ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2025. Т. 29, № 3 (109). С. 61-71 http://journal.ugatu.su

УДК 004.45, 004.8 doi 10.54708/19926502_2025_29310961
Код ГРНТИ 28.23.17, 28.23.25, 28.23.35

Программная архитектура системы мониторинга состояний кластеров баз данных на основе агентно-ориентированного подхода

А.В. Шундеев[1], Д.П. Саблин[1], В.А. Глущенко[1], А.С. Ковтуненко[2*]

[1]Уфимский университет науки и технологий, г. Уфа, Республика Башкортостан, Россия
[2]ООО ЮНИТС, г. Уфа, Республика Башкортостан, Россия

Аннотация. В статье рассматривается система мониторинга состояний узлов распределенных баз данных Представлена архитектура системы, описаны основные модули, включая модуль обработки экспертных оценок, модуль иммунных алгоритмов, а также модуль детектора неизвестного поведения. Разработаны методы сбора и анализа метрик, выявления аномалий и формирования рекомендаций для оперативного устранения последствий сбоев. Система направлена на повышение эффективности мониторинга и администрирования баз данных, а также на снижение рисков, связанных с их работой.

Ключевые слова: *мониторинг, распределенные базы данных, искусственный интеллект, машинное обучение, экспертные системы, иммунные системы, большие языковые модели*.

***askovtunenko@mail.ru**

Введение

В наше время базы данных являются неотъемлемой частью информационной инфраструктуры большинства организаций, поэтому бизнес-процессы, связанные с обеспечением их стабильной работы и оперативным реагированим на возникающие сбои, приобретают высокий приоритет. С ростом объемов данных и сложности архитектуры систем управления базами данных (СУБД) традиционные методы мониторинга и администрирования часто оказываются недостаточно эффективными. Они не всегда способны вовремя выявлять аномалии, анализировать большие объемы метрик и предоставлять рекомендации для устранения проблем. В связи с этим актуальной становится разработка систем мониторинга, в которых которые процессы сбора и анализа данных были бы максимально автоматизированы и которые бы предоставляли бы расширенную аналитическую поддержку администраторам баз данных с применением всех современных средств искусственного интеллекта.

В настоящей работе рассматривается вопрос архитектурного проектирования системы мониторинга состояний кластеров баз данных, обеспечивающей информационно-аналитическую поддержку процессов мониторинга и администрирования [1]. Разрабатываемая система позволяет интегрировать разнообразные технологии, такие как экспертные системы, искусственные иммунные системы и большие языковые модели (LLM), для достижения наибольшей эффективности и эргономичности в экусплуатации, автоматически анализировать метрики, выявлять аномалии и формировать рекомендации для предотвращения и устранения сбоев. Это способствует повышению стабильности и надежности работы кластеров баз данных.

Обзор существующих решений мониторинга

В области программно-аппаратных средств мониторинга информационных систем накоплен значительный опыт. В настоящее время доступны как коммерческие, так и открытые решения для отслеживания состояния инфраструктуры. Широко распространены открытые системы мониторинга общего назначения, такие как Zabbix и Prometheus, а также специализированные инструменты для мониторинга баз данных, такие как, например, Percona Monitoring and Management (PMM). Рассмотрим кратко возможности этих решений, особенно в контексте мониторинга СУБД PostgreSQL.

Zabbix представляет собой свободную систему мониторинга состояний разнообразных сервисов, серверов и сетевого оборудования [2]. Это комплексное решение «из одного источника», которое предоставляет возможность сбора метрик, оповещения и формирования отчетности в единой системе [3].

Сильной стороной Zabbix является степень готовности к использованию непосредственно после установки, минимальный порог вхождения для персонала, а также минимальные требования к конфигурированию. Система включает множество предустановленных шаблонов мониторинга, что упрощает ее настройку и развертывание [3]. Zabbix применяет модель с агентами: на контролируемые узлы устанавливаются агенты – специальные программные компоненты, передающие данные на центральный сервер для сохранения в реляционной СУБД (поддерживаются MySQL, PostgreSQL, Oracle и др.) [2].

Веб-интерфейс Zabbix (разработанный на PHP) обеспечивает доступ к данным мониторинга и настройкам. Таким образом, Zabbix предоставляет богатый функционал для мониторинга и управления классической ИТ-инфраструктурой, однако крупномасштабный мониторинг с его помощью может требовать значительных вычислительных ресурсов, а в случае с динамически меняющейся средой (например, при контейнеризации микросервиов) – еще и трудозатрат на непрерывную реконфигурацию [3].

Prometheus – современная открытая система мониторинга и оповещения, изначально разработанная в SoundCloud (2012) для мониторинга микросервисов. В отличие от Zabbix, Prometheus ориентирован на сбор метрических данных в виде временных рядов и хранение их в высокопроизводительном встроенном хранилище временных рядов (time-series database – TSDB) [4].

Prometheus использует технологию «pull» – запрос значений метрик происходит по протоколу HTTP со специальных программ-экспортеров, работающих рядом с сервисами. Для мониторинга PostgreSQL обычно применяется экспортер postgres_exporter. Простота развертывания (запуск экспортеров в Docker-контейнерах) делает его популярным инструментом DevOps-инженеров [5].

Главными преимуществами Prometheus являются модульность и гибкость: сам Prometheus отвечает за сбор, хранение и первичную обработку метрик, а визуализация отдается на сторону клиента. Prometheus отлично масштабируется под большие и динамичные нагрузки и обладает мощным языком запросов PromQL для аналитики метрик. Однако отсутствие встроенного графического интерфейса и полной системы прав требует применения дополнительных компонентов для построения полнофункциональной системы мониторинга [3].

По умолчанию Prometheus хранит данные за достаточно короткий период (недели), хотя возможна интеграция с внешними хранилищами для сохранения долговременной истории. Тем не менее, благодаря развитой экосистеме и активной поддержке сообщества, связка Prometheus + Grafana стала де-факто стандартом мониторинга в облачных и контей-нерных средах [3].

Percona Monitoring and Management (PMM) – открытая платформа мониторинга баз данных от компании Percona. Она ориентирована на СУБД (MySQL, PostgreSQL, MongoDB и др.) и включает в себя стек на базе Prometheus и Grafana с большим количеством дополнительных функций, ориентированных на администраторов баз данных. PMM предоставляет единый веб-интерфейс для просмотра метрик узлов и отдельных запросов, обладает встроенной системой анализа запросов (Query Analytics) для выявления «тяжелых» SQL, позволяет получать рекомендации по настройкам производительности и безопасности через механизм Advisors [6].

Система распространяется как в виде установочного пакета, так и в виде готового Docker-образа, что существенно упрощает развертывание в различных средах [7]. Для мониторинга PostgreSQL PMM предлагает готовые дашборды, сбор метрик осуществляется посредством postgres_exporter, однако дополнительно система позволяет агрегировать информацию, например о медленных запросах, блокировках, репликации и т.п., предоставляя более «узкоспециализированный» вид мониторинга. В отличие от универсальных решений, PMM сфокусирована на СУБД, что позволяет глубже анализировать состояние базы данных, однако при этом требует выделения отдельного сервера мониторинга [6].

Помимо вышеперечисленных, существует множество других систем, однако они в меньшей степени отвечают рассматриваемой задаче, поэтому рассмотрим их бегло. Классические Nagios и Icinga используются преимущественно для инфраструктурного мониторинга, графические платформы на основе стека ELK (Elasticsearch/Logstash/Kibana) могут быть применены для логирования и мониторинга, облачные сервисы вроде AWS CloudWatch, а также новейшие SaaS-решения (Datadog, New Relic и др.), предлагают интеграцию с СУБД. Многие из них также могут быть развернуты в контейнерах Docker для удобства интеграции в CI/CD.

Кроме того, в инструментах мониторинга активно используется язык Python, например для написания скриптов агентов, обработки данных или реализации алгоритмов машинного обучения на основе собранных данных. Популярность Python обусловлена наличием большого количество функциональных и эффективных библиотек для анализа данных и ML, это упрощает создание пользовательских расширений мониторинга.

Сравнение ключевых характеристик популярных систем мониторинга приведено в Табл 1.

Таблица 1. Сравнительный анализ систем мониторинга баз данных.

Система	Особенности архитектуры	Преимущества	Недостатки
Zabbix	Централизованный сервер + агенты; хранение данных в SQL-БД	Комплексность: единая система (метрики, алерты, отчеты) [3]; готовые шаблоны: быстрый старт настройки мониторинга [3]	Ресурсоемкость: высокая нагрузка на БД хранения при большом масштабе [3]; статика: менее приспособлен для динамических контейнерных сред [3]
Prometheus	Самостоятельный сервер сбора метрик + экспортеры; хранение в TSDB	Масштабируемость: рассчитан на облачные и контейнерные среды (микросервисы, Kubernetes) [3]; гибкость: мощный язык запросов PromQL для аналитики метрик [3]	Функционал «по частям»: нет собственного UI (нужна Grafana) и расширенной аутентификации/ролей из коробки [6]
Percona Monitoring and Management	Веб-платформа (Docker-контейнер); в основе Prometheus + Grafana, специальные плагины	Специализация: заточена под СУБД (PostgreSQL, MySQL), глубокий анализ запросов, рекомендации по конфигурации [6]; удобство: единый интерфейс, быстрая установка (образ Docker) [7]	Сложность кастомизации: готовые дашборды Grafana, поставляемые с PMM, трудно модифицировать без глубокого знания PromQL [6, 7]

Как видно из анализа, существующие в настоящее время современные системы мониторинга обеспечивают сбор обширных данных о работе СУБД и инфраструктуры, однако даже самые продвинутые из них не всегда имеют встроенные средства интеллектуального анализа. Они могут, к примеру, обнаруживать аномальные метрики по статическим порогам или по предварительно заложенным шаблонам, но не предоставляют функции формирования автоматических советов по устранению неисправностей, либо не способны использовать накопленные данных для обучения и адаптации к различным ситуациям без дополнительного ручного конфигурирования. Поэтому актуальным направлением развития методов автоматизации управления информационной инфраструктурой является интеграция методов искусственного

интеллекта и машинного обучения в системы мониторинга. В частности, комбинация традиционного мониторинга с ML-алгоритмами позволяет построить адаптивную модель, помогающую своевременно реагировать на инциденты и определять их первопричины [8].

Архитектура системы мониторинга состояний кластеров БД

Предлагаемая система мониторинга предназначена для комплексного наблюдения за кластерами баз данных (в первую очередь PostgreSQL) в реальном времени с помощью интеллектуальных модулей анализа. Она реализована как набор взаимосвязанных сервисов, развернутых в контейнерах Docker для облегчения масштабирования и обновления [9]. На Рис. 1 представлена общая архитектура системы и отражает состав и взаимосвязь модулей и компонент системы в режиме мониторинга одного кластера PostgreSQL.

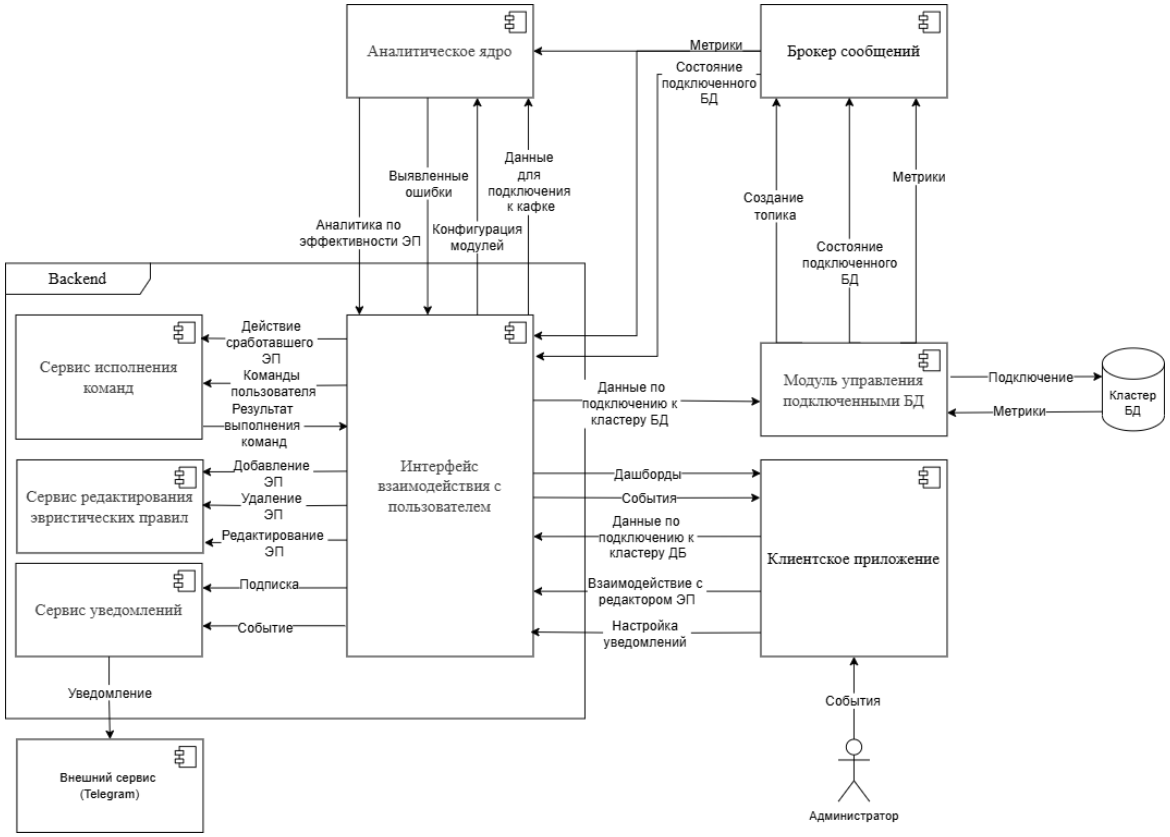

Рисунок 1. Архитектура системы мониторинга состояний кластеров БД.

Система включает следующие компоненты (Рис. 1): интерфейс взаимодействия с пользователем, аналитическое ядро, сервис исполнения команд, сервис уведомлений, сервис редактирования эвристических правил (ЭП) и модуль управления подключенными базами данных. Потоки данных между компонентами организованы через брокер сообщений.

Интерфейс взаимодействия с пользователем обеспечивает авторизацию, визуализацию состояния кластеров и конфигурирование системы. Аналитическое ядро составляют три независимых модуля: экспертную систему, иммунную систему и детектор неизвестного поведения на основе LLM. Каждый кластер баз данных обслуживается автономным программным агентом – экземпляром каждого из модулей, что гарантирует изолированность обработки данных.

Сервис исполнения команд отвечает за безопасное выполнение управляющих команд от администраторов, которые он получает через интерфейс пользователя. Сервис уведомлений обеспечивает своевременное оповещение пользователей о критических событиях и аномалиях в работе кластеров. Сервис редактирования эвристических правил предоставляет средства для создания, изменения и управления проверкой ЭП в экспертной системе. Модуль управления подключенными базами данных отвечает за конфиденциальное хранение и маршрутизацию

параметров подключения к кластеру баз данных, обеспечивая доступ к этим данным другим сервисам системы.

Представленная архитектура системы обеспечивает масштабируемость, независимость анализа по каждому кластеру и гибкость конфигурирования за счет модульной структуры.

Аналитическое ядро системы: архитектура и принципы работы

В системе мониторинга кластеров баз данных ключевую роль играет модуль «Аналитическое ядро», объединяющий несколько взаимодополняющих подходов к анализу состояния кластеров. Этот модуль обеспечивает комплексную обработку метрик за счет параллельной работы трех специализированных подсистем, каждая из которых использует собственные методы выявления аномалий. Такой подход позволяет повысить надежность диагностики за счет перекрестной проверки результатов разными методами.

На Рис. 2 представлена детализированная схема модуля «Аналитическое ядро», демонстрирующая его внутреннюю структуру и взаимодействие компонентов. Модуль выполняет основную аналитическую работу по контролю за состоянием кластеров баз данных и реализует три независимых направления анализа: «Мастерская сценариев», «Детектор отклонений и анализа на Т-клетках» и «Детектор неизвестного поведения», которая выполняет функции детектора неизвестного поведения. Каждому отслеживаемому кластеру баз данных соответствует отдельный программный агент-экземпляр каждого из этих трех сервисов, что обеспечивает масштабируемость и независимость анализа. Таким образом, система может параллельно анализировать множество кластеров, адаптируясь под особенности каждого из них.

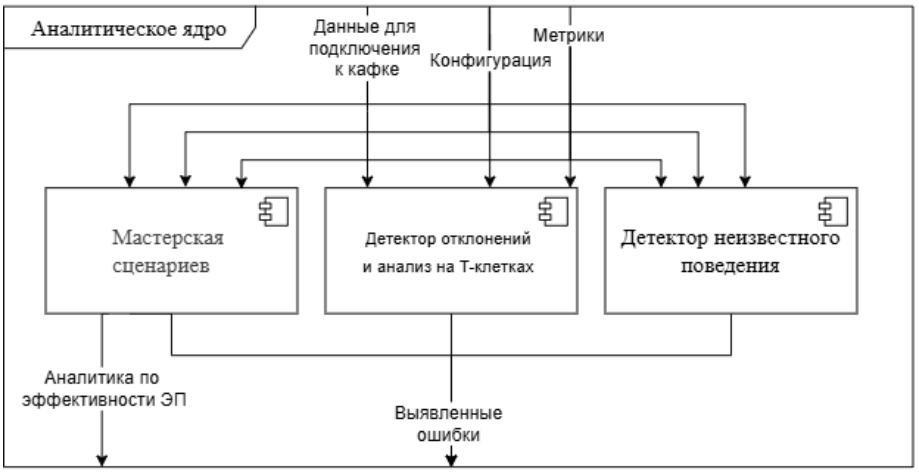

Рисунок 2. Структура модуля «Аналитическое ядро».

Аналитическое ядро включает следующие ключевые компоненты:

– Мастерская сценариев осуществляет обработку эвристических правил, проверяет условия срабатывания и формирует список необходимых действий (выявленных ошибок или отклонений), которые необходимо отработать системе. Модуль анализа эффективности ЭП агрегирует статистику выполнения правил. Если какое-либо правило часто срабатывает, но не приводит к результату (не обнаруживает значимых проблем), оно считается неэффективным. В этом случае модуль формирует уведомление об эффективности данного эвристического правила.

– Детектор отклонений и анализ на Т-клетках реализует методы искусственного иммунитета и занимается выявлением аномалий путем анализа отклонений от обученных паттернов нормального поведения.

– Детектор неизвестного поведения на базе LLM применяет методы Retrieval-Augmented Generation (RAG) и многозадачные агенты на базе больших языковых моделей для интерпретации аномалий и формирования текстовых объяснений и рекомендаций.

Модуль «Аналитическое ядро» является ядром аналитической части системы мониторинга и объединяет различные подходы к анализу и интерпретации данных о состоянии кластера. Все компоненты модуля взаимодействуют через единый интерфейс обмена данными, что обеспечивает согласованную работу при сохранении независимости каждого метода анализа.

Мастерская сценариев

Мастерская сценариев предназначена для сбора и хранения экспертных знаний, сформулированных в виде правил (эвристик), а также за обработку с их помощью данных мониторинга для автоматического контроля состояния кластера БД [10]. Каждое эвристическое правило представляет собой пару «условие – действие»: если условие истинно, то срабатывает соответствующее действие. Условия правил формулируются как логические выражения на основе метрик и событий, поступающих от кластера, а действия представляют собой либо уведомления (предупреждения), либо выполнение некоторых предопределенных алгоритмов (скриптов), использование API СУБД (например, PostgreSQL pg_reload_conf для применения настроек) и т.п. Например:

– «Если процент использования CPU превышает 90% в течение 5 минут, отправить предупреждение администраторам».

– «Если задержка репликации превышает порог X, автоматически переключить кластер на реплику».

Мастерская сценариев реализует следующий механизм проверки и выполнения правил: при поступлении нового набора метрик он последовательно проверяет условия всех активных правил и выявляет те, чьи условия истинны. По сработавшим правилам запускаются соответствующие действия. Чтобы избежать лавинообразного срабатывания, возможна организация правил по приоритетам и группировка правил, связанных с одним событием.

В системе предусмотрена библиотека предустановленных правил (набор типовых сценариев мониторинга). Пользователь (администратор) через соответствующий сервис может отключать или изменять эти правила, а также добавлять новые, характерные для своей инфраструктуры. Правила хранятся в базе знаний (репозитории экспертной системы) и могут быть активированы и деактивированы динамически [11].

На Рис. 3 представлена структура мастерской сценариев: поток метрик поступают на вход модуля проверки ЭП. Вычисляются логические выражения всех активных правил, затем с учетом приоритетов и группировок формируется перечень необходимых к выполнению действий, который, в свою очередь, передаются в сервис исполнения команд.

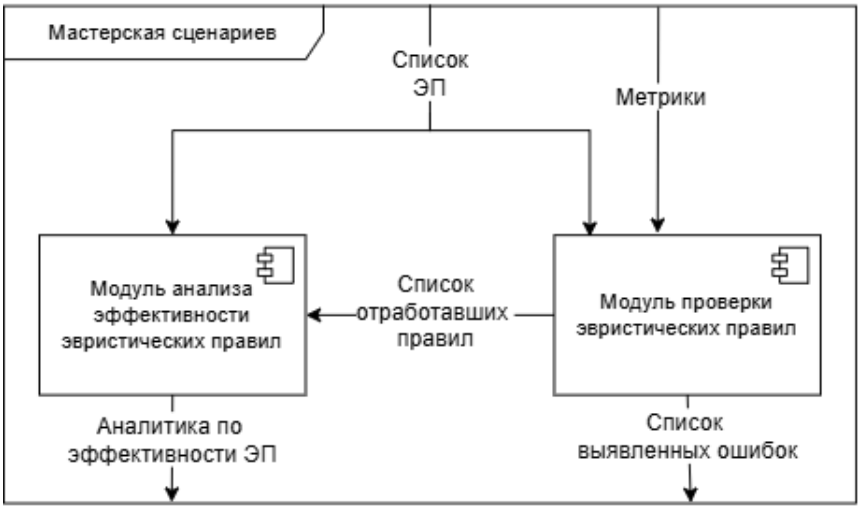

Рисунок 3. Третий уровень декомпозиции «Мастерская сценариев».

При очевидном достоинстве технологии экспертных систем, которое заключается в понятности действий и решений, главное их ограничение в том, что они оперируют только известными заранее шаблонами проблем. Новые, неизвестные экспертам ситуации вероятнее всего останутся незамеченными. Для снятия подобных ограничений кроме мастерской сценариев в составе аналитического ядра системы мониторинга параллельно используются еще технологии искусственных иммунных систем и большая языковая модель.

Детектор отклонений и анализа на Т-клетках

Детектор отклонений и анализа на Т-клетках отвечает за применение методов искусственных иммунных систем (Artificial Immune System, AIS) искусственного иммунитета для мониторинга и выявления аномалий в работе кластера баз данных. Подход, вдохновленный биологической иммунной системой, проявляет способность обнаруживать отклонения («not-self» сигнатуры) без необходимости заранее знать вид конкретной проблемы. Иными словами, детектор обучается распознавать «нормальное» состояние системы и сигнализировать при достаточном отклонении от него [12].

В разработанном детекторе отклонений реализован алгоритм негативной селекции и кластеризации паттернов метрик. На этапе обучения модуль собирает метрики кластера за продолжительный период стабильной работы и формирует множество «самоописаний» нормального состояния. Затем синтезируется множество детекторов, способных реагировать на несовпадение с нормой. Когда система работает в режиме мониторинга, каждый новый набор метрик сравнивается с эталонами: если обнаруживается достаточно «чужеродный» паттерн (несвойственное сочетание значений метрик), детектор отклонений и анализа на Т-клетках расценивает это как потенциальную аномалию и генерирует предупреждение.

Важно, что искусственная иммунная система обладает способностью адаптивного обучения. Наш модуль поддерживает механизмы интерактивной настройки: администратор может пометить найденный модулем инцидент как подтвержденный сбой или ложное срабатывание, на основе этой обратной связи алгоритм подстраивает свои детекторы. Кроме того, модуль может переходить в режим дообучения на актуальных данных, чтобы адаптироваться к изменившемуся «здоровому» состоянию системы (например, когда нагрузка выросла планомерно и прежний порог срабатывания в новых условиях считается нормой).

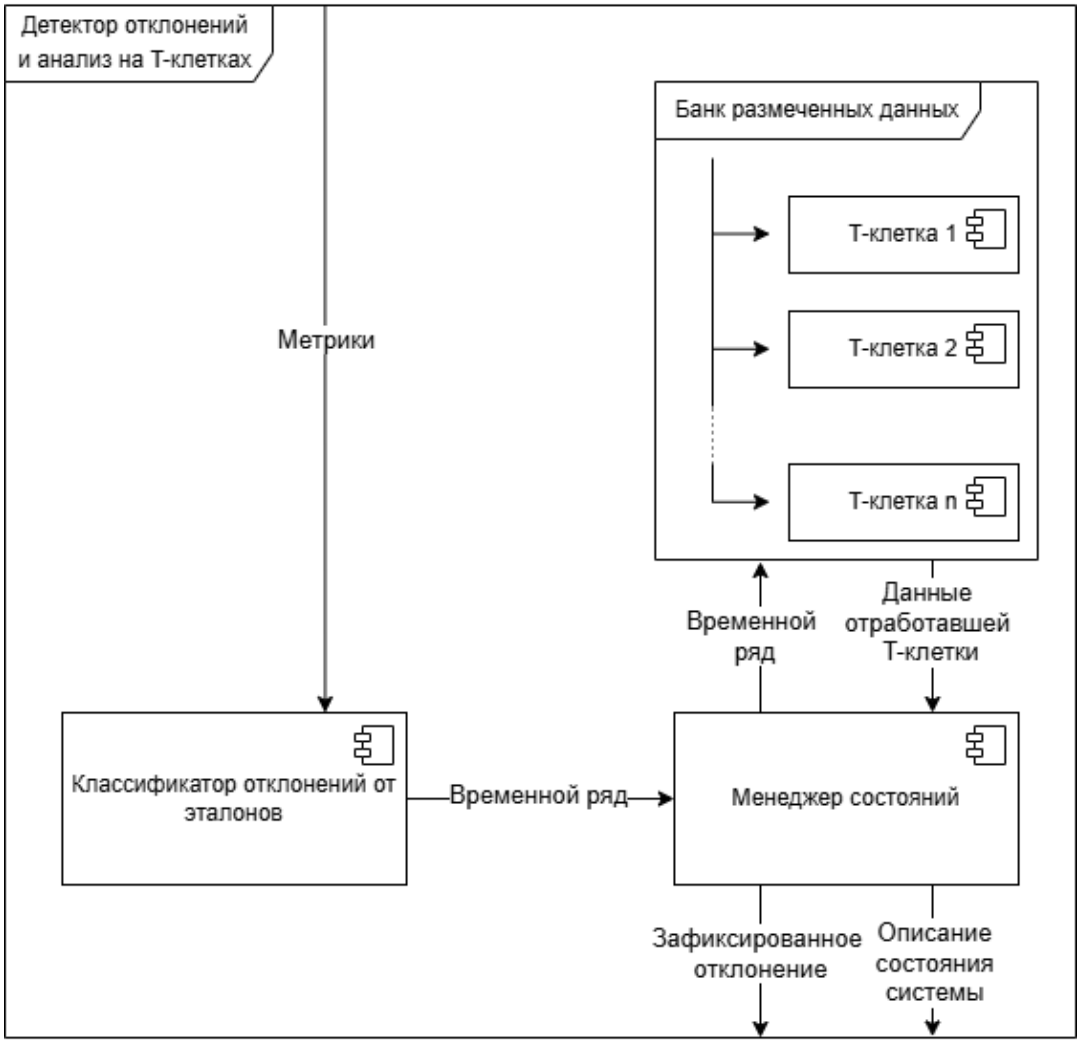

Рисунок 4. Третий уровень декомпозиции «Детектора отклонений и анализа на Т-клетках».

Структура детектора отклонений и анализа на Т-клетках представлена на Рис. 4: на этапе обучения формируется база «self»-паттернов, на этапе мониторинга поток метрик пропускается через набор детекторов, результаты агрегируются и вычисляется мера «аномальности». Если она превышает порог – генерируется событие аномалии, которое передается в систему уведомлений и в модуль LLM для генерации предупреждения на естественном языке.

Следует отметить, что применение AIS в ИТ-мониторинге изучается в ряде исследований по кибербезопасности и обнаружению сбоев [13]. Наш модуль адаптирует эти идеи к домену СУБД, фокусируясь на метриках производительности и отказах, а не только на вторжениях. В сочетании с мастерской сценариев детектор отклонений и анализа на Т-клетках образует двухуровневый механизм: правила ловят известные проблемы, иммунитет – неизвестные.

Детектор неизвестного поведения

Детектор неизвестного поведения (на основе большой языковой модели) интегрирован в систему для интеллектуальной обработки текстовой информации и обогащения выводов мониторинга. Его задачи включают: анализ журналов и сообщений ошибок базы данных, генерацию описаний обнаруженных проблем на естественном языке.

Основой функционирования данного модуля является подход Retrieval-Augmented Generation (RAG) – сочетание поиска релевантной информации и генерации текста с опорой на нее [14]. В контексте нашей системы это реализовано следующим образом: детектор неизвестного поведения, использующий большую языковую модель, имеет доступ к базе знаний, состоящей из документации, руководств, записей инцидентов, накопленных системой, в том числе, описаний сбоев, действия правил, решений, которые принимались ранее. При возникновении нового события (например, иммунная система обнаружила аномалию, либо сработало экспертное правило) LLM-модуль получает описание ситуации (набор метрик, идентификатор правила или параметров аномалии) и делает поиск по базе знаний (например, ищет похожие случаи или соответствующую документацию – так реализуется retrieval). Далее найденная информация подается во вход LLM, которая генерирует пояснение и рекомендации на естественном языке, основанные как на собственной модели (предобученной на широком корпусе ИИ), так и на извлеченных данных. Благодаря этому ответы системы могут быть конкретными и точными, с учетом актуальной внутренней информации, а риск «галлюцинаций» LLM снижается [14].

Кроме того, детектор неизвестного поведения включает многозадачных агентов, решающих вспомогательные подзадачи: один агент может отвечать за классификацию входящего инцидента (типа сбоя), другой – за поиск нужных разделов документации, третий – за формирование итогового отчета. Такая декомпозиция позволяет лучше структурировать работу LLM и повышает надежность (каждый агент имеет ограниченную зону ответственности). Оркестрация агентов также выполнена в Python с использованием фреймворков для создания систем с LLM (например, LangChain или аналогичного).

Схема работы Детектора неизвестного поведения представлена на Рис. 5. Показано, как инцидент проходит через стадию поиска данных, затем через ядро LLM и как формируется ответ.

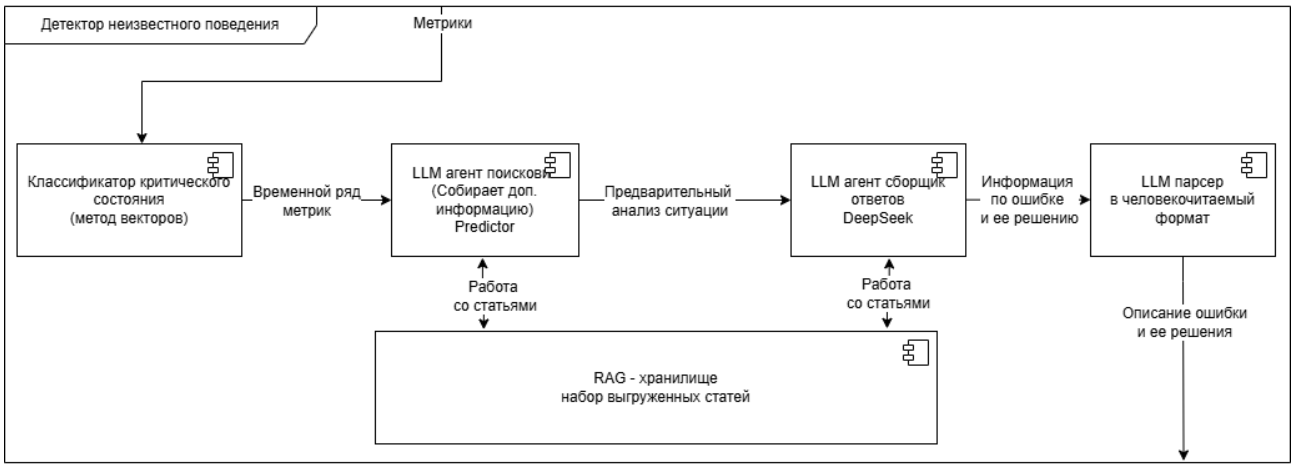

Рисунок 5. Третий уровень декомпозиции «Детектор неизвестного поведения».

Заключение

Предложенная функциональная схема системы мониторинга состояний кластеров баз данных объединяет классические и интеллектуальные подходы для обеспечения проактивного и адаптивного мониторинга. Интеграция мастерской сценариев (эвристических правил), детектора отклонений и анализа на Т-клетках и модуля детекции неизвестного поведения позволяет автоматически обнаруживать аномалии, оперативно реагировать на сбои и минимизировать возможные риски для бизнеса. Экспериментальное прототипирование системы на тестовом кластере PostgreSQL показало, что комбинированное использование правил и обучаемых алгоритмов действительно повышает точность выявления проблем: уменьшилось число пропущенных событий и снизилось число ложных тревог (благодаря адаптации порогов детектора отклонений и анализа на Т-клетках). Детектор неизвестного поведения продемонстрировал способность генерировать полезные рекомендации на основе внутренней базы знаний, тем самым сокращая время анализа инцидентов администратором.

Система основана на технологиях искусственного интеллекта и может со временем самостоятельно улучшать качество мониторинга по мере накопления данных. Такой подход соответствует современной тенденции внедрения AI/ML в ИТ-операции (AIOps) [8]. Автоматизация мониторинга кластеров баз данных призвана не заменить администратора, а вооружить его эффективным инструментом для своевременного обнаружения и устранения проблем. Использование такой системы позволит повысить устойчивость и производительность баз данных, минимизировать время простоя при сбоях и оптимизировать трудозатраты на сопровождение. Сочетание правил, самообучающихся алгоритмов и возможностей больших языковых моделей может стать основой нового поколения интеллектуальных систем мониторинга, востребованных в различных отраслях, работающих с большими данными и критичными информационными инфраструктурами.

Литература:

1. Валеев С.С., Масленников В.А., Ковтуненко А.С. Проектирование промежуточного программного обеспечения на основе агентно-ориентированных технологий для САУ сложными техническими объектами. В сб.: «Параллельные вычисления и задачи управления» PACO 2012: Шестая международная конференция. Труды: в 3 т., Москва, 24–26 октября 2012 года. Т.1. М.: ИПУ РАН, 2012. С. 250–259. [Valeev S.S., Maslennikov V.A., Kovtunenko A.S. Design of middleware based on agent-based technologies for automated control systems of complex technical objects. In: “Parallel Computing and Control Objectives” PACO 2012: Sixth International Conference. Proceedings: in 3 volumes, Moscow, October 24–26, 2012. Vol. 1. Moscow: Institute of Control Sciences of the Russian Academy of Sciences, 2012. P. 250–259]. EDN SODVIZ.

2. Zabbix – свободная система мониторинга. Википедия [электронный ресурс]. [Zabbix – a free monitoring system. Wikipedia [webpage] (in Russian; accessed: 01.04.2025)]. URL: https://ru.wikipedia.org/wiki/Zabbix (дата обращения: 01.04.2025).

3. Zabbix vs Prometheus. Что выбрать для гетерогенной инфраструктуры? Хабр [электронный ресурс]. [Zabbix vs. Prometheus: Which to choose for a heterogeneous infrastructure? Habr [webpage] (in Russian; accessed: 01.04.2025)]. URL: https://habr.com/ru/articles/852394/ (дата обращения: 01.04.2025).

4. Prometheus (software). Wikipedia [webpage]. URL: https://en.wikipedia.org/wiki/Prometheus_(software) (accessed: 01.04.2025).

5. postgres_exporter и мониторинг экземпляров PostgreSQL с несколькими БД. Хабр. [электронный ресурс]. [postgres_exporter and monitoring PostgreSQL instances with multiple databases. Habr [webpage] (in Russian; accessed 01.04.2025)]. URL: https://habr.com/ru/articles/468573/ (дата обращения: 01.04.2025).

6. Database Monitoring Tools for MySQL, MongoDB, PostgreSQL & More. Percona [webpage]. URL: https://www.percona.com/software/database-tools/percona-monitoring-and-management (accessed: 01.04.2025).

7. Installing Percona Monitoring & Management (PMM) with Postgres. Medium [webpage]. URL: https://ozwizard.medium.com/installing-percona-monitoring-management-pmm-with-postgres-972386d38b81 (accessed: 01.04.2025).
8. Машинное обучение в IT-мониторинге. Хабр [электронный ресурс]. [Machine learning in IT monitoring. Habr [webpage] (in Russian; accessed: 01.04.2025)]. URL: https://habr.com/ru/companies/netcracker/articles/442620/ (дата обращения: 01.04.2025).
9. Kovtunenko A.S., Klimova A.V. Agent-based distributed system for the realtime data-driven simulation of complex dynamic systems. In: 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). IEEE, 2019. Art. 8970985. DOI: 10.1109/ICUMT48472.2019.8970985. EDN JIXFZE.
10. Ковтуненко А.В., Ковтуненко А.С. Управление реализацией индивидуальных образовательных траекторий в высшей школе на основе онтологической модели данных // Системная инженерия и информационные технологии. 2023. Т. 5. № 6(15). С. 17–23. [Kovtunenko A.V., Kovtunenko A.S. Management of the implementation of individual educational trajectories in higher school based on ontological data model // Systems Engineering and Information Technologies. Vol. 5. No. 6(15). P. 17–23 (in Russian)]. DOI: 10.54708/2658-5014-SIIT-2023-no6-p17. EDN OEMVIT.
11. Минасова Н.С., Тархов С.В., Тархова Л.М. Модели формирования и практическая реализация скомпилированных учебных модулей в системе электронного обучения // Открытое образование. 2006. № 5. С. 21–29. [Minasova N.S., Tarkhov S.V., Tarkhova L.M. Models of formation and practical implementation of compiled training modules in the e-learning system // Otkrytoe Obrazovanie. 2006. No. 5. P. 21–29 (in Russian)]. EDN JWLIJF.
12. A Survey of Artificial Immune System Based Intrusion Detection. PMC. [webpage]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3981469/ (accessed: 01.04.2025).
13. Bejoy B.J., Raju G., Swain D., Acharya B., Hu Y.-Ch. A generic cyber immune framework for anomaly detection // Applied Soft Computing. 2022. Vol. 130. Art. 109680. DOI: 10.1016/j.asoc.2022.109680.
14. Retrieval-augmented generation. Wikipedia. [webpage]. URL: https://en.wikipedia.org/wiki/Retrieval-augmented_generation (accessed: 01.04.2025).

Об авторах:
ШУНДЕЕВ Артем Владимирович, обучающийся специалитета, ФГБОУ ВО «Уфимский университет науки и технологий», artem_shundeev@mail.ru.
САБЛИН Дмитрий Павлович, инженер по машинному обучению, ООО «ЮНИТС», sablin.dmitrii2024@gmail.com.
ГЛУЩЕНКО Валерий Андреевич, аспирант, ФГБОУ ВО «Уфимский университет науки и технологий», val_g_2001@bk.ru.
КОВТУНЕНКО Алексей Сергеевич, к.т.н., доцент, доцент кафедры информатики, ФБГОУ ВО «Уфимский университет науки и технологий», askovtunenko@mail.ru.

Metadata:
Title: Software architecture of a database cluster state monitoring system based on an agent-oriented approach
Author 1: Artem Vladimirovich Shundeev, student, Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, artem_shundeev@mail.ru.
Author 2: Dmitry Pavlovich Sablin, machine learning engineer, LLC “UNITS”, 15 Entuzazistov st, 450098 Ufa, Republic of Bashkortostan, Russia, sablin.dmitrii2024@gmail.com.
Author 3: Valery Andreevich Gluschenko, Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, val_g_2001@bk.ru.
Author 4: Alexey Sergeevich Kovtunenko, Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, askovtunenko@mail.ru, ORCID ID 0000-0002-3814-7310, Web of Science ResearcherID AAH-5198-2019, Scopus Author ID 57204176565.

Abstract: The article considers a system for monitoring the states of distributed database nodes. The system architecture is presented, the main modules are described, including the expert assessment processing module, the immune algorithms module, and the unknown behavior detector module. Methods for collecting and analyzing metrics, identifying anomalies, and generating recommendations for promptly eliminating the consequences of failures are developed. The system is aimed at increasing the efficiency of monitoring and administering databases, as well as reducing the risks associated with their operation.

Keywords: monitoring, distributed databases, artificial intelligence, machine learning, expert systems, immune systems, large language models.