DOI 10.37882/2223–2966.2021.08.41

МЕТОД И АЛГОРИТМ АВТОМАТИЧЕСКОГО ВОССТАНОВЛЕНИЯ ИНФОРМАЦИОННЫХ СЕРВИСОВ НА ОСНОВЕ ОБЪЕКТИВНЫХ ПРОГНОСТИЧЕСКИХ ДАННЫХ МОНИТОРИНГА

Щемелинин Дмитрий Александрович
К.т.н., Санкт-Петербургский политехнический университет Петра Великого, г. Санкт-Петербург
dshchmel@gmail.ru

METHOD AND ALGORITHM FOR AUTOMATIC RECOVERY OF INFORMATION SERVICES BASED ON OBJECTIVE PREDICTIVE MONITORING DATA

D. Schemelinin

Summary. This paper presents the results of analyzing the performance and evaluating the use of the RAM of virtual machines used for Java-based applications and the developed method and algorithm for automatic recovery of services of globally distributed computing systems (GDVK) based on objective monitoring data, as well as predicting anomalies for a given period of time. in the future using new forecasting methods. To analyze data on the performance of computing resources, production statistics were collected from the studied GDVK using the continuous monitoring system Zabbix, which is used to regularly receive counters from several VMs (Virtual Machine) and store performance data in database management systems (DBMS) SQL (Structured Query Language) for analysis and troubleshooting. The production servers under study are organized into pool groups, each of which performs specific business services. The purpose of the analysis was to develop control methods, performance assessments and automatic elimination of anomalies in the software (software) of software components built using the JEDI technology (Java Environment for Distributed Invocation). The new monitoring method takes into account the specifics of Java memory leaks in Internet applications (Memory Leak), is based on predictive models of anomaly detection and provides secure automatic recovery of cloud services. Unlike existing solutions, the developed method and algorithm for automatic restoration of GDVK services is based on objective monitoring data, as well as predicting an anomaly for a given period of time in the future using new forecasting methods and is universal and can be generalized to a wider class of studied metrics and monitoring systems. The method was implemented programmatically at RingCentral on the example of detecting Java memory leaks in Internet applications and automatically restoring services in order to achieve a world level of 99.999% availability in 24/7 mode.

Keywords: monitoring, big data, modeling, forecasting function, monitoring metrics, Java, cloud technologies.

Аннотация. В данной работе представлены результаты анализа производительности и оценки использования оперативной памяти виртуальных машин используемых для приложений на основе Java и разработанный метод и алгоритм автоматического восстановления сервисов глобально-распределенных вычислительных комплексах (ГРВК) основан на объективных мониторинговых данных, а также прогнозировании аномалии на заданный период времени в будущем с использованием новых методов прогнозирования. Для анализа данных о производительности вычислительных ресурсов, была собрана производственная статистика из исследуемых ГРВК с использованием непрерывной системы мониторинга Zabbix, которая используется для регулярного получения счетчиков от нескольких VM (англ. Virtual Machine) и сохранения данных о производительности в системах управления базами данных (СУБД) SQL (англ. Structured Query Language) для анализа и устранения неполадок. Исследуемые производственные серверы объединены в группы пулов, каждая из которых выполняет определенные бизнес-сервисы. Целью проведенного анализа стала разработка методов управления, оценок производительности и автоматического устранения аномалий в программном обеспечении (ПО) программных компонент, построенных по технологии JEDI (англ. Java Environment for Distributed Invocation). Новый метод мониторинга учитывает специфику утечки Java памяти в Интернет-приложениях (англ. Memory Leak), основан на прогностических моделях обнаружения аномалии и обеспечивает безопасное автоматическое восстановление облачных сервисов. В отличие от существующих решений, разработанный метод и алгоритм автоматического восстановления сервисов ГРВК основан на объективных мониторинговых данных, а также прогнозировании аномалии на заданный период времени в будущем с использованием новых методов прогнозирования и является универсальным и может быть обобщен на более широкий класс исследуемых метрик и мониторинговых систем. Метод внедрен программно в компании RingCentral на примере обнаружения утечки Java памяти в Интернет-приложениях и автоматического восстановления сервисов с целью их достижения мирового уровня доступности 99,999% в режиме 24/7.

Ключевые слова: мониторинг, большие данные, моделирование, функция прогнозирования, метрики мониторинга, Java, облачные технологии.

Введение

На сегодняшний день широк используются различного рода Java приложения. Приложения Java работают автономно на клиентском сервисе или собственном сервере. Виртуальная машина Java интерпретирует инструкции, и, как любой язык программирования, работающий в своей родной среде, программы Java имеют полный доступ ко всем ресурсам компьютера. Каждому приложению Java требуется память для работы на JVM [1]. Эта память берется из доступной оперативной памяти системы, в которой запущено приложение. Есть 2 вида памяти: stack и heap:

1. Stack (стек). Это область ОЗУ, которая используется для хранения временных переменных или примитивных типов данных в Java. Он также хранит ссылки на объекты, которые физически создаются в куче. Он сохраняет переменные, созданные функциями, в формате «последним вошел, первым ушел» (LIFO) и освобождает всю выделенную память при выходе из функции. Стек управляется для каждого потока в Java, поэтому его область действия находится внутри потока. Он меньше размера кучи. Стек быстрее, потому что выделение / освобождение памяти более тривиально по сравнению с кучей. Когда происходит вызов метода, в стеке этого потока создается новый кадр стека. Стек будет содержать локальные переменные, параметры, адреса возврата и т.д.
2. Heap. Это большая область оперативной памяти, которая используется для распределения динамической памяти. Все объекты Java хранятся в куче, а объем объектов — все приложение. Управление памятью осуществляется нами в куче, но неиспользуемые объекты автоматически очищаются сборщиком мусора [2, 3].

Сложность использования оперативной памяти является хорошо известной проблемой Java приложений, особенно в облачных вычислительных средах с более высокой загрузкой центрального процессора и оперативной памяти из-за запуска VMware на платформе MS Windows Server. Постоянная утечка памяти наблюдается в Java API (англ. Application Program Interface), потому что объекты и классы загружаются динамически, вызывая существенное ухудшение использования оперативной памяти.

Утечки оперативной памяти часто связаны с ошибками программирования. Ручное управление ресурсами является основным направлением для разработчиков Java, а исправление кода API является долгосрочным решением. Служба GC (англ. Garbage Collector) является самым популярным и эффективным инструментом для автоматического обнаружения и высвобождения объектов данных, которые впоследствии больше не будут использоваться.

В информатике сборка мусора (GC) — это форма автоматического управления памятью. Garbage Collector пытается освободить память, которая была выделена программой, но больше не используется, что также называется мусором. Garbage Collector была изобретена американским ученым-компьютерщиком Джоном Маккарти примерно в 1959 году для упрощения ручного управления памятью в Лиспе.

Garbage Collector освобождает программиста от выполнения ручного управления памятью, когда программист указывает, какие объекты освободить и вернуть в систему памяти и когда это сделать. Другие аналогичные методы включают выделение стека, вывод области, владение памятью и комбинации нескольких методов. Garbage Collector может занять значительную часть общего времени обработки в программе и, как следствие, может существенно повлиять на производительность.

Другие ресурсы, помимо памяти, такие как сетевые сокеты, дескрипторы базы данных, окна взаимодействия с пользователем, дескрипторы файлов и устройств, обычно не обрабатываются сборкой мусора. Методы управления такими ресурсами, особенно деструкторами, также могут быть достаточными для управления памятью, не оставляя необходимости в сборке мусора. Некоторые системы GC позволяют ассоциировать такие другие ресурсы с областью памяти, которая при сборе вызывает работу по возврату этих ресурсов [4, 5].

Несмотря на то, что GC включен, цикл его работы периодически частично высвобождает оперативную память, не обеспечивая ее первоначального размера. После цикла работы GC наблюдаются постоянные утечки памяти Java. Тенденция деградации зависит от активности пользователей и рабочих часов. Рано или поздно, после того как объем свободной памяти достигнет критического порогового значения, либо потребуется перезапустить службу JBoss, либо перезагрузить VM, чтобы предотвратить сбой работоспособности сервера.

Помимо памяти Java, теряются и другие ресурсы VM, такие как сетевые каналы, соединения с системой управления базами данных (СУБД), использование центрального процессора (ЦПУ), взаимодействие с пользователем и т.д.

Таким образом, на сегодняшний день актуальной задачей является разработка эффективных методов и алгоритмов автоматического восстановления ин-

формационных сервисов на основе объективных прогностических данных мониторинга. На данный момент выполнен ряд исследований, посвященных методика создания распределенной компьютерно-вычислительной системы, однако требуется проведение новых исследований с целью поиска более эффективных решений [6, 7, 8, 9, 10, 11]

Цель данной работы

Разработка алгоритма автоматического восстановления информационных сервисов на основе объективных прогностических данных мониторинга.

Материалы и методы

Разработка метода автоматического восстановления Java сервисов базировалась на программно-реализованной процедуре автоматического перезапуска службы JBoss на ПО JEDI, когда свободная память Java подходит к критическому пороговому значению. Соответствующие триггеры были реализованы в системе мониторинга Zabbix [11]. В ходе выполнения исследования процесс автоматического устранения аномалий запускается, если происходит хотя бы один из следующих случаев:

1. Объем свободной оперативной памяти Java ниже критического порога, установленного в 5 МБ;
2. Объем оперативной памяти VM ниже критического порога, установленного в 5 МБ;
3. Выделенная виртуальная память Java выше допустимой (2 ГБ для 32-разрядной VM и 6 ГБ для 64-разрядной VM);
4. Статус сервиса JBoss не отвечает.

В ходе исследования рабочие метрики для приложений Java были реализованы с использованием технологии JMX (англ. Java Management Extensions). Преимущества технологии JMX показаны на рисунке 5, где соединения с СУБД не используются активно, в течение длительного периода времени. Метрики JMX, такие как активные сеансы СУБД, соединения пула серверов, лучше отражают реальную рабочую нагрузку и ресурсы. Общее количество подключений (от 5 до 20, в зависимости от механизма подключения к СУБД) может быть установлено с сервером, но пользователи имеют различную активность рабочих процессов.

В исследуемых инфраструктурах ГРВК введены следующие новые метрики JMX для анализа работоспособности вычислительных компонентов JEDI:

1. Объем памяти Java (используется, свободный, средний, максимальный, виртуальный);
2. Статистика GC (последний раз, продолжительность, количество срабатываний, результат);
3. Системные ресурсы (загрузка процессора для процессов Java, использование системной памяти для API Java);
4. Метрики активности пользователя (HTTP-запросы за время, количество активных веб-потоков, время сеанса);
5. Количество соединений с СУБД (общее количество установленных соединений, количество объектов СУБД, количество активных соединений с СУБД и количество сеансов).

Результаты и обсуждение

Был разработан алгоритм метода авто-устранения неполадок на ПО JEDI. Он выглядит следующим образом:

Шаг 1. Сбор данных: сервер мониторинга Zabbix собирает все данные JEDI и выдает предупреждение в случае обнаружения аномалии в работе сервера;

Шаг 2. Проверка: на Zabbix-сервере выполняется пользовательский сценарий для проверки общей доступности служб JEDI в пуле;

Шаг 3. Подтверждение безопасного перезапуска проблемного сервера без возможного перерыва работоспособности ИС с точки зрения пользователя сервисов;

Шаг 4. Инициация сценария автоматического перезапуска служб JBoss или перезагрузки виртуальной машины;

Шаг 5. Нештатная ситуация: в случае сбоя процедуры автоматического исправления срабатывает специальный триггер для устранения неисправностей вручную.

В ходе исследования, было экспериментально установлено, что стандартные счетчики, такие как загрузка системного процессора и использование памяти, не показывают все внутренние процессы в приложениях Java и не помогают при устранении неполадок. Было выявлено, что несмотря на то, что используемая оперативная память Java находится в пределах от минимальных до максимальных значений, использование виртуальной памяти возросло с 631 МБ до 2,37 ГБ. Это может быть объяснено либо проблемой GC, либо ростом активности пользователей, но не может быть точно определено с использованием только системных показателей.

Таким образом, в дополнение к системным метрикам было предложено дополнительно учитывать прогностические данные мониторинга, такие как почасовая оплата и распределение запросов на обслуживание, активность пользователей установленных веб-сеансов и соединений с СУБД, количество обработанных страниц и многие другие.

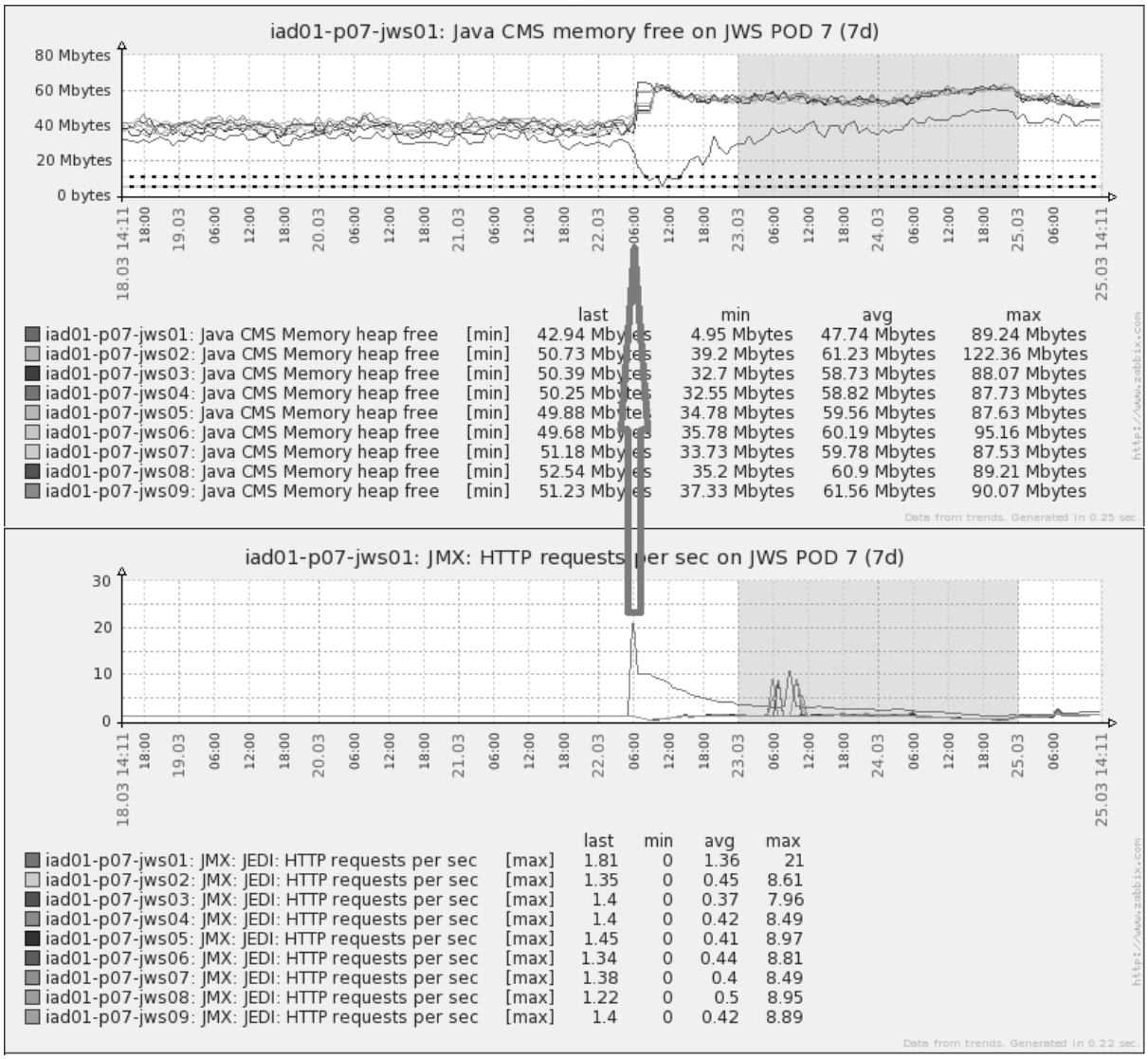

Рис. 1. Пример всплеска активности пользователя в пуле

С одной стороны, вся информация для непрерывного мониторинга специальных программных средств (СПС) может быть получена из журналов приложений. Однако, с другой стороны, анализ большого количества файлов журнала записей, для расчета каждой метрики, приведет к значительной задержке (от минут до часов), прежде чем результат будет доступен в базе данных мониторинга.

Файлы ежедневных журналов различаются по размеру, до 20 ГБ, в зависимости от бизнес-сервисов, дня недели и активности пользователя, которые требуют мощных компьютерных ресурсов для обработки полученных больших данных.

Предложенные новые метрики JMX позволяют измерять активность пользователей в пуле. На рисунке 1 показан пример статистики HTTP-запросов в секунду, свободной памяти Java для всех серверов JEDI на одном графике. Цель состоит в том, чтобы исследовать, как весь пул справляется с растущей рабочей нагрузкой пользователя.

Несмотря на большое количество серверов JEDI в пуле, скачок HTTP-запроса одного пользователя вызвал резкое ухудшение свободной памяти Java на одном сервере до критического значения, ниже указанного порога в 5 МБ. В результате была начата процедура автоматического исправления для предотвращения сбоя работоспособности сервера. После перезапуска службы JBoss, на проблемном сервере заданные параметры свободной памяти Java были восстановлены.

В то же время все остальные серверы JEDI работали должным образом, не влияя на использование памя-

ти Java, так как сбой работоспособности веб-службы в пуле не был обнаружен. Это означает, что на данный момент число серверов в пуле JEDI было достаточным. Тем не менее, требуется дополнительное исследование, чтобы убедиться в отсутствии проблем с пропускной способностью системы.

Выводы

Новый метод мониторинга учитывает специфику утечки Java памяти в Интернет-приложениях (англ. Memory Leak), основан на прогностических моделях обнаружения аномалии и обеспечивает безопасное автоматическое восстановление облачных сервисов. В отличие от существующих решений, разработанный метод и алгоритм автоматического восстановления сервисов ГРВК основан на объективных мониторинговых данных, а также прогнозировании аномалии на заданный период времени в будущем с использованием новых методов прогнозирования и является универсальным и может быть обобщен на более широкий класс исследуемых метрик и мониторинговых систем. Метод уже внедрен программно в компании RingCentral на примере обнаружения утечки Java памяти в Интернет-приложениях и автоматического восстановления сервисов с целью их достижения мирового уровня доступности 99,999% в режиме 24/7 [12].

ЛИТЕРАТУРА

1. Петрищев С.С. Проблемы производительности разработки Java приложений // МНИЖ. 2012. № 7–1 (7). URL: https://cyberleninka.ru/article/n/problemy-proizvoditelnosti-razrabotki-java-prilozheniy (дата обращения: 03.07.2021).
2. Сухомлин Владимир Александрович Дистанционная подготовка разработчиков корпоративных Java-приложений // International Journal of Open Information Technologies. 2013. № 1. URL: https://cyberleninka.ru/article/n/distantsionnaya-podgotovka-razrabotchikov-korporativnyh-java-prilozheniy (дата обращения: 03.07.2021).
3. Дашук В.А., Намиот Д.Е. Сравнительный анализ моделей работы с данными в Java-приложениях // International Journal of Open Information Technologies. 2020. № 7. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-modeley-raboty-s-dannymi-v-java-prilozheniyah (дата обращения: 02.07.2021).
4. Harold Abelson and Gerald Jay Sussman and Julie Sussman (2016). Structure and Interpretation of Computer Programs (PDF) (2nd ed.). Cambridge, MA: MIT Press. Here: p.734–736.
5. McCarthy, John (1960). "Recursive functions of symbolic expressions and their computation by machine, Part I". Communications of the ACM. 3 (4): 184–195. doi:10.1145/367177.367199. S2CID1489409. Retrieved 2009–05–29.
6. Shchemelinin D. Capacity Management of Java-based Business Applications Running on Virtualized Environment / D. Shchemelinin, S. Mescheryakov // [Электронный ресурс] Proceedings of the 39th International Conference for the Performance and Capacity by CMG. La Jolla, CA, USA,— 2013, Режим доступа: http://www.cmg.org/conference/cmg2013/
7. Shchemelinin D., Ardulov Y., Mescheryakov S. Monitoring and Remediation of Cloud Services Based on 4R Approach. The 41 International IT Conference by CMG, San Antonio, USA, 2015. Режим доступа: http://www.cmg.org/publications/conference-proceedings/conference-proceedings2015/
8. Щемелинин Д.А. Прогностическое моделирование и визуализация в облачной системе мониторинга / К.Н. Кучерова, С.В. Мещеряков, Д.А. Щемелинин // Распределенные компьютерные и теле-коммуникационные сети: управление, вычисление, связь (DCCN-2016): Материалы 19 междунар. науч. конф., Т. 1, М: РУДН,— 2016 Режим доступа: http://dccn.ru
9. Щемелинин Д.А. Программные модели и методы мониторинга состояния процессинговых узлов в облачной инфокоммуникационной системе с использованием Zabbix // Программные системы и вычислительные методы.— 2021.— № 2. DOI: 10.7256/2454–0714.2021.2.35617 URL: https://nbpublish.com/library_read_article.php?id=35617
10. Щемелинин Д.А. Методика создания распределенной компьютерно-вычислительной системы для программного инфокоммуникационного коммутатора // Программные системы и вычислительные методы.— 2019.— № 1.— С. 91–97. DOI: 10.7256/2454–0714.2019.1.28782 URL: https://nbpublish.com/library_read_article.php?id=28782
11. Щемелинин Д.А. Методика управления конфигурационными параметрами, программными артефактами и метриками состояния вычислительных компонент в глобально распределенных облачных информационных комплексах // Программные системы и вычислительные методы.— 2019.— № 1.— С. 98–106. DOI: 10.7256/2454–0714.2019.1.29757 URL: https://nbpublish.com/library_read_article.php?id=29757
12. Zabbix Enterprise-class Monitoring System, [Электронный ресурс] // Режим доступа: http://www.zabbix.com
13. Официальный Интернет-сайт RingCentral [Электронный ресурс]. URL: http://www.ringcentral.com/

Журнал «Современная наука: актуальные проблемы теории и практики»