наличием промежуточных элементов в схеме вложенной виртуализации, необходимых для ее запуска.

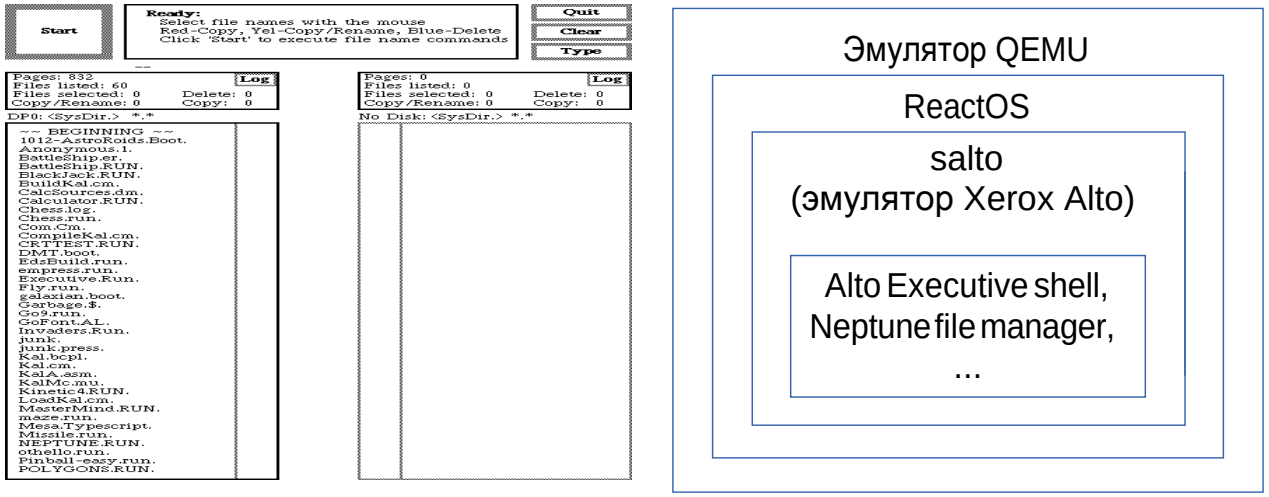

Рисунок 4 – Рабочий экран и схема запуска программ платформы Xerox Alto

Список цитированных источников

1. Костюк, Д.А. Особенности использования виртуализованных окружений, внедренных в презентационные материалы // Восьмая конференция «Свободное программное обеспечение высшей школе»: тез. докл. / Переславль, 26-27 января 2013 года. ─ М.: Альт Линукс, 2013. – С. 83–86.

2. Костюк, Д.А. Построение прозрачных виртуализованных окружений для изоляции уязвимых программных систем / Д.А. Костюк, С.С. Дереченник // Комплексная защита информации: матер. XVI научно-практич. конф., Гродно, 17-20 мая 2011 г. Гродно, 2011. – С. 209-212.

3. Костюк, Д.А. Применение виртуальных машин в составе иллюстрированных обзоров истории программного обеспечения / Д.А. Костюк, П.Н. Луцюк // Девятая конференция «Свободное программное обеспечение в высшей школе»: тезисы докладов / Переславль, 25-26 января 2014 года. ─ М.: Альт Линукс, 2014. – С. 19-23.

УДК 004.514.62

Власенко С.С., Желудок В.А.
Научный руководитель: к.т.н., доцент Костюк Д.А.

АРХИТЕКТУРА НАГЛЯДНЫХ МАТЕРИАЛОВ ПО ИСТОРИИ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА НА БАЗЕ ВИРТУАЛИЗАЦИИ

В данной работе рассматривается подход к использованию виртуализации для повышения наглядности учебных материалов за счет встраивания окон виртуальных машин (ВМ) в поясняющие материалы по истории графических операционных систем. Медийная насыщенность демонстрационных материалов и электронной документации обычно ограничивается копиями экранов и, возможно, вставками анимационных фрагментов. «Живая» демонстрация обеспечивается частым переключением между окном, отображающим слайды или страницы документа, и окнами демонстрируемых программ. При этом для более простого развертывания демонстрируемое ПО может помещаться в контейнер ВМ. Импортируемое виртуализованное окружение может содержать любую готовую конфигурацию системного и прикладного программного обеспечения, а механизм снимков (snapshots) позволяет быстро выполнить откат ВМ к нужному моменту работы для повторной демонстрации ключевых элементов либо пропуска длительных процедур [1].

Современные ВМ способны работать в «невидимом» (headless) режиме, предоставляя вместо графического окна программу-сервер для удаленного доступа. Клиентское же программное обеспечение является сравнительно несложным, а некоторые клиенты написаны на скриптовых языках либо в виде легко встраиваемых апплетов. Благодаря этому факту и представляется возможным встроить окна ВМ непосредственно в построенные на основе веб-технологий материалы учебных курсов, обеспечив тем самым их более полную интеграцию.

Для построения архитектуры такого интегрированного решения требуется обосновать выбор следующих компонент: используемой ВМ, обладающего достаточной функциональностью HTML-фреймворка, а также совместимых с веб-технологиями средств удаленного доступа к рабочему столу виртуализованной операционной системы. Рассмотрим далее варианты, потенциально отвечающие поставленной цели.

HTML5-шаблоны, работающие на JavaScript с помощью 2D/3D-переходов CSS3 и анимации, все чаще используются в последнее время для построения интерактивных демонстрационных материалов, тесня классические решения наподобие Microsoft PowerPoint. В первую очередь это объясняется богатой медийной насыщенностью CSS (Cascading Style Sheet) – технологии, которая предоставляет возможность изменения цвета, шрифта, размера, фона, границы и т.д. в HTML-документе.

Рассмотрим несколько наиболее популярных HTML5/CSS3 фреймворков, с помощью которых можно создать презентационные материалы:

- Reveal.js – позволяет создавать визуально привлекательные слайды, связанные между собой горизонтальным и/или вертикальным направлением навигации; также легко изменяется CSS для добавления своих собственных эффектов; реализован большой выбор 3D-переходов между слайдами;
- Slides Presentation with HTML5 – модульный фреймворк, который использует слайды в виде обычного HTML в сочетании с макетами и переходами между слайдами в CSS, и небольшой MVC-Framework на JavaScript для их показа;
- Impress.js – один из наиболее сложных фреймворков, в котором слайды расположены в трехмерном пространстве, а их содержимое определено в виде div-элементов с атрибутами управления x, y, r (координатами положения и вращения); при создании материалов требуется мысленно визуализировать их трехмерную схему и определять значения координат.

Наиболее подходящая ВМ для данного программного проекта – QEMU. Пользователь может использовать его с гостевыми платформами x86, x86-64, SPARC, PowerPC и с другими процессорами. Эксперименты, представленные в [1], показали, что QEMU с модулем аппаратной акселерации KVM, доступным при использовании ОС GNU/Linux в качестве хост-системы, позволяет решить задачу подобного класса на среднем мобильном либо настольном процессоре, имеющем аппаратную поддержку виртуализации. Такие низкие требования вызваны тем, что в каждый момент предполагается активное использование только ВМ, отображаемой на активном экране. Требуемый объем ОЗУ более критичен и определяется нуждами конкретных гостевых ОС. Однако при большом числе однотипных гостевых систем он может быть уменьшен использованием модуля KVM в связке с технологией Kernel Samepage Merging (KSM), позволяющей объединять одинаковые страницы памяти для различных приложений (одно из типовых применений – как раз сервера виртуальных машин). KSM также реализован только при использовании

GNU/Linux в качестве хост-системы; однако он существенно снижает потребление памяти.

Доступ к изображению графического рабочего стола ВМ будет удобнее всего осуществлять по протоколу VNC. VNC или Virtual Network Computing – система удаленного доступа к рабочему столу компьютера, использующая протокол RFB (Remote FrameBuffer, удаленный кадровый буфер). Управление осуществляется путем передачи нажатий клавиш на клавиатуре и движений мыши с одного компьютера на другой и ретрансляции содержимого экрана через компьютерную сеть. VNC-клиент, называемый также иногда VNC viewer, запущенный на одной операционной системе, может подключаться к VNC-серверу, работающему на любой другой ОС. Существуют реализации клиентской и серверной части практически для всех операционных систем, в том числе и для Java. К одному VNC-серверу одновременно могут подключаться множественные клиенты. Наиболее популярные способы использования VNC – удаленная техническая поддержка и доступ к рабочему компьютеру из дома, а также доступ к графическим окружениям серверов.

Так как протокол RFB для удаленного доступа к графическому рабочему столу работает на уровне кадрового буфера, то его можно применять для графических оконных систем, например X Window System, Windows, Quartz Compositor. По умолчанию RFB использует диапазон TCP-портов с 5900 до 5906, однако порты могут быть изменены.

Для целей данного проекта требуется VNC-клиент, который мог бы быть встроен в веб-браузер для показа средствами выбранного HTML-фреймворка. На текущий момент существует несколько VNC-клиентов, обладающих требуемым функционалом.

- Guacamole – средство просмотра VNC, построенное на связке HTML5 и асинхронного JavaScript (AJAX), использующее прокси-сервер, написанный на Java. На стороне сервера Guacamole требуется контейнер сервлетов (например, Apache Tomcat), в то время как на стороне клиента достаточно браузера, поддерживающего тег canvas HTML5 и AJAX. Текущая версия обладает практически такой же отзывчивостью интерфейса, как нативные VNC-клиенты.
- NoVNC – другой проект, который также представляет собой полнофункциональную реализацию VNC-клиента на Javascript/HTML5 (в отличие Guacamole, где протокол VNC реализован в прокси-сервере). Однако noVNC точно так же ограничен невозможностью для Javascript создавать простые TCP-соединения; noVNC использует для связи с сервером веб-сокеты. Для работы в noVNC включен универсальные прокси из WebSockets в TCP, написанные на python и C), один из которых должен быть запущен на сервере либо на клиенте и которые не требуют для своей работы внешних зависимостей.
- TightVNC.com является чистым веб-решением, однаком совместим только с VNC-сервером TightVNC (сервер TightVNC может обрабатывать входящие веб-запросы). В результате клиент нельзя использовать с любым другим VNC-сервером, а ни одна система виртуализации не обладает функционалом TightVNC.

Как видно, ни одно из имеющихся решений не является на 100% чистым веб-приложением, способным напрямую соединяться с чистым VNC-сервером. Главной проблемой является то, что браузеры не могут выполнять простые соединения. Ближайший тип соединений, доступных веб-браузеру – стандарт WebSockets, неполно описывающий построенный на фреймах протокол, требующий HTTP-подобной синхронизации [1, 2].

Таким образом, по результатам обзора можно сформулировать следующий набор компонент для «живой» демонстрации истории операционных систем:

- ВМ QEMU с аппаратной поддержкой виртуализации;
- GNU/Linux в качестве операционной системы, обеспечивающей максимальную производительность гостевых приложений в QEMU;
- VNC-клиент noVNC, написанный на JavaScript и HTML5, а также WebSockets-proxy;
- JavaScript-фреймворк Reveal.js для постраничного показа информационных материалов и навигации между ними.

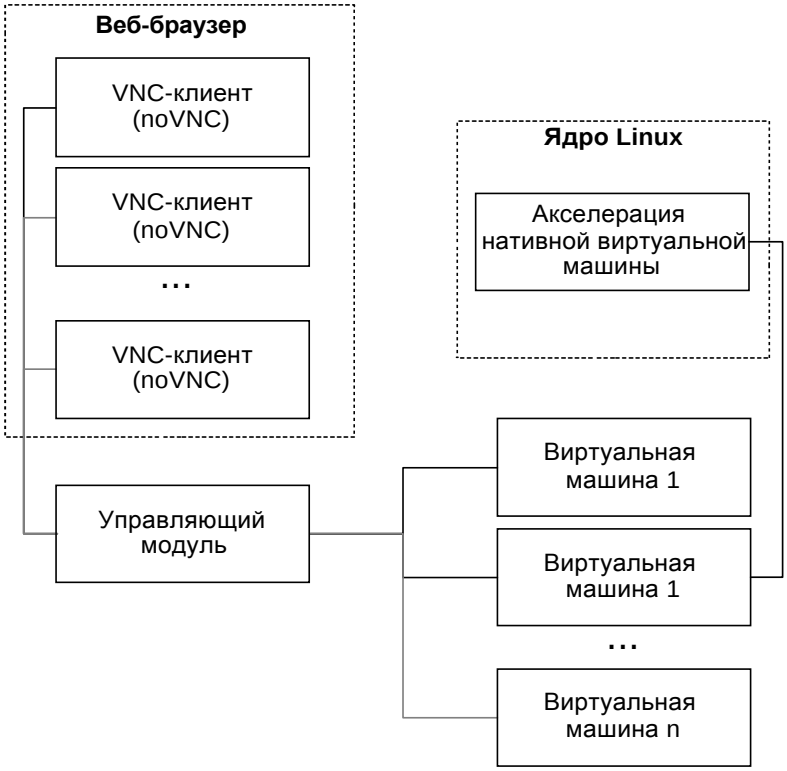

Рисунок 1** – **Архитектура системы

Архитектура системы, построенной на основе выбранных компонент, представлена на рис. 1. Рассмотрим подробнее взаимодействие ее элементов.

VNC-клиент noVNC обеспечивает ретрансляцию содержимого экрана между хост-системой и гостевой ОС внутри ВМ. Входные данные клиента – передаваемые веб-браузером нажатия клавиш клавиатуры и события мыши, с помощью которых пользователь посылает команды гостевому графическому интерфейсу. Выходные данные – изображение графического интерфейса.

Управляющий модуль запускается пользователем для старта всей системы. Его выходные данные – параметры командной строки, которые передаются всем ВМ, а также веб-браузеру. Например, параметры для ВМ – это образ диска, размер памяти, специфические устройства, сетевой порт, с которым будет взаимодействовать VNC-клиент.

ВМ используется как полностью изолированный контейнер для хранения образов операционных систем. Входные данные – параметры запуска ВМ, поступающие от управляющего модуля, образы виртуальных систем на диске, а так же события мыши и клавиатуры, поступающие от VNC-клиента через протокол VNC. Входные данные – изображение экрана, передаваемое по протоколу VNC.

Наконец, модуль акселерации ускоряет работу ВМ, если архитектура эмулируемого процессора совпадает с реальной архитектурой. Соответственно, его входные данные –

машинные команды, а выходные данные – результаты их выполнения.

Компоновка материала, включающего встроенные ВМ, предполагает поэкранную организацию, где каждый экран соответствует определенной графической ОС или ее значительной в историческом плане версии. Компоновка экрана представлена на рис. 2.

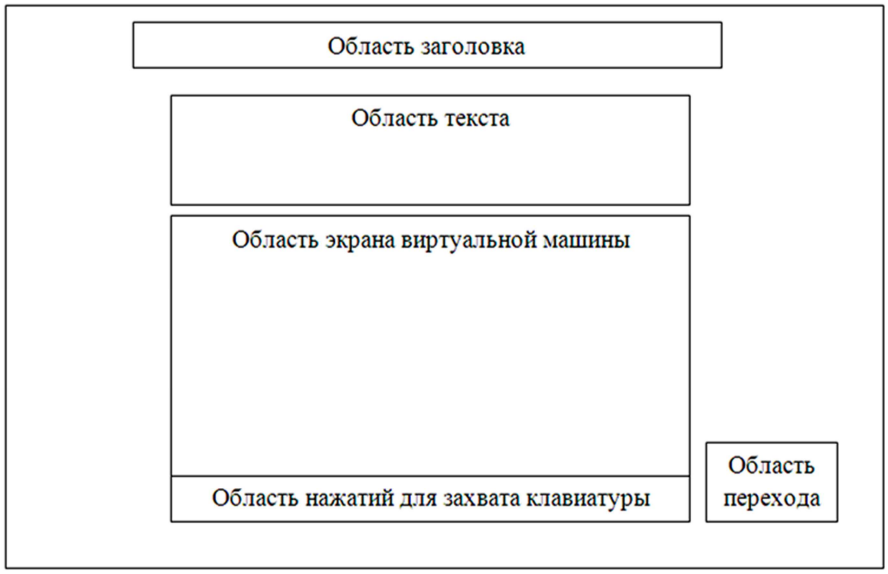

Рисунок 2 – Компоновка материала

Помимо области экрана ВМ, обновляемой noVNC, предусмотрен заголовок, показывающий год и название ОС, область текста с краткой исторической справкой, и служебные элементы навигации: область перехода с экранными кнопками для навигации по последовательности экранов и область захвата клавиатуры, переключающая на ВМ фокус клавиатурного ввода.

Список цитированных источников

1. Костюк, Д.А. Особенности использования виртуализованных окружений, внедренных в презентационные материалы // Восьмая конференция «Свободное программное обеспечение высшей школе»: тез. докл. / Переславль, 26-27 января 2013 года. – М.: Альт Линукс, 2013. – С. 83-86.

2. Костюк, Д.А. Применение виртуальных машин в составе иллюстрированных обзоров истории программного обеспечения / Д.А. Костюк, П.Н. Луцюк // Девятая конференция «Свободное программное обеспечение в высшей школе»: тезисы докладов / Переславль, 25-26 января 2014 года. – М.: Альт Линукс, 2014. – С. 19-23.

УДК 539.23; 539.216.1

Войтович А.Г., Балабанович А.Н.
Научный руководитель: ст. преподаватель Чугунов С.В.

МОДЕЛИРОВАНИЕ РАСПРЕДЕЛЕНИЯ ЭЛЕКТРОМАГНИТНОГО ПОЛЯ В ДИЭЛЕКТРИЧЕСКИХ МИКРОСТРУКТУРАХ

Данная работа направлена на исследование распределения электромагнитного поля, как за пределами диэлектрических микроструктур, так и внутри них, применяя различные методики моделирования.

Целью данной работы является моделирование узконаправленного фотонного пучка