Портирование ОС Фантом на Genode OS Framework

Антон Антонов

О себе

Антонов Антон

- Студент магистратуры УИ
- Инженер лаборатории ИИ в промышленности
- Год назад защитил дипломную работу по теме доклада

План

1. Идея Фантом ОС
2. Процесс портирования и текущий статус
3. Выводы
4. Планы на будущее

1 - Идея Фантом ОС

Идеальная память

Все в одном:

- Быстрая
- Дешевая
- Энергонезависимая

Идеальная память

Тогда будет достаточно одного слоя виртуальной памяти где будет храниться все сразу

Включая состояние потоков исполнения

Но что это нам может дать?

Идеальная память

Более отказоустойчивое ПО с меньшими трудозатратами

Меньше (де-)инициализации

Меньше (де-)сериализации

Только одно хранилище

Больше фокуса на бизнес логику

Но...

Вряд ли такая идеальная память появится

Однако, перспектива разрабатывать ПО с меньшими силами звучит довольно интересно

“Идеальная память”

Одной из характеристик такой памяти является то...

Что все объекты в памяти хранятся одинаково и будут там находиться пока они нужны

Ортогональная персистентность

Иными словами,

Срок хранения объектов не зависит от их использования

Ортогональная персистентность

Персистентность – период времени во время которого объект существует в системе и его можно использовать

Ортогональная – не зависит от того для чего объект был предназначен

Обычно применяется к объектным системам

Ортогональная персистентность

Как следствие,

Система заботиться о перемещении данных между устройствами вместо программиста

Ортогональная персистентность

Концепт реализован в нескольких языках

PS-Algol, Nappier88

В нескольких операционных системах

KeyKOS, EROS, Coyotos, Grasshopper OS

И даже была попытка реализовать персистентный JVM — проект PJama

Ортогональная персистентность

Но есть ряд проблем:

- Обновления ПО
- Персистентные ошибки
- Связь с не персистентным миром

Ортогональная персистентность

Однако прошло довольно много времени

- Нам доступно намного более производительное железо
- Контекст разработки ПО изменился

Ортогональная персистентность

На чем можно проводить эксперименты и тестировать гипотезу ортогональной персистентности?

ОС Фантом

- Экспериментальная ортогонально персистентная ОС
- ОС общего назначения
- Исполняет только управляемый код
- Разрабатывалась Дмитрием Завалишиным
- Активно разрабатывалась с 2009 по 2011. Работы продолжались с 2016 по 2019

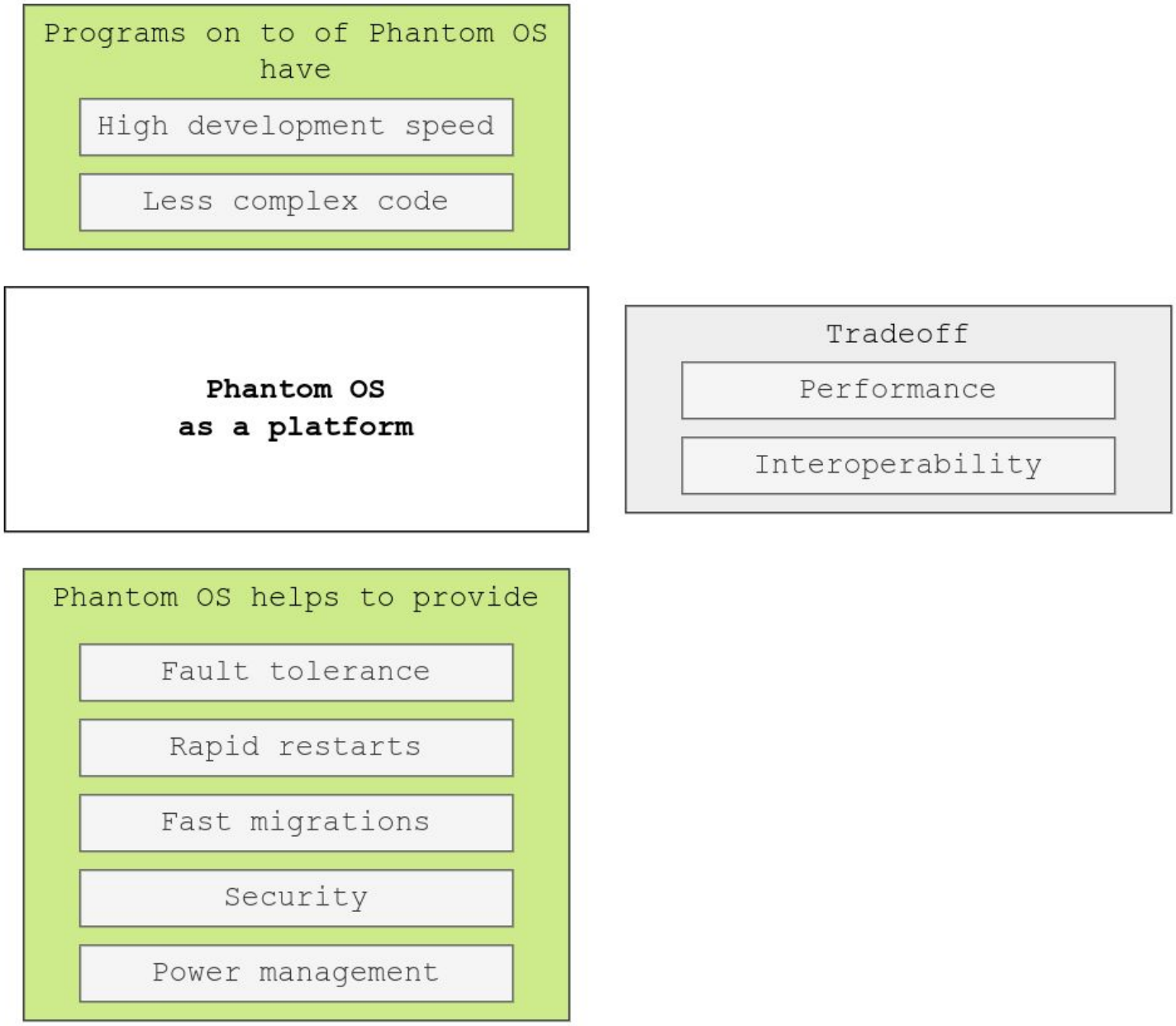
ОП в ОС Фантом

- Реализована благодаря механизму снапшотов
- В снапшот попадает не все, а только минимальное состояние для восстановления пользовательских программ
- Снапшоты производятся периодически
- Пользовательские программы запускаются в языковой виртуальной машине

Почему виртуальная машина?

- Способ поддержать больше высокоуровневых языков
- Абстрагируемся от API операционной системы который не должен быть персистентным
- Потенциально, нужен меньший объем информации для восстановления

- На данный момент поддерживает только собственную PVM похожую на JVM


Programs on to of Phantom OS have
High development speed
Less complex code
Phantom OS
as a platform
Tradeoff
Performance
Interoperability
Phantom OS helps to provide
Fault tolerance
Rapid restarts
Fast migrations
Security
Power management

2 - Процесс портирования и текущий статус

Цель портирования

Цель:

Адаптировать Фантом для реальных нагрузок

Для этого нужно улучшить его

- Стабильность
- Безопасность

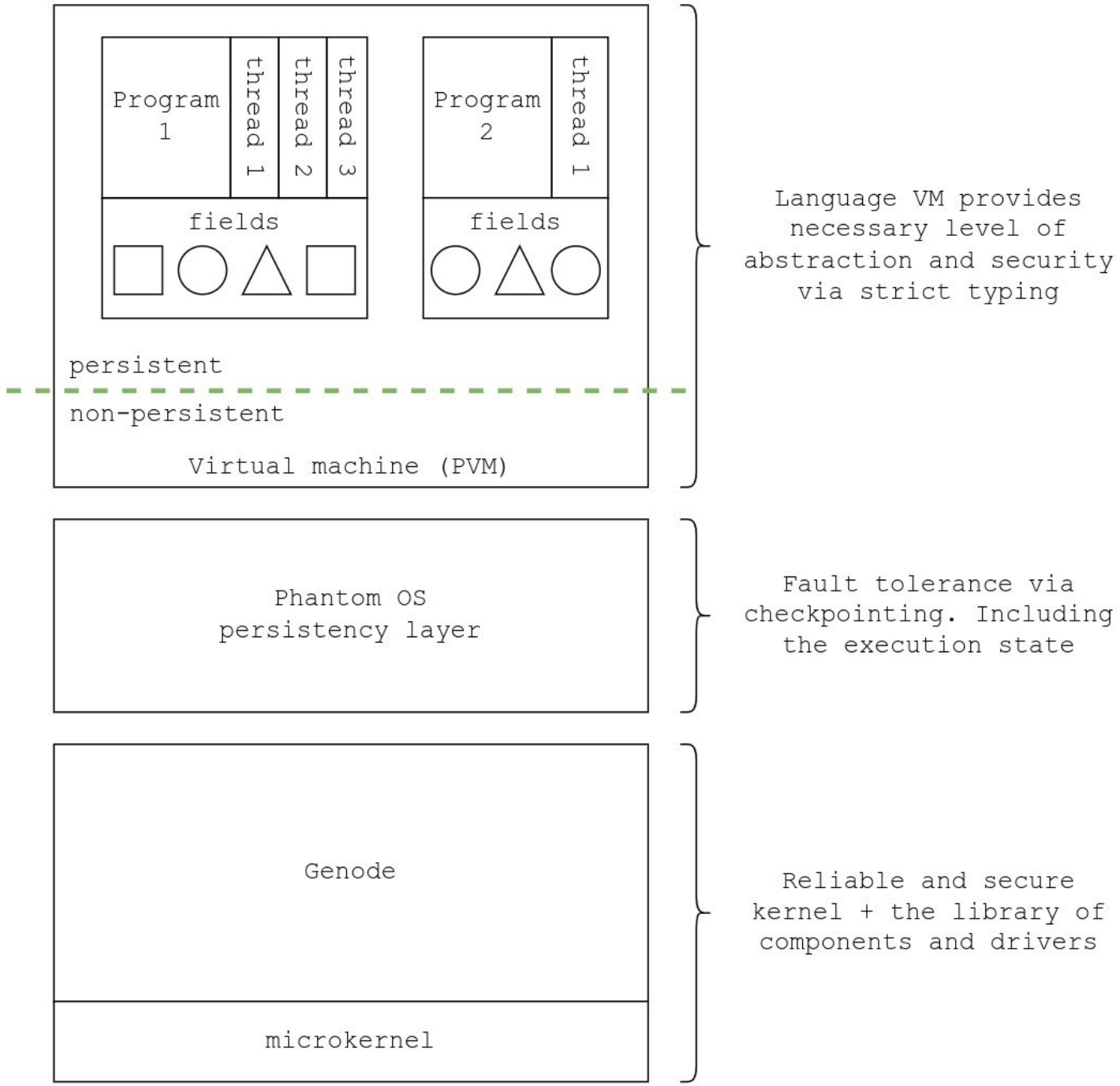
Цель портирования

Хорошее решение - заменить ядро на более протестированное и надежное

Ожидаемый результат: прототип который сможет запускать простые программы в персистентной среде (делать и восстанавливаться из снапшотов)

Genode OS Framework

- Microkernel based
- Application specific trusted computing base
- Trading and tracking of resources
- Provides basic services of OS as separate components
- “Lego constructor” for operating systems
- Supports multiple microkernels including seL4 and NOVA


Program 1
thread 1
thread 2
thread 3
Program 2
thread 1
fields
fields
persistent
non-persistent
Virtual machine (PVM)
Language VM provides necessary level of abstraction and security via strict typing
Phantom OS persistency layer
Fault tolerance via checkpointing. Including the execution state
Genode
microkernel
Reliable and secure kernel + the library of components and drivers

Проблемы

- Работает только на 32-битных системах
- Глобальный лок во время снапшота (хоть и недолгий)
- Драйвера интегрированы в ядро
- Много не протестированного кода написанного в разное время

3 этапа работ

1. Понять можно ли запустить PVM на Genode 64-бит
2. Найти и “вырезать” минимальный набор кода нужный для запуска PVM в персистентной среде
3. Понять какие низкоуровневые интерфейсы ему нужны и реализовать их в Genode

Disclaimer

Процесс портирования еще не завершен!

Этап 1

1. Интегрировать сборочные системы
2. Заменить странный libc на libc из Genode
3. Проверить работает ли standalone PVM на Genode на x86_64

...

PVM & libc

- В итоге получилось запустить PVM и все встроенные тесты успешно проходят
- Достаточно было заменить некоторые заголовки чтобы использовать libc из Genode
- Но пришлось писать отдельные сборочные скрипты для Genode так как сборочные системы были несовместимы

Этап 2

...

4. Определить минимальное подмножество модулей/функций которые нужны для полноценной работы механизма чекпоинтинга
5. Заменить нужные им интерфейсы на заглушки

...

Задача

- На входе имеем
- ~618k LoC (только C код) кодовой базы
- Не так много понимания о том как она работает
- Не так много времени и ресурсов

Надо как-то упрощать задачу

Граф вызовов

- Относительно быстро получилось построить граф вызовов и разметить компоненты на нем
- Использовался cflow и python
- А также jupyter notebook для удобства

Граф вызовов

Гипотеза: подавляющее большинство нужных вершин лежит в подграфе который можно построить обходом начиная с

- Вызовов исходящих из виртуальной машины

С условием прекращения в интерфейсах

- hardware abstraction layer
- внутренних вызовах phantom_*
- libc

Граф вызовов

- После получения подграфа файлы содержащие нужные функции были скопированы в отдельную директорию вместе с процедурами инициализации системы
- Большой лог с undefined reference’ами получилось относительно быстро разрешить вручную с помощью заглушек и копирования недостающих файлов

Граф вызовов (Итог)

Было: ~270k LoC

Стало: 10.7k LoC включая заглушки

Этап 3

...

6. По одному реализовывать адаптеры на Genode

7. Отладить механизм чекпоинтинга

...

Текущий статус

На данный момент реализовано большинство необходимых низкоуровневых интерфейсов включая интерфейсы:

- Работы с потоками (через pthread)
- Синхронизации (через pthread)
- Виртуальную память
- Физическую
- Драйвера диска

Текущий статус

- Юнит тесты для реализованных интерфейсов успешно проходят
- Инициализируется большая часть сервисов Фантом ОС
- Работаем над отладкой механизма снапшоттинга и переходом на новую версию Genode

3 - Выводы

Граф вызовов

- Граф вызовов может дать много инсайдов о том как работает проект
- Также в задачах “трансплантации” на его основе можно получить неплохое приближение подмножества кода нужного для переноса

Тестирование

- В среде где тяжело отследить что именно вызвало ошибку, тестирование может сэкономить довольно много времени
- Тесты также могут служить документацией

Планирование

- Можно как-то измерять прогресс
- Если тяжело дать точную оценку, лучше обновлять регулярно план и она будет постепенно улучшаться
- Если фреймворк который вы используете недостаточно зрел, то желательно закладывать время на возможные проблемы

4 - Планы на будущее

Обозримое будущее

1. Получить прототип который сможет запускать программы в персистентной среде
2. Добавить сеть
3. Включить графическую подсистему
4. Доработать и интегрировать с Genode систему драйверов
5. Интегрировать WebAssembly рантайм в Фантом ОС

Не очень обозримое будущее

- Взаимозаменяемые стратегии для создания снапшотов
- Общая персистентная память между несколькими системами
- Persistency engine как универсальный компонент
- Лайв миграции
- Запуск на конкретном микроядре без Genode

Спасибо за внимание!

Email для связи:

a.antonov@innopolis.ru

Github:

S7rizh/phantomuserland

Ресурсы

- M. Atkinson and R. Morrison, “Orthogonally persistent object systems,” VLDB Journal, 1995
- G. N. K. Alan Dearle and R. Morrison, “Orthogonal persistence revisited”, presented at the International Conference on Object Databases, 2009