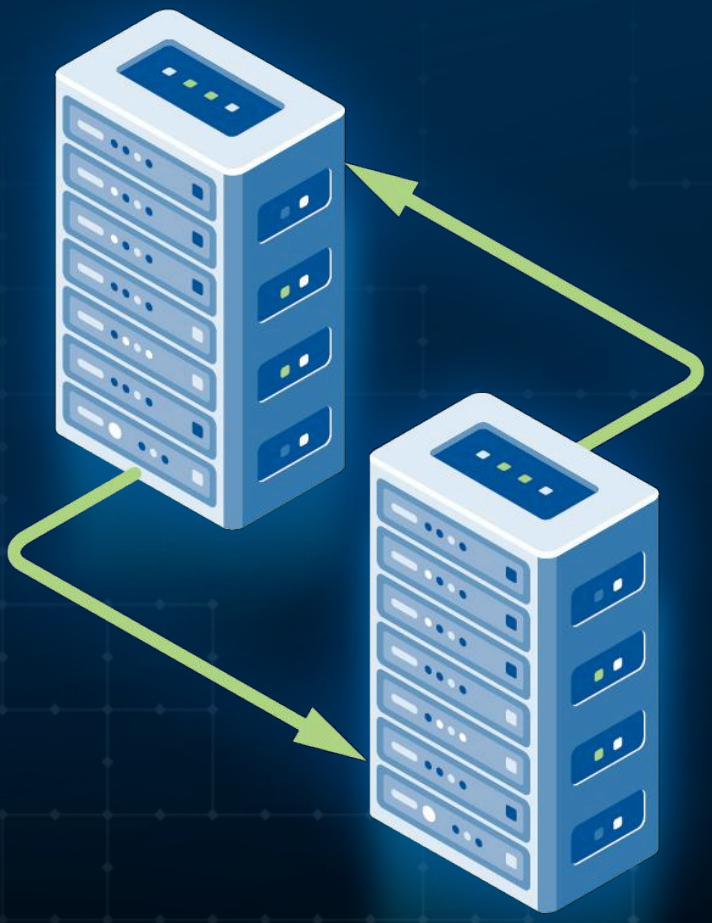
Среда исполнения Альфа платформы

Модель резервирования и новые механизмы



CONTROL ENGINEERING Россия

Атомик Софт

СРЕДА ИСПОЛНЕНИЯ АЛЬФА ПЛАТФОРМЫ: НОВЫЕ МЕХАНИЗМЫ И МОДЕЛЬ РЕЗЕРВИРОВАНИЯ

ДЕНИС РАКИТИН
ЕКАТЕРИНА ВЕРТОПРАХОВА

Альфа платформа — это инструментальная программная платформа, позволяющая реализовать проекты автоматизации технологических и производственных процессов практически любой сложности.
В конце 2023 г. мы провели масштабное обновление Альфа платформы, которое затронуло Alpha.Server — компонент среды исполнения, отвечающий за оперативный сбор, обработку и предоставление данных. Основой изменений в Alpha.Server стал журнал транзакций — центральный высокопроизводительный тракт для обмена событиями между модулями сервера. На его базе мы реализовали новую транзакционную модель исполнения, которая повлияла на резервирование, механизм передачи данных и подсистему сигнализаций.
В статье, продолжающей цикл публикаций об Альфа платформе, речь пойдет о журнале транзакций, особенностях новой транзакционной модели на примере процесса резервирования и принципах работы новых механизмов Alpha.Server.
Обзор архитектуры и возможностей использования был опубликован ранее в Control Engineering № 1'2023.

ЖУРНАЛ ТРАНЗАКЦИЙ И НОВЫЙ МЕХАНИЗМ РЕПЛИКАЦИИ

О журнале транзакций

Журнал транзакций представляет собой единый высокопроизводительный тракт для обмена любыми событиями между модулями-поставщиками и модулями-потребителями. С его внедрением любые события внутри сервера стали рассматриваться как транзакции.

Принцип работы журнала транзакций представлен на рис. 1. В первую очередь в журнал транзакций помещаются обновления данных, получаемые от источников. Модули-процессоры (модули Alpha.Server, вычисления, AE и др.) осуществляют параллельное чтение этих данных из журнала и в него же записывают результаты обработки: вычисленные значения, активации сигнализаций, изменения состояний. Таким образом, весь поток событий в Alpha.Server проходит через журнал, откуда он становится доступным для потребителей.

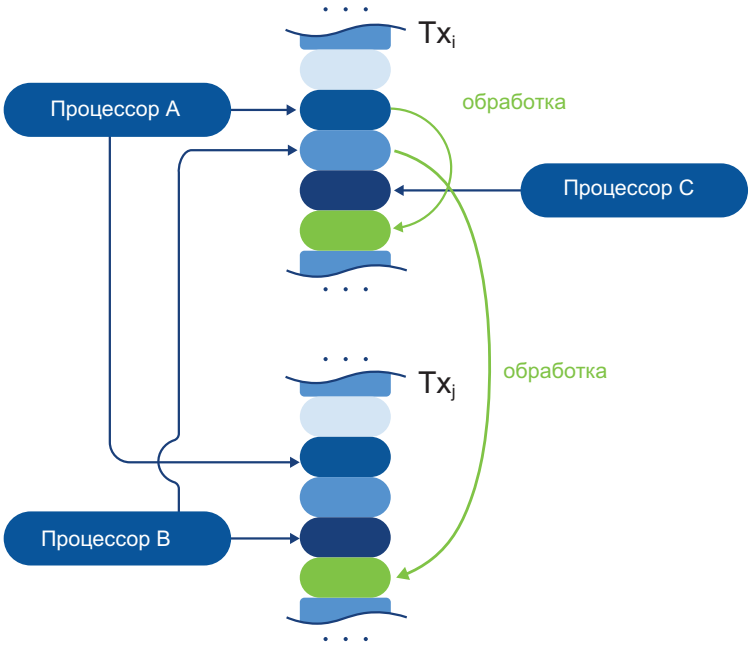


РИС. 1. ▶ Принцип работы журнала транзакций

При реализации журнала транзакций мы учли случаи, когда важны как каждое событие в отдельности, так и их очередность. События, поступающие в журнал, выстраиваются в строгую последовательность, что гарантирует единый порядок их обработки.

«Прародителем» журнала транзакций стал модуль вычислений в Alpha.Server с его характерной чертой — строгим порядком выполнения задач. Так, задачи, генерируемые в модуле в ответ на каждое событие, размещаются в очередь в порядке их генерации. Модуль последовательно обрабатывает их, используя актуальные на момент постановки конкретной задачи значения зависимых переменных, получая всю необходимую информацию из кэша внутренних состояний. Описанный подход и был обобщен на весь сервер.

Журнал транзакций сыграл главную роль в последних изменениях на Alpha.Server: на его основе мы реализовали и новую модель резервирования, и два новых механизма — буферизацию и оперативную таблицу событий. Рассмотрим их более подробно.

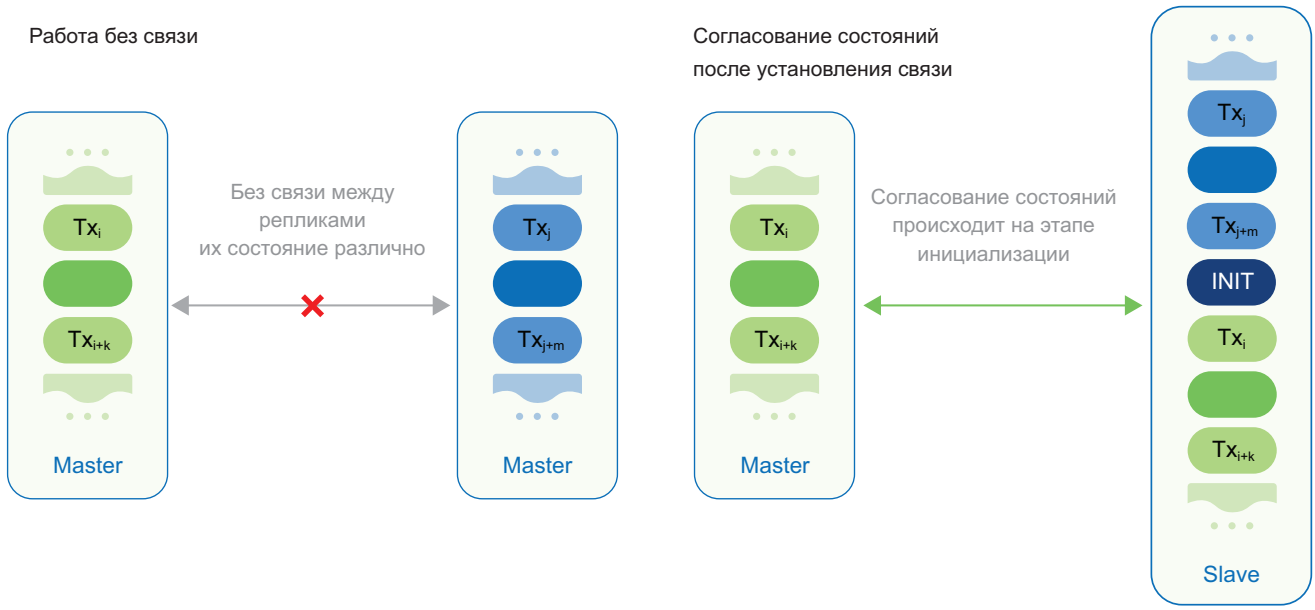


РИС. 2. Принцип инициализации

Новая модель резервирования: Master-Slave-репликация

Новая модель резервирования является полноценной Master-Slave-репликацией: в ней активный сервер приобретает статус Master, а пассивный — Slave. Журнал транзакций в данной модели играет роль механизма, с помощью которого реплики согласовывают свои состояния: он непрерывно реплицируется с Master на Slave и применяет изменения на пассивном сервере, обеспечивая абсолютную идентичность состояний резервной пары.

До обновления производительность резервной пары была в целом ниже: модуль вычислений и модуль AE обрабатывали события на обеих репликах. Сейчас же в штатном режиме генерация событий и все необходимые вычисления производятся только на одной из реплик — той, что работает в режиме Master. Затем вычисленные значения отправляются на Slave через журнал транзакций. Таким образом, пассивный сервер больше не дублирует вычисления, а только принимает и доставляет до клиентов транзакции. Кроме того, Slave гарантированно приходит в идентичное состояние с Master с помощью непрерывной репликации журнала транзакций, но это происходит с незначительной задержкой.

ИНИЦИАЛИЗАЦИЯ

Мы кратко рассмотрели, как с помощью журнала транзакций реплики согласовывают свои состояния в новой транзакционной модели, но не ответили на вопрос, с какого момента начинается процесс Master-Slave-репликации. Отправной точкой взаимодействия резервной пары стала инициализация — обязательный этап, по завершении которого начинается репликация журнала транзакций.

Когда серверы резервной пары впервые подключаются друг к другу, Master собирает актуальный срез своего состояния и отправляет его на Slave, обеспечивая постепенное согласование состояний — инициализацию. После того как все инициализационные данные переданы, начинается штатная репликация журнала транзакций — с той позиции, с которой Slave присоединился к Master.

Инициализация осуществляется не только при первом подключении реплик, но и после временной потери связи между ними. Это вызвано тем, что при отсутствии связи обе реплики работают в режиме Master и производят независимые вычисления, поэтому их состояния могут быть различными и при восстановлении штатного режима работы нуждаются в согласовании. Процесс инициализации, осуществляемый после восстановления связи между резервной парой, показан на рис. 2.

Важно отметить, что во время инициализации работа Master не останавливается: он продолжает прием данных и обработку событий. Продолжительность инициализации зависит от сложности и размеров конфигурации, но на типовых конфигурациях (десятки тысяч сигналов) этот процесс проходит за доли секунды.

ПРИОРИТЕТ КАК ОСНОВА РЕЗЕРВНОГО ПЕРЕХОДА

Алгоритм резервного перехода также претерпел изменения. Прежняя модель основывалась на синхронных переходах, а сейчас в основе механизма лежит новая характеристика — приоритет. Он распределяется парой реплицируемых серверов самостоятельно по критериям связи с полем, времени старта или приоритету, заданному администратором вручную. Реплика, обладающая приоритетом, будет стремиться оставаться в состоянии Master, а неприоритетная — в Slave.

Резервный переход в новой модели исполнения является операцией передачи приоритета. Рассмотрим, как этот процесс происходит в двух возможных случаях: штатном и нештатном.

Штатный резервный переход, инициируемый пользователем, происходит по следующему алгоритму:

1. Оператор вручную обновляет (переназначает) приоритет одной из реплик.
2. Резервная пара получает информацию о внесенных изменениях и подтверждает необходимость резервного перехода.
3. После подтверждения Master последовательно переходит в Slave, а Slave в Master через состояния «резерв-резерв», причем во время резервного перехода бывший Master не прекращает передачу событий журнала транзакций.
4. Когда бывший Slave завершит переход к состоянию Master, он начнет обрабатывать транзакции с позиции, следующей за последней обработанной парным сервером (рис. 3).

Штатный автоматический резервный переход происходит при потере Master связи с полем. В данном случае, как и при переходе, инициируемом пользователем, Slave и Master меняются состояниями, и позиция,

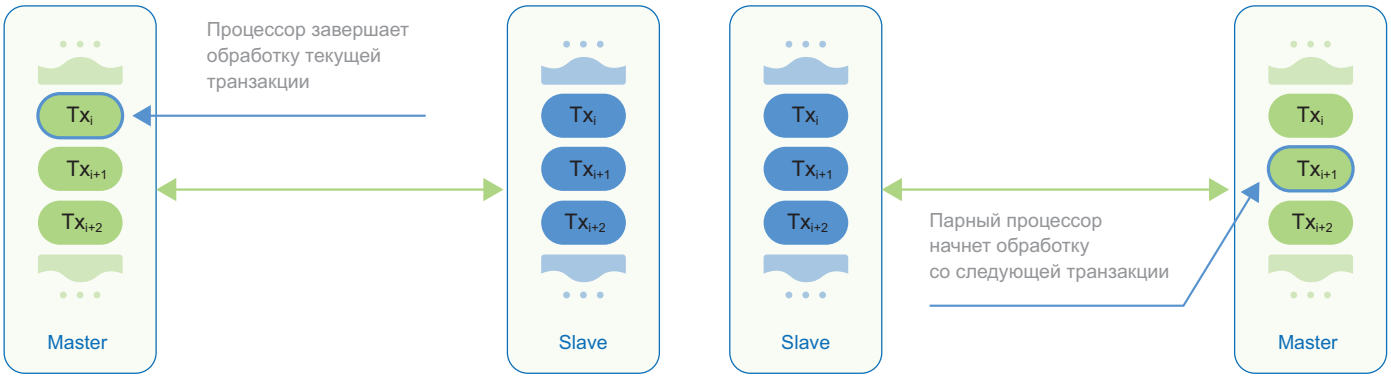


РИС. 3. Принцип штатного резервного перехода

с которой бывший Slave продолжит чтение, будет следующей за последней обработанной Master.

Нештатный резервный переход осуществляется, когда Slave теряет связь с Master. В этом случае Slave переходит в режим «аварийного» Master и начинает обрабатывать транзакции. Если отсутствие связи вызвано сбоем соединения, обе реплики выполняют независимые вычисления; при работе в таком режиме возможно дублирование данных, однако потери сводятся к нулю. Если произошел сбой в работе Master, то потери ограничиваются временем определения отсутствия связи между репликами. После восстановления связи или перезапуска бывшего Master реплики самостоятельно определят приоритет, основываясь на своем фактическом состоянии, и продолжат штатную работу.

Итак, реализованный на основе приоритета резервный переход исключает потерю данных при штатном переходе и минимизирует их в случае сбоя работы одного из серверов резервной пары.

ДЕТАЛЬНАЯ СТАТИСТИКА

Новая транзакционная модель позволяет предоставлять больший объем информации о состоянии пары реплицируемых серверов. Оценить состояние реплик дает возможность «статус аварии», который важен прежде всего для конечных пользователей Альфа платформы.

Так, взаимодействие пары реплицируемых серверов описывает один из трех индикаторов:

- штатное состояние — работа реплицируемых серверов происходит в обычном режиме;
- предупреждение — наблюдаются отклонения (например, приоритетный сервер потерял связь с полем или слишком долго выполняет операцию);
- авария — состояние, из которого реплика не может выйти самостоятельно, и требуется вмешательство оператора.

Вместе со статусом аварии в статистике сервера и сервисных сигналах отображается информация о причинах, которые привели к нештатной ситуации в работе реплик. На рис. 4 для примера схематично показано, что в случае потери связи между репликами оператор увидит соответствующий предупреждающий знак и информацию о работе реплик в режиме «аварийного» Master из-за отсутствия связи.

Детальная статистика не ограничивается одним лишь статусом аварии и ее причинами. Клиент получает подробную информацию о состоянии работы на обеих репликах вплоть до идентификатора транзакции, до которой дошла каждая из реплик.

А теперь перейдем к механизмам, которые мы смогли реализовать благодаря возможностям новой транзакционной модели и журнала транзакций.

Изменения в Alpha.Server, которые мы реализовали на основе журнала транзакций и новой модели исполнения, привели к следующим положительным эффектам:

- упорядоченность данных — реплики соблюдают единый порядок обработки транзакций, а их состояние идентично;
- согласованность состояний реплик после возобновления связи — достигается благодаря этапу инициализации;
- минимизация потерь при нештатных ситуациях — в случае разрыва связи между реплицируемыми серверами бывший Slave-сервер переходит в режим Master и продолжает обработку транзакций с максимально близкой позиции;
- сокращение потребления вычислительных ресурсов — Slave-сервер больше не дублирует вычисления, что освобождает ресурсы на полезную работу в целом;
- предоставление детальной информации о состоянии реплик.

БУФЕРИЗАЦИЯ

Буфер и журнал транзакций. Принципы буферизации

Одной из основных производных журнала транзакций и новой модели исполнения стал механизм буферизированной передачи потока данных между

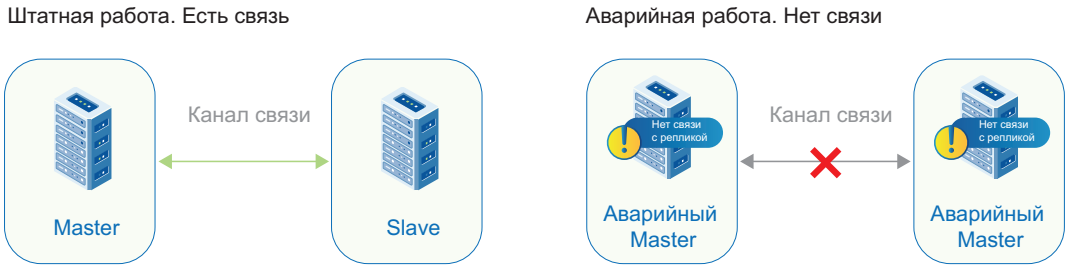


РИС. 4. Статус аварии

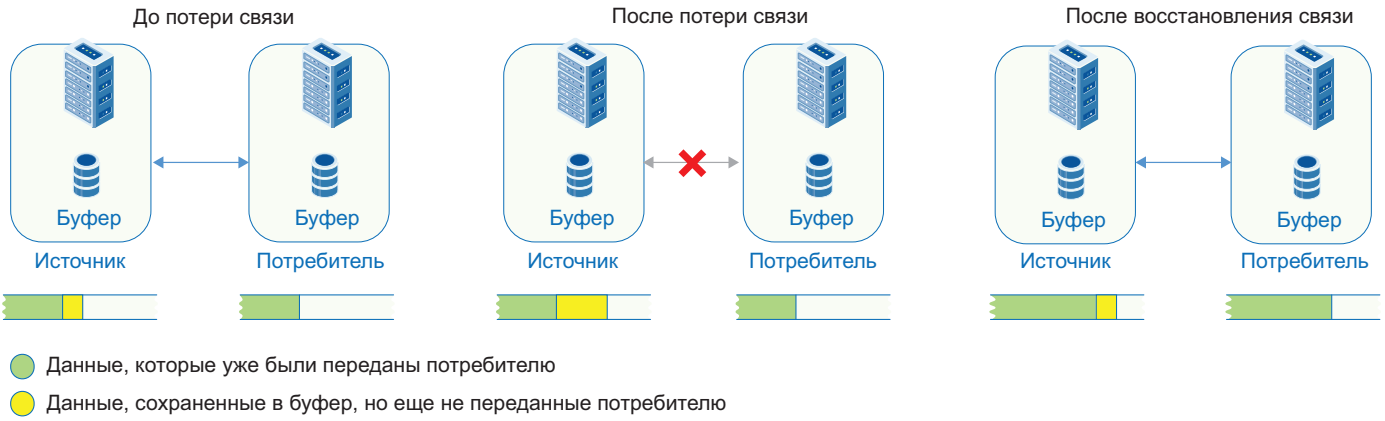


РИС. 5. ▲ Идея буферизации

узлами Alpha.Server, осуществляемый по модели «источник-потребитель». Механизм отвечает за сохранность актуальных данных в случае временной потери связи между отдельными экземплярами сервера — узлами, выступающими в роли источников и потребителей буферизируемых данных.

Буфер представляет собой очередь событий, которые сохраняются на стороне узла-источника. Он отражает, хоть и не в полном объеме, журнал транзакций, включая события только по тому подмножеству данных, которое определил пользователь.

Буфер хранится локально на диске каждого сервера: его размер ограничивается объемом свободного пространства и пользовательскими настройками. Он имеет свой внутренний бинарный формат, а значит, не требует установки дополнительного программного обеспечения.

Механизм буферизации, схематично показанный на рис. 5, осуществляется по следующему алгоритму:

1. На содержимое буфера узла-источника подписываются узлы-потребители, которых может быть несколько.
2. При создании подписки для потребителей в буфере определяется позиция, с которой начинается чтение.
3. Узлы-потребители получают срез буферизованных данных, актуальных для позиции в буфере, к которой было произведено подключение.
4. Начинается штатное чтение буфера; его данные стираются после того, как их получат все подписавшиеся узлы-потребители.

Подписка и, соответственно, позиция сохраняются на время потери связи между узлами, а узел-источник продолжает копить и хранить нужные события в буфере. При восстановлении связи потребитель запрашивает накопившиеся для него данные и начинает их обрабатывать, подтверждая обработку и перемещая свою позицию в буфере.

Но как определяется позиция, с которой начинается чтение буфера? В случае, когда подключение к узлу-источнику происходит впервые или его буфер успел очиститься, узел-потребитель начинает читать буфер либо с его начала, либо с конца, что зависит от конфигурации, заданной пользователем. Если узел-источник был на связи ранее, а его буфер не успел очиститься до восстановления связи, то узел-потребитель подключается к узлу-источнику и возобновляет чтение буфера с последней подтвержденной позиции.

Буферизация — это принципиально новый, но важный механизм в Alpha.Server. При его реализации мы придерживались следующих принципов:

- минимизация использования оперативной памяти — если потребителя нет на связи долгое время, буфер не будет занимать ОЗУ поставщика, так как его содержимое хранится локально на диске;
- устойчивость к перезагрузкам — данные для потребителя сохраняются на жестком диске, а значит, останутся доступны после перезагрузки сервера;
- единый порядок данных на поставщике и потребителе — события сохраняются и читаются в буфере с соблюдением единого строгого порядка и в том же порядке передаются подписчикам;
- обеспечение полноты данных на потребителе — при временном разрыве и восстановлении связи позиция подписчика сохраняется, и он продолжает читать буфер с того же места.

Далее рассмотрим особенности механизма буферизации в различных схемах резервирования источников и потребителей.

Буферизация и процесс резервирования

При использовании схемы «источник-потребитель» без резервирования (рис. 6) буферизация является относительно простой задачей:

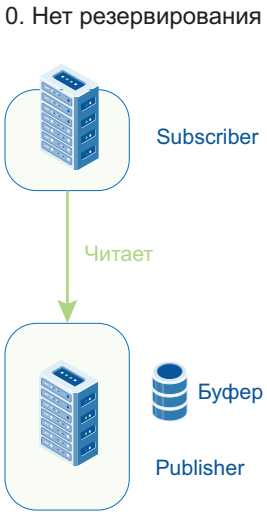


РИС. 6. ◀ Буферизация в схеме без резервирования

1. Узел-потребитель (Subscriber) подключается к буферу узла-источника (Publisher) и, если это подключение первое, получает срез актуальных для своей позиции буферизованных данных.
2. Происходит штатное чтение буфера.

При резервировании потребителя согласованность содержимого буфера образованной резервной пары Master (Subscriber.0) и Slave

РИС. 7. ▼ Буферизация в схеме с резервированием потребителя

РИС. 8. Буферизация для схемы с резервированием источника и потребителя

(Subscriber.1) осуществляется с помощью механизма репликации.

Буферизация в данной схеме (представлена на рис. 7) происходит по следующему алгоритму:

1. Узел-потребитель Master (Subscriber.0) запрашивает и читает буферизованные данные узла-источника (Publisher).
2. С помощью механизма репликации узел-потребитель Master (Subscriber.0) передает буферизованные данные на Slave (Subscriber.1).
3. Slave переставляет позицию в буфере узла-источника (Publisher) вслед за своим Master.

Такая логика взаимодействия с источником гарантирует, что в случае резервного перехода новый Master продолжит чтение буфера с актуальной позиции.

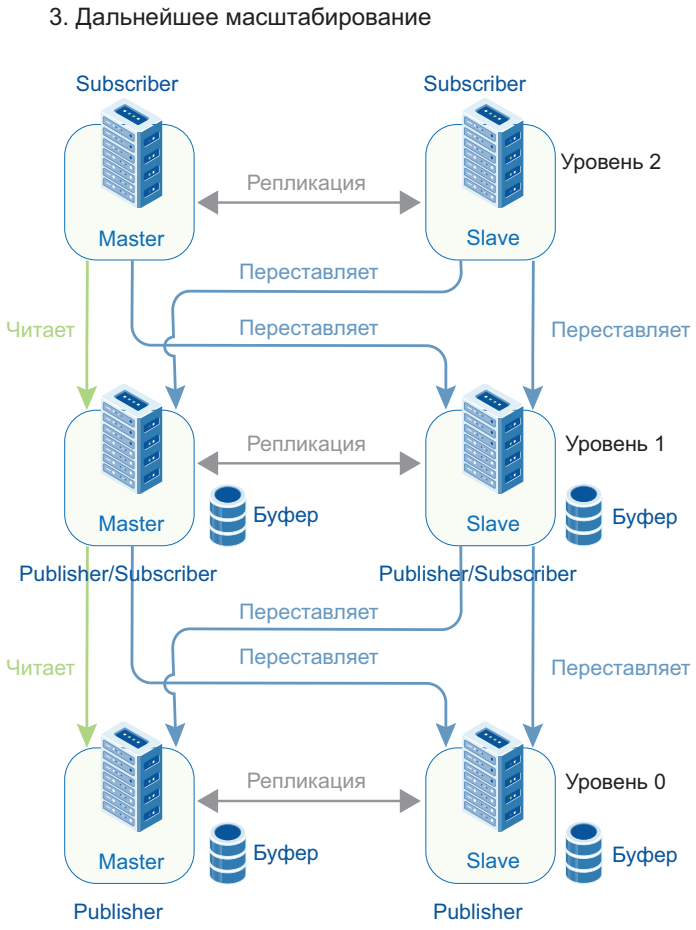


РИС. 9. Буферизация в многоуровневой схеме

Если мы зарезервируем в предыдущей схеме узел-источник, то сохранится аналогичный принцип буферизации, но для потребителя будет уже два равноценных поставщика буфера: Master (Publisher.0) и Slave (Publisher.1).

Алгоритм буферизации для рассматриваемой схемы (рис. 8):

1. Узел-потребитель Master (Subscriber.0) запрашивает и читает буфер одного из узлов-источников, например тоже Master (Publisher.0).
2. По мере того как Subscriber.0 вычитывает буферизованные данные на Publisher.0, он также переставляет свою позицию в буфере Publisher.1.
3. Вычитанные данные буфера путем репликации отправляются с узла-потребителя Master (Subscriber.0) на Slave (Subscriber.1).
4. Узел-потребитель Slave следует за своим Master и переставляет позицию на источниках Publisher.0 и Publisher.1.

Так позиции узлов-потребителей синхронизируются в буфере узлов-источников.

При дальнейшем масштабировании системы сохраняется аналогичная логика буферизации. Главное отличие многоуровневой схемы буферизации (представлена на рис. 9) заключается в том, что узлы уровня 1 будут одновременно потребителями буферизируемых данных уровня 0 и источниками буфера для уровня 2.

Все описанные схемы взаимодействия узлов-источников и узлов-потребителей позволяют исключить потерю данных при штатной работе и свести к нулю в случае сбоев в работе Master.

ОПЕРАТИВНАЯ ТАБЛИЦА СОБЫТИЙ

Механизм работы OAT

Следующим механизмом, реализованным на основе новой модели исполнения, стала оперативная таблица событий (Operative Alarm Table, OAT). Таблица показывает сгенерированные события, пока они остаются актуальными для клиента.

Как и в случае с буфером, файловое хранилище OAT имеет собственный внутренний бинарный формат. OAT хранится локально на диске и не требует установки дополнительного ПО.

Рассмотрим, какие изменения появились вместе с OAT, на примере активаций абстрактного датчика Sensor A, который срабатывает при достижении определенного порога значений (Hi_Level).

На рис. 10, иллюстрирующем принцип работы OAT, можно заметить, что за время наблюдения произошло три срабатывания, два из которых уже не активны. Без OAT клиент увидит только последнюю активацию, которая произошла в момент времени T_3, а предыдущие события не отобразятся, хотя они остались не квитированы. В случае перезапуска сервера исчезнут вообще все срабатывания: их все еще можно будет получить из истории активаций, но в наборе актуальных событий они не отобразятся.

OAT позволяет увидеть все актуальные события — активные или неквитированные. Таблица будет выводить срабатывания сенсора в моменты T_1, T_2 и T_3, пока пользователь не отреагирует на них.

Информация об активациях хранится на жестком диске, а значит, будет доступна и после перезапуска сервера. При этом в оперативной памяти удерживаются лишь небольшие индексные блоки, поэтому даже большое количество событий не приведет к существенному возрастанию потребления ОЗУ.

События, потерявшие актуальность, подлежат удалению из базы данных OAT. Чтобы удалить неактивные цепочки событий, достаточно принудительно их квитировать, а для предотвращения разрастания OAT можно настроить глубину хранения по давности или предельному количеству активаций.

Репликация OAT

В условиях репликации у каждого из резервируемых серверов есть своя копия OAT. На этапе инициализации Master передает OAT на Slave, и тот замещает собственную таблицу полученной. Так и достигается начальная идентичность OAT резервной пары.

Дальнейшее согласование OAT резервной пары, продемонстрированное на рис. 11, обеспечивается штатной репликацией журнала транзакций. Процессор таблицы читает журнал и на его основе формирует состояние OAT на Slave, которое будет идентично Master.

Таким образом, оперативная таблица событий позволяет видеть не только активные события, но и все, которые остаются актуальными, то есть активными или неквитированными.

ЗАКЛЮЧЕНИЕ

Мы рассказали о главных изменениях и нововведениях подсистемы исполнения Альфа платформы, которые пришли с глобальным обновлением в конце 2023 г. Ключевую роль в реализации всех описанных механизмов Alpha.Server сыграл журнал транзакций — единый высокопроизводительный тракт для обмена событиями как внутри одного узла, так и между резервируемыми узлами. На его основе мы осуществили переход к полноценной Master-Slave-репликации.

Новая модель исполнения задает однозначный порядок обработки данных на репликах и обеспечивает идентичность состояний резервной пары. Кроме того, Slave-сервер больше не дублирует вычисления, благодаря чему сокращается потребление ресурсов в условиях резервирования.

На базе журнала транзакций и Master-Slave-репликации мы смогли реализовать механизмы, отвечающие за сохранность и предоставление актуальных данных, — буферизацию и OAT. Буферизация гарантирует доставку всех событий, произошедших на источнике за время отсутствия связи с потребителем. OAT, в свою очередь, предоставляет оператору информацию обо всех срабатываниях — даже тех, которые уже не активны, но еще не были квитированы.

За последнее время расширились и функциональность Альфа платформы, и возможности ее применения. Программная платформа совместима с отечественными операционными системами и другими дистрибутивами семейства Linux и работает на них нативно, без вспомогательных технологий, таких как Wine, что упрощает ее использование в проектах автоматизации российских производств. Кроме того, Альфа платформа прошла проверку на соответствие требованиям ФСТЭК, предъявляемым к программному обеспечению для значимых объектов критической информационной инфраструктуры (КИИ), и широко применяется на объектах подобного рода. Но сегодня Альфа платформа — это не только конечный продукт, но и компонент производных продуктов, разрабатываемых партнерами. В июле 2024 г. был зарегистрирован первый программно-аппаратный комплекс (ПАК), созданный на базе Альфа платформы и контроллеров Инкомсистем, — «АБАК Платформа» (Реестр Минцифры, № 23143).

О расширении функциональности Альфа платформы и новых возможностях ее применения, описанных выше, мы расскажем в следующих статьях цикла. В публикациях будут подробно раскрыты вопросы интеграции со сторонними системами посредством API и SDK, принципы работы обновленной базы данных временных рядов Alpha.Historian, а также возможности разработки визуальной части проекта, предоставляемые Альфа платформой. ⬣

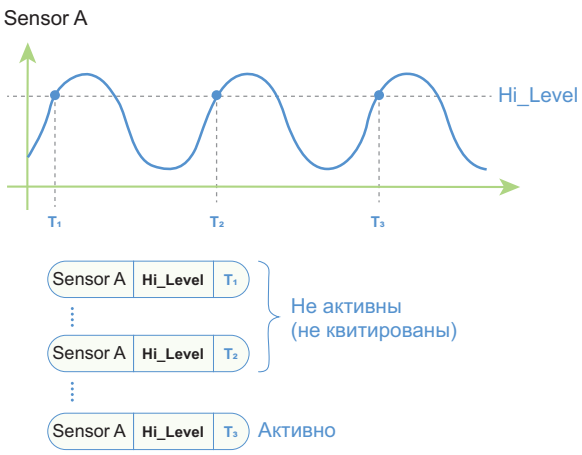


РИС. 10. ▲ Принцип работы оперативной таблицы событий (OAT)

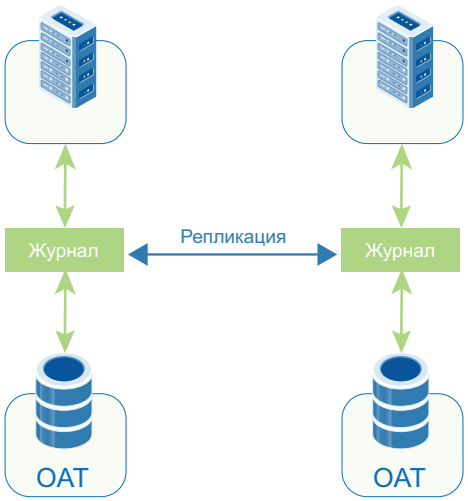


РИС. 11. ▼ OAT в условиях репликации

Описанные в статье нововведения, затронувшие среду исполнения Альфа платформы:

- сделали более прогнозируемыми как взаимодействие реплик, так и обработку потока данных в Alpha.Server;
- повысили устойчивость системы к нештатным ситуациям;
- позволили внедрить механизмы, гарантирующие доставку данных между отдельными экземплярами серверов и единую последовательность их обработки.

В дальнейших планах — развитие модели резервирования, в частности поддержка N-кратной репликации.