Clsync — инструмент живой синхронизации данных

Андрей Савченко[1] Дмитрий Окунев[2]

[1]ООО «Базальт СПО»

[2]НИЯУ МИФИ

17 Мая 2018

Задачи

Что нужно:

- Организовать HA/LB кластеры:
  - хостинг
  - ip-телефония
  - корпоративные сервисы
- Управление и синхронизация HPC-систем
- Резервное копирование всего этого

Ограниченные ресурсы

Что имелось:

- Ограниченные аппаратные ресурсы:
  - Около 10 блейдов
  - 1 Gb/s интерконнект (и нередко падает)
  - нехватка памяти и дисков
- Сотни 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 организации HA/LB кластеров

Требования к синхронизации

- Высокая производительность (минимизация издержек)
- Высокая доступность (отказ сервиса не дольше нескольких секунд)
- Надёжность (минимизация отказов)
- Универсальность (широкий спектр решаемых задач)
- Необходимость агрегации накопленных изменений

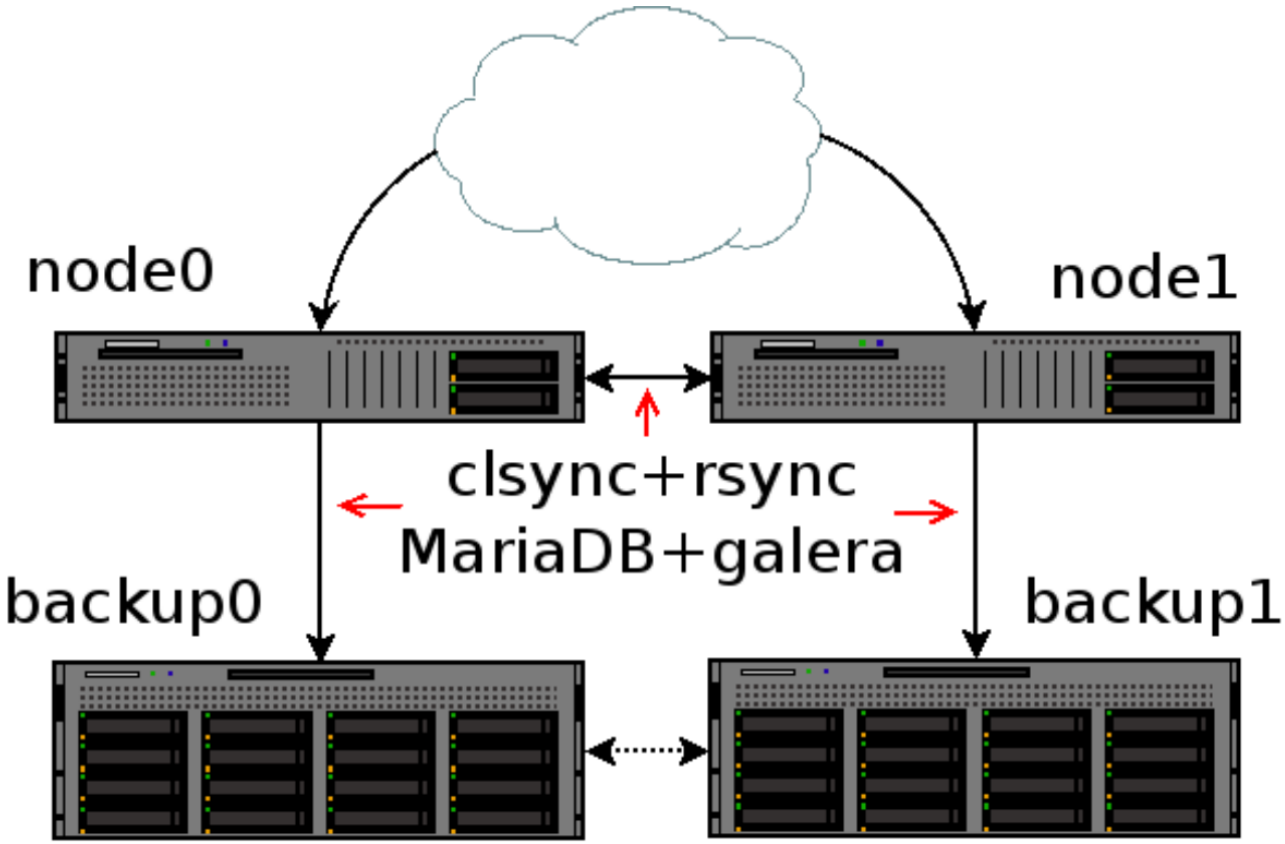
Возможные подходы

- 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)
  - 3x требования к хранилищу
- Файловая репликация

Выбор решения

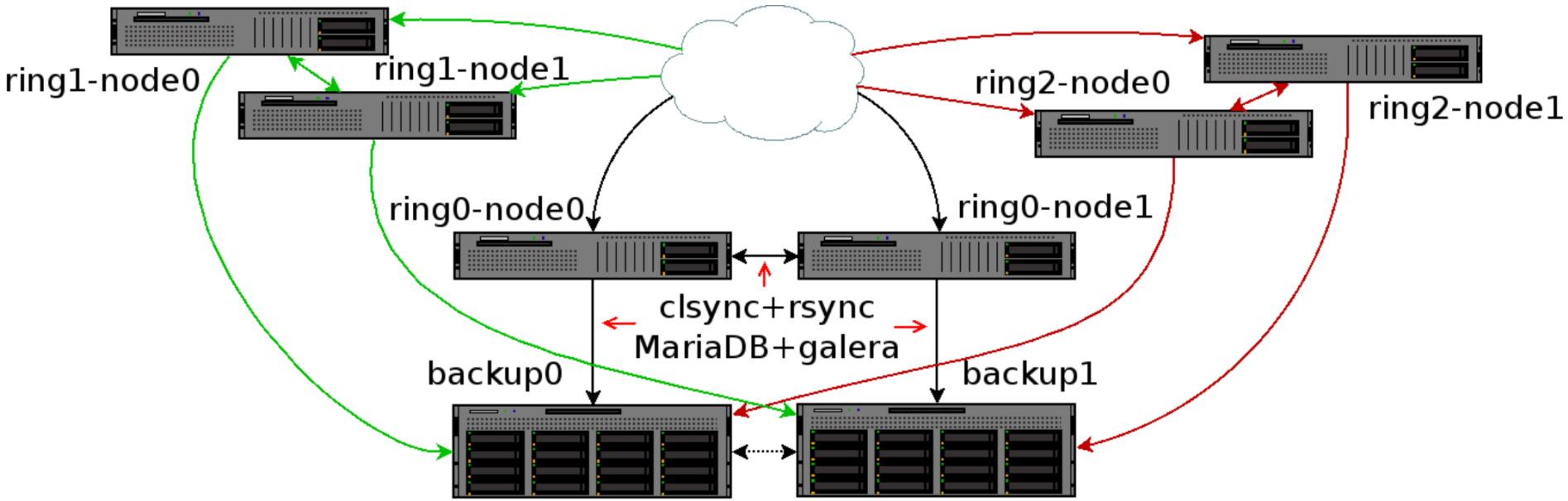
- lsyncd: лучше всего подходит для задачи, но
  - сильная загрузка CPU ($> \frac{2}{3}$ кода на LUA)
  - баги и сложность исправления LUA кода
  - нет тредов, нужно для больших директорий ($> 10^6$ объектов)
  - нет агрегации событий
  - нет поддержки so
  - нет поддержки BSD
- incron
  - нет рекурсии, событийности
- csync2
  - нет событийности, очень медленный
- librnotify
  - не существовало :)
  - нет обвязки (голый inotify интерфейс)

Что получилось



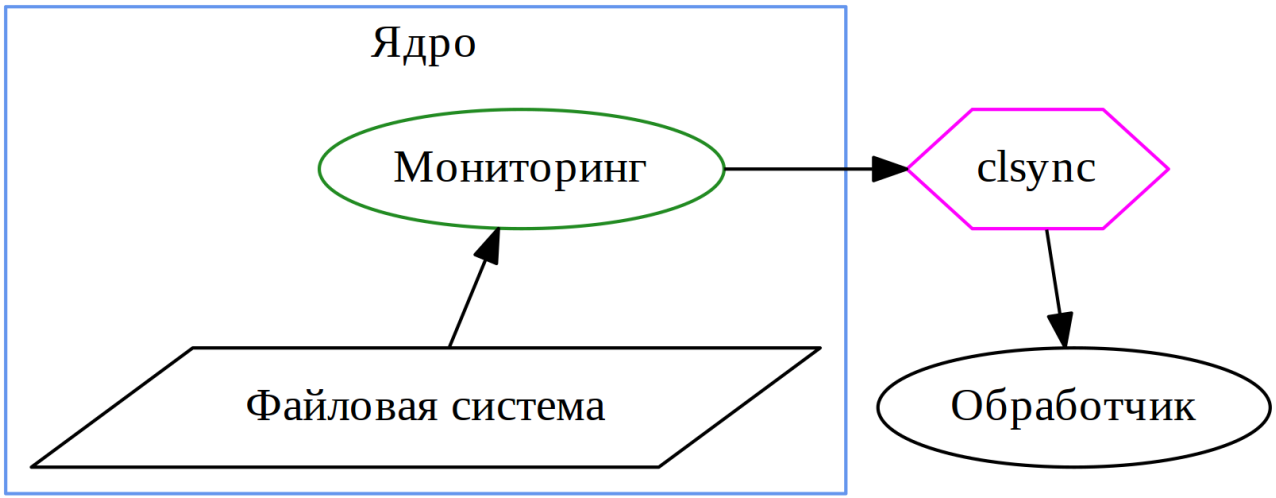
- LXC
- Ext4 с clsync + rsync
- Percona

Инфраструктура



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

Как работает 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)

В FreeBSD нет достойной замены Linux inotify.
Наиболее пригоден к эксплуатации 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 (synchandler)

- *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* — загрузка библиотеки и передача списка rsync через обратный вызов clsyncapi_rsync()
- *so* — загрузка библиотеки и передача простого списка через обратный вызов clsyncapi_sync()

Безопасность

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

- сброс привилегий
- использование capabilities
- изоляция по namespace
- изоляция с использованием cgroups
- разделение на привилегированный и обычный тред
- seccomp изоляция
  - mprotect() недоступен $\Rightarrow$ невозможно разделение на треды
- разделение на привилегированный и обычный процесс

Сброс привилегий

setgid, setuid:

- нужно сохранить CAP_DAC_READ_SEARCH для clsync
- обработчику нужны права чтения

Проблема: возможно чтение всех данных раздела

Пространства имён:

- unshare

Проблема: возможно чтение всех данных раздела

Решение: разделение тредов

Сброс привилегий

setgid, setuid:

- нужно сохранить CAP_DAC_READ_SEARCH для clsync
- обработчику нужны права чтения

Проблема: возможно чтение всех данных раздела

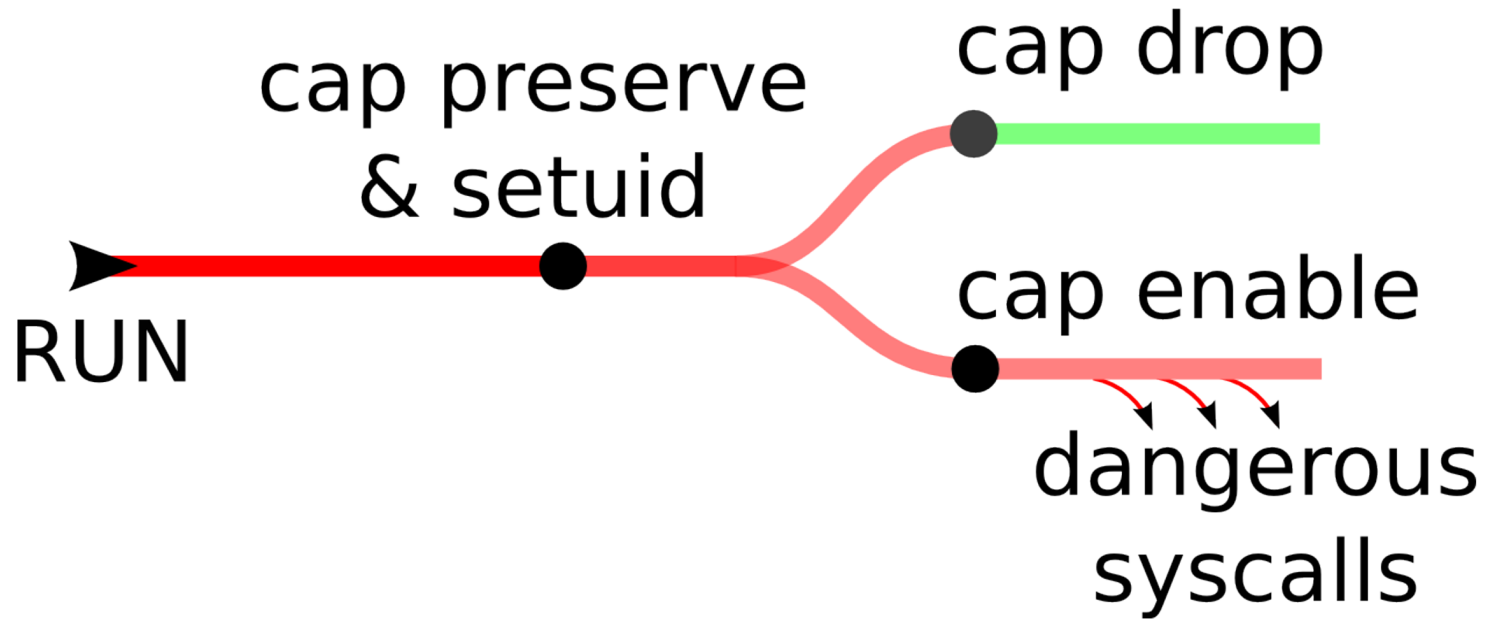
Пространства имён:

- unshare

Проблема: возможно чтение всех данных раздела

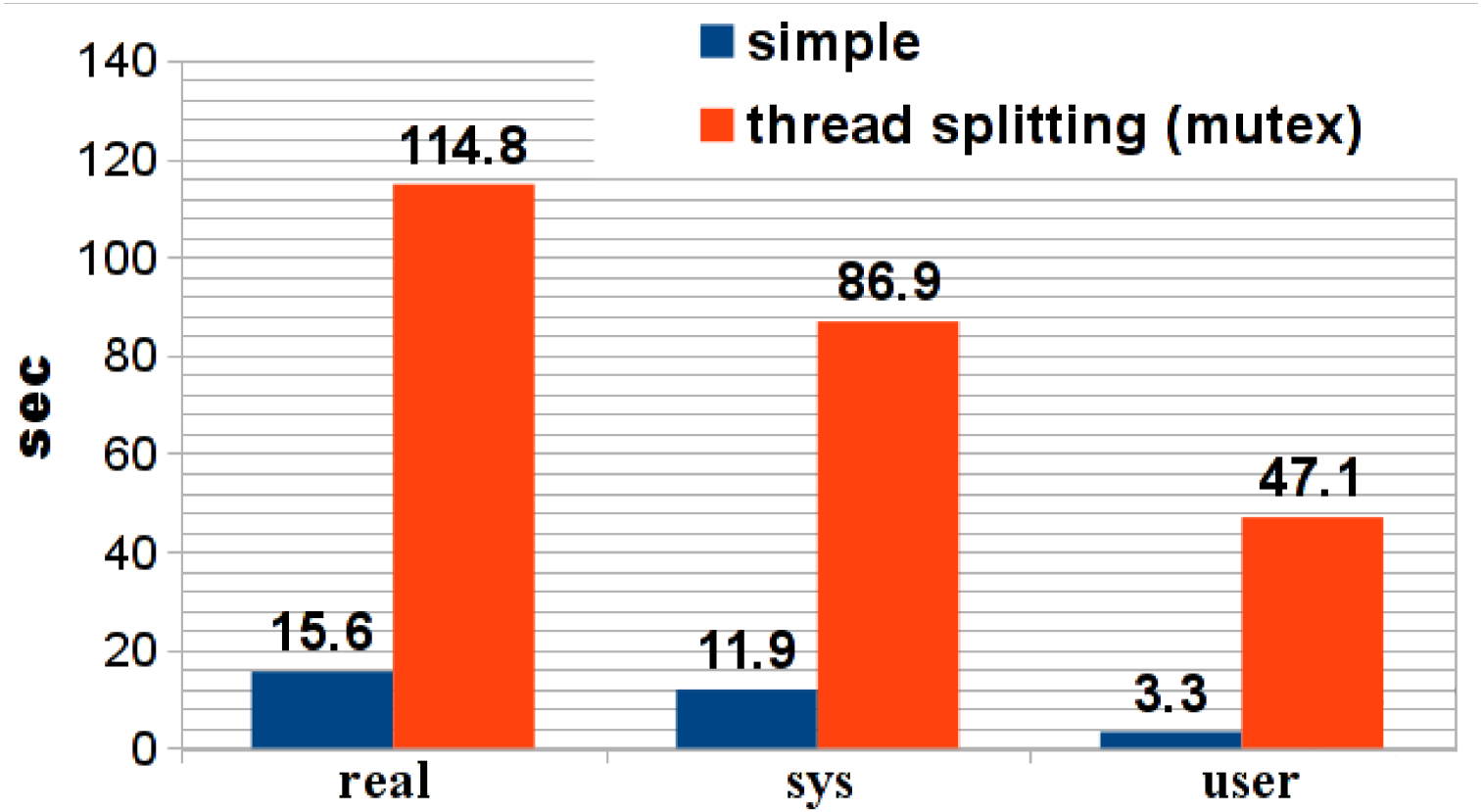
Решение: разделение тредов

Разделение тредов



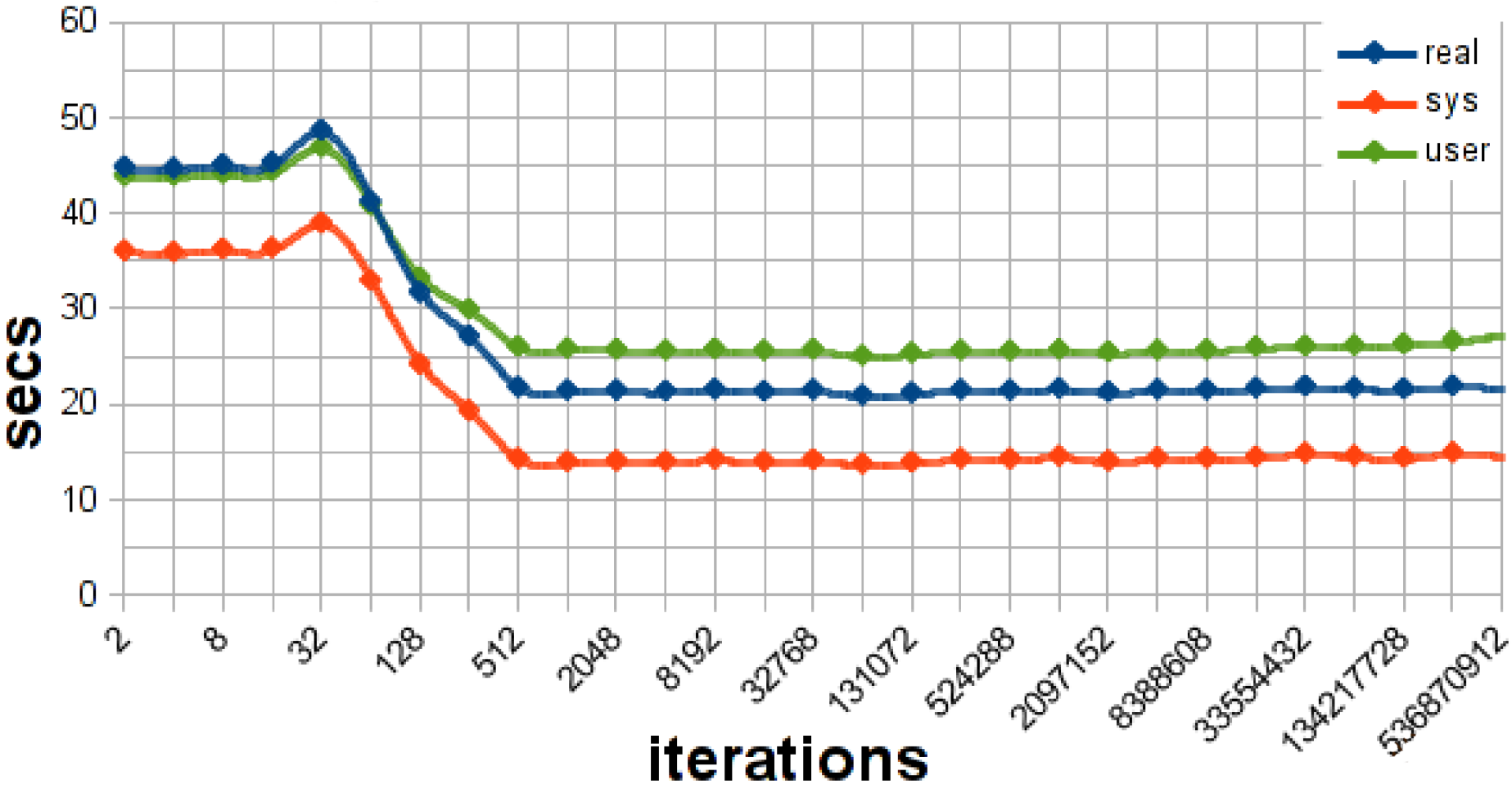
Создание дополнительного потока для обработки системных вызовов, требующих повышенных привилегий.

Производительность тредов



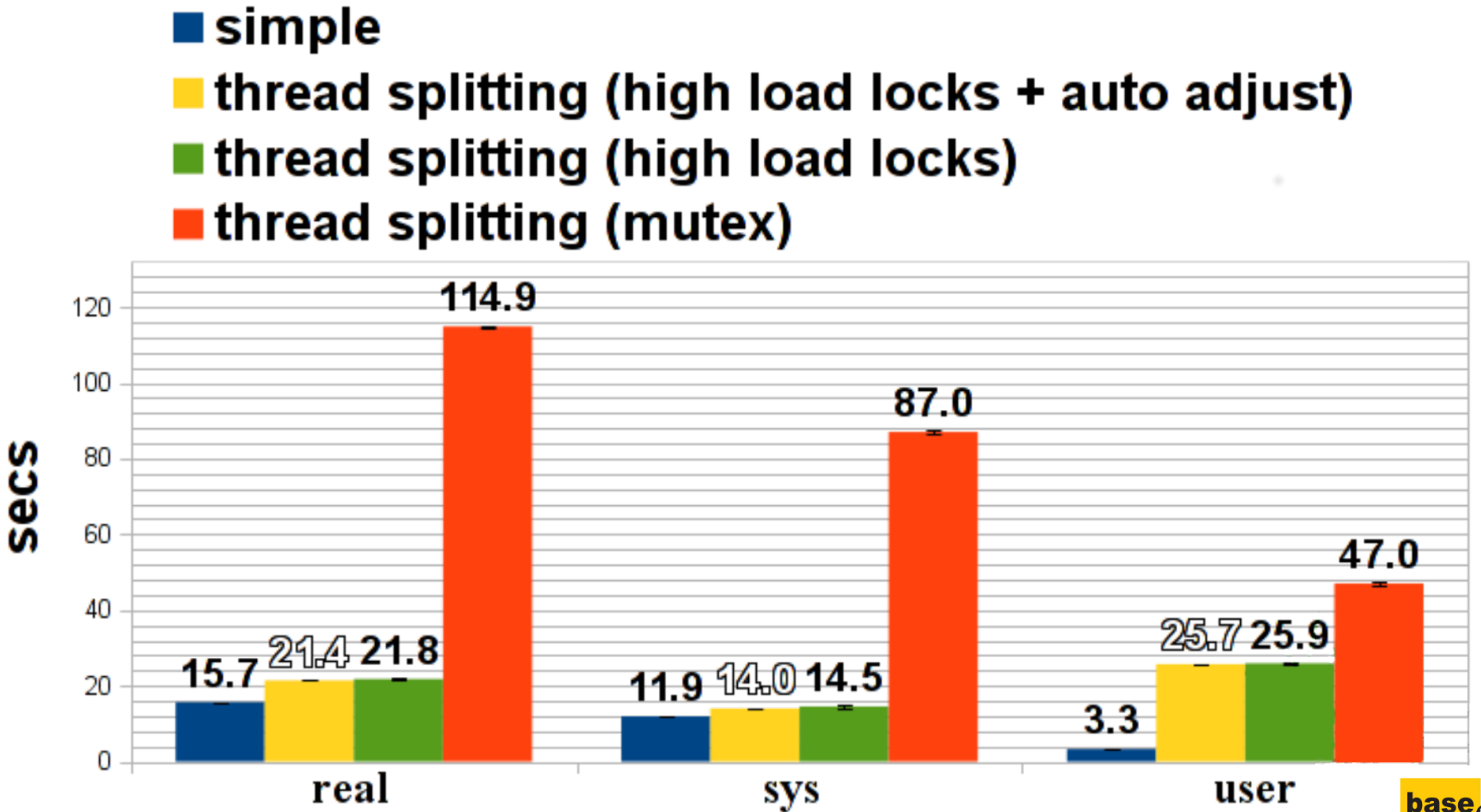
pthread mutex не предназначен для высокоскоростных блокировок

Переключение spinlock → mutex

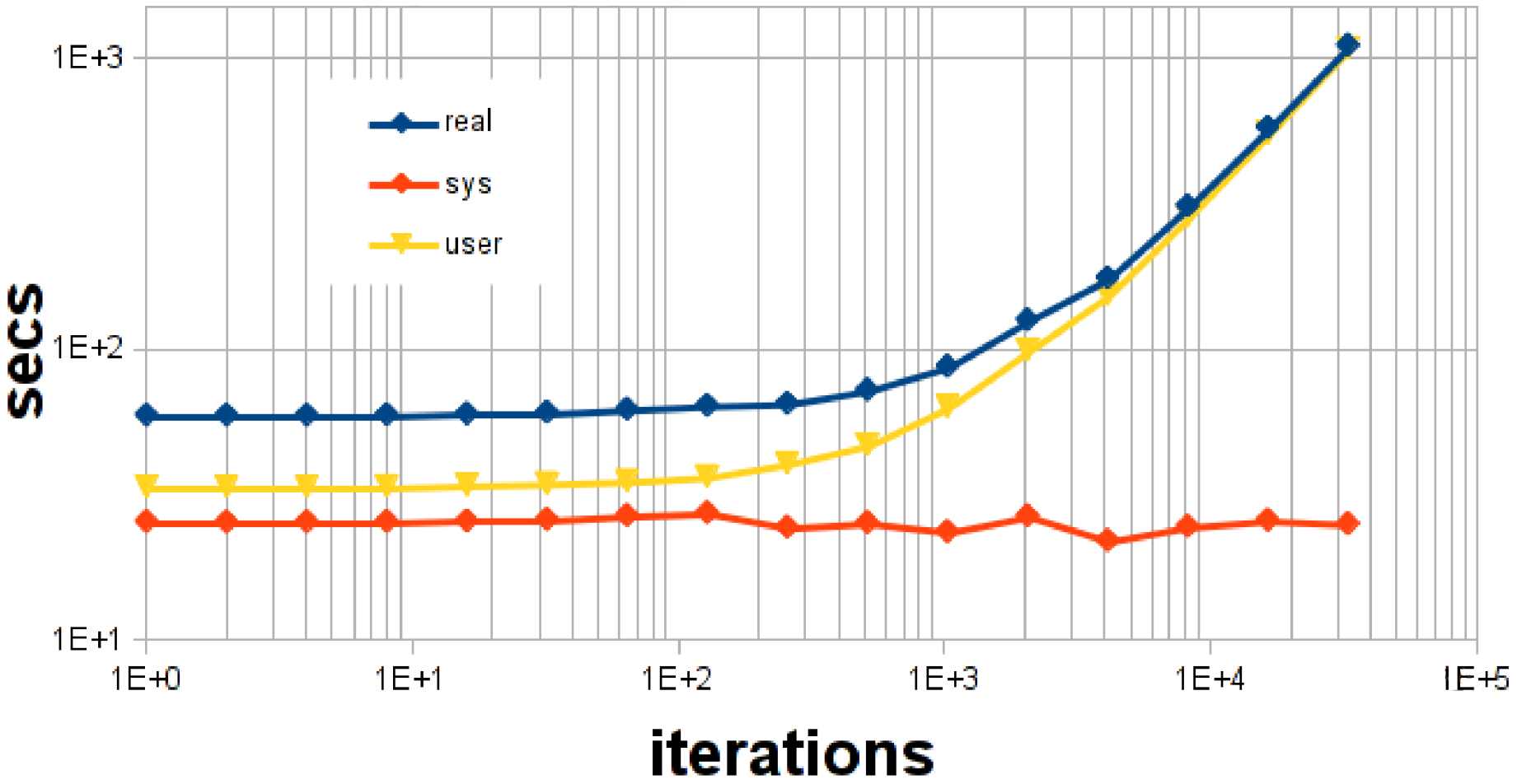


протестировано на первичной синхронизации

Умные блокировки



Spinlock на одном ядре



Безопасность

seccomp:

+ белый список системных вызовов
- запрет mprotect $\rightarrow$ невозможен pthread_create()

cgroups:

- ограничение доступа к /dev

fork():

+ отдельный процесс для (setuid, setgid, exec)
- медленнее (shm)
- более сложный код

Безопасность

seccomp:

+ белый список системных вызовов
- запрет mprotect $\rightarrow$ невозможен pthread_create()

cgroups:

- ограничение доступа к /dev

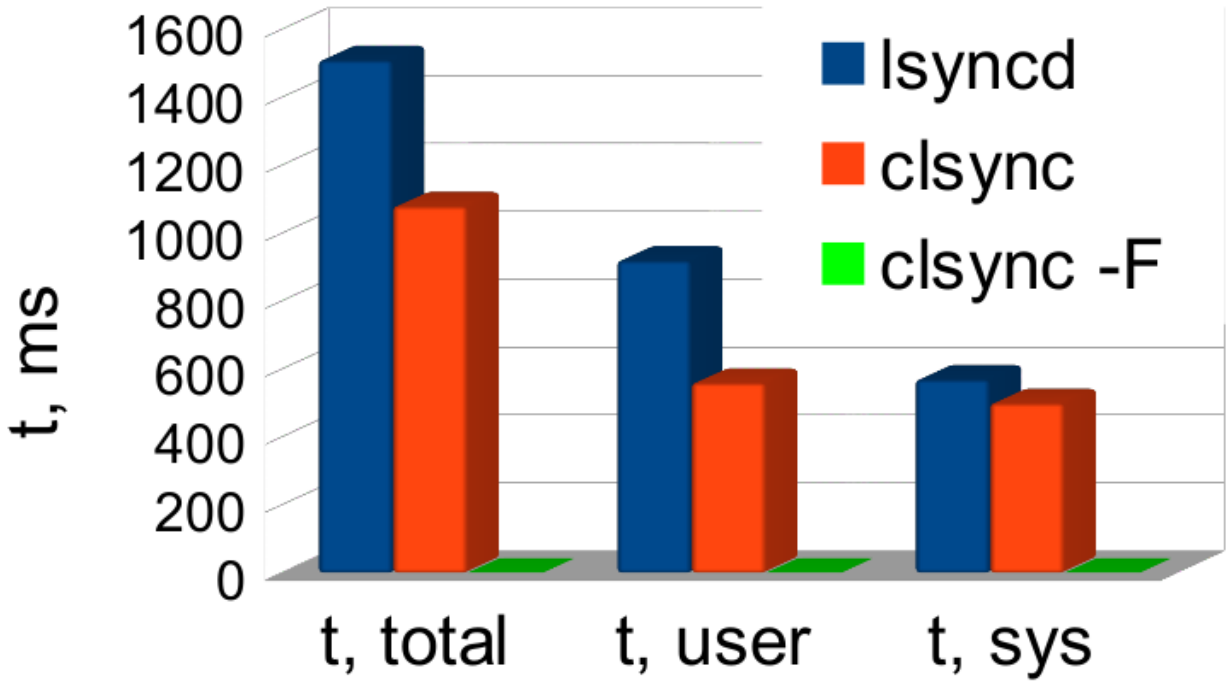
fork():

+ отдельный процесс для (setuid, setgid, exec)
- медленнее (shm)
- более сложный код

Дополнительные возможности

- Выделение тредов на каждый процесс синхронизации
- Поддержка регулярных выражений и выбор типа объектов файловой системы
- Управление через UNIX-сокет
- Поддержка кластеризации через multicast
- Быстрая первичная синхронизация

Измерение накладных расходов



- lsyncd v.2.1.x
- clsync v.0.4.x
- 4789558 файлов и директорий на tmpfs
- -F: быстрая первичная синхронизацию, игнорируя правила фильтрации

Примеры использования

Зеркалирование директории

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

Одноразовая 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

Зеркалирование директории

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

Одноразовая 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 в реальном времени

```
clsync -w1 -t1 -T1 -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%
```

Решение проблем

```
Cannot inotify_add_watch() on [...]:
No space left on device (errno: 28)
```

Увеличьте sysctl **fs.inotify.max_user_watches**:

- один хендлер на директорию
- нет рекурсии
- умолчание ядра 8192

Примеры использования

Коррекция прав доступа в реальном времени

```
clsync -w1 -t1 -T1 -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%
```

Решение проблем

```
Cannot inotify_add_watch() on [...]:
No space left on device (errno: 28)
```

Увеличьте sysctl **fs.inotify.max_user_watches**:

- один хендлер на директорию
- нет рекурсии
- умолчание ядра 8192

Статус

- Зрелый проект
- В режиме сопровождения

Поддержка в дистрибутивах:

Gentoo	официальная
Alt	официальная
Debian	официальная
RH-based	есть spec/rpm
FreeBSD	доступны порты

Единственная обязательная зависимость: *glib*.

Заключение

Контакты:

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

Благодарности:

- Дмитрий Окунев — главный разработчик
- Артём Аникеев и Barak A. Pearlmutter — пакетирование и тестирование
- oldlaptop и Enrique Martinez за их помощь

Спасибо за внимание!