Подсистема архивации данных системы мониторинга Botikmon3

Н.С. Живчикова[1], Ю.В. Шевчук[1]

1 ИПС им. А.К. Айламазяна РАН, ИЦМС

Аннотация. В статье рассматривается организация подсистемы хранения сенсорных данных общецелевой системы мониторинга Botikmon3. Система Botikmon3 сочетает в себе функциональность брокера сообщений и базы данных временных рядов (TSDB). Botikmon3 это распределённая система, характеризующаяся отказоустойчивостью, горизонтальной масштабируемостью, автоматической балансировкой нагрузки на узлы вычислительного кластера. Система отличается возможностью приема нерегулярно поступающих данных и данных с нарушением хронологического порядка. Botikmon3 использует TSDB Akumuli для хранения данных на каждом из узлов и интегрирует независимые хранилища в единую распределённую систему. При этом необходимо решать задачи сбалансированного распределения данных по узлам при записи и нахождения нужных данных в распределённой системе при обработке запросов на чтение. Подсистема хранения использует распределённое хранилище типа «ключ-значение» Riak KV для хранения карт распределения данных по узлам и другой служебной информации, а также организации взаимодействия экземпляров Botikmon3, работающих на каждом из узлов. В статье предложен механизм «рециркуляции» данных, дающий возможность задавать различное время хранения данных от разных источников и обеспечивающий балансировку нагрузки на узлы кластера. Проект находится на стадии реализации.

Ключевые слова: сенсорные данные, распределённые системы, RiakKV, Akumuli

Data storage in Botikmon3 monitoring system

N.S. Zhivchikova[1], Yu. V. Shevchuk[1]

1 The Program Systems Institute of RAS

Abstract. The article treats the sensor data storage subsystem of Botikmon3, a general-purpose monitoring system. Botikmon3 works both as a sensor data broker and as a time series data base (TSDB). Botikmon3 is a distributed system that features high availability, fault tolerance, horizontal scalability and cluster load balancing. The system works with both periodic and non-periodic time series, and can

accept data arriving out of chronological order. Botikmon3 uses Akumuli TSDB as a time series data storage backend on every cluster node, integrating the individual storage segments into one distributed system. To do this one has to distribute data evenly over the individual storage segments when inserting new data, and to locate requested data when handing read requests. Botikmon3 utilizes Riak KV, the distributed fault-tolerant key-value store, to store and exchange the data placement maps between the cluster nodes. A data recirculation mechanism is proposed to facilitate different retention periods for different metrics and also to help solving the problem of cluster load balancing. The project is a work in progress.

Keywords: sensor data, write performance, distributed storage, time series, Riak, Erlang

1. Введение

Система мониторинга реализует сбор и хранение сенсорных данных, а также предоставление их пользователям для визуализации и анализа. Системы мониторинга позволяют получать информацию, необходимую для успешной эксплуатации больших объектов, таких как телекоммуникационные сети, диагностики, анализа причин возникающих неисправностей, прогнозирования неисправностей и их предупреждения. Они могут также применяться для автоматизации научных исследований.

Сенсорные данные, с которыми работает система мониторинга, поступают от большого числа датчиков (порядка 10^6); каждая порция данных содержит идентификатор датчика, отметку времени и числовое значение.

В настоящей статье рассматривается проект распределённой, горизонтально масштабируемой системы хранения сенсорных данных Botikmon3, в которой в качестве базового хранилища используется база данных временных рядов Akumuli [2], а в качестве средства интеграции — распределённое NoSQL хранилище типа «ключ-значение» Riak KV.

Одно из запланированных практических применений – усовершенствование существующей системы мониторинга телекоммуникационного оборудования в сети небольшого регионального Интернет-провайдера.

2. Akumuli: краткое описание

Akumuli это отечественная разработка с открытым исходным кодом (лицензия Apache2). Автор Евгений Лазин. Для хранения данных использует оригинальное дерево (NB+Tree [3]), в котором сохранение данных происходит эффективно за счет того, что дерево пополняется только справа, так что при пополнении не требуются операции «чтение-модификация-запись». Записанные на диск узлы дерева, как листья, так и внутренние узлы, никогда не модифицируются. Можно рассматривать хранилище Akumuli как лог-файл с индексом: журнальная организация обеспечивает эффективную запись, индекс обеспечивает эффективный поиск данных. При сохранении данных применяется специализированный алгоритм компрессии. Во внутренних узлах

дерева сохраняется дополнительная информация, позволяющая «на лету» эффективно вычислять агрегированные значения для выборок за большие интервалы времени с пониженным разрешением. Данные буферизуются в оперативной памяти перед сохранением на диск, но расход памяти сравнительно небольшой, порядка 16КБ на датчик. Хранилище имеет циклическую структуру: самые старые данные удаляются с диска, новые данные записываются на их место. Данные хранятся без потери разрешения.

Ограничения:

1) Работает на одиночном компьютере, но не на распределённых системах, поэтому трудно обеспечить отказоустойчивость и масштабирование.

2) Данные в каждом временном ряду должны поступать в хронологическом порядке, запись старых данных невозможна. Это ограничение несущественно при нормальной работе системы, но создаёт неудобства при практической эксплуатации. Пример: сбой на системе-источнике данных, после перезагрузки система сразу начинает генерировать новые данные, а потом удаётся восстановить лог-файл с более старыми данными, но Akumuli уже не может их принять из-за нарушения хронологического порядка.

3) Невозможно задать разное время хранения для разных датчиков.

3. Принципы организации Botikmon3

Мы строим систему Botikmon3 на кластере из нескольких узлов и имеем целью высокую доступность системы и неограниченное горизонтальное масштабирование. Конфигурация кластера может меняться без остановки системы: могут возникать ситуации «разделения» – нарушения связи или выход из строя узлов кластера. Также могут добавляться новые узлы с целью увеличения ёмкости и производительности системы.

На каждом узле работает ОС Linux, экземпляр Akumuli и процесс Botikmon. Кроме того, на тех же узлах работает распределённое хранилище типа «ключ-значение» Riak KV, которое Botikmon использует для хранения метаданных и для межузлового взаимодействия (рис. 1).

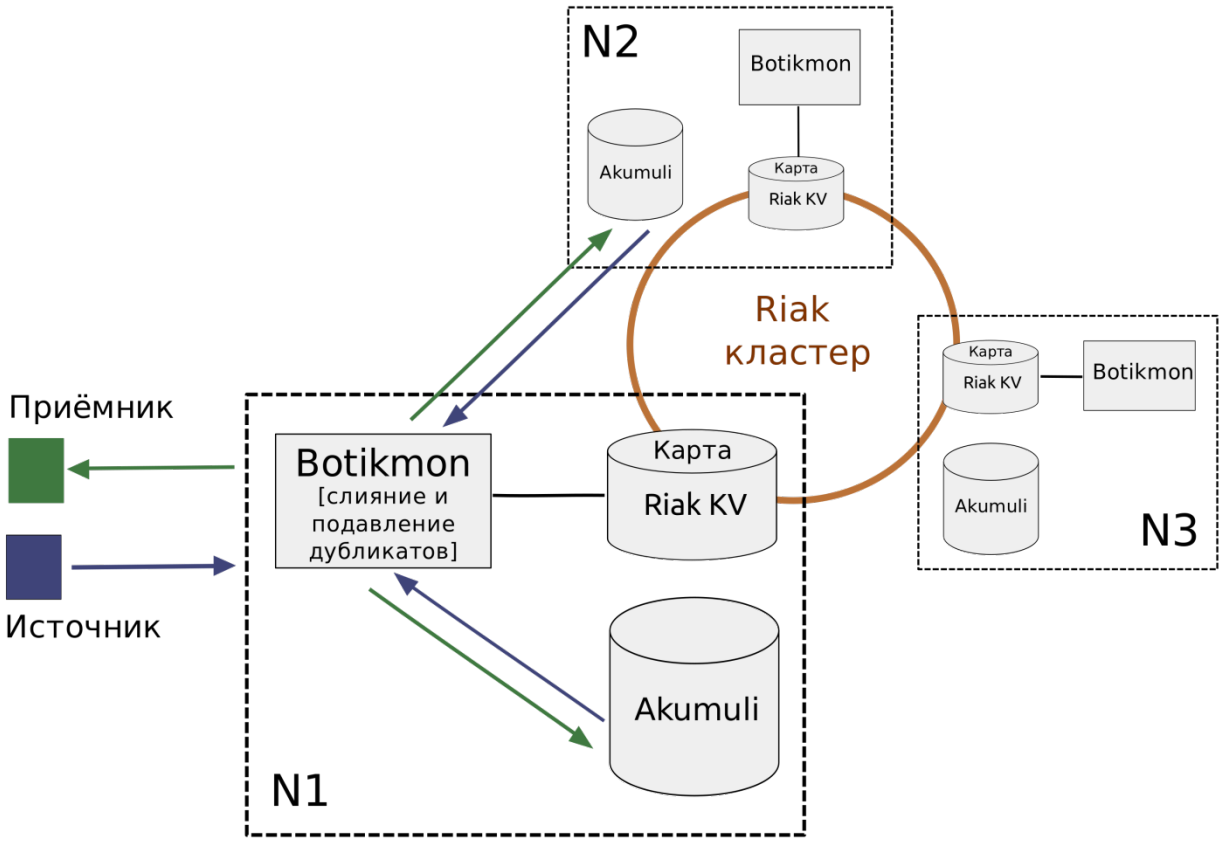

Рис. 1. Архитектура распределённого хранилища

При поступлении на вход системы данных от нового датчика Botikmon выбирает, в каком экземпляре Akumuli (на каком узле) будут сохраняться его данные. Возможно сохранение с репликацией данных на нескольких узлах; коэффициент репликации задаётся в конфигурации Botikmon и чем он выше, тем выше доступность системы на чтение.

В ходе работы системы привязка датчика к узлам может меняться. Например, если один из назначенных датчику узлов становится недоступен, система должна назначить взамен другой узел, чтобы поддержать заданный конфигурацией коэффициент репликации. Если в кластер добавляется новый узел, часть датчиков должна быть переназначена на него для равномерного использования объёмов Akumuli на всех узлах. Botikmon хранит историю (карту) назначений для каждого датчика в отдельном объекте Riak (рис.2), таким образом карта каждого датчика доступна всем узлам кластера на условиях «согласованности в конечном счёте».

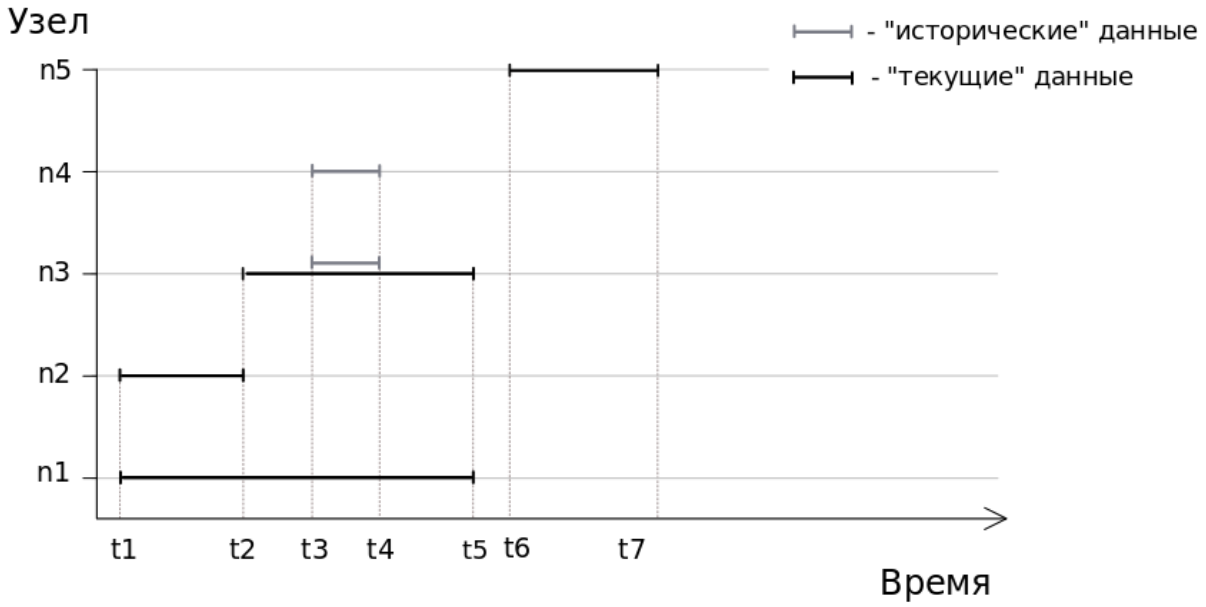

Рис. 2. Пример карты распределения данных одного датчика по узлам

Карты используются при обработке запросов на чтение. Узел, получивший запрос, запрашивает в Riak карту требуемого датчика и по ней определяет, к каким экземплярам Akumuli следует обращаться за данными. Карты также запрашиваются при каждом подключении к системе источника данных. Если данный источник ранее появлялся в системе (подключался к тому же или другому узлу), карты для его датчиков уже существуют и Botikmon в первую очередь пытается продолжить имеющиеся в карте интервалы, чтобы избежать изменения привязки датчика к узлам. Это позволяет избежать увеличения размера карты и использовать возможности Akumuli в части выборок со снижением разрешения данных. Если продолжить интервал не удаётся (например, указанный в карте узел недоступен), выбирается новый узел и в карту добавляется запись о новом интервале.

Использование карт позволяет обойти требование хронологического порядка данных. При подключении к системе источника «исторических» данных от уже известного системе датчика в Akumuli создается временная серия с новым именем, а в карте датчика делается ссылка на неё. С точки зрения Akumuli ситуация выглядит так: появился новый датчик и поступили данные от него, со старыми датами но в хронологическом порядке. А на уровне Botikmon к карте существующего датчика добавляется запись о наличии данных за некоторый интервал в прошлом, возможно пересекающийся с ранее существовавшими в карте интервалами.

При получении запроса на чтение Botikmon выполняет слияние данных из перекрывающихся интервалов (сортировка слиянием с подавлением дубликатов). Строго говоря, карту нужно обновлять после поступления каждой порции данных — продвигать дату конца текущего интервала. Но записывать изменения карт в Riak после каждой порции данных затратно. Поэтому мелкие

изменения карт (только дата текущего интервала) буферизуются в оперативной памяти и сохраняются в Riak раз в минуту. Важные изменения (добавления новых привязок датчиков к узлам) сохраняются немедленно.

Рассмотрим пример карты, показанный на Рис. 1. Согласно этой карте, в системе нет данных от данного датчика, датированных ранее чем t1. В момент времени t1 поступила первая порция данных от датчика, коэффициент репликации был равен 2 и датчику были назначены два узла – n1 и n2. В момент времени t2 произошло переназначение (возможно, из-за того что узел n2 стал недоступен, или узел n3 был недогружен), и вместо узла n2 датчику был назначен узел n3. Таким образом, в период [t2, t5] данные датчика сохранялись на узлах n1 и n3. В момент t6 произошло разделение системы, причем такое, что источник данных оказался в одной части сети с узлом n5, а все остальные узлы в другой части. В этой ситуации сохраняется доступность системы на запись: данные датчика стал принимать узел n5. Хотя коэффициент репликации был установлен равным 2, в рассматриваемой ситуации разделения узел n5 оказался один и поэтому временно используется коэффициент репликации 1; система исправит ситуацию после того как разделение закончится.

Также в карте отмечено наличие «исторических» данных в интервале [t3,t4]. Источник этих данных подключился к системе в интервале [t3,t5], после прихода текущих данных с временной отметкой t3. В первой же порции исторических данных система обнаружила нарушение хронологической последовательности и для их сохранения создала новые временные серии в Akumuli. При этом узлы для сохранения этих серий были выбраны из соображений выравнивания нагрузки между узлами, без всякой связи с тем, какие узлы назначены для сохранения текущих данных.

4. Длительность хранения

Akumuli реализует циклическое хранение данных по принципу FIFO-буфера. Хранилище состоит из нескольких (N) заранее созданных файлов постоянного размера. При записи в текущий файл Akumuli отслеживает заполненность файла. Когда файл Fi заполняется целиком, запись переключается на файл Fj, где j=(i+1) mod N. Если файл Fj был ранее заполнен, он помечается как пустой; бывшие в нём данные с этого момента недоступны. Это простая и эффективная реализация, хотя не гибкая: у данных всех датчиков одинаковое время жизни, а на практике бывает нужно задать увеличенное или уменьшенное время хранения для некоторых датчиков.

При переходе к распределённому хранилищу с несколькими экземплярами Akumuli ситуация осложняется: из-за неравномерности распределения датчиков по узлам и разного темпа поступления информации от разных датчиков темп заполнения хранилищ разный, и длительность хранения данных на разных узлах получится не просто разная, но ещё и непредсказуемая. Ситуация ещё усугубляется при изменениях конфигурации кластера.

Предлагаемое решение использует «рециркуляцию» данных. Все старые данные, приближающиеся к удалению, вновь подаются на вход Botikmon и фильтруются по заданной в конфигурации политике длительности хранения. Данные с вышедшим сроком хранения отбрасываются, а остальные вновь сохраняются в Akumuli «ещё на один срок».

В конфигурации политики длительности хранения, кроме явного задания срока, допускается также значение «автоматически», которое действует и для всех датчиков, для которых срок хранения не задан явно. Для таких датчиков система автоматически выбирает порог рециркуляции данных так, чтобы темп удаления старых данных соответствовал среднему темпу поступления новых данных в систему. Таким образом, обеспечивается максимально возможный период хранения.

Литература

1. Живчикова Н.С., Шевчук Ю.В. Масштабируемая распределенная система архивирования сенсорных данных // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2017. — С. 157-169. — URL: http://keldysh.ru/abrau/2017/35.pdf doi:10.20948/abrau-2017-35.
2. Akumuli Time Series Database. — URL: http://akumuli.org
3. Лазин Е. Numeric B+tree reference. — URL: https://docs.google.com/document/d/1jFK8E3CZSqR5IPsMGojm2LknkNyUZA7tY51N6IgzW_g/pub
4. Basho Technologies. — URL: http://basho.com/products/#riak .