Малые компьютерные системы со свойствами масштабируемости и высокой доступности

Ю.В.Шевчук <sizif@botik.ru>, А.В.Елистратов, А.Ю.Пономарев

OSEDUCONF-2023, 2023-01-27

ИЦМС ИПС им. А.К.Айламазяна РАН, Переславль-Залесский

Задача

Поддерживать работу ряда информационных сервисов для нужд организации:

- www
- mail = SMTP + IMAP
- wiki
- DNS
- . . . всего около 70 сервисов

Требования

1. Масштабируемость: возможность наращивать производительность системы по мере увеличения числа пользователей, нагрузки, объёма хранимых данных
2. Высокая доступность: отсутствие заметных пользователям перерывов в работе информационных сервисов

$$T_{\text{недоступности}} = T(1 - A)$$

$$\begin{aligned} 99.95\% &= 22 \text{ минуты недоступности в месяц} \\ 99.99\% &= 4.5 \text{ минуты недоступности в месяц} \\ 99.999\% &= 26 \text{ секунд недоступности в месяц} \end{aligned}$$

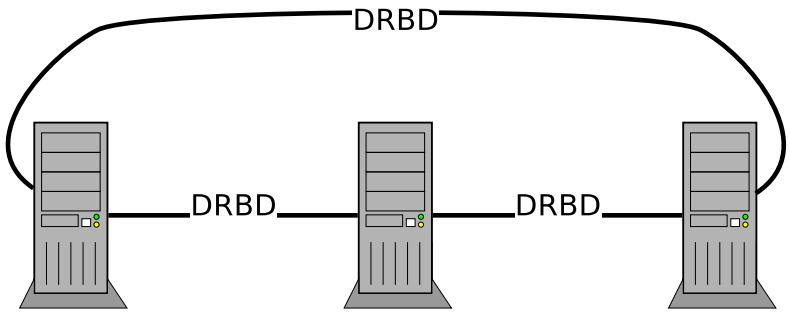
История

1994: единственный сервер Slackware Linux 1.1.2

2005: 2×Debian GNU/Linux, LVM + DRBD + reiserfs + vserver

2019: 3×Debian GNU/Linux, LVM + DRBD + ext4 + Xen + LXC

2019: 3×Debian GNU/Linux, LVM + DRBD + ext4 + Xen + LXC



- каждый сервис работает в отдельном контейнере LXC
- группы из нескольких LXC работают в Xen VM,
- на каждом физическом сервере работает 2 или более Xen VM
- каждый сервис имеет собственный раздел LVM + DRBD
- Xen обеспечивает «живую миграцию» VM на соседние (парные по DRBD) физические серверы
- => можно без перерыва в работе сервиса разгрузить и выключить, обслужить, заменить любой физический сервер

Недостатки варианта 2019

1. **ограниченная масштабируемость:** невозможно дать одной VM больше ресурсов, чем есть на физическом сервере;
2. **требование гомогенности:** живая миграция требует идентичных процессоров (флаги в `/proc/cpuinfo`) => нельзя добавить более современный сервер в кольцо;
3. **обновление ядра на VM** требует перезагрузки VM, расходует бюджет времени недоступности;
4. **трудное разрешение конфликтующих изменений** при репликации на уровне блочных устройств (split brain).
5. **ручное распределение ресурсов,** борьба с фрагментацией

Уйти в публичное облако? есть препятствия

коффициент доступности канала связи с облаком: внутренние пользователи попадают в зависимость от внешней коннективности

масштабируемость ограничена размером физического сервера, используемого облачным провайдером (хотя это большой сервер);

нет подходящих тарифов

лимитированный объём трафика и/или пропускная способность

риск ухудшения условий

конфиденциальность данных

Распределенные программные системы

- каждый сервис предоставляется не одним, а множеством Linux-серверов
- общее информационное пространство – объектное хранилище (не блочное) со средствами автоматического разрешения конфликтов
- потоки событий как средство синхронизации процессов на разных серверах
- средства для распределения запросов по серверам:
  - `haproxy` / `nginx` / ...
  - множественные записи A в DNS (DNS load balancing)
  - anycast IP адресация
  - трансляция запросов более свободному узлу

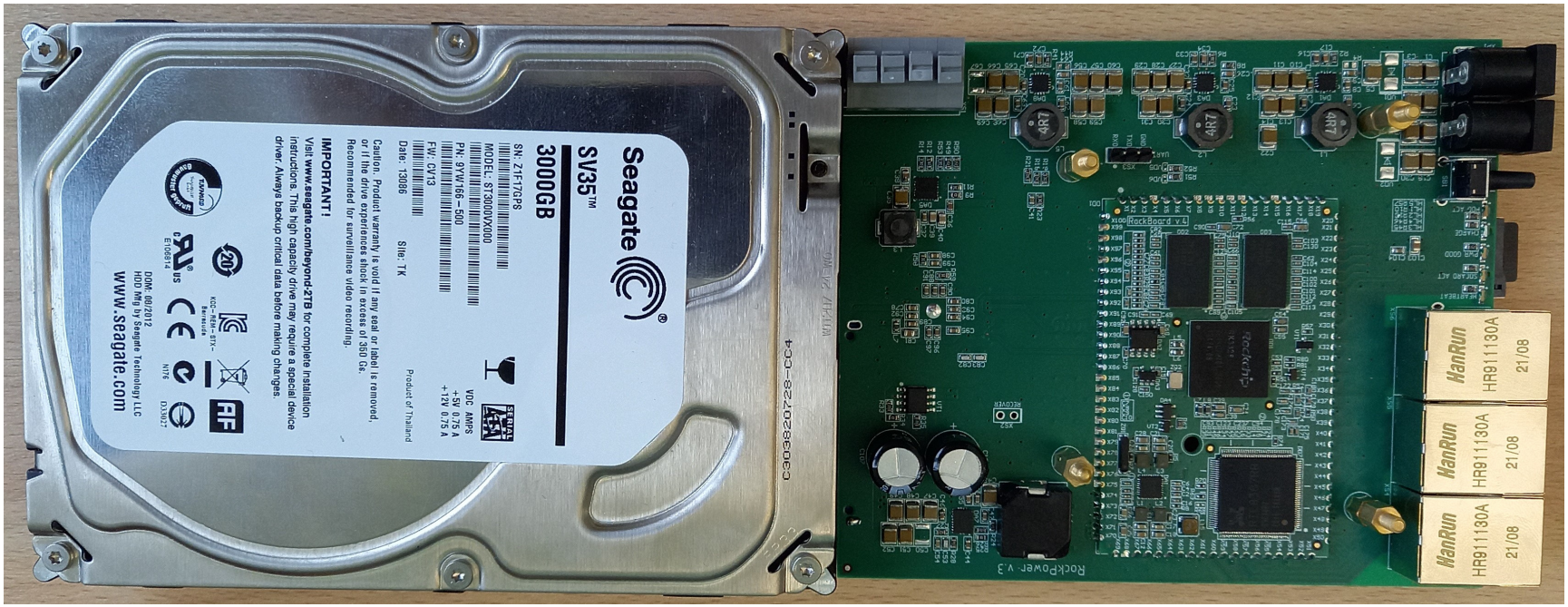
Аппаратная платформа для распределенных ПС

множество физических серверов – дорого для маленьких организаций

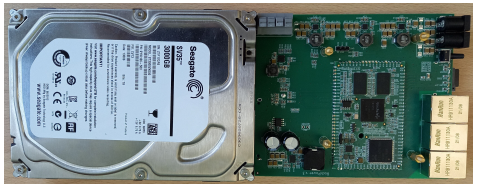
множество виртуальных серверов в облаке – частично возможно, но есть препятствия

множество *маленьких* физических серверов – как Raspberry Pi, но специализированных для построения отказоустойчивых мультикомпьютерных систем.

Botik SBC: внешний вид



Botik SBC: конструктивные особенности



SoC Rockchip RK3328 (4×ARM Cortex A53, 1.5 ГГц)

оперативная память: 4ГБ DDR4

flash-память: 8ГБ eMMC

flash-память: microSD

интерфейс SATA III

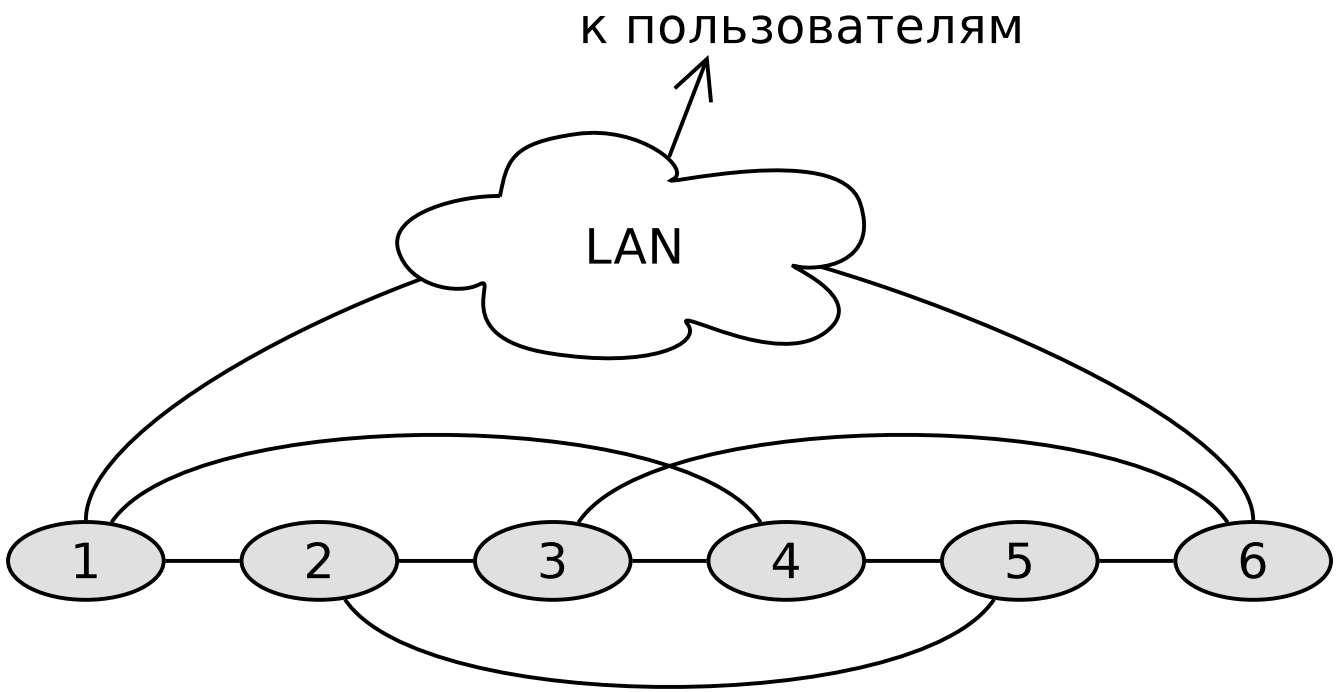
подсистема питания HDD: 5В, 12В

встроенный аккумулятор Li-Ion для «мягкой посадки»

встроенный коммутатор Gigabit Ethernet, 3 внешних порта

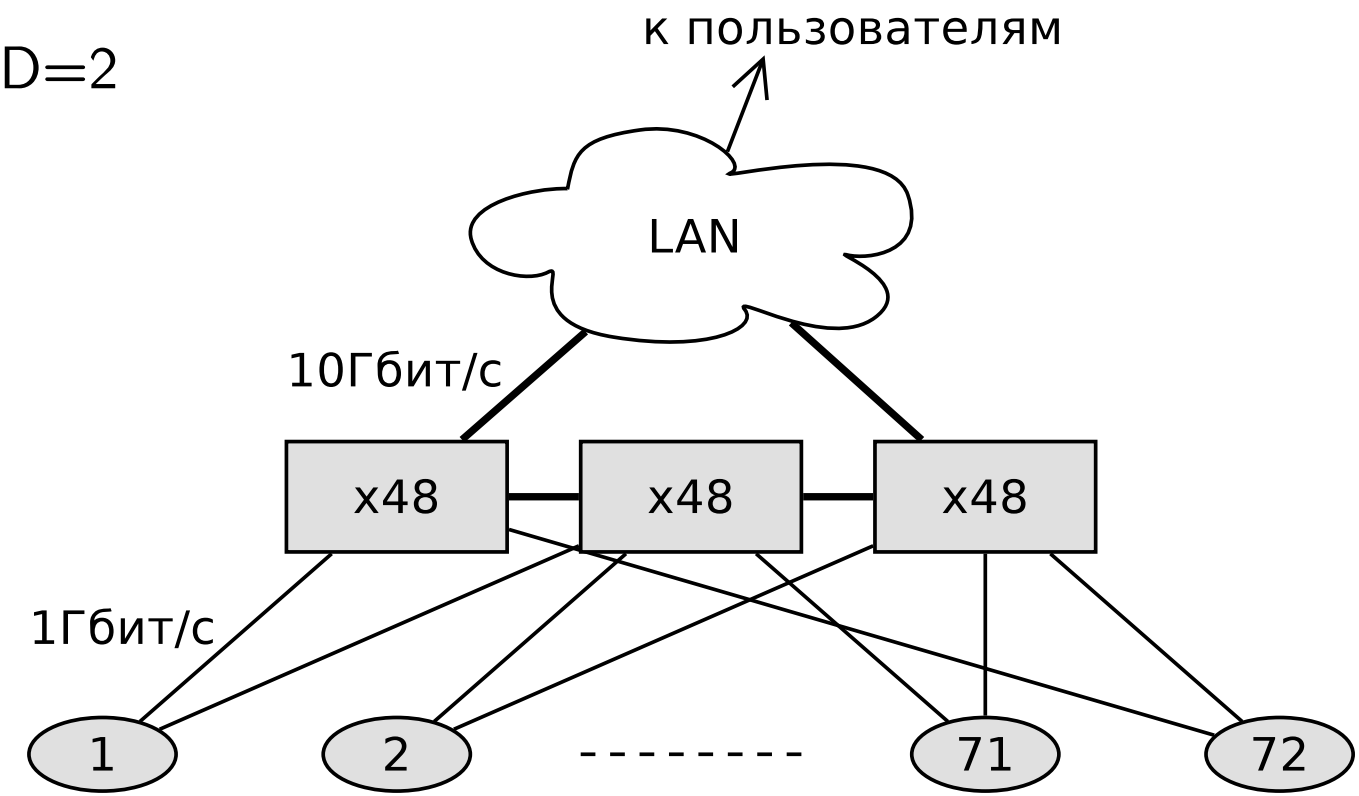
питание 10..15В 1А, два альтернативных входа

Пример сети на встроенных коммутаторах: 6 узлов



Пример сети на внешних 48-портовых коммутаторах

72 узла
Диаметр D=2



Свободное ПО для мультикомпьютерных систем

масштабируемые хранилища:

- Ceph
- GlusterFS
- Openstack Swift
- Hadoop
- Riak KV + Riak CS

Riak Core: библиотека для создания распределенных отказоустойчивых программных систем на языке **Erlang**

Apache Kafka: распредёленный отказоустойчивый брокер (архитектура издатель-подписчик) и архив сообщений. Может быть основой для мультиязыковых распределенных программных систем.

Заключение: основные тезисы

1. Распределенные программные системы как решение проблем высокой доступности и неограниченной масштабируемости
2. Сети малых компьютеров как общедоступная аппаратная платформа для построения систем с высокой доступностью и горизонтальной масштабируемостью