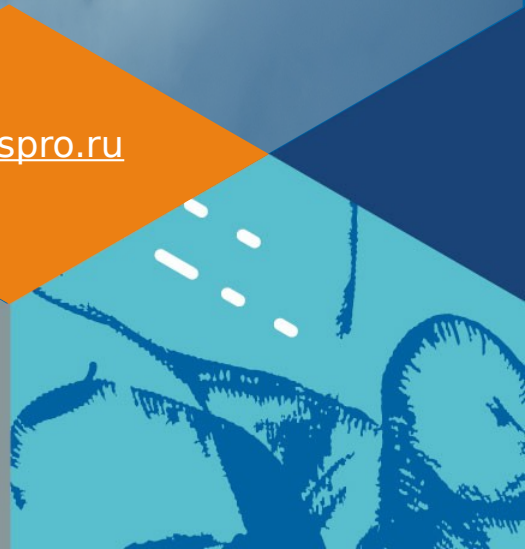
Надежная реализация сложной бизнес-логики с помощью pgpro_scheduler

Фролков Иван
Постгрес Профессиональный

postgrespro.ru



pgpro_scheduler

- Аналог crond
  - Выполнение задач по расписанию
- Одноразовые задания
  - Асинхронное выполнение
  - Зависимые задания
    - Задание не начнет выполняться, пока не выполнятся другие

Задания — зачем это надо

PROFESSIONAL Postgres

Как реализовать сложный и долгий бизнес-процесс?

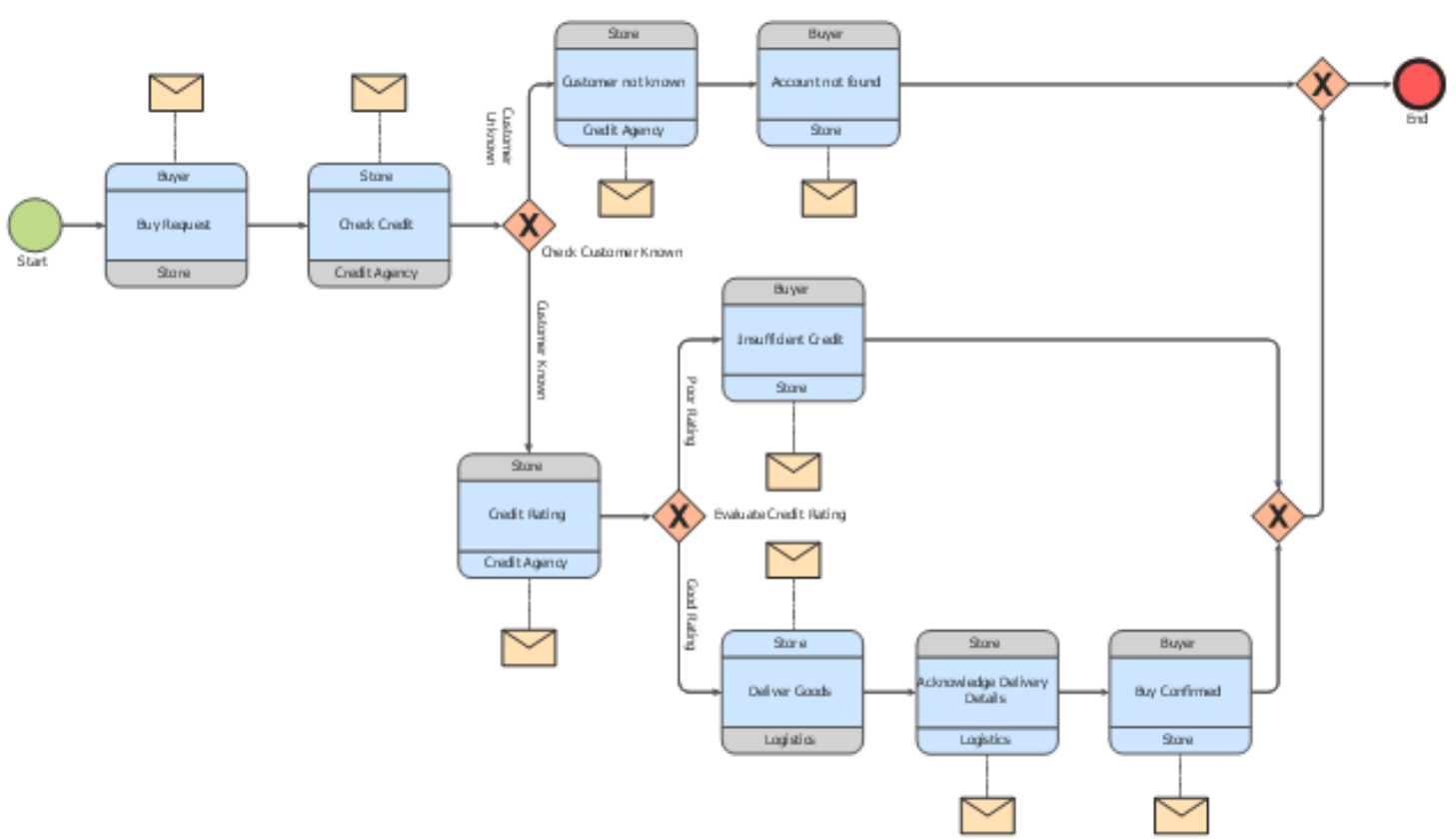
- Сначала создать документ
- Потом его зарегистрировать
- Потом его отправить
  - Если отправка не удалась, повторить
  - Если документ неактуален, то создать его снова
- После отправки убедиться, что документ обработан верно
- Уведомить о завершении обработки

Способы реализации

- BPMN 2.0
  - Camunda BPM
  - Activiti
  - jBPM
  - и др.
- Коммерческие
- Громоздкие
- BPMN 2.0 не всем нравится
- Капризные
- Требуется время на обучение

BPMN 2.0

- Пример диаграммы



pgpro_scheduler

Одноразовые задания

- Выполняются только один раз, вне расписания
- Можно установить время выполнения — задание будет выполнено не раньше указанного времени
- Задание может потребовать повторного выполнения
- Так как это СУБД, то задание выполняется строго в рамках транзакции
  - С точки зрения СУБД оно либо выполнилось, либо не выполнялось вообще
  - В случае ошибки от задания остается только сообщение об ошибке

pgpro_scheduler

- Это не BPMN 2.0
- Просто код
- Все вообще очень и очень просто:
  - `schedule.submit_job(query := 'select $1, $2', params := array['text 1', 'text 2'], depends_on := '{23, 15, 334}')`
  - `schedule.get_self_id()`
  - `schedule.resubmit(run_after interval default NULL)`
  - `select * from schedule.job_status`
  - `schedule.set_run_after(job_id, now() +make_interval(secs:=3600)`

Надежность.
Отправка транзакции в Ethereum

Почему?

- С виду простой случай, но есть подводные камни
- Есть публичные сети для разработки, можно прогнать в совершенно реальных условиях

Как?

- Создать и подписать транзакцию
- Отправить ее в блокчейн
- Дождаться появления ее там
- Отрапортовать об успехе

Предметная область - Ethereum

Известная криптовалюта

- По слухам, крипторубль будет на ее основе

Есть REST API, авторизация в общем случае не требуется

Account-based

- Есть адреса (хеш публичного ключа), у адреса есть баланс

Nonce (первый подводный камень)

- У каждого адреса есть счетчик отправленных им транзакций. Каждая следующая транзакция должна быть на единицу больше этого счетчика. Все транзакции попадают в блокчейн в порядке nonce, транзакция с nonce=N не попадет в блокчейн до тех пор, пока не будет обработана транзакция с nonce=N-1

Анатомия транзакции Ethereum

У транзакции есть

- **hash — уникальный идентификатор транзакции**
- Подпись
- И другое

TX POOL

TX POOL — второй подводный камень

- Майнеры берут транзакции из общей кучи (TX POOL/иногда его называют mempool)
- У транзакции есть комиссия — вознаграждение майнеру.
- В случае пиков транзакция с низкой комиссией часами-сутками не попадает в блокчейн и в конце концов выкидывается из TX POOL
- Транзакцию можно отправить снова с тем же nonce, но с другой комиссией

Форки

Третий подводный камень - два (или более) майнера нашли следующий блок

- У узла есть два (или более) корректных следующих блока
- Обработчик должен убедиться, что транзакция не только попала в блокчейн, но там еще и осталась
- Как? Подождать еще несколько блоков, и убедиться, что транзакция в блочейне

Ограничение

Надежность

- Подсистема отправки должна быть устойчива к любым сбоям
  - По питанию
  - Сбои сети
  - Сбои оборудования
  - Ошибки в коде (99% причин всех сбоев :-))
  - Аварийная остановка приложения
- Предполагается, что БД всегда может быть восстановлена до состояния последней зафиксированной транзакции — либо обычным восстановлением после аварийного окончания работы, либо из бекапа.

Это штатная возможность Postgres!

Как тут может помочь pgpro_scheduler?

Одноразовые задания!

- Они находятся в БД и спокойно переживут аварии

Как?

1. Главное задание каким-то способом получает nonce
2. И порождает дочернее с созданием, подписанием и отправкой транзакции
3. Главное задание ведет список отправленных им транзакций, и, если им долго не удается отправить транзакцию в блокчейн, порождает новое, с увеличенной комиссией, если же кому-то удалось - завершается
4. Дочернее задание проверяет статус отправки транзакуии
    1. В случае попадания транзакции в блокчейн дочернее задание ждет еще несколько блоков, и если транзакция продолжает оставаться в блокчейне, то считает задачу выполненной, сообщает об этом родительскому и завершается
    2. Если другому дочернему заданию удалось отправить — просто завершается

Что у нас получилось

PROFESSIONAL Postgres

Решение, устройчивое к сбоям:

- Состояние отправки находится в БД
- Все внешние изменения идемпотентнты
- Обрабатываются случаи
  - Отказа сети
  - Аварийного завершения работы
  - Недостатка средств

Масштабируемость. Хранилище электронных документов.

Постановка задачи

- Архив электронных документов
- Требуется хранение неограниченного объема документов неограниченное время
- Три уровня
  - Оперативный
  - Временное хранилище
  - Постоянное хранение
- Бизнес-задачи
  - Сохранение документа в Оперативный уровень
  - Создание описи документов и сохранение их во Временном или Постоянном уровне
  - Перенос описи документов из Временного уровня в постоянный
  - Переподписание документов
  - Запрос на получение документа/описи

Возможности для мастшабирования

Единицы взаимодействия

- Документ — просто zip определенной структуры
- Опись документов — просто список таких документов

Возможности для масштабирования

- Самый плохой случай — можно заплатить с любого счета на любой и баланс должен быть корректен всегда
  - **Его тут нет!**
- Документы не зависят друг от друга
- Следовательно, в пределе можно для каждого документа и для каждой операции с описью выделить свой сервер

Что можно сделать

Задачи

- Помещение документа на Оперативный уровень
  - Пара согласований, проверка подписи и просто передача файла
- Помещение описи на Оперативный/Временный/Постоянный уровни
  - Несколько согласований
  - Передача файлов из списка с проверкой подписи
- Переподписание
  - Совсем уж тривиально, просто перебираем файлы и переподписываем

Консенсус и супернадежность

Например, реализовать raft на plpgsql

(***italic*** — задания, **green** — таблицы, **blue** — вспомогательные функции)

- ***heartbeat***
  - Если **я — лидер**, то поставить задание для каждого **пира** «вызвать функцию **raft.heartbeat**», если такое еще не стоит
- ***elector***
  - Если **я - лидер**, то ничего не делаем, иначе
  - Проверяем наличие свежего **heartbeat**
  - Если нет — поставить задание для каждого **пира** «**vote_for_me**». По их окончанию проверить, выиграны ли выборы, и проставить **лидера**
  - перевыполнится с указанным временем (см. протокол)
- ***log_sync***
  - Если сменился лидер — **синхронизируем лог (см. форки в Ethereum(!))**
- **read_leader_log**
  - Читаем лог лидера и копируем его себе

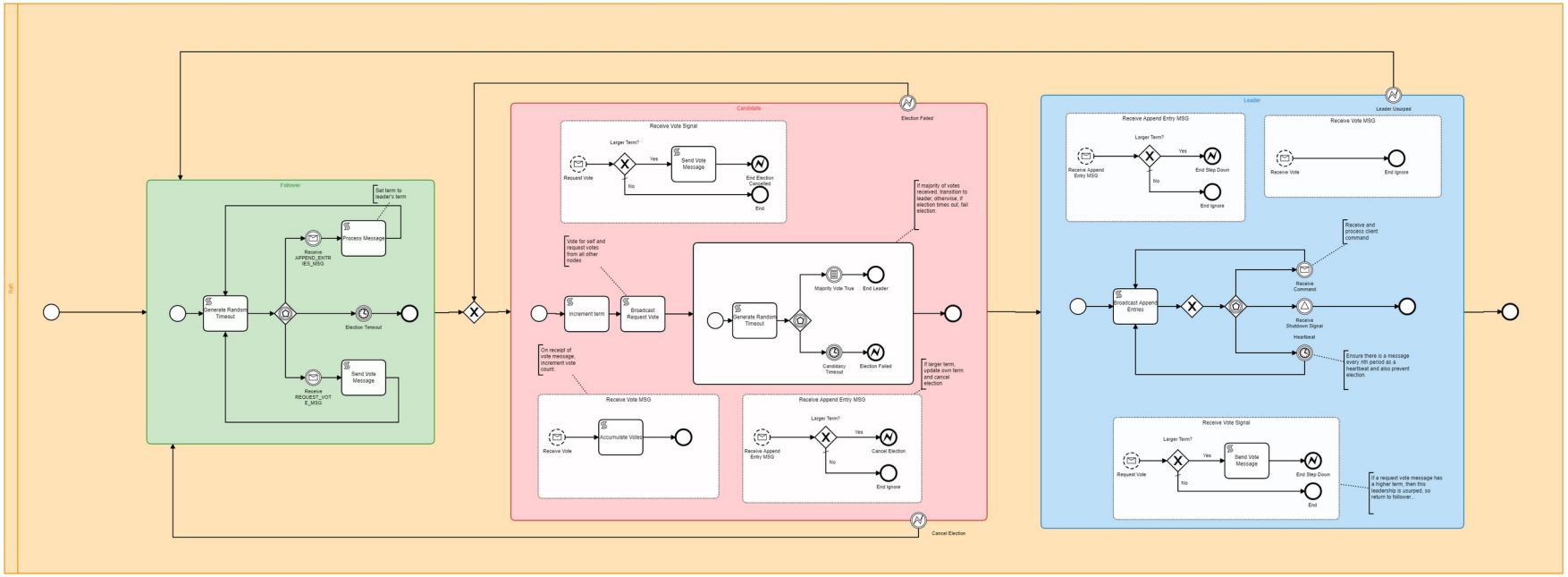
Супернадежность

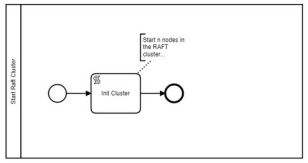
Все опять же очень просто:

- В каждое задание добавляем условие «если не **я — лидер**, то ничего не делаем»
- Задания для выполнения берем из лога raft с некоторым лагом (1-2 терма)
- Контексты берем оттуда же

Raft в BPMN 2.0

- https://forum.camunda.org/t/event-driven-state-machine-raft-demo/9078





Выводы

pgpro_scheduler, очевидно, не уникальное средство, но, но...

- Но он сильно, очень сильно проще BPMN и его реализаций
- Несмотря на простоту, может использоваться весьма замысловатая логика: так, BPMN работает по жесткой схеме и ее динамическое изменение не предусмотрено, тут же это реализуется интуитивно
- Совершенно не монструозен
- Низкий порог вхождения — вы, например, уже готовы его применять
- Нет отдельного сервиса — его не надо администрировать и настраивать
- Это относительно несложно сделать самостоятельно, но это уже есть и оно готово
- Все выполняется в БД, следовательно, есть гарантии целостности

Выводы (продолжение)

Простое использование pgpro_scheduler и нормально настроенный бекап уже дают очень высокую надежность

Для невзаимосвязанных OLTP-задач использование заданий позволяет достичь линейного масштабирования

Использование протоколов консенсуса может обеспечить катастрофоустойчиовсть

Всего-то пять функций

- Честно говоря, это не обязательно должен быть pgpro_scheduler

Вопросы?

Postgres Professional

http://postgrespro.ru/

+7 495 150 06 91

info@postgrespro.ru

postgrespro.ru