Как из головоногих моллюсков построить хранилище для частного облака

Межов Игорь Александрович, CIO, Уютерра, 2015

О нас

уютерра

товары для дома

- федеральная сеть
- 44 региона
- 120 магазинов
- 10 лет на рынке

О чем не будем говорить

- О стратегиях и концепциях *aaS
- О том, как строить cloud-aware сервисы
- О том, Linux – признанная платформа для облачной инфраструктуры
- О решениях Microsoft ;)

О чем будем говорить

Где взять хранилище для небольшого частного облака?

The Right Way

- Вендоры первого эшелона
- Быстрые, надежные и мощные СХД
- А еще надо позвать интеграторов! ;)

... и готовить N чемоданов купюр известного цвета

The Right Way – обратная сторона

- Midrange - потолок масштабирования
- HiEnd СХД стоят как небольшой самолет
- Поддержка SAN/CNA инфраструктуры
- Support fees, vendor lock и EOL

... и готовить N/5 чемоданов купюр ежегодно

А давайте пойдем «налево»?!

- SDS – software defined storage

(программные хранилища)

- Работа на «commodity hardware»
- Кластерная архитектура (=масштабирование)
- Использование обычной инфраструктуры
- Программная реализация функций

(снапшоты, репликация)

CEPH = Cephalopoda (семейство моллюсков)

- Распределенное хранилище
- Масштабирование до эксабайт
- Архитектура не содержит SPoF
- 3-in-1: объектное, блочное и файловое
- Работает на обычном оборудовании
- Быстроразвивающийся OpenSource

История CEPH

- Сэйдж Вейл (Sage Weil) - создатель
  - 2006 - дипломная работа, концепт
  - 2007 – реализация на fuse
- Компания InkTank (Weil/Yehuda Sadeh/Gregory Farnum)
  - 2012.07 - первый стабильный релиз - Argonaut
  - 2013.01 – Bobtail, 2013.05 – Cuttlefish, 2013.07 – Dumpling,
  - 2013.11 – Emperor , 2014.05 – Firefly, 2014.10 – Giant
- RedHat приобрела InkTank (2014.04)

Идеология CERN

- Данные распределены по всем узлам
- Распределение:
  - алгоритмически ВЫЧИСЛЯЕМО и равномерно
  - может динамически изменяться
  - учитывает топологию
- Клиент знает сам, куда обращаться
- Самопроверка и самовосстановление узлов
- Пауза лучше, чем ошибка

Что храним? Объекты!

Объект		
ID	Двоичные данные	Метаданные
1234	0100101000101001121010100 1001	name1 = value1 name2 = value2 ... nameN = valueN

Что можно делать с объектами:

- прочитать, записать и удалить объект
- прочитать, записать и удалить метаданные объекта
- поддерживается синхронное и асинхронное обращения

ID объекта уникален для всего кластера

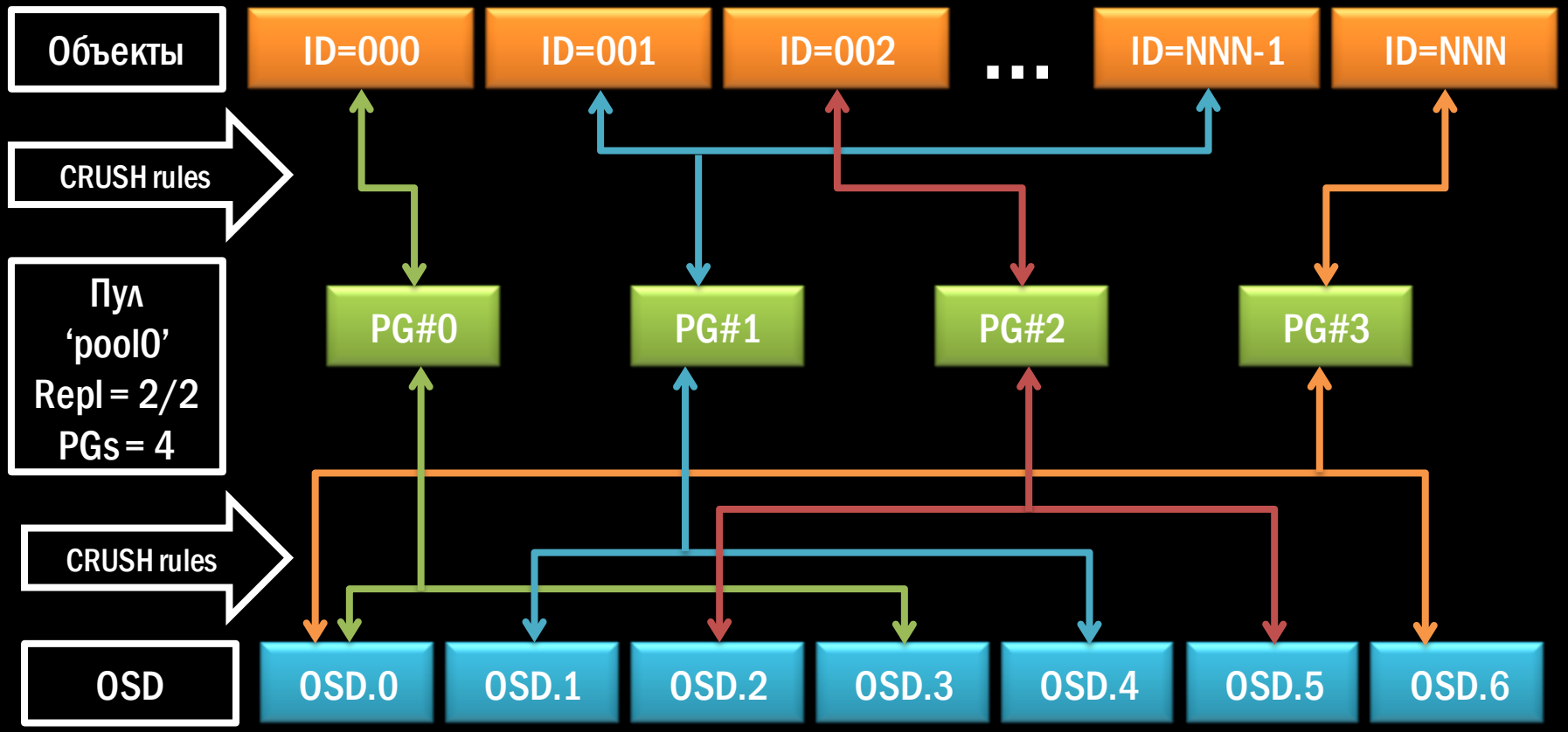
Что внутри: CRUSH и немного магии!

CRUSH = Controlled Replication Under Scalable Hashing

- Псевдослучайный (но детерминированный!) алгоритм определения размещения
- Места хранения ВЫЧИСЛЯЮТСЯ КЛИЕНТАМИ самостоятельно
- На размещение может влиять набор правил (CRUSH ruleset):
  - задается необходимый/желаемый фактор репликации (число реплик объекта)
  - задается топология инфраструктуры хранения (дерево: диски, узлы, шкафы, комнаты, ЦОДы)
  - задаются сами правила распределения
  - задаются веса OSD, а, следовательно, и распределение
- Результат распределения – достаточно равномерный (<7-10%)
- Изменение правил динамически перераспределяет данные
- Реплики одного объекта разносятся как можно дальше по топологии (fail.domain)
- Небольшие изменения правил => пропорциональный объем перераспределения

Что внутри: пулы и PG

Pool – пул, логический раздел для размещения объектов
PG = Placement Group (группа размещения),
OSD = Object Storage Daemon (сервис хранения объектов)



Что внутри: «А давайте добавим!»

Pool – пул, логический раздел для размещения объектов
PG = Placement Group (группа размещения),
OSD = Object Storage Daemon (сервис хранения объектов)

Что внутри: CRUSHmaps

CRUSHmap – описывает структуру и топологию хранилища

CRUSHmap = дерево кластера + правила размещения

Дерево = диск(OSD) > хост > шасси > шкаф >

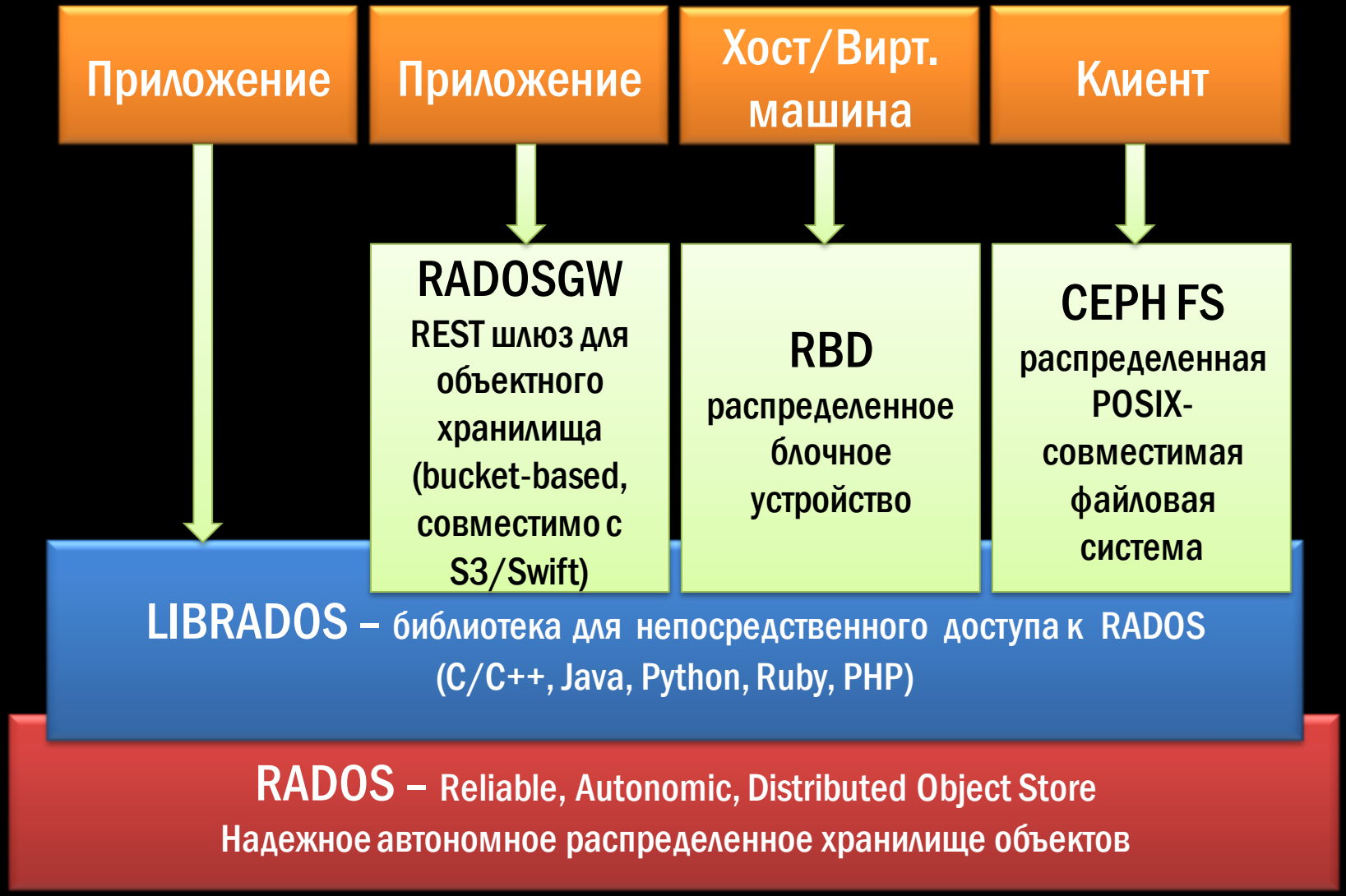
> ряд > PDU > комната > ЦОД > регион

- Листья дерева имеют весовые коэффициенты (1=1Тб)
- Весовые коэффициенты узла – сумма весов узлов нижнего уровня
- Веса влияют на распределение (емкие и быстрые – больше, медленные - меньше)

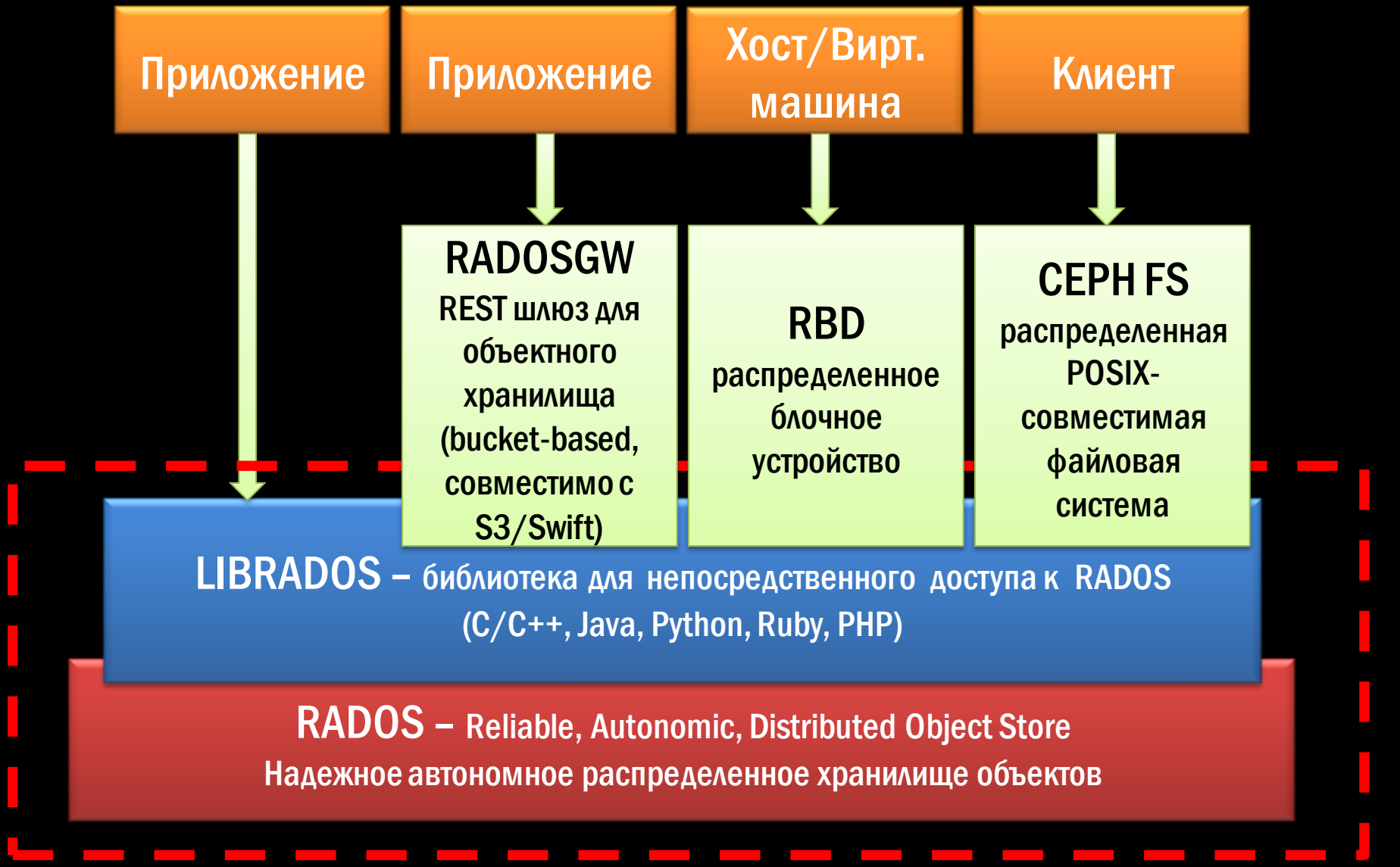
Правила размещения = как выбирать узел

- Правила по умолчанию обеспечивают хорошее распределение
- Правила задаются для пула
- Пользовательские правила используются для тонкой настройки

Архитектура CEPH: подсистемы



Архитектура CEPH: подсистемы



Архитектура CEPH: RADOS

Всё хранится в RADOS как объекты

- Два программных компонента: OSD и MON
  - OSD – Object Storage Daemon – хранение объектов
  - MON – Ceph Monitor –поддержание карты кластера и контроль состояния
- Алгоритм CRUSH
  - используется для определения нахождения объекта
  - нахождение – ВЫЧИСЛЯЕМО
  - нет никакого центрального сервера метаданных

Архитектура CEPH: OSD - функции

OSD = Object Storage Daemon

- «строительный блок» хранилища CEPH
- обслуживает запросы клиентов к хранимым объектам
- поддерживает атомарные транзакции
- обеспечивает взаимодействие с другими OSD

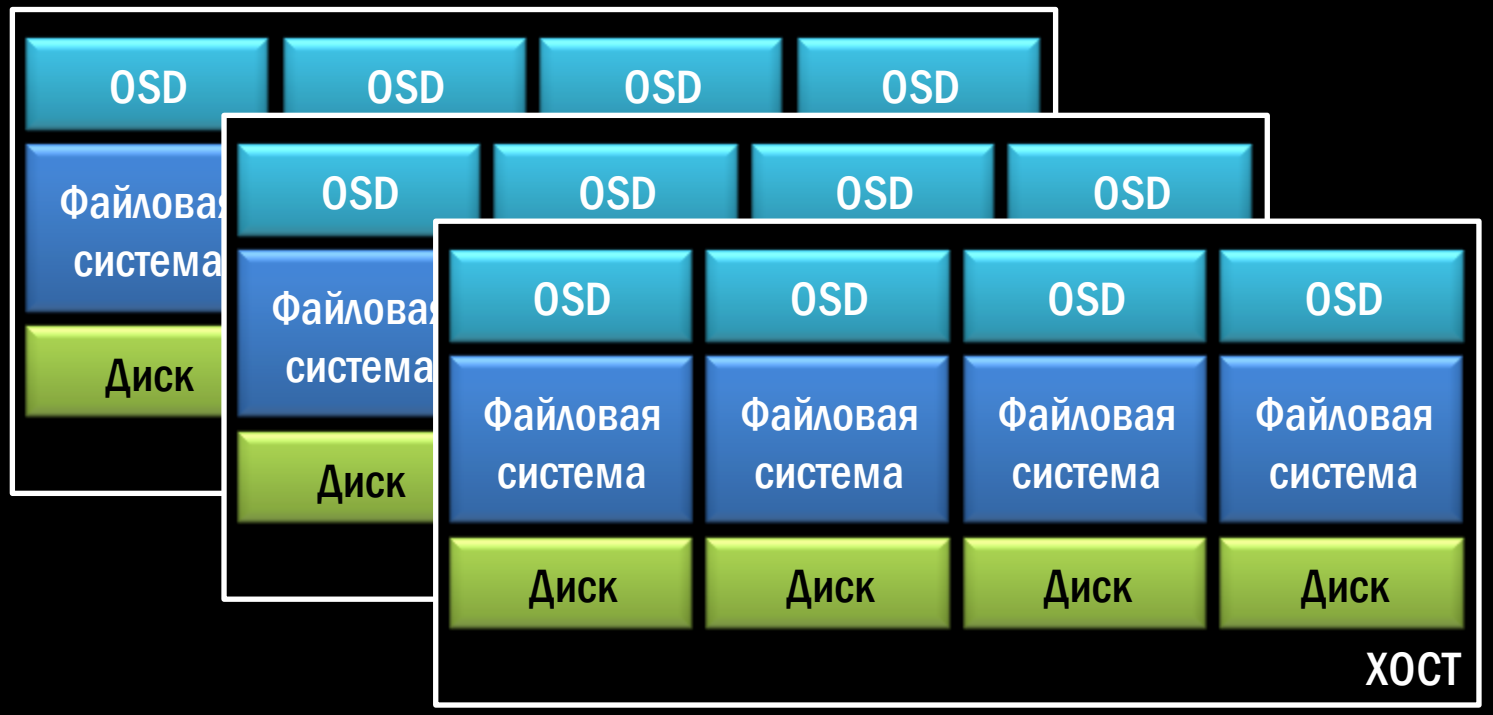
Если OSD первичный для некоторых объектов, то он:

- обеспечивает клиентский ввод – вывод
- обеспечивает репликацию на вторичные OSD
- обеспечивает идентичность всех реплик объектов
- отвечает за процедуры восстановления (recover) и балансировки (rebalance)

Если OSD вторичный для некоторых объектов, то он:

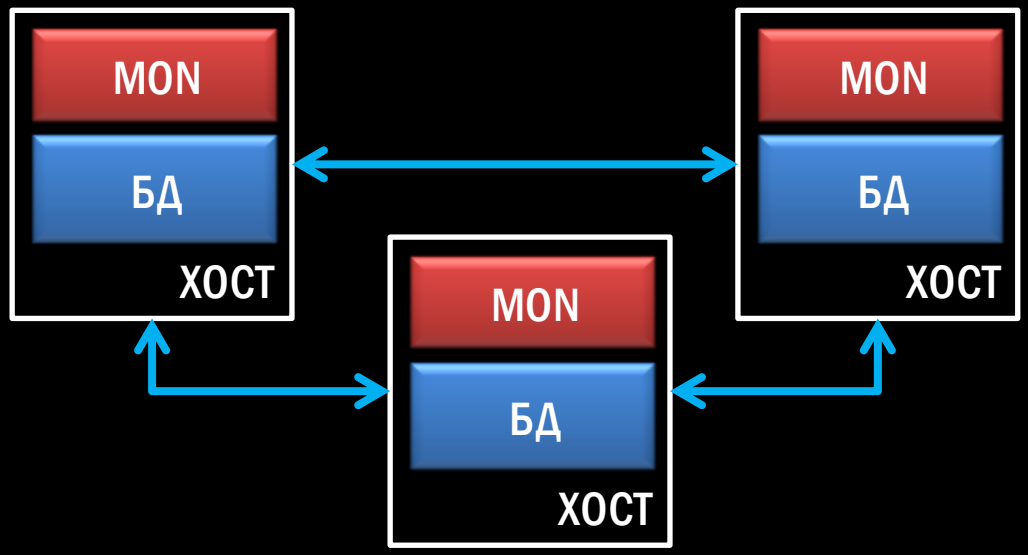
- управляется первичным OSD
- может стать первичным (remap)

Архитектура CEPH: OSD - ландшафт



- 1 OSD = 1 диск, ~8-20 OSD на хост, до десятков тысяч OSD на кластер
- Разные backend-ы для хранения: ext4, xfs, btrfs, LevelDB
- Отдельные write-ahead журналы на каждый OSD для целостности
- Журналы на SSD для увеличения производительности
- Peer-to-peer репликация с другими OSD

Архитектура CEPH: MON - функции

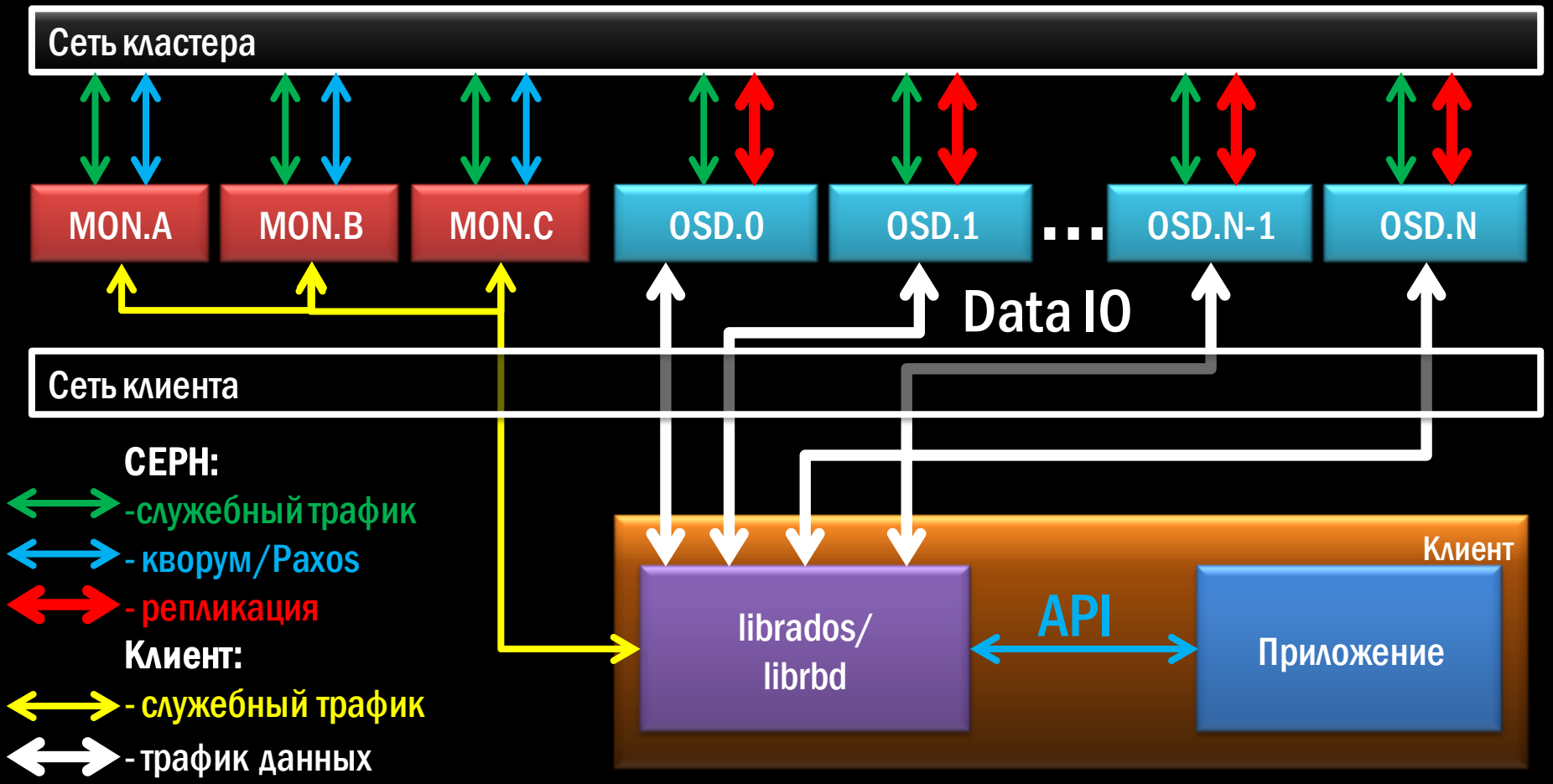


Монитор – «стюард» кластера CEPH:

- Поддержание мастер-копий карт кластера: мониторов (monmap), OSD (osdmap), групп размещения (pgmap), CRUSH (crushmap), серверов метаданных (mdsmap)
- High Availability – мониторы образуют кластер на мажоритарной основе (Paxos)
- Точка входа для клиентов: HA аутентификация и авторизация (cephx протокол)
- Мониторинг OSD, координация и сбор статистики

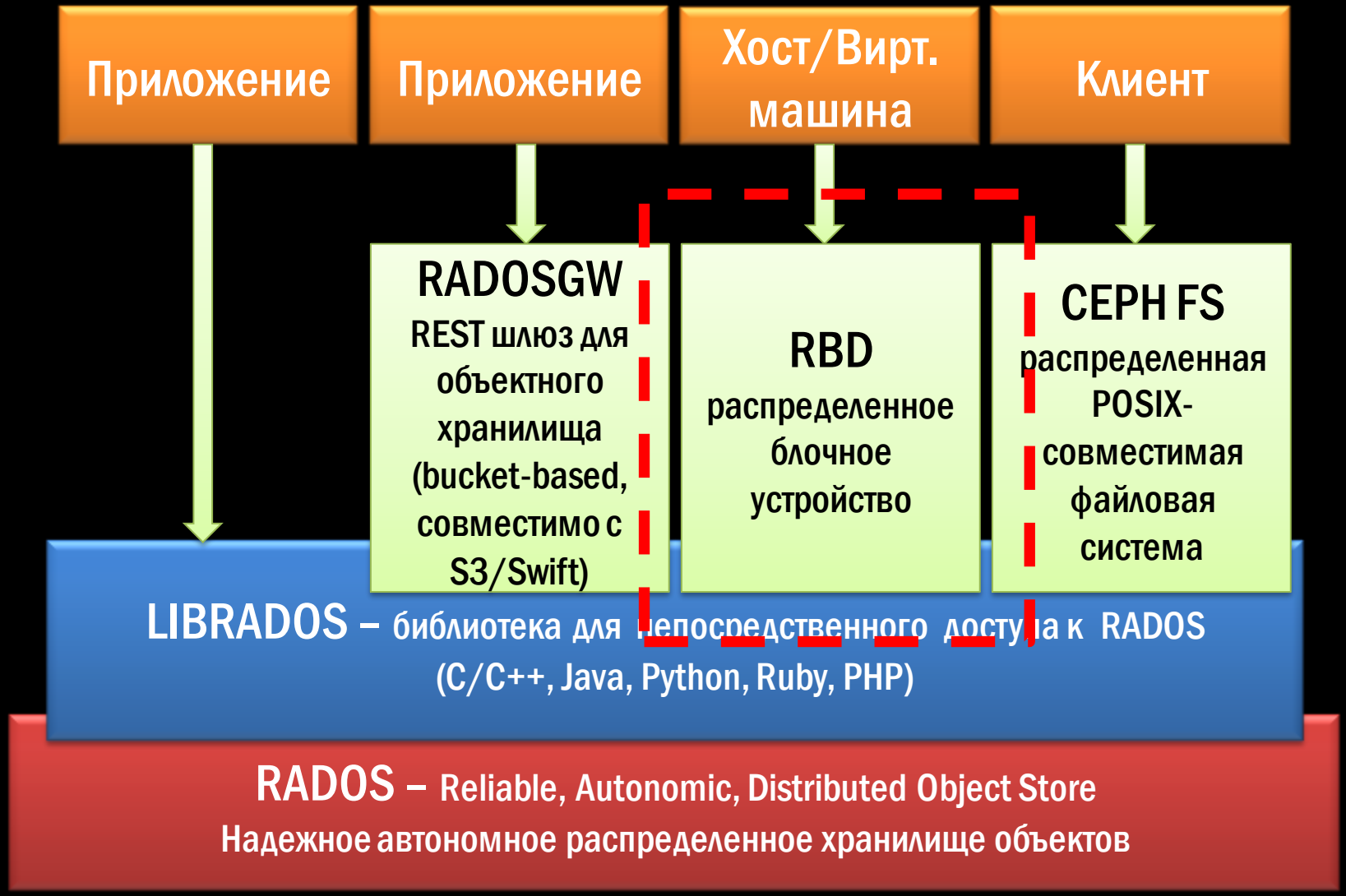
МОНИТОРЫ НЕ УЧАСТВУЮТ В КЛИЕНТСКОМ ВВОДЕ-ВЫВОДЕ

Архитектура CEPH: кластер в сборе

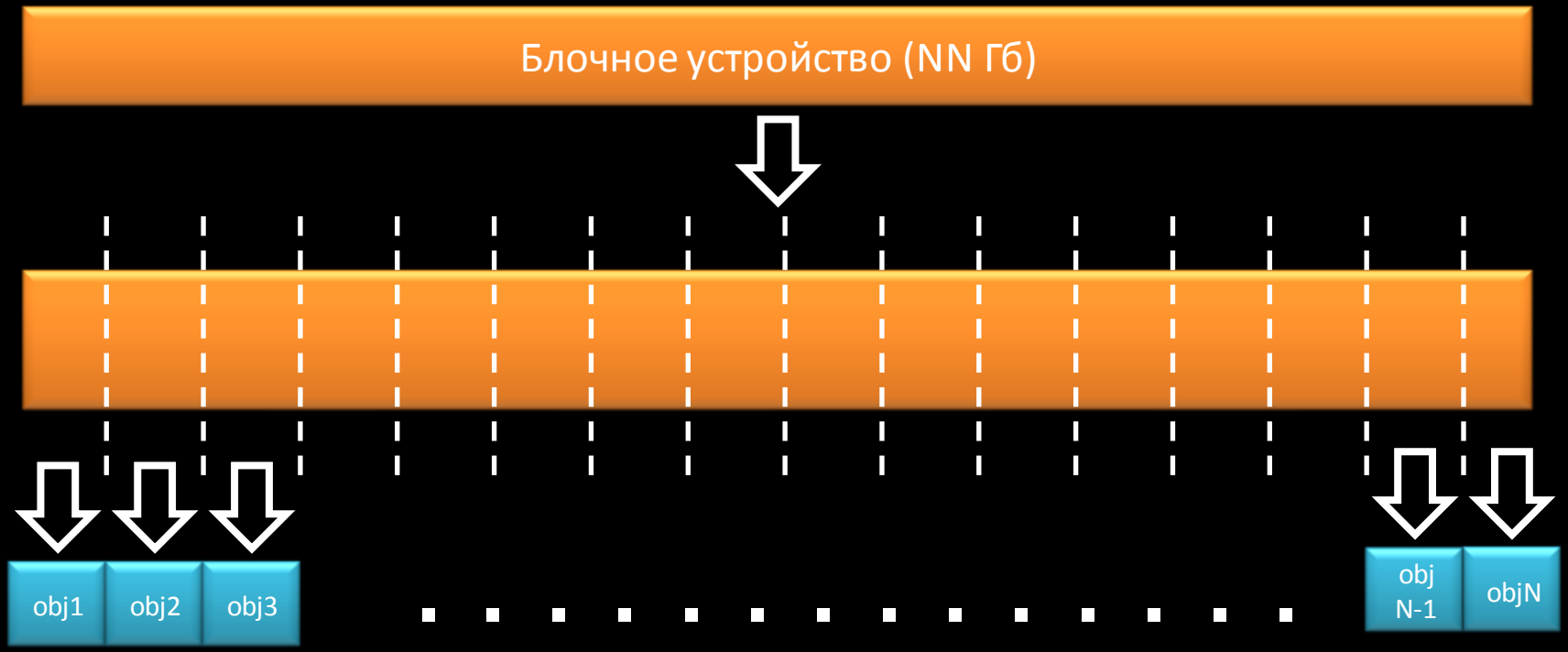


=> Клиенты сами распараллеливают IO.

Архитектура CEPH: RBD



Что внутри CEPH: как устроено RBD



- Пространство блочного устройства делится на части
- Каждая часть – отдельный объект
- Объекты размещаются в RADOS
- Тривиальные объекты (=0) можно не размещать

RBD features

- Снапшоты (readonly, с откатом)
- Layering - дочерние снапшоты (CoW)
- Клон – полная копия (flatten)
- Sparse images / Lazy allocation
- Kernel module + fuse mount
- Прямая поддержка qemu (storage backend)

=> Удобная основа для облачного хранилища

Преимущества CEPH

- **Масштабируемость:**
  - Calc !=Lookup, «просто добавь OSDшек»*, использование всех дисков
- **Отказоустойчивость:**
  - Репликация (2-8), учет топологии, распределенное управление:

    «Отказ – это норма, а не ЧП»
- **“Плюшки”:**
  - 3 в 1, kernel+qemu, thing provisioning , снапшоты, SSD-ready, tiered pools, быстрый ребилд и автовосстановление

* - навеяно древним рекламным лозунгом сухих напитков «Invite»

«RAID? Нет, не слышал...»

Действительно ли концепция RAID умерла?

- 1 OSD – 1 HDD. RAID не требуется
- Атомарные транзакции – целостность при изменении
- CRUSH разносит реплики как можно дальше по дереву
  (уход от точки отказа выше уровня дисков и отдельных узлов)
- Потеря OSD = «А давайте добавим!», но наоборот:

IO всех затронутых групп размещения, моментально переключаются (remap) на вторичные OSD, при отсутствии OSD более таймаута начинается перебалансировка.

- Потеря узла = потеря всех его OSD. Вполне допустима.
- По неполным PG (доступных реплик меньше, чем нужно) IO останавливается. IO возобновляется, как только PG станет полной.
- Фоновые процессы scrub и deep-scrub проверяют целостность

=> Резервируются не диски а совокупные ресурсы кластера

Проблемы CERN

- “Детка! Это все-таки кластер!”
  - общая сложность, не «out-of-the-box», сети - отдельно
- Интерконнект решает все
  - «Мама, купи мне на День Рождения небольшой Mellanox!» ☺
- Репликация «крадет IO» (записать N раз)
- Масштабирование наоборот
  - Производительность не полностью раскрывается на малых инсталляциях
- Есть еще, где улучшать:
  - Например: чтение – только с первичной для группы размещения OSD (0.80)

«Куда приводят мечты»

Масштабирование

Кластер хранения в ЦЕРН:

47 узлов, 1128 OSD
- -2 x E5-2650 + HT
- -2 x 10 Gbit Ethernet
- -24 x 3TB HDD (5k9)
- -64 Gb RAM

5 мониторов:
- -2 x Intel L5640 + HT
- -2 x 1Gbit Ethernet
- -2 x 2TB HDD
- -БД на SSD
- -48 Gb RAM

3 петабайта CephFS + RBD для образов VM (OpenStack) + RADOSGW

7 месяцев в работе (по состоянию на март 2014)

НЕ БЫЛО: ни отказов, ни потерь данных,

ни неустранимых проблем с производительностью

Зачем это нужно нам?

- Инфраструктура уже была виртуализована (KVM/QEMU)
- Всё уже управлялось OpenNebula

=> Не хватало только приличного хранилища

- Расширение и поддержка midrange SAN выливалась в серьезные затраты.
- Были свободные диски и недогруженные по IO серверы
- “Хочу петабайт, живую миграцию и много-много iops’ов!” Ну и JFF! ;)

=> А давайте запустим CEPH ?!

Ты помнишь, как все начиналось...

Proof-of-concept (PoC) CEPH в Уютерре

- 4 узла по 3x1Тб OSD, только sata hdd
- Пара MON и несколько тестовых VM на тех же узлах
- 1Гбит сеть

Итоги PoC:

- ЗАРАБОТАЛО!! Живая миграция
- Переживает потерю до 3 OSD или 1 узла (из 4х)
- Не быстро (~500 iops)

«Через тернии к звездам!»

Как проходил запуск в production

- Учли уроки PoC и ошибки:
  - 1 Гбит – только для PoC. 10Гбит и выше – необходимы
  - мониторы тоже, оказывается, жадные до CPU
- Вывод в prod проводился итеративно и без остановки
  - «ползучее»добавление узлов с 12xOSD в 10Гбит сегмент
  - «ползучий» перенос роли MON на хосты с 10Гбит
  - освобождение дисков в 1Гбит сегменте
  - перенос журналов на SSD

«Через тернии к звездам!»

Что достигнуто сейчас

Кластер хранения Уютерры:

5 узлов, 58 OSD
- -2 x E5520/E5620
- -2 x 10 Gbit Ethernet (failover)
- -6x2TB + 6x1TB HDD (SATA 7k2)
- -32-48 Gb RAM

- 3 MON (совмещены с узлами хранения)
- 10Гбит интерконнект
- кластерная и клиентская сети совмещены

78 TB – Хранилище RBD для образов VM (KVM/OpenNebula)

В эксплуатации – с октября 2014

Отличная гибкость и надежность

Хорошая производительность (>60VM, ~2kiops avg)

Спасибо за внимание!

Ваши вопросы?

Межов И. А., Директор ДИТиО, Уютерра, 2015

megov@yuterra.ru +7 915 855 3139