Живая синхронизация данных с помощью clsync

Андрей Савченко Дмитрий Окунев

НИЯУ МИФИ, Москва, Россия

Basealt-2016
1 октября 2016 г.

Постановка задачи

Что нужно:

- Организовать HA/LB кластеры — системы высокой доступности и/или с распределением нагрузки:
  - хостинг
  - ip-телефония
  - корпоративные сервисы
- Управление и синхронизация HPC-систем
- Резервное копирование всего этого

Ограниченные ресурсы

Что есть:

- Ограниченные аппаратные ресурсы:
  - Около 10 блейдов
  - 1 Gb/s интерконнект (и нередко падает)
  - нехватка памяти и дисков
- Более 100 задач (контейнеров):
  - сайт ВУЗа и разных подразделений
  - сайты кафедр
  - ip-телефония
  - vpn для общежитий
  - внутренние сервисы (почта, ntp, sks,...)

Способы решения

Что делать?

- Уход с виртуальных машин в контейнеры (LXC)
- Дедупликация дисковых образов:
  - базовый образ + aufs/overlayfs
- Дешёвый способ организация HA/LB кластеров

Файловая синхронизация

Основные проблемы:

- производительность (минимизация издержек)
- высокая доступность (отказ сервиса не дольше нескольких секунд)
- надёжность (минимизация отказов)
- универсальность (широкий спектр решаемых задач)
- необходимость агрегации накопленных изменений

Файловая синхронизация

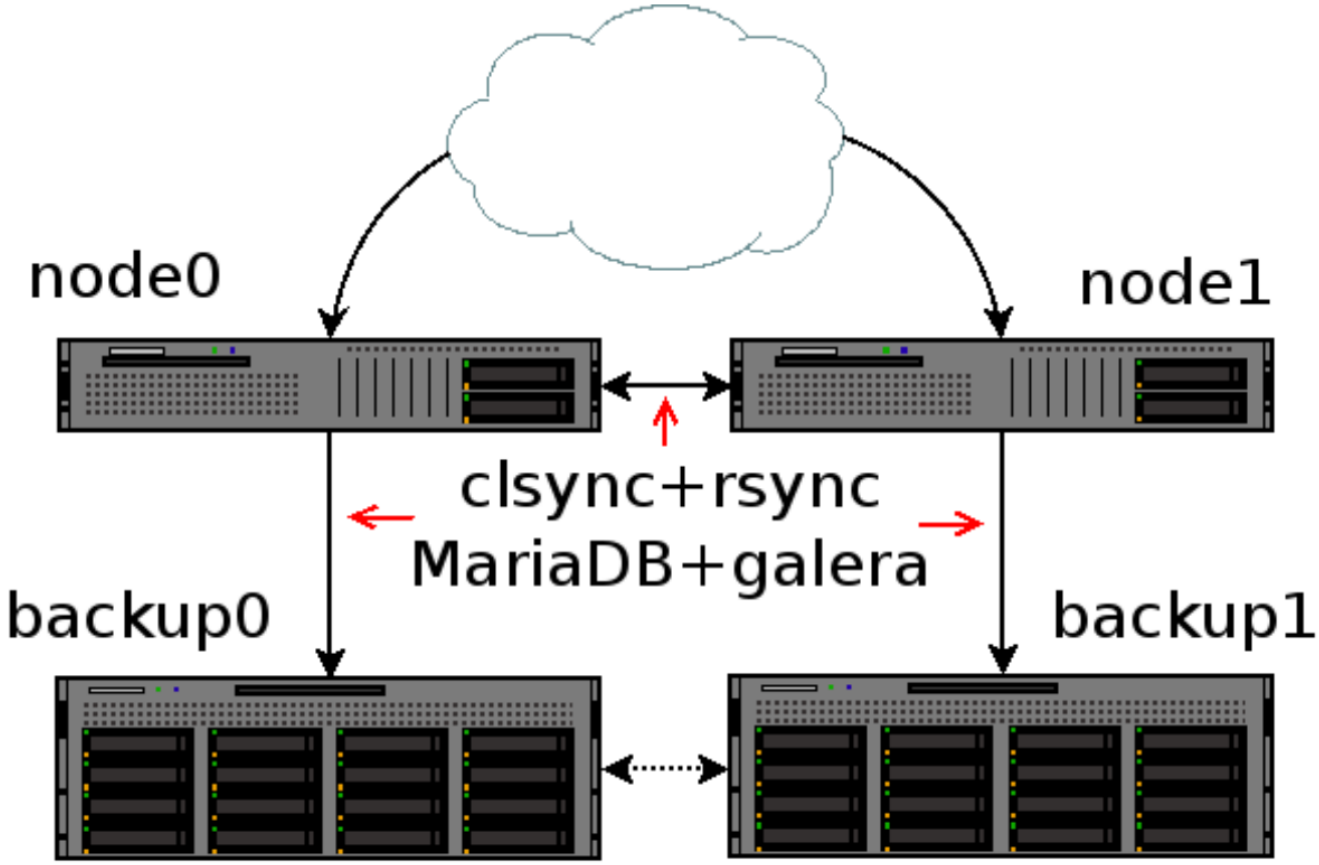
Способы решения:

- файловые системы только для чтения
  - малая область применимости
- блочная репликация
  - очень низкая скорость работы (DRBD + OCFS2)
  - неустойчивость к разрывам линков (split brain)
- единые сетевые файловые системы (CEPH)
  - низкая скорость работы (latency)
  - высокая сложность, неустойчивость (2012 год)
- файловая репликация

Выбор ПО

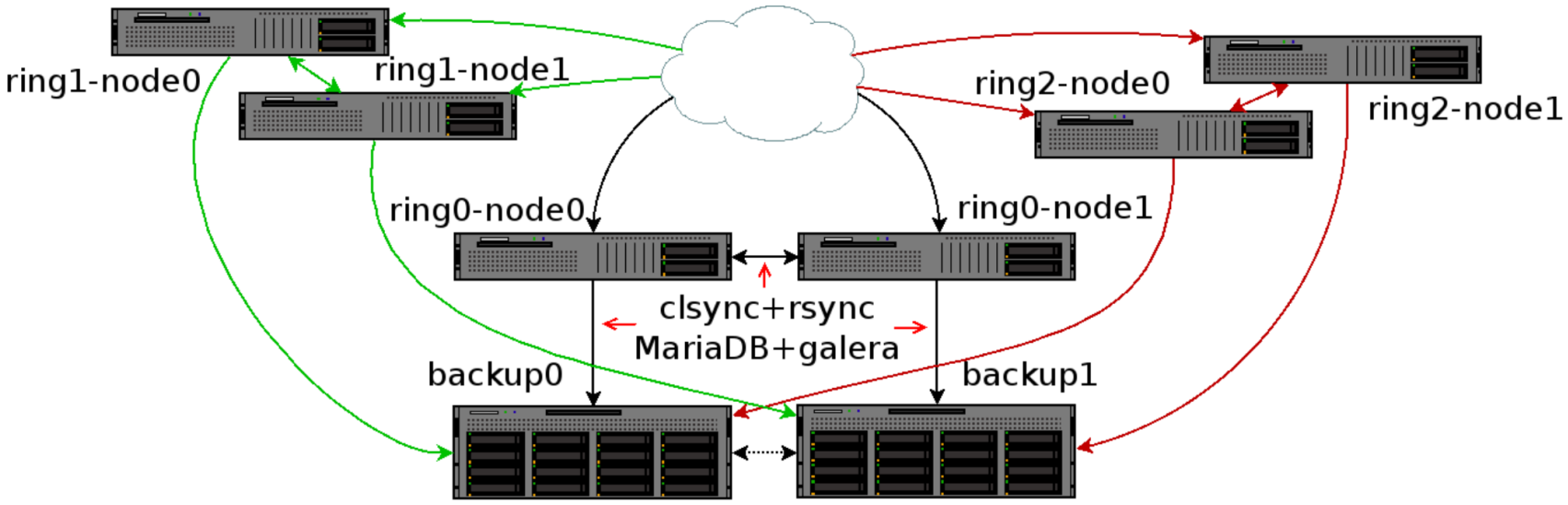
- lsyncd: лучше всего подходит для задачи, но
  - сильная загрузка CPU ($> ½$ кода на LUA)
  - баги и сложность исправления LUA кода
  - нет тредов, нужно для больших директорий ($> 10^6$ объектов)
  - нет агрегации событий
  - нет поддержки so
  - нет поддержки BSD
- incron
  - нет рекурсии, событийности
- csync2
  - нет событийности, очень медленный
- librnotify
  - не существовало :)
  - нет обвязки

Что получилось



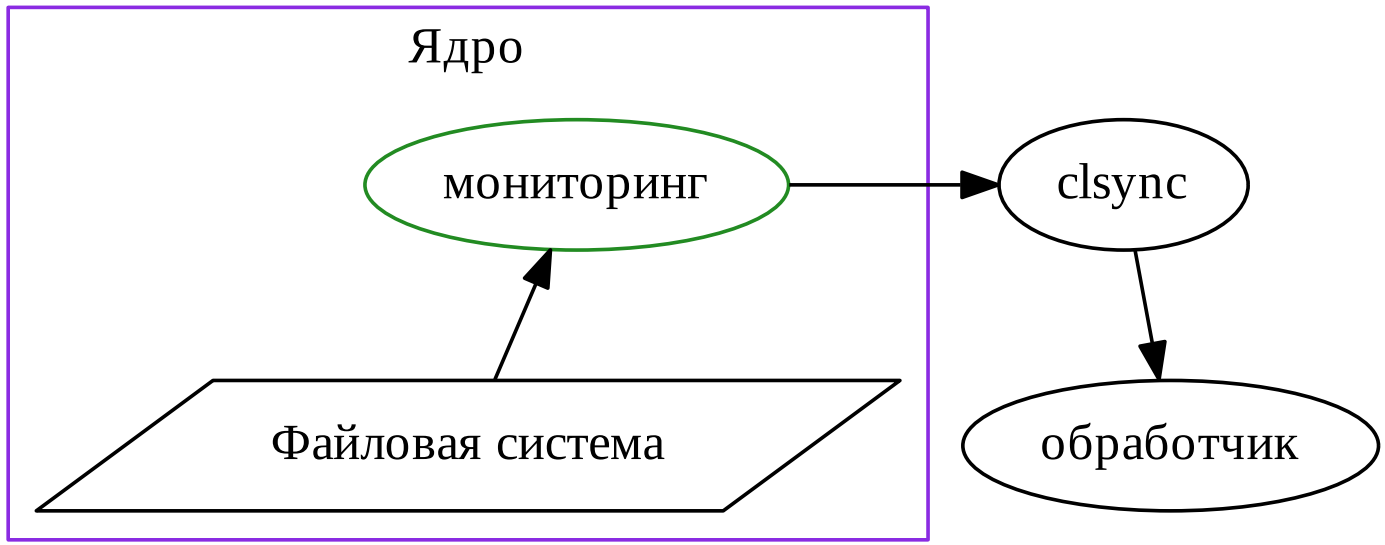
- LXC
- Ext4 c clsync + rsync
- MariaDB + Galera

Что получилось



- Разделение на кольца по значимости и доверяемости
- HA и бэкапы для каждого кольца

Принцип работы clsync



Обработчиком может быть что угодно, но чаще всего используется rsync.

Выбор системы мониторинга

Linux:

- dnotify
  - \+ можно отследить любое событие
  - \- нельзя следить за отдельными файлами
  - \- блокируется umount
  - \- уведомление через сигналы
  - \- нужно делать много stat(), т.к. есть только fd
- inotify [выбор]
  - \+ уведомление через epoll
  - \+ есть вся нужная информация
  - \- нет рекурсии
  - \- нужно хранить соответствие watch $\Leftrightarrow$ путь

Выбор системы мониторинга

Linux:

- fanotify
  - \+ есть рекурсия
  - \+ получается fd и pid *Rightarrow* знаем путь
  - \- нет отслеживания удаления, перемещения, переименования файлов $;-($

FreeBSD:

- kqueue/kevent
- libinotify (поверх kqueue)
- BSM API (не по назначению)
- dtrace (непригодно, т.к. путь неизвлекаем)

Достойный аналогов Linux inotify нет, из имеющегося лучше всего работает kqueue.

Как работает clsync

При настройках по-умолчанию:

1. Инициализация
2. Маркировка в подсистеме inotify
3. Синхронизация всего файлового дерева
4. Досинхронизация новых событий
5. Ожидание, агрегация событий и ⇑

Обработчики

Режимы synchandler (обработчиков):

- *simple* — вызов приложения на каждое событие
- *shell* — вызов приложения со списком изменений на каждую синхронизацию
- *rsyncdirect* — прямой вызов rsync
- *rsyncshell* — вызов обработчика для дальнейшей работы с rsync [рекомендуется]
- *rsyncso* — подгрузка библиотеки и передача списка rsync через обратный вызов clsyncapi_rsync()
- *so* — подгрузка библиотеки и передача простого списка через обратный вызов clsyncapi_sync()

Безопасность

Наблюдение может быть за всем деревом с правами root, поэтому опционально используются:

- сброс привелегий
- использование capabilities
- изоляция по namespace
- разделение на привелигированный и обычный thread
- fork() на привелигированный и обычный процесс
- seccomp изоляция
- использование cgroups

Что ещё умеем?

- выделение тредов на каждый процесс синхронизации
- правила поддерживают регулярные выражения и выбор тип объектов файловой системы
- управление через UNIX-сокет
- умное переключение между spin_lock и mutex
- и много другого!

Статус

- Основные функции реализованы
- Исправление ошибок
- Доработка мелочей
- Последний резил $0.4.2$ вчера $;-)$

Поддержка в дистрибутивах:

Gentoo	официальный репозиторий
Debian	официальный репозиторий
RH-based	предоставляется RPM spec файл
FreeBSD	есть порты

Итоги

Доступен гибкий, безопасный и эффективный инструмент живой синхронизации данных: clsync.

Контакты:

Github	https://github.com/xaionaro/clsync
IRC	Freenode: #clsync
e-mail	dyokunev@ut.mephi.ru
	aasavchenko@ut.mephi.ru

Спасибо за внимание!

Примеры использования

Зеркалирование директории:

```
clsync -Mrsyncdirect -W/path/to/source_dir \
-D/path/to/destination_dir
```

Разовая синхронизация:

```
clsync --exit-on-no-events --max-iterations=20 \
--mode=rsyncdirect -W/var/www_new -Srsync -- \
%RSYNC-ARGS% /var/www_new/ /var/www/
```

Примеры использования

Коррекция прав доступа файлов web-сайта:

```
clsync -w3 -t3 -T3 -x1 \
-W/var/www/site.example.org/root \
-Mdirect -Schown --uid 0 --gid 0 -Ysyslog -b1 \
--modification-signature uid,gid -- \
--from=root www-data:www-data %INCLUDE-LIST%
```