с разделяемыми библиотеками

на стадии компоновки и загрузки. Модификация релокации состоит в реализации средств перехвата вызовов профилируемых функций.

Реализация механизма перехвата вызовов. Перехват вызовов профилируемых функций является основой реализации методики неинвазивного профилирования. В общем случае данный механизм состоит из двух частей: кода перенаправления вызова (редиректор, redirector) и функции-обертки (wrapper).

Редиректор представляет собой набор машинных команд для следующих ассемблерных инструкций (пример для платформы x86) [3]:

push $fcnPtr

jmp $wrapper_addr

Функции редиректора: сохранение адреса профилируемой функции в стеке программы и безусловный переход по адресу функции-обертки. Особенность кода перенаправления вызовов состоит в том, что он создается индивидуально для каждой профилируемой функции и размещается непосредственно в сегменте данных виртуальной памяти процесса. При этом средства ОС позволяют сделать данный набор инструкций исполняемым, что позволяет ему выполняться при вызове вместо интересующей функции (подробнее это описано в [4]).

Функция-обертка служит для вызова профилируемой функции и измерения времени ее работы при передаче управления от редиректора.

Алгоритм работы данной функции:

1. Получение управления от редиректора с сохранением состояния всех регистров в стеке программы и вызовом функции предварительного профилирования.

2. Создание функцией предварительного профилирования специальной структуры – контекста вызова, в которую сохраняется адрес профилируемой функции, адрес возврата и время начала работы.

3. Замена адреса возврата профилируемой функции, находящегося в стеке процесса, на адрес некоторой метки внутри функции-обертки и восстановление сохраненного состояния регистров.

4. Безусловный переход по адресу профилируемой функции.

5. Выполнение профилируемой функции и возврат в функцию-обертку по замененному адресу возврата.

6. Сохранение состояния регистров в стеке завершающего профилирования.

7. Восстановление функцией завершающего профилирования контекста вызова и измерение времени окончания работы, а также передача в функцию-обертку настоящего адреса возврата профилируемой функции.

8. Размещение настоящего адреса возврата в стеке и восстановление сохраненных регистров.

9. Возврат в вызывающую программу.

Главная особенность механизма «обертки» функций – то, что при профилировании не нарушается состояние регистров процессора и стека программы. Это позволяет работать с большим множеством функций, написанных на языках C/C++, независимо от архитектуры, соглашения о вызовах либо способа оптимизации программ.

Ограничения разработанного инструмента при измерении времени. Задача измерения времени работы функций ложится на функцию-обертку. При этом для измерения времени с максимальной разрешающей способностью применяется чтение состояния счетчика тактов процессора (Time Stamp Counter – TSC).

Однако использование такого метода измерений в многопроцессорной системе имеет побочный эффект, связанный с немонотонностью работы отдельно взятого процессорного таймера и асинхронностью совместной работы нескольких таймеров [5].

В связи с этим общие результаты, получаемые при профилировании с таким измерением времени, следует считать относительными. Иными словами, более точные результаты профилирования можно получить, измеряя время работы функций для двух состояний системы: нагруженного и ненагруженного.

В результате исследования и разработки инструмента неинвазивного профилирования разработан инструмент профилирования вызовов функций из разделяемых библиотек, основанный на использовании

динамического компоновщика и библиотеки динамической загрузки в ОС Linux; инструмент протестирован при профилировании системных утилит и сторонних приложений в ОС Debian GNU/Linux 6.0.

В дальнейшем планируется решить проблемы измерения времени в многопроцессорных системах и распространить решения на другие версии ОС Linux.

Адаптивный алгоритм обновления ресурсов в отказоустойчивых кластерах

Кластеры, состоящие из нескольких равноправных вычислительных систем (узлов), могут использоваться для оптимизации распределения вычислительной мощности, а также для повышения отказоустойчивости приложений (ресурсов). На каждом вычислительном узле кластерной системы запускается ряд служебных ресурсов, предназначенных для поддержания целостности служб кластера, слежения за состоянием аппаратного обеспечения и мониторинга работоспособности приложений, непосредственно выполняющих задачи по обработке информации.

Существующие системы управления кластерами требуют наличия идентичных версий ресурсов на всех узлах кластера [7], что не позволяет обновлять ресурсы не прерывая синхронизацию состояния узлов в кластере, что в свою очередь может привести к некорректному запуску ресурсов. Возможность автоматического обновления ресурсов с минимизацией времени простоя узла необходима для реализации отказоустойчивых систем, в которых длительное отключение одного узла ведет к возможной потери функциональности на других узлах кластера.

В данной статье представлен алгоритм, позволяющий обновлять ресурсы без нарушения целостности кластера (принудительной остановки узлов) на примере кластер-менеджера Pacemaker. Этот кластер-менеджер представляет собой одну из последних реализаций Linux High Availability (HA) кластеров, следующих спецификации Open Cluster Framework (OCF) [6].

В процессе разработки алгоритма обновления были выделены следующие этапы: 1) определение факторов, существенных в процессе обновления ресурсов кластера; 2) реализация алгоритма для кластера с двумя узлами; 3) расширение алгоритма для кластера с произвольным числом узлов. Кроме того, были определены способы задания версий кластерной конфигурации ресурса и его кода на каждом узле.

Процесс установки новой версии приложения на узел кластера характеризуется следующим: все ресурсы мигрируют на другие узлы; узлы поочередно отключаются от кластера; происходит обновление приложения; узел присоединяется к кластеру; алгоритм обновления записывает новые версии ресурсов в конфигурацию узла; выполняются необходимые действия по обновлению кластерной конфигурации ресурса; ресурсам позволяется стартовать на данном узле.

Для реализации предложенного процесса необходимы следующие функции кластер-менеджера: функция единовременной миграции ресурсов с конкретного узла кластера с сохранением зависимостей между ресурсами (в Pacemaker это свойство выполняется автоматически при отключении узла от кластера) и функция старта кластер-менеджера на узле без запуска ресурсов [9] (в Pacemaker эта функциональность не реализована [7]).

Предложенная функциональность автоматизирует следующие действия:

- при запуске процесса обновления на узле кластера происходит остановка кластер-менеджера, влекущая за собой миграцию ресурсов с узла;
- после установки новых версий приложений и конфигурационных файлов выполняется старт кластер-менеджера;
- первым стартует служебный ресурс, который сравнивает версии ресурсов на системе и в конфигурационных файлах;
- служебный ресурс запускает обновление кластерной конфигурации изменившихся ресурсов, обновляет версии ресурса на узле и зависимости, позволяющие блокировать одновременную работу несовместимых версий одного и того же ресурса;
- по завершении запуска служебного ресурса другие ресурсы могут стартовать на кластере в соответствии со вновь определенными зависимостями.

В итоге можно сформулировать список свойств, необходимых для автоматического обновления ресурсов кластера и отсутствующих как в спецификации OCF, так и в текущих реализациях Linux HA кластер-менеджерах:

1) возможность задания версий конфигурации и версий кода ресурсов;

2) возможность запускать кластер-менеджер на определенном узле без автоматического запуска ресурсов на этом узле;

3) возможность добавлять ресурсам зависимости от фантомных ресурсов.

При добавлении указанных свойств появляется возможность реализовать общий алгоритм обновления для произвольного количества узлов в кластере.

Анализ файлов журнала

Файл регистрации, протокол, журнал или log-файл – синонимы файла, хранящего записи о событиях в хронологическом порядке. Сведения хранятся с необходимой детальностью в определенном стандартном формате в текстовом виде.

Такие файлы (отчеты) содержат, в т. ч. и записи об ошибках, простоях системы и прочих проблемах. Часто анализа подобных отчетов достаточно для нахождения искомой неисправности или ошибки. Поэтому при разработке и модификации программного обеспечения анализ журнала тестирования является важной задачей. Как правило, ручная процедура анализа выполняется разработчиком или системным администратором и крайне трудозатратна. Есть ряд моментов, которые делают «ручной» поиск необходимых строк в log-файле длительной процедурой.

Log-файлы, обычно, достаточно велики и содержат большое количество однообразных строк. Например, чтобы разобрать одну запись методом визуального просмотра нужно не менее 5 с, но на практике приходится прибегать к одновременному просмотру нескольких файлов, что существенно увеличивает время просмотра. Сложность анализа усугубляется тем, что трудно уловить причинно-следственные связи между событиями, приходится вручную производить поиск ошибок и отладочной информации, кроме того, в процессе просмотра невозможно оставлять пометки.

В рамках данной статьи сформулирован подход к решению описанных проблем, заключающийся в автоматизации, позволяющей:

- избежать проблемы обработки громоздких журналов;
- автоматически выделить важную информацию из строк журнала;
- создать сервисы для автоматизированного поиска и анализа журнала;
- создать удобное визуальное представление информации;
- создать возможность пользователю оставлять пометки и комментарии в тексте файлов, не нарушая целостности исходных файлов журнала.

В настоящее время подобный инструмент находится в разработке.

СПИСОК ЛИТЕРАТУРЫ

1. The ELF Object File Format by Dissection [Электронный ресурс] / Режим доступа http://www.linuxjournal.com/node/1060/print

2. **Michael Kerrisk.** The Linux Programming Interface [Text] / Kerrisk Michael. –San-Francisco: No Starch Press, 2010. –P. 833–859.

3. Intel x86 Function-call Conventions – Assembly View [Электронный ресурс] / Режим доступа http://www.unixwiz.net/techtips/win32-call-conv-asm.html

4. SSE (Streaming SIMD Extentions) [Электронный ресурс] / Режим доступа http://www.songho.ca/misc/sse/sse.html

5. **Matz Michael.** System V Application Binary Interface. AMD64 Architecture Processor Supplement [Электронный ресурс] / Michael Matz, Jan Hubicka, Andreas Jaeger [et al.] // AMD64 ABI Draft 0.96 – June 14, 2005.

6. Corosync Configuration [Электронный ресурс] / Режим доступа http://www.corosync.org/

7. **Beekhof Andrew.** Pacemaker 1.1 Configuration Explained. An A-Z guide to Pacemaker's Configuration, Options Edition [Электронный ресурс] / Andrew Beekhof, 2012.

8. **Geist Al.** PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing [Text] / Al Geist, Adam Beguelin, Jack Dongarra [et al.]. –Massachusetts Institute of Technology, 1994.

REFERENCES

1. The ELF Object File Format by Dissection. Availiable http://www.linuxjournal.com/node/1060/print

2. **Kerrisk M.** The Linux Programming Interface. – San-Francisco: No Starch Press, 2010. – P. 833–859.

3. Intel x86 Function-call Conventions – Assembly View. Availiable http://www.unixwiz.net/techtips/win32-callconv-asm.html

4. SSE (Streaming SIMD Extentions). Availiable http://www.songho.ca/misc/sse/sse.html

5. **Matz Michael, Hubicka Jan, Jaeger Andreas, Mitchell Mark.** System V Application Binary Interface. AMD64 Architecture Processor Supplement; AMD64 ABI Draft 0.96; 14.06.2005.

6. Corosync Configuration. Availiable http://www.corosync.org/

7. **Andrew Beekhof.** Pacemaker 1.1 Configuration Explained. An A-Z guide to Pacemaker's Configuration, Options Edition, 2012.

8. **Geist Al, Beguelin Adam, Dongarra Jack, Jiang Weicheng, Manchek Robert, Sunderam Vaidy.** PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing. – Massachusetts Institute of Technology, 1994.

ЗАСЛАВСКИЙ Марк Маркович – *студент кафедры системного анализа и управления Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: mark.zaslavskiy@gmail.com

ZASLAVSKIY, Mark M. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: mark.zaslavskiy@gmail.com

КРИНКИН Кирилл Владимирович – *доцент кафедры математического обеспечения и применения ЭВМ Санкт-Петербургского государственного электротехнического университета «ЛЭТИ», кандидат технических наук.*
197376, Россия, Санкт-Петербург, ул. Профессора Попова, д. 5.

KRINKIN, Kirill V. *Saint Petersburg Electrotechnical University «LETI».*
197376, Professora Popova Str. 5, St. Petersburg, Russia.

РЯБИКОВ Эдуард Михайлович – *студент кафедры математического обеспечения и применения ЭВМ Санкт-Петербургского государственного электротехнического университета «ЛЭТИ.*
197376, Россия, Санкт-Петербург, ул. Профессора Попова, д. 5.
E-mail: Edward.ryabikov@gmail.com

RYABIKOV, Eduard M. *Saint Petersburg Electrotechnical University «LETI».*
197376, Professora Popova Str. 5, St. Petersburg, Russia.
E-mail: Edward.ryabikov@gmail.com

ЧЕРКАЛОВА Виктория Владимировна – *магистр кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: Vitoria@bk.ru

CHERKALOVA, Viktoria V. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: Vitoria@bk.ru

ЦАМУТАЛИ Алексей Борисович – *магистр кафедры информационных и управляющих систем Санкт-Петербургского государственного политехнического университета.*
195251, Россия, Санкт-Петербург, ул. Политехническая, д. 29.
E-mail: indrik_rush@mail.ru

TSAMUTALI, Aleksey B. *St. Petersburg State Polytechnical University.*
195251, Politekhnicheskaya Str. 29, St. Petersburg, Russia.
E-mail: indrik_rush@mail.ru