# Практический опыт использования решений репликации MySQL

Александр Чистяков
bOombate
http://alexclear.livejournal.com

# Докладчик?

- Разработчик серверных приложений
- Администратор баз данных
- Эксплуатационщик
- Архитектор серверных приложений
- Просто хороший человек

# Аудитория?

- Разработчики серверных приложений
- Администраторы баз данных
- Эксплуатационщики
- Архитекторы серверных приложений
- Просто хорошие люди

# Цель

- Традиционно СУБД является SPOF
- Время восстановления после сбоя СУБД может составлять несколько часов при отсутствии как минимум горячего резерва
- Перспектива несколько часов ничего не продавать очень не радует топ-менеджеров

# MySQL? А имеет ли смысл?

- Главный open source конкурент - PostgreSQL
- Надо как-то оценить статистику использования
- http://www.indeed.com/jobs?q=postgresql&l=CA
- http://www.indeed.com/jobs?q=mysql&l=CA
- 575 против 5728
- Кажется, у нас есть победитель
- Это была не самая корректная метрика, я в курсе

# Что мы хотим обеспечить?

- Несколько MySQL-серверов
- Несколько клиентов
- При отказе одного MySQL-сервера клиенты работают с другими
- Знакомая задача!
- Имеет несколько традиционных решений

# Платформа

- Хостинг среднего ценового диапазона
- Подключение к сети 100Мбит
- Машины в одном датацентре
- Крайне желательно, чтобы через WAN подключение тоже работало

# Что такое «репликация»?

- Процесс синхронизации нескольких копий данных
- Репликация возможна на нескольких уровнях:
  - Уровень блочного устройства
  - Уровень строк в таблице базы данных
  - Уровень SQL-запросов

# Виды репликации

- Синхронная (копии данных на нодах гарантированно одинаковые)
- Асинхронная (операция завершается раньше, чем о ней узнают все ноды)
- Какая лучше?
- А каковы метрики?

# Метрики

- Простота настройки
- Простота поддержки
- Быстродействие
- Простота восстановления после сбоя
- Скорость восстановления после сбоя
- Возможность автоматического восстановления
- Целостность данных

# На уровне блочного устройства

- MySQL + DRBD + Heartbeat
- Упомянуто в официальной документации
- DRBD – сетевой RAID1
- Может быть как sync, так и async
- DRBD может быть active-active
  - Но для СУБД это не подходит

# На уровне блочного устройства

- Минусы:
  - Для нашей платформы не очень подходит (очень медленно)
  - Одна из нод полностью простаивает
  - Heartbeat устарел, и его кодом никто не владеет
- Плюсы:
  - Донастройка MySQL не нужна

# Метрики - DRBD

- Простота настройки
- Простота поддержки
- Быстродействие
- Простота восстановления после сбоя
- Скорость восстановления после сбоя
- Возможность автоматического восстановления
- Целостность данных (sync/async ?)

# На уровне базы данных

- Встроенная в MySQL
- rubyrep
- Galera Cluster for MySQL
- Tungsten Replicator
- MMM
- PRM

# Встроенная в MySQL

- до версии 5.1 – только statement-based
- 5.1 и выше – row-based
- Плюсы:
  - Может работать между разными версиями сервера (между 5.0 и 5.5)
  - Очень проста в настройке

# Встроенная в MySQL

- Минусы
  - Информация о состоянии slave записывается в обычный файл – может потеряться (со мной такое было)
  - При использовании statement-based slave и master результаты запросов различаются – INSERT…. VALUES(NOW(),….)

# Встроенная в MySQL

- Можно настроить master-master и даже кольцевую репликацию
- auto_increment_increment, auto_increment_offset
- log-slave-update=TRUE
- Минус: split brain
- Выход: На разных узлах писать только в разные таблицы

# Split brain?

- Пусть в кластере есть два узла
- Или даже три
- Между узлами нарушается связность, при этом оба узла остаются в рабочем состоянии и обрабатывают запросы
- Рассинхронизация данных

# Метрики - встроенная

- Простота настройки
- Простота поддержки
- Быстродействие
- Простота восстановления после сбоя
- Скорость восстановления после сбоя
- Возможность автоматического восстановления
- Целостность данных

# rubyrep

- Trigger-based
- Ruby-based
- Из-за того, что основана на триггерах, изменения структуры базы требуют перезапуск репликации
- Несовместима с pt-online-schema-change
- Версии MySQL могут быть разными

# Метрики - rubyrep

- Простота настройки
- Простота поддержки
- Быстродействие
- Простота восстановления после сбоя
- Скорость восстановления после сбоя
- Возможность автоматического восстановления
- Целостность данных

# Взаимная совместимость

- Краткий ответ – лучше никогда не пытайтесь
- Однажды я настроил rubyrep между серверами со штатной master-slave репликацией
- Возникла петля

# Galera Cluster for MySQL

- Синхронная репликация
- Производитель заявляет active-active multi-master
- Поддержка MySQL 5.1, 5.5
- InnoDB only (MyISAM все равно не нужен)

# Galera Cluster - установка

- MySQL w/wsrep patch .deb/.rpm
- wsrep provider .deb/.rpm
- По умолчанию wsrep provider отключен
- Поменять установки в файле /etc/mysql/conf.d/wsrep.cnf

# Galera Cluster - настройка 1

- binlog_format=ROW
- default-storage-engine=InnoDB
- innodb_locks_unsafe_for_binlog=1
- Отключить query_cache
- innodb_autoinc_lock_mode=2

# Galera Cluster - настройка 2

- wsrep_cluster_name
- wsrep_provider
- wsrep_cluster_address – можно устанавливать динамически в случае, если state transfer method не rsync
- wsrep_retry_autocommit=1
- wsrep_certify_non_PK=1

# Galera Cluster - передача состояния

- mysqldump – обычный обмен дампом (очень медленно)
- rsync – передача самих файлов DB, гораздо быстрее
- В момент передачи состояния от ноды к ноде кластер недоступен

# Galera Cluster - производительность

- Один и тот же дамп базы ~3 Gb
- Два узла MySQL, один арбитратор
- 100Мб сеть, сервера в одном ДЦ
- 60 мин – заливка дампа в кластер
- 7 мин – заливка дампа на отдельно стоящий сервер

# Galera Cluster - балансировка

- Блокировка на уровне строк – можно попробовать использовать балансировщик уровня TCP
- Мы использовали HAProxy
- Python-based скрипт проверки состояния MySQL

# Galera Cluster - split brain

- Если узла только два, при нарушении связности оба перестанут обрабатывать запросы
- Поэтому узла должно быть три (или любое нечетное число)
- Арбитратор – приложение, которое участвует в обмене данными репликации, но не пишет на диск

# Galera Cluster - проблемы

- При конкурентных вставках в одну и ту же таблицу возможен deadlock (и у нас он возникал постоянно)
- Варианты:
  - Переписать бизнес-логику
  - Поменять балансировщик
- (кстати, Python скрипт с предыдущего слайда зависал)

# Балансировщик уровня приложения - yybal

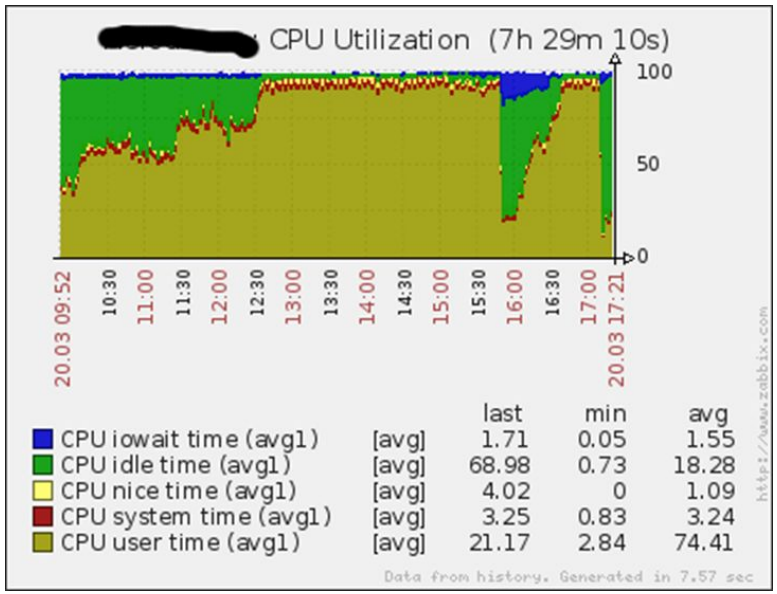
- Написан на python/greenlets
- Не готов для публичного использования
- Перенаправляет запросы с учетом их смысла (все запросы на изменение данных идут на один и тот же узел)

# Galera Cluster - проблемы 2

- ID у суррогатных ключей при вставке перескакивал на несколько тысяч
- Разработчик Galera сообщил, что проблема в самом MySQL
- Так как от конкурентной вставки уже отказались, отключили смещение в конфигурационном файле

# Galera Cluster - проблемы 3

- Внезапное и необъяснимое увеличение потребления CPU
- Разбираться было некогда



# Метрики - Galera

- Простота настройки
- Простота поддержки
- Быстродействие
- Простота восстановления после сбоя
- Скорость восстановления после сбоя
- Возможность автоматического восстановления
- Целостность данных

# MMM

- MMM – Multi-Master Replication Manager
- http://www.google.ru/#q=MMM+MySQL+problems
- http://www.xaprb.com/blog/2011/05/04/whats-wrong-with-mmm/

# Percona Replication Manager

- Позиционируется как замена MMM
- Основан на Pacemaker
- Pacemaker предоставляет надежный коммуникационный канал и занимается арбитражем
- Pacemaker оперирует виртуальными IP-адресами

# PRM - настройка

- Как быть с виртуальными IP-адресами, если машины в разных подсетях?
- IPsec туннель, поверх него – GRE туннель с разрешенным multicast
- Quagga с включенным OSPF
- Виртуальные адреса на алиасе локального интерфейса (lo)

# PRM – split brain

- PRM очень консервативен и делает выбор нового мастера только один раз
- Логика переключения IP ложится на Pacemaker
- Документация pacemaker
- Welcome to Vietnam! (ключевые слово: STONITH device)

# Метрики - PRM

- Простота настройки
- Простота поддержки
- Быстродействие
- Простота восстановления после сбоя
- Скорость восстановления после сбоя
- Возможность автоматического восстановления
- Целостность данных (semisync?)

# Планы на будущее

- Drizzle – очень большое внимание уделено репликации:
  - формат Google protobuf
  - replication log – таблица InnoDB
  - Один slave для нескольких master
  - Replication state записывается транзакционно
- Semisync plugins для MySQL 5.5

# Выводы

- Репликация MySQL требует жертв
- Универсального решения не существует
- Galera Cluster – неплохое решение при не очень большой нагрузке (Alexa rank in RU < 500)
- Следите за сообществом

# Вопросы?

- 
- 
- 
- 
- 
- 

# Спасибо за внимание!

- С вами был Александр Чистяков
- http://alexclear.livejournal.com
- alexclear@gmail.com
- http://github.com/alexclear