JAVA

Как же добиться стабильной производительности?

Кочубеев Юрий
Консультант по решениям,
World IT systems

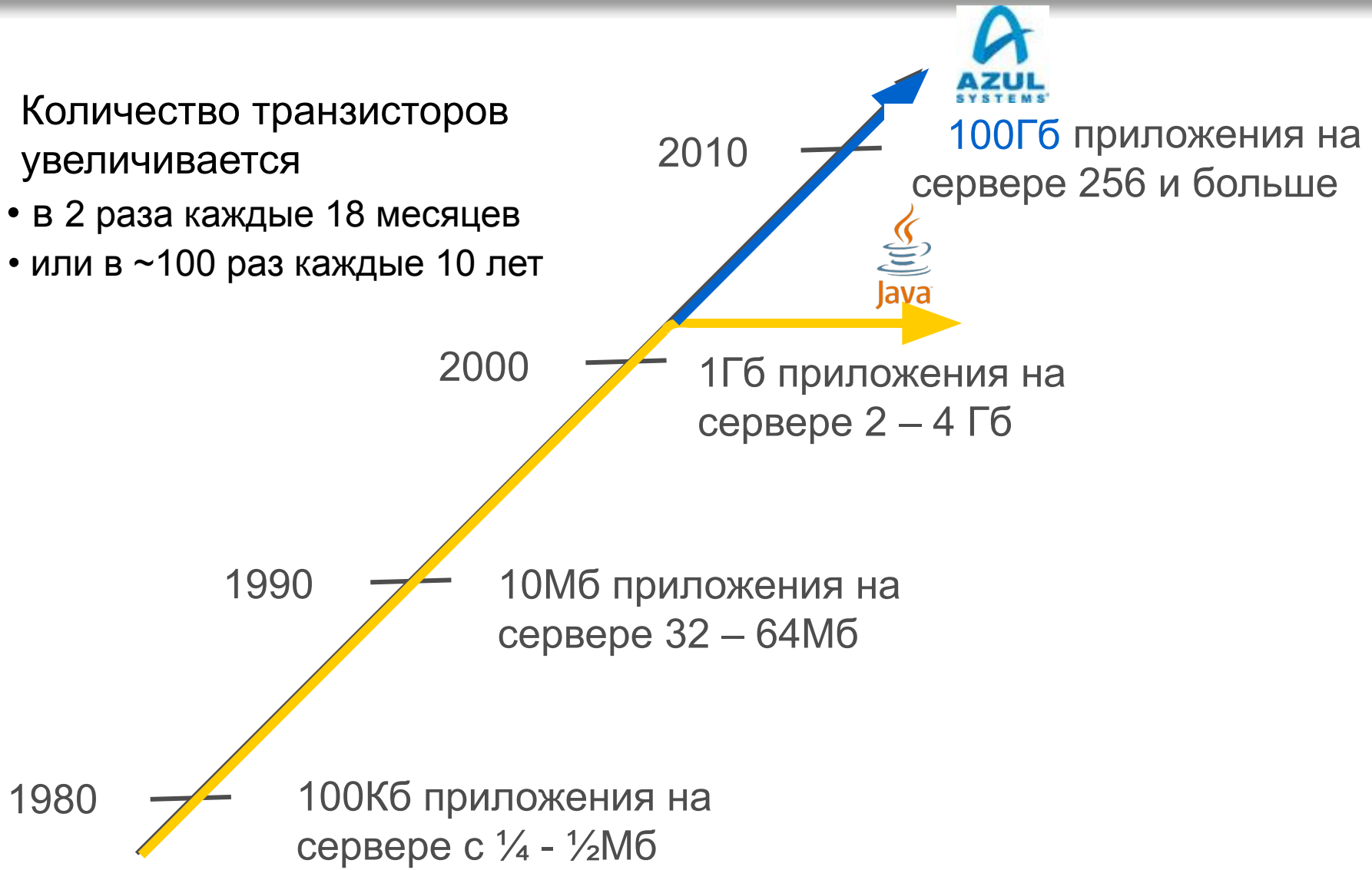
Ограничение 2GB:
Так уже было – вспомните про 640K

- 2001-й год: отдельные экземпляры приложения получают ~1-2ГБ памяти
  - Некоторые занимают меньше памяти
  - Ничтожное количество занимает больше
- 2011-й год: отдельные экземпляры приложения получают ~1-2ГБ памяти
  - Очень мало кому хватает меньшего объема
  - Очень мало кто может работать с большим объемом
- Практически, объем памяти, доступный отдельному приложению, не изменился за последние 10 лет.

Закон Мура

Количество транзисторов увеличивается

- в 2 раза каждые 18 месяцев
- или в ~100 раз каждые 10 лет



Почему ~2Гб?
Все из-за механизма очистки памяти GC

- Является выявленным на практике пределом выше которого приложения перестают быть предсказуемыми в поведении
- Объем памяти 100Гб не приведет к катастрофическому падению приложения. Он просто приведет к появлению “пауз” в работе приложения, длящихся порой несколько минут.
- Все существующие на сегодняшний день реализацииJVM страдают периодическими «зависаниями» на секунды и даже десятки секунд при работе с объемами памяти порядка 2GB.
  - Вопрос не ”Будут ли паузы?” - вопрос “Когда?” они будут.
  - Все известные способы тюнинга GC всего лишь влияют на то “Когда?” и “Как часто?” будут происходить паузы
- “*Compaction is done with the application paused. However, it is a necessary evil, because without it, the heap will be useless…*” (JRockit RT tuning guide).

Решение Azul – специализированная платформа для Java

- Используйте платформу специально разработанную для JAVA…..
  - Гибко масштабируйте приложения
    - Гибко выделяйте ресурсы, в зависимости от запросов приложения, повышая тем самым масштабируемость, эффективность и отказоустойчивость
  - Эффективно используйте ресурсы виртуализованной Java платформы
    - На порядки большая масштабируемость, и производительность
  - Упрощайте развертывание приложений
    - Упростите управление за счет уменьшения количества узлов в кластере сервера приложений.

Два варианта развертывания AZUL

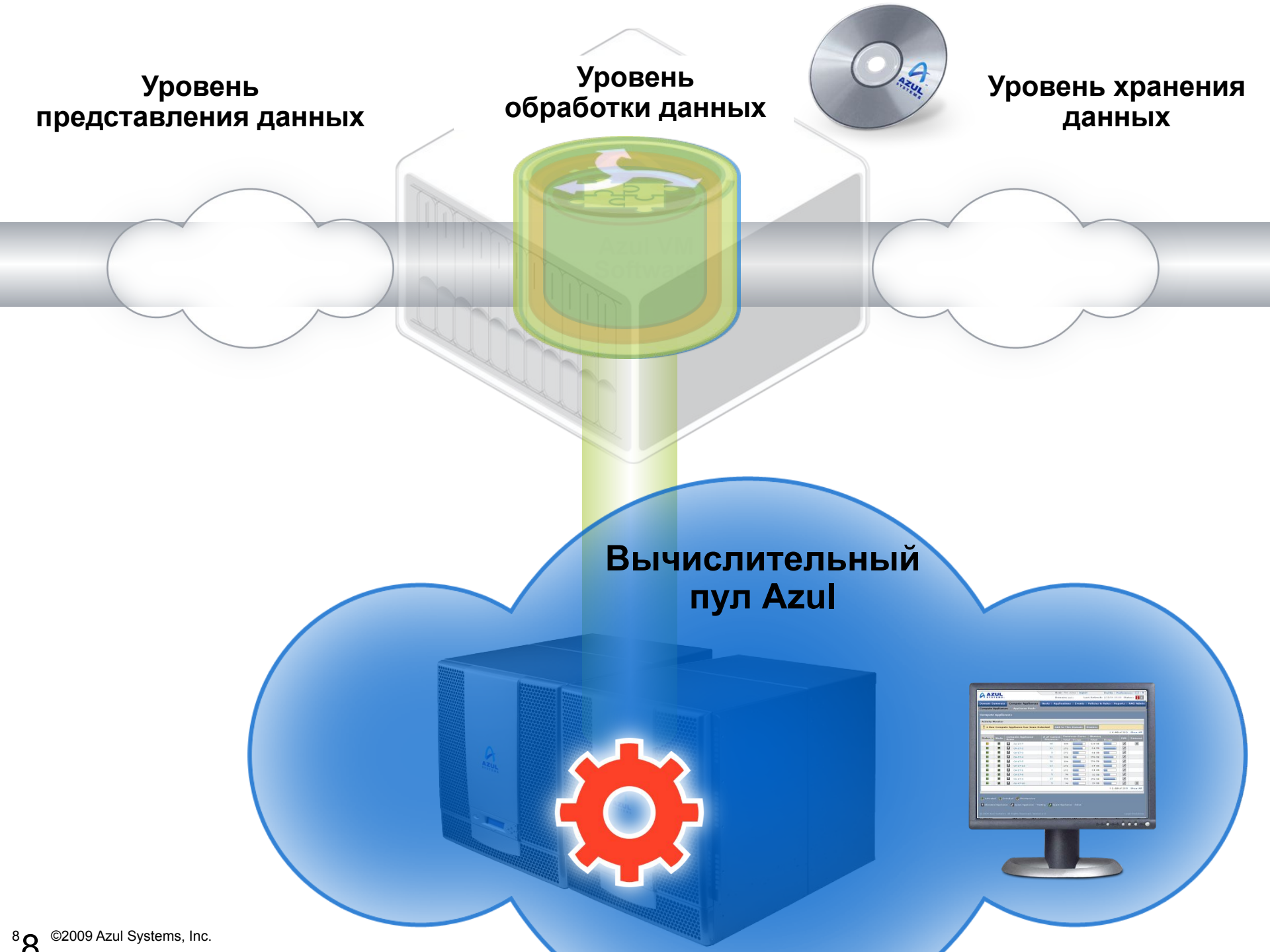
Существует два варианта развертывания AZUL:

- Специализированное аппаратное решение Vega Appliance
  - Огромные объемы памяти (768Гб)
  - Огромное количество процессорных ядер (864 CPU cores)
  - Малый размер (5 юнитов или 14 юнитов)
  - Низкое энергопотребление (< 1 КВт или < 3.3КВт)
- Виртуальный Appliance для x86 с Intel VT или AMD-v
  - Cloud ориентированная платформа
  - Использование эффективных и недорогих аппаратных компонент
  - Простота использования стандартной платформы x86
  - Наиболее полное использование плюсов виртуализации

AZUL – JVM в кремнии

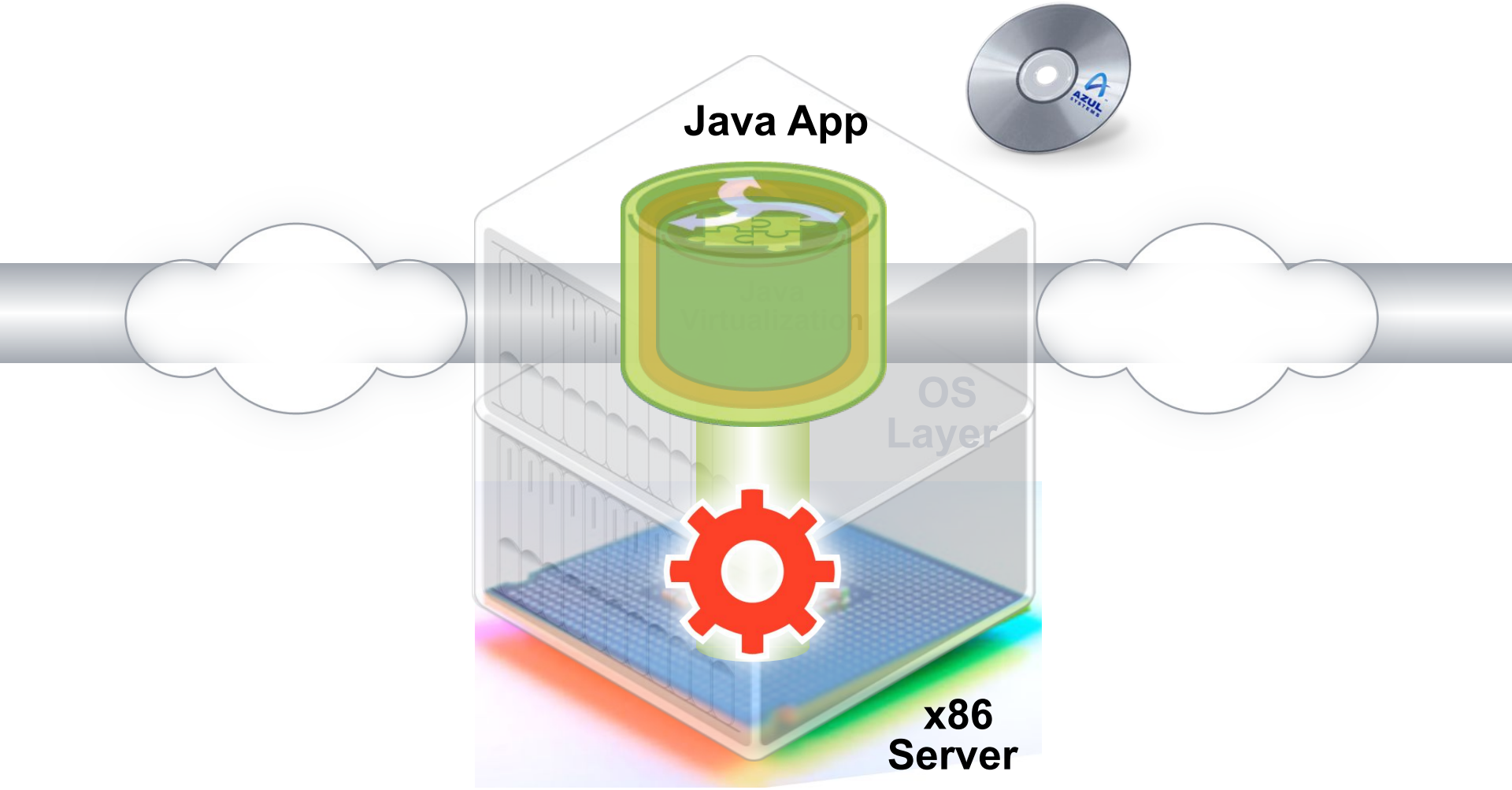
- 54-х ядерный процессор
- До 16-ти процессоров
- Управление распределенной памятью
- До 864 процессоров и

  768 Гб ОЗУ в одном суперкомпьютере
- Поддержка на аппаратном уровне:
  - Garbage collector
  - Управление памятью
  - Optimistic thread locking («оптимистичный алгоритм блокировки»)



©2009 Azul Systems, Inc.

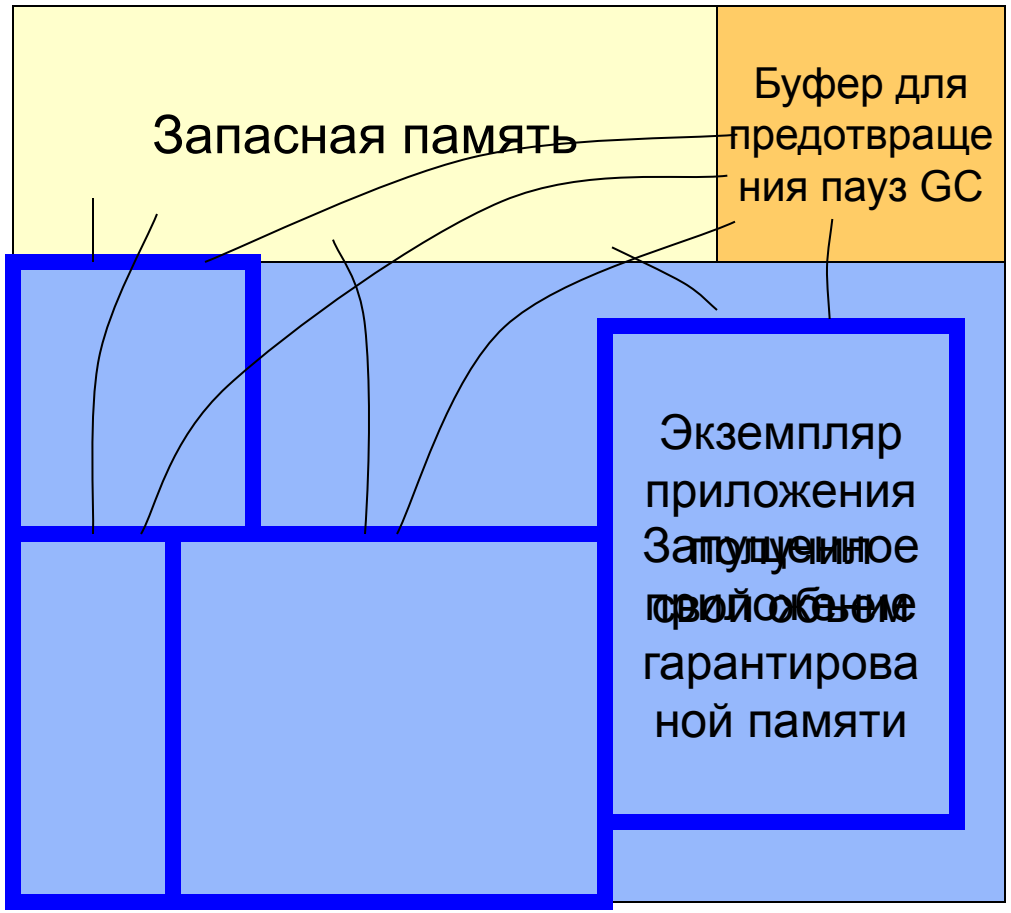
Виртуальный Java Appliance



Механизм «Запасной памяти»

Кооперативная память

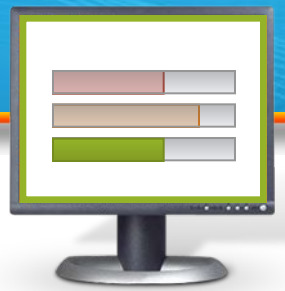
Гарантированная память

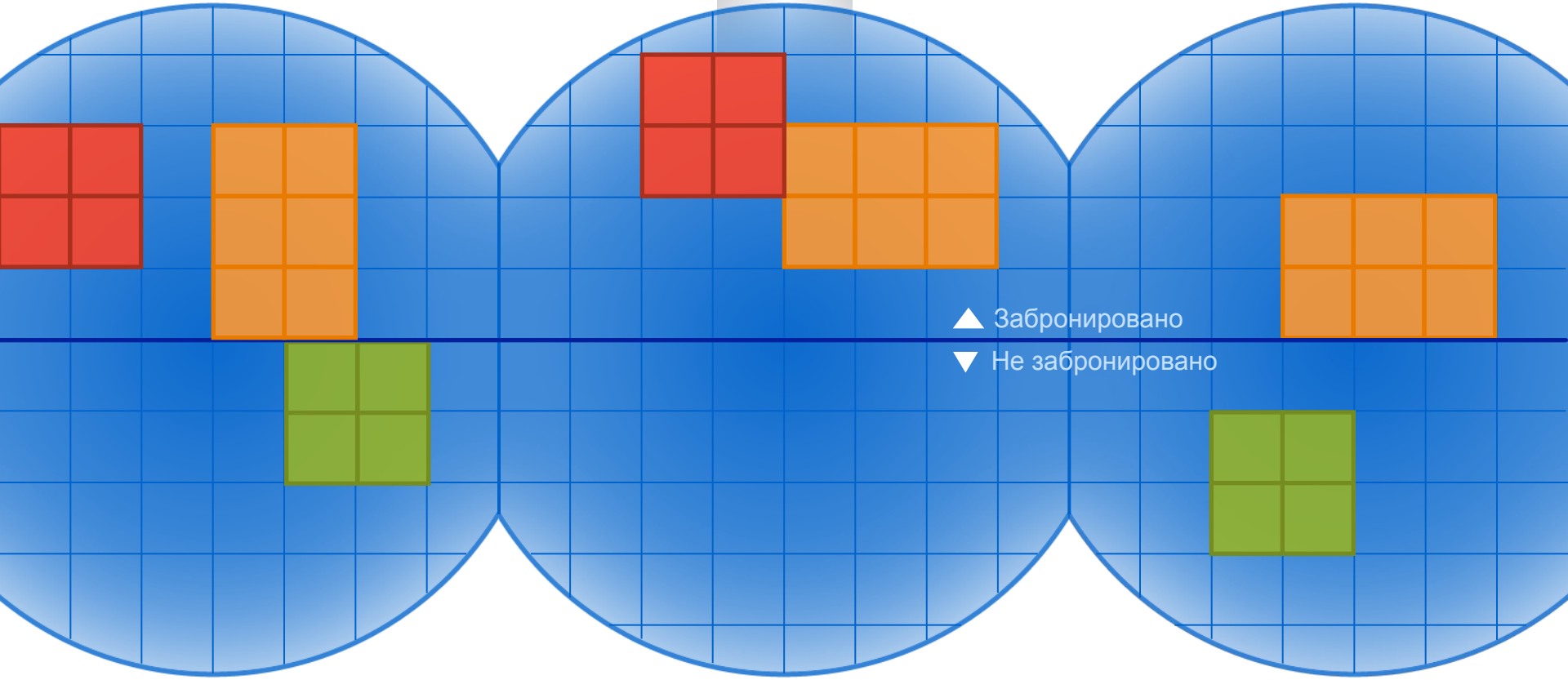


Запуск нового приложения

Разделяемая виртуализованная среда

Менеджер ресурсов





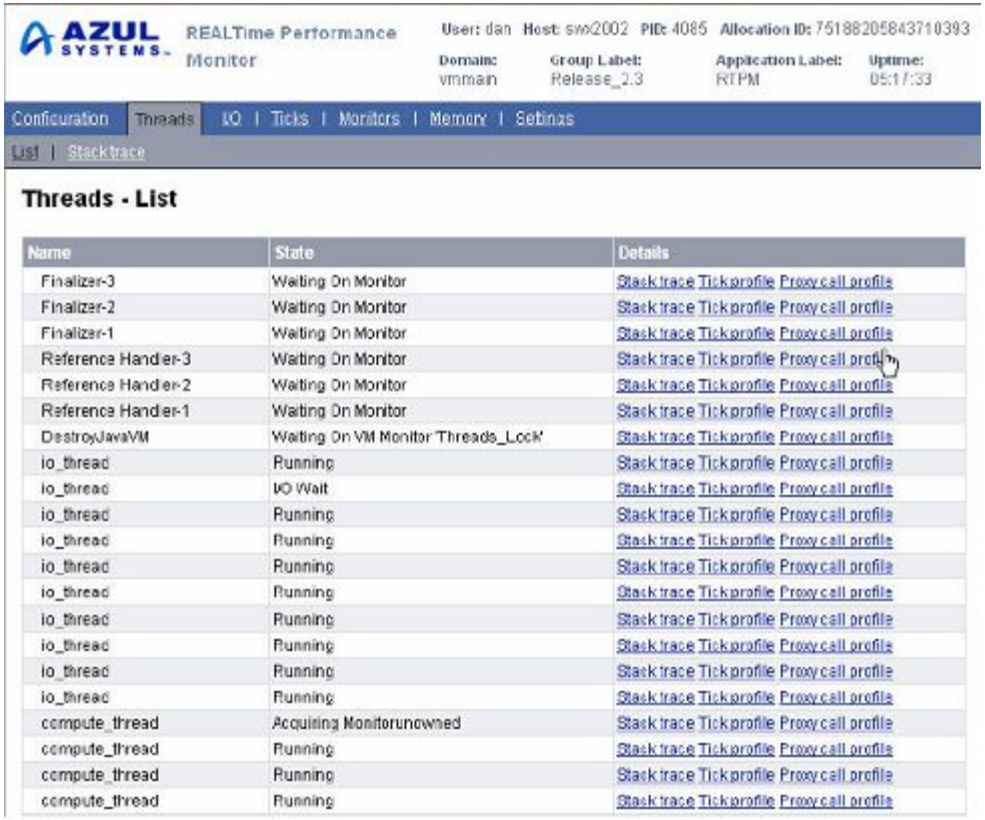
Azul1

Azul 2

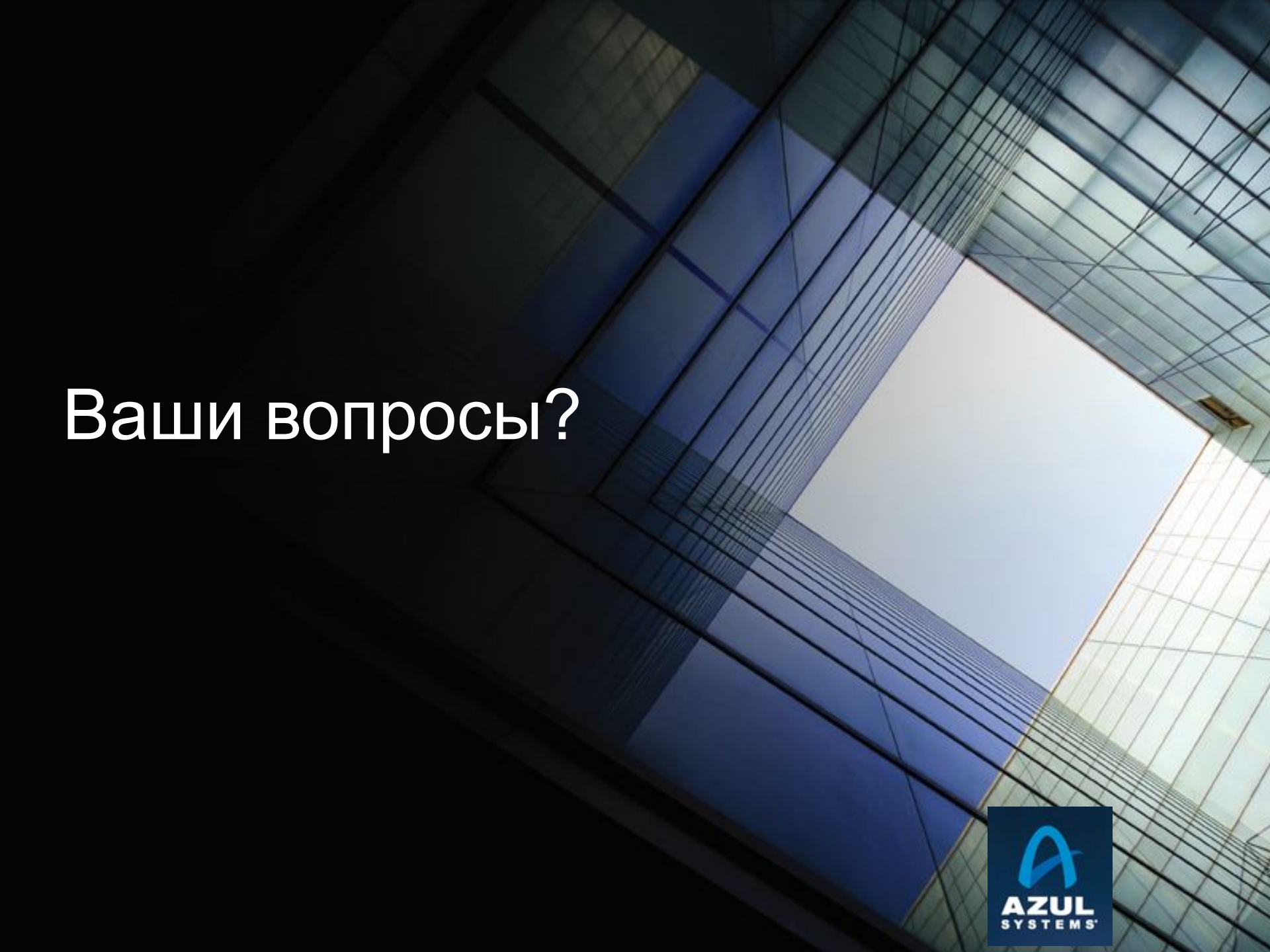
Azul 3

Управление производительностью

- Информация о потоках
  - Имя класса
  - Статус потока
- I/O статистика
  - Вызовы прокси
  - Информация о кеше
- Критические сессии
  - Имя монитора
  - Время ожидания
- Мониторинг ОЗУ
  - Статистика загруженности
  - Информация о выделенных объектах



Примеры заказчиков

Supporting mission-critical deployments around the globe

Ваши вопросы?