Hot Topics in Operating Systems

Евгений Иванович Клименков

osisp2019@gmail.com

Белорусский Государственный Университет
Информатики и Радиоэлектроники

2019

MapReduce

MapReduce представляет собой модель программирования для обработки больших объемов данных.

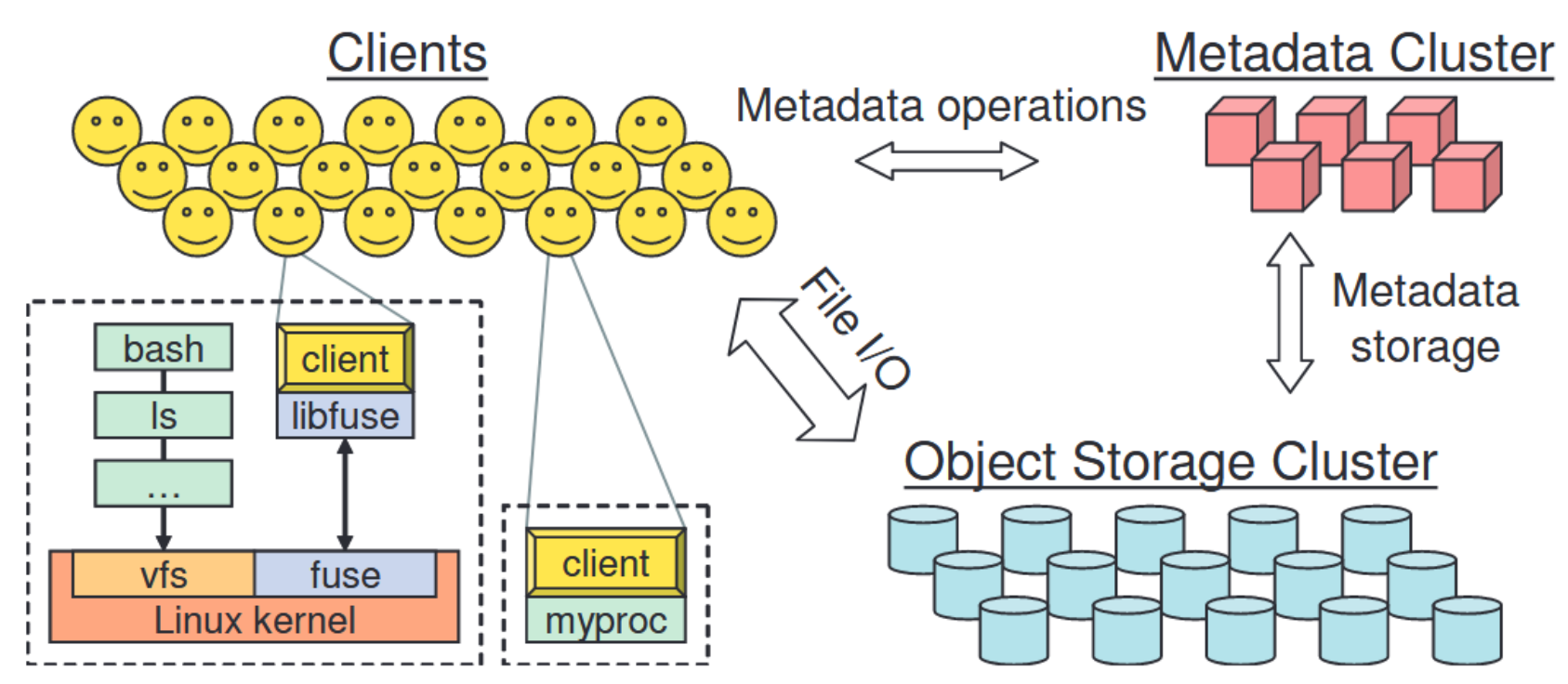
MapReduce реализует стратегию “раздели-обработай-подведи итоги”.
MapReduce опирается на локальность данных. Фактическая обработка данных происходит на тех узлах системы, на которых располагаются обрабатываемые данные.
По этой же теме можете посмотреть на BigTable.

Распределенные вычисления

Одним из ключевых компонентов является распределенная файловая система:

- специализированная: MapReduce, BigTable, Dynamo, Cassandra
- общего назначения: Ceph

Принципиальная модель DFS



Виртуализация и Безопасность

Гипервизоры для защиты традиционных операционных систем (SecVisor).

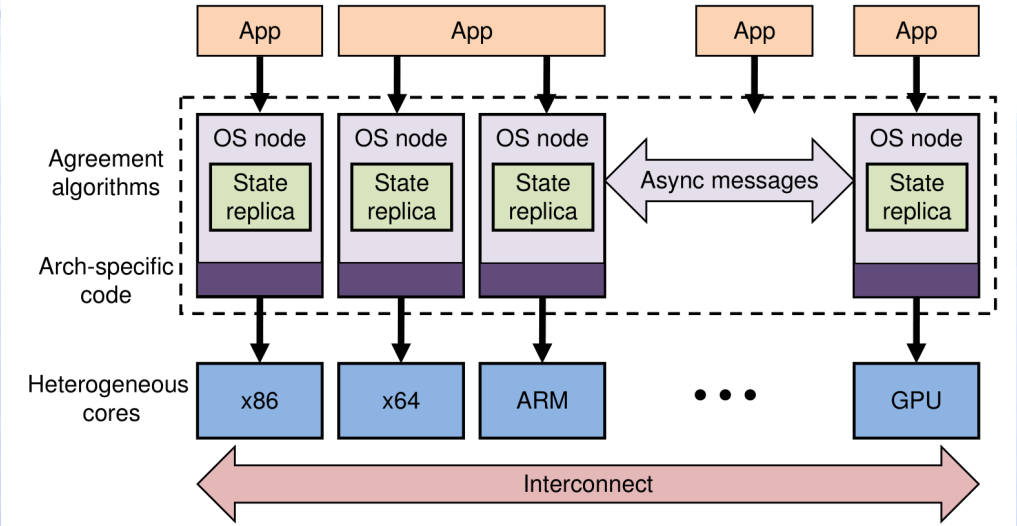
Перехватывает точки входа/выхода в/из ядра и контролирует всю память системы:

- Валидация загружаемых модулей
- Строгая политика Non-Executable $\oplus$ Writable

Параллелизм

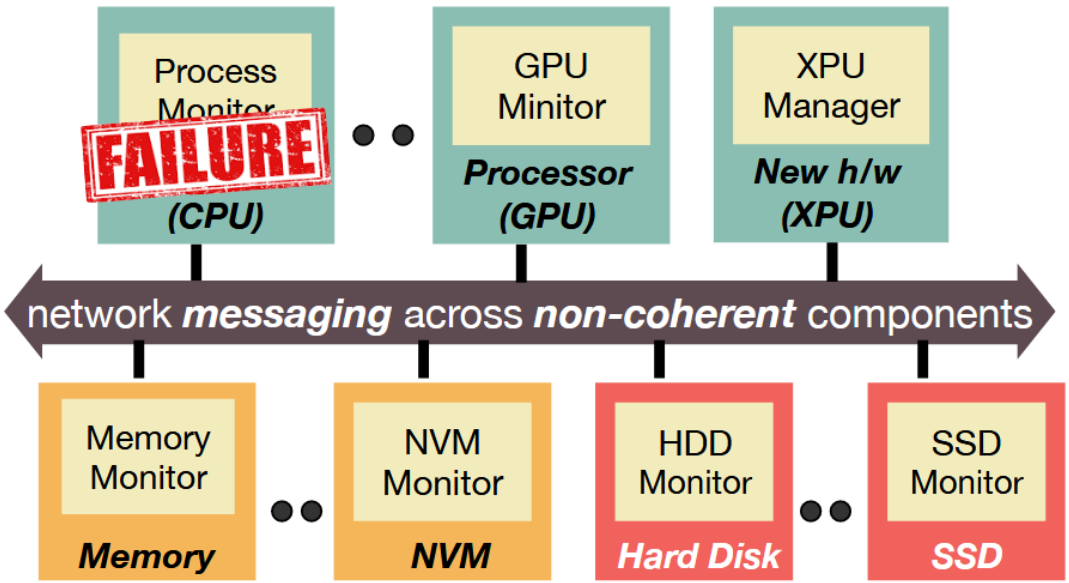
Corey (2008) предложила привязать сервисы ОС, а главное! связанные с этими сервисами данные к ядрам процессора. За счет этого была увеличена масштабируемость и производительность системы.

В ответ в 2009 был предложен Barrelfish – многоядерная операционная система организованная как сеть. Ядра ОС привязаны к ядрам процессора и используют только частную память. Все взаимодействие между компонентами системы сводится к передаче сообщений.



Параллелизм

В 2018 идея получила развитие в архитектуре Splitkernel. Инновационный подход (на самом деле нет) состоит в том чтобы разнести и специализировать ядра ОС непосредственно на аппаратные компоненты.

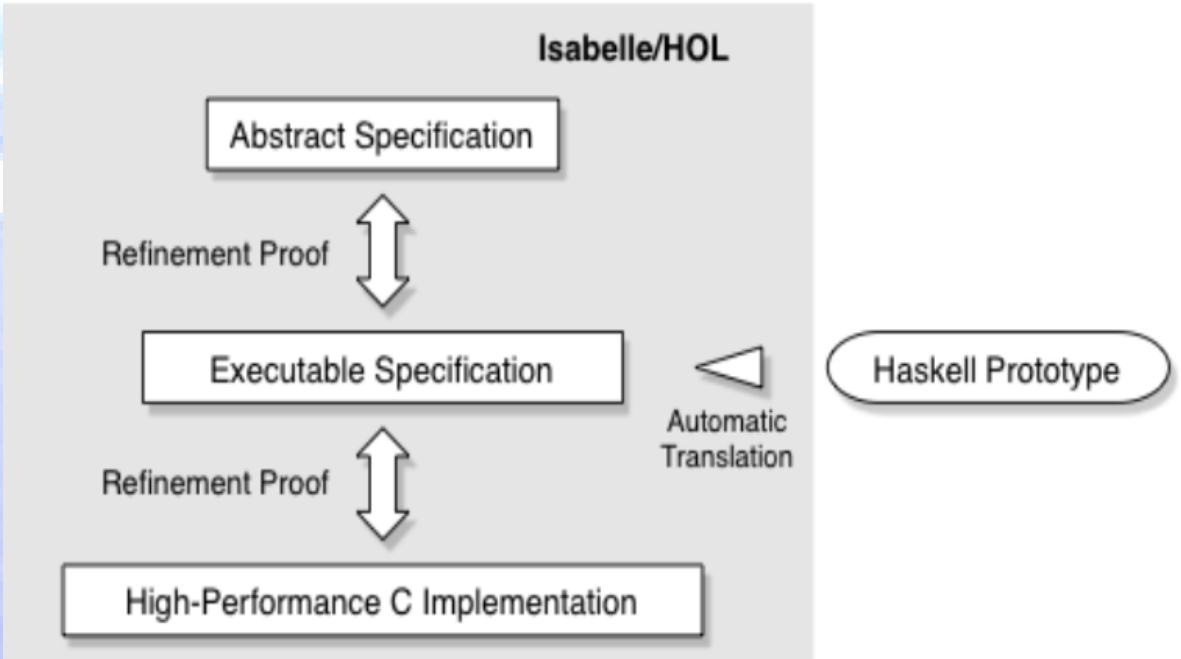


- Split OS functions into *monitors*
- Run each monitor at h/w device
- Network messaging across non-coherent components
- Distributed resource mgmt and failure handling

Надежность через верификацию

В 2009 году было представлено seL4 – первое полностью формально верифицированное микроядро. Минимализм ядра и ряд специальных трюков позволили добиться такого результата (язык C, Big Kernel Lock, и т.д.). Была поднята волна исследований в формальной ферификации.

Микроядра развивают в Германии, Швейцарии и США. Взрыв инетереса в последние годы в связи с Self-Driving Car хайпом.



Безопасность базирующаяся на SGX

В 2015 году был представлен ряд систем реализующих безопасность на основе SGX-анклавов.

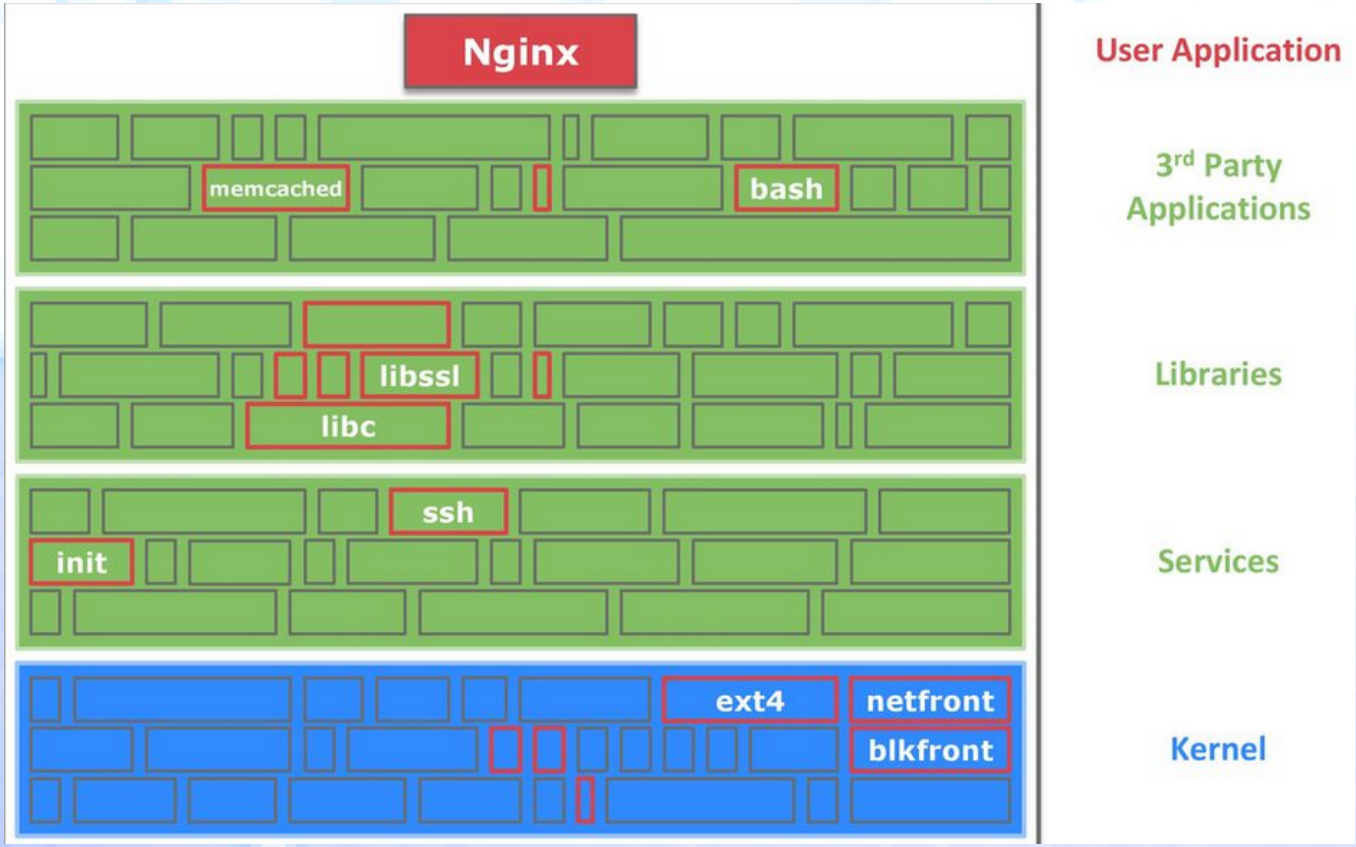
Идея состоит в том чтобы получить возможность выполнять доверенный код в среде недоверенной операционной системы.

Похожий эффект может быть достигнут с помощью использования гипервизора.

Требуется аппаратная поддержка от Intel.

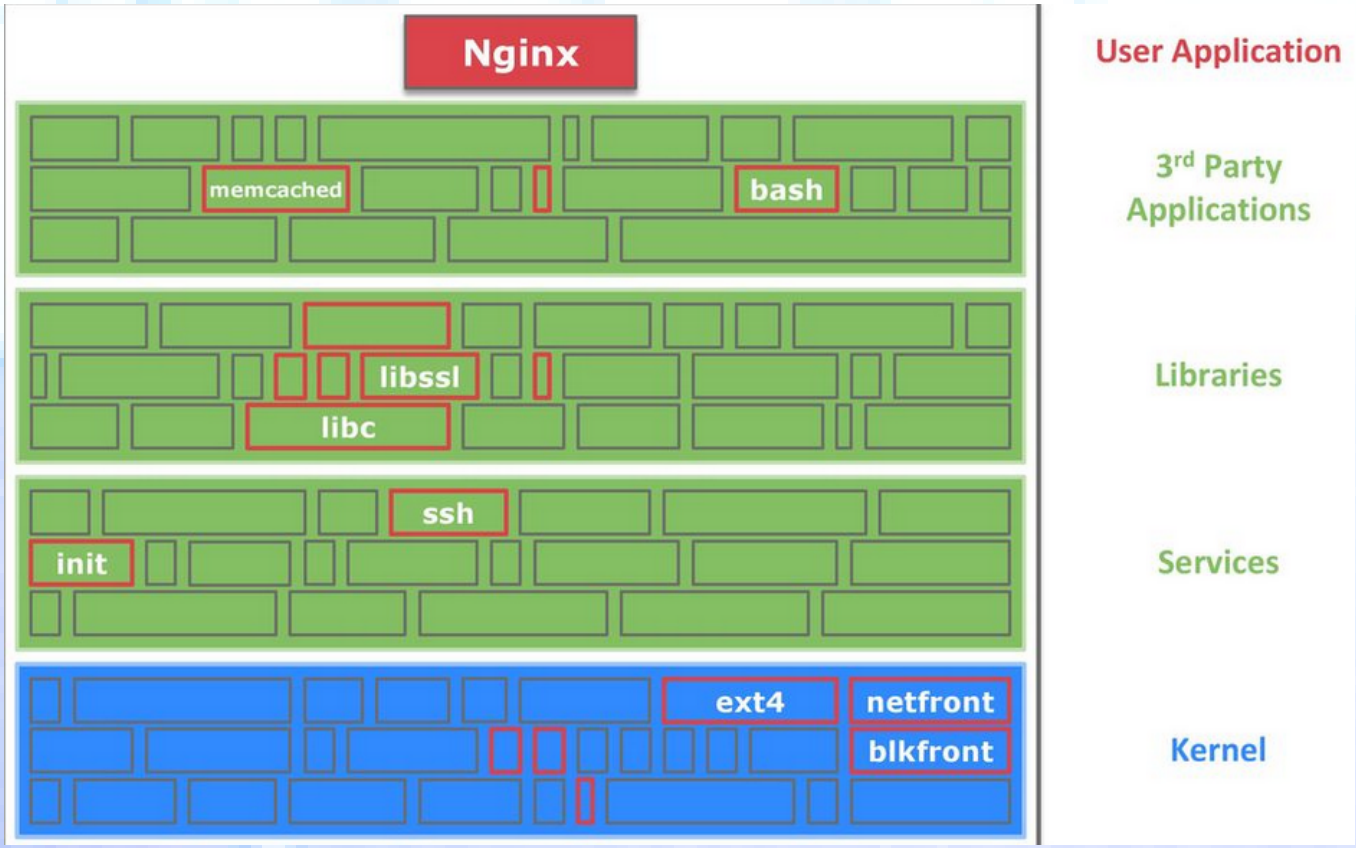
LibOS

LibOS представляют собой активно развивающуюся область и рассматриваются как альтернатива контейнерам.



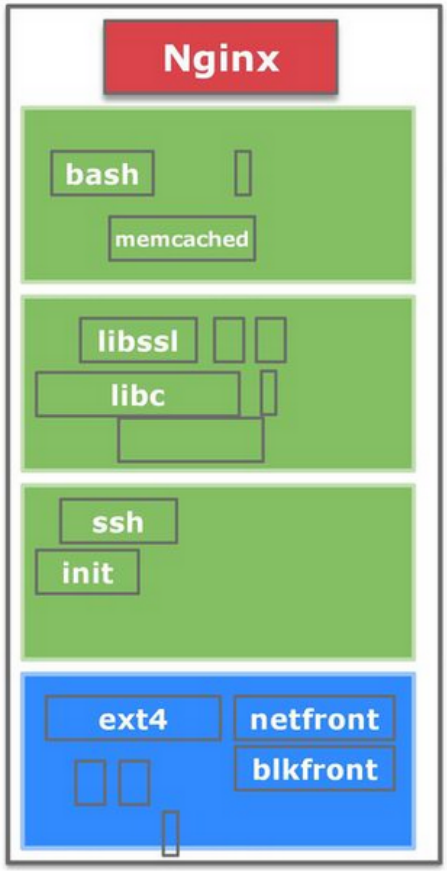
LibOS

LibOS представляют собой активно развивающуюся область и рассматриваются как альтернатива контейнерам.



LibOS

Основная идея заключается в соинтеграции всех компонентов программного стека в единый легковесный образ.

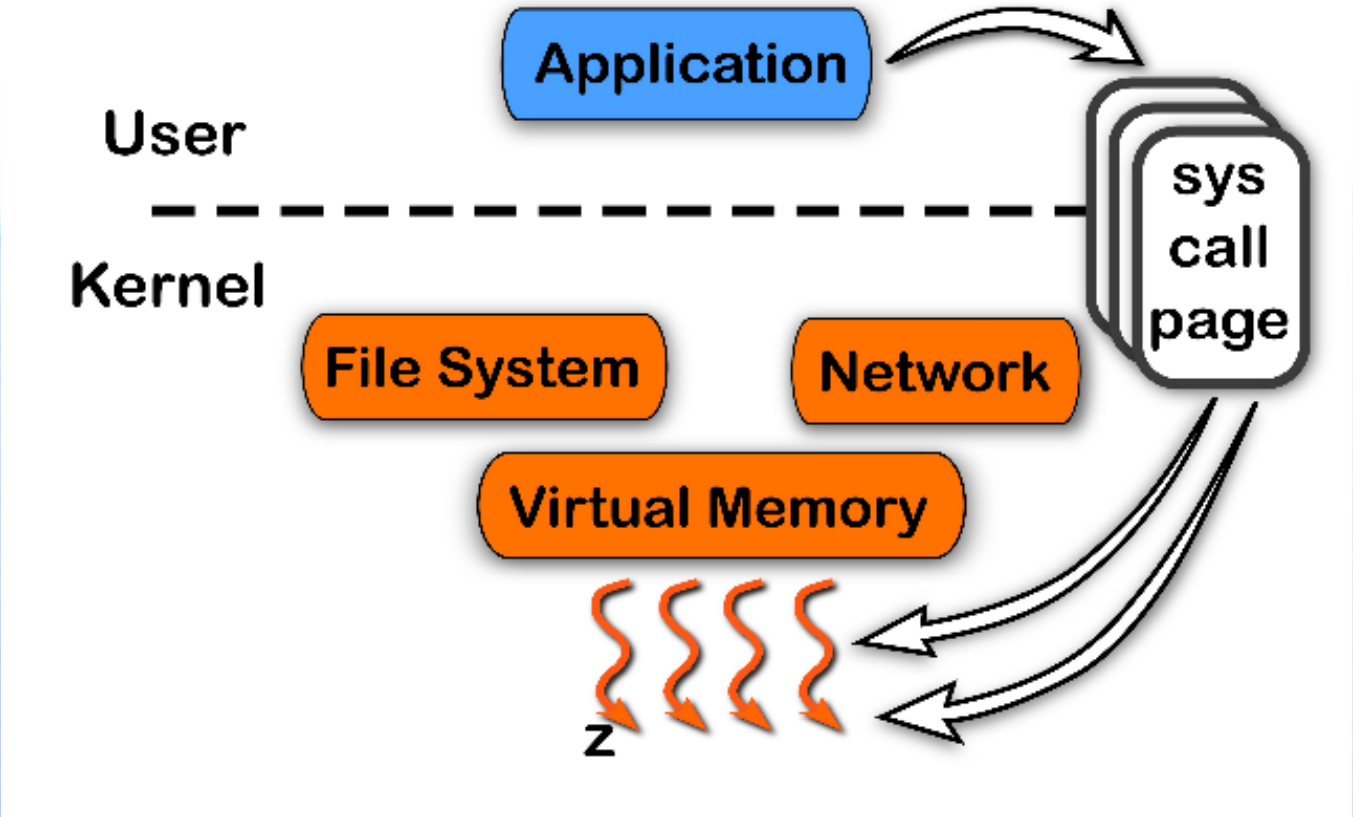


Tinyx VMs are also lightweight:

- Kernel: 1.5MB (compared to 8MB)
- Image size – 10-30MB (compared to 1GB).
- Boot: 200ms instead of 2s.

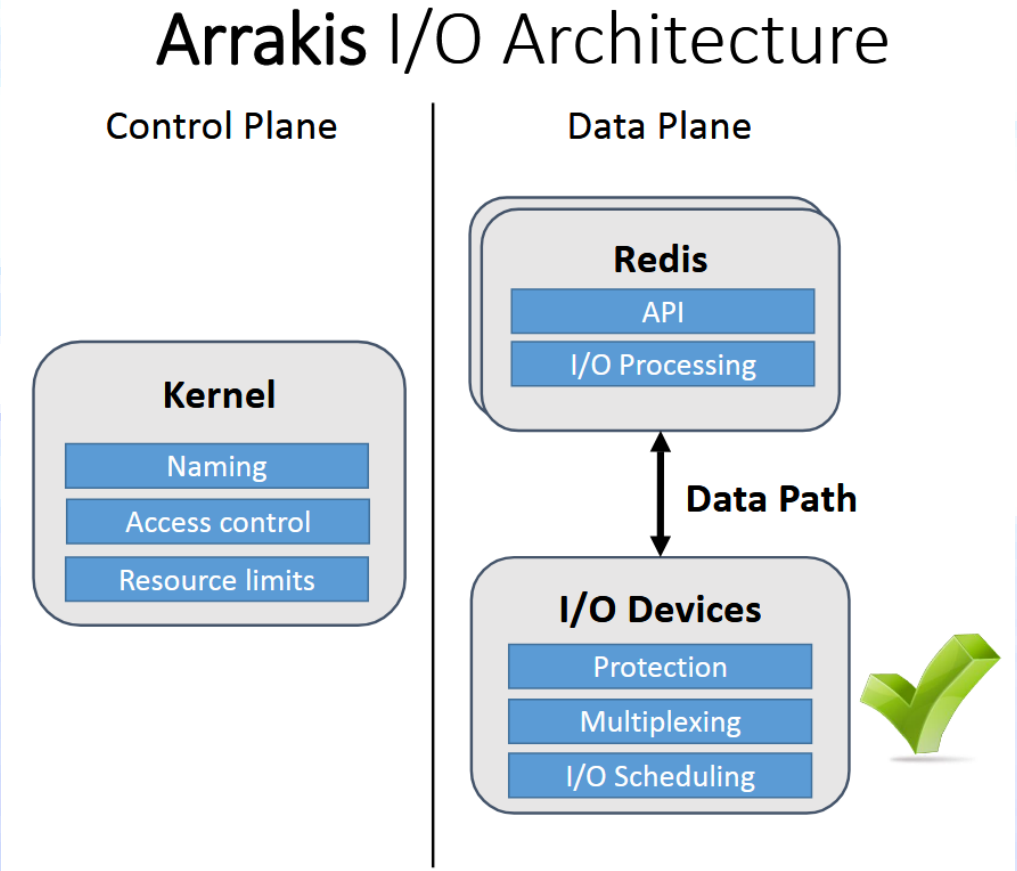
Производительность через многопоточность

В 2010 году был предложен механизм FlexSC. FlexSC реализует системные вызовы не через переключения режимов процессора (INT/SYSENTER/SYSCALL), а через механизм разделения памяти и HyperThreading. В отдельных случаях производительность увеличивалась до двух раз.

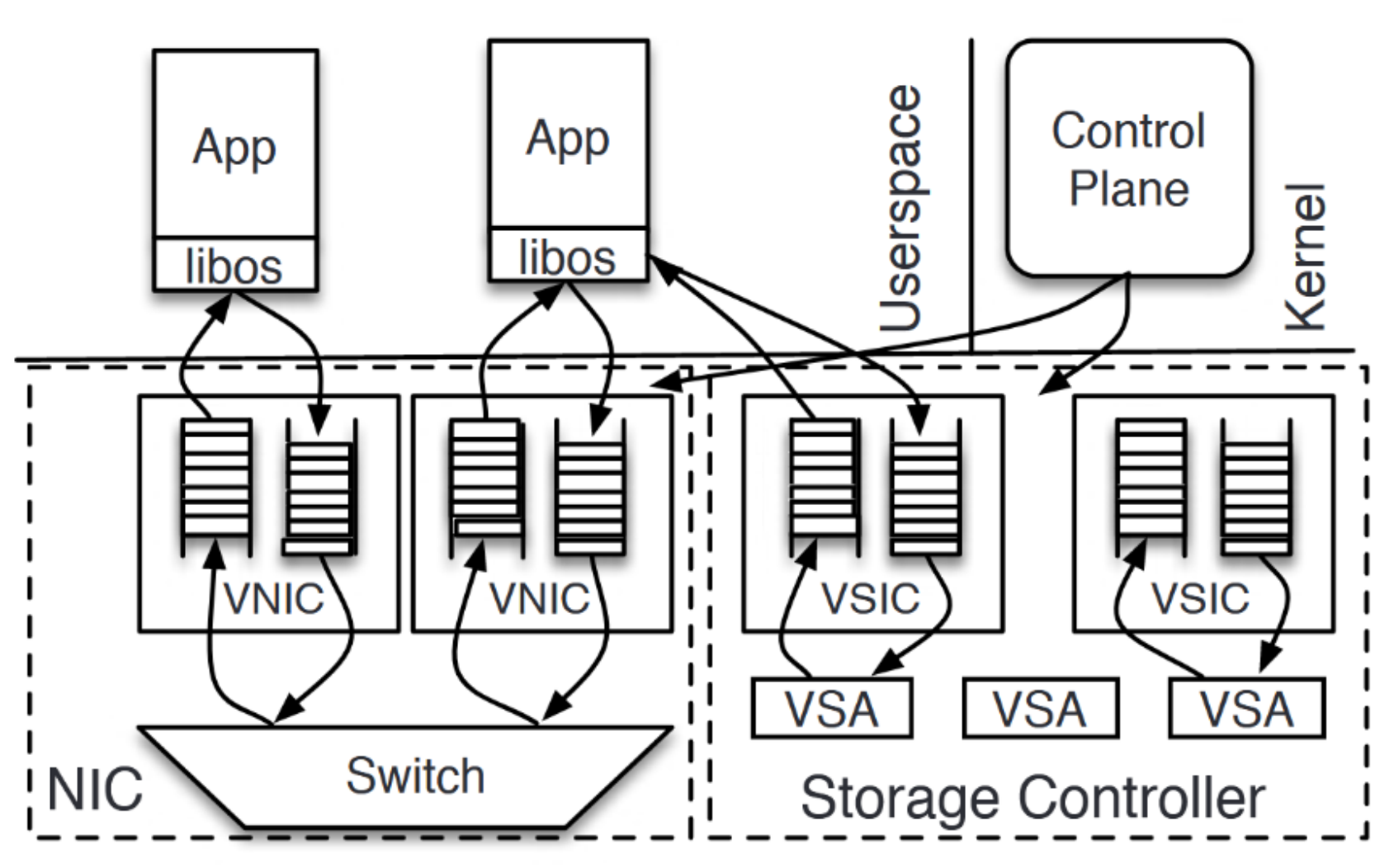


Производительность ввода-вывода

В 2014 году была предложена OS Arrakis. Основной идеей ОС является реализация прямого ввода вывода для двух основных стеков устройств – сети и жестких дисков.



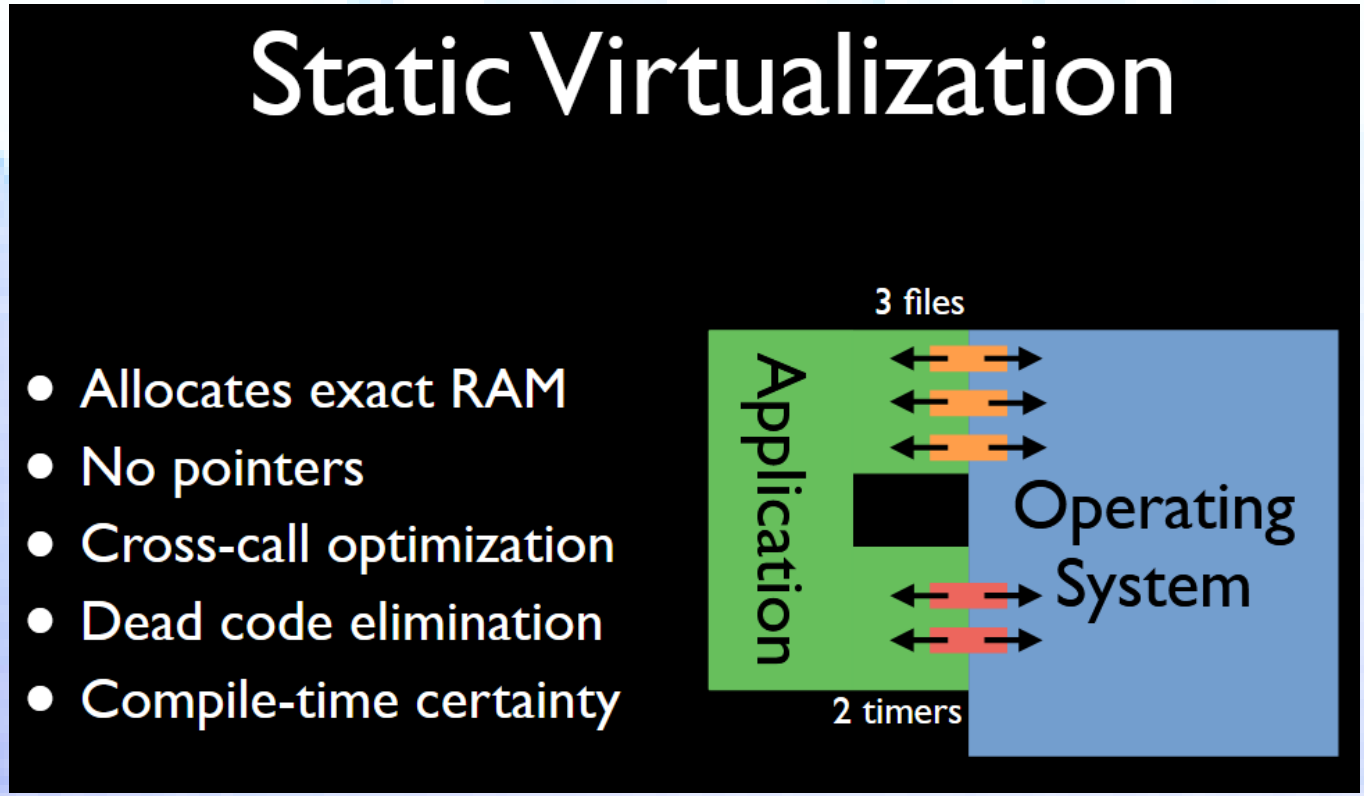
Производительность ввода-вывода



TinyOS, Contiki, Tock были разработаны специально для маломощных и энергоэффективных устройств подключенных к сетям.

Два основных принципа:

- Минимизация использования ресурсов системы
- Структурирование кода и интерфейсов



Использование новых языков программирования

В последние годы ученые обратили внимание на новые языки программирования и оценили их применимость для реализации операционных систем.

- Rust используется в Tock для написания приложений.
- Biscuit была написана на Go.

Why Rust?

- **Type and memory safe**
  - No buffer overflows, dangling pointers, type confusion…
- **Compile-time enforced type system**
  - No type artifacts at run time
- **No garbage collection**
  - Control over memory layout and execution
- **Runtime behavior similar to C**

Использование новых языков программирования

Go is a good choice:

- Easy to call asm
- Compiled to machine code w/good compiler
- Easy concurrency
- Easy static analysis
- GC

Pros:

- Reduction of bugs
- Simpler code

Cons:

- HLL safety tax
- GC CPU and memory overhead
- GC pause times

The HLL worked well for kernel development

Performance is paramount $\Rightarrow$ use C (up to 15%)

Minimize memory use $\Rightarrow$ use C ($\downarrow$ mem. budget, $\uparrow$ GC cost)

Safety is paramount $\Rightarrow$ use HLL (40 CVEs stopped)

Performance merely important $\Rightarrow$ use HLL (pay 15%, memory)