Рисунок 4 – Часть выборки из множетства рукописных символов MNIST

Была выбрана нейронная сеть с 5 слоями размерностью 784-500-500-2000-10. Входной слой имеет линейную функцию активации, скрытые слои – сигмоидную, выходной – softmax функцию активации. Предобучение каждого из слоёв ограничено 200 эпохами. После этого сеть обучается алгоритмом обратного распространения ошибки (backpropagation) и проверяется на множестве из 10000 картинок.

Было проведено сравнение скорости обучения с предобучением и без него, а также обобщающая способность. Результаты обучения приведены в таблице 1 (было проведено несколько прогонов):

Таблица 1 – Результаты сравнения обучения DBN и обычной многослойной нейросети

Back Propagation с предобучением		Back Propagation без предобучения	
Количество эпох	Обобщающая способность, %	Количество эпох	Обобщающая способность, %
46	91.2	853	90.85
50	91.8	1235	89,30
48	91.5	1196	89,75

Как видно из результатов, предобучение дало значительный выигрыш.

Список цитированных источников

1. Hinton, G.E., Osindero, S., Teh, Y. A fast learning algorithm for deep belief nets. Neural Computation, 18, 1527-1554 (2006)

2. Hinton, G. Training products of experts by minimizing contrastive divergence. Neural Computation, 14, 1771-1800 (2002).

3. Hinton, G., Salakhutdinov, R. Reducing the dimensionality of data with neural networks. Science, 313 (5786), 504-507 (2006).

4. Hinton, G.E. A practical guide to training restricted Boltzmann machines.(Tech. Rep. 2010-000). Toronto: Machine Learning Group, University of Toronto (2010)

5.Golovko V., Kroshchanka F., Rubanau U., Jankowski S. A Learning Technique for Deep Belief Neural Networks. Lecture Notes in Computer Science, Springer, 2014 (in press).

УДК 004.514.62

Власенко С.С., Желудок В.А.

Научный руководитель: к.т.н., доцент Костюк Д.А.

ИСПОЛЬЗОВАНИЕ ВЛОЖЕННОЙ ВИРТУАЛИЗАЦИИ В ЭЛЕКТРОННОЙ ВЫСТАВКЕ ИСТОРИИ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Быстро растущие потребности в средствах автоматизации интеллектуального труда привели к стремительному развитию компьютерной техники и программного обеспечения, к превращению вычислительных машин в главный инструмент во многих областях человеческой деятельности. На их основе создаются сети обмена информацией, что в свою очередь, дает новые импульсы для развития научно-технического прогресса.

Хотя сейчас для того чтобы работать с новыми информационными технологиями и не обязательно знать историю их возникновения и развития, однако, как говорил великий немецкий философ Гегель, без истории предмета нет теории предмета. Создание теоретических постулатов и воплощающих их практических решений всегда начинается с изучения предыдущих достижений, их эволюции. Специалист, который формулирует либо применяет современную теорию, не зная ее истории, рискует лично повторять ошибки предшественников одну за другой.

Поэтому тема данной работы актуальна и может представлять интерес как для широкого круга общественности, так и для специалистов в области IT. Идея заключается в том, чтобы, благодаря изрядной производительности современных ноутбуков и десктопов, заменить копии экранов (screenshots) работающей программой на перенаправление вывода из окна виртуальной машины [1].

В силу необходимости универсального совмещения на одном компьютере множества виртуальных машин с операционными системами, создававшимися для самых разных платформ, в качестве системы виртуализации выбран эмулятор QEMU. Это программа с открытым исходным кодом для эмуляции различного аппаратного обеспечения, включая процессоры Intel x86, 80386, 80486, Pentium, Pentium Pro, AMD64 и другие x86-совместимые процессоры, а также PowerPC, ARM, MIPS, SPARC, SPARC64, m68k. QEMU является многоплатформенным приложением: работает на Syllable, FreeBSD, FreeDOS, Linux, Windows 9x, Windows 2000, Mac OS X, QNX, Android. Важным достоинством QEMU является также поддержка механизма мгновенных снимков (снапшотов, от англ. snapshots), позволяющая при старте виртуальной машины избежать процесса загрузки гостевой ОС, восстановив вместо этого ранее сохраненный образ ее оперативной памяти и регистров [2].

Многие устаревшие ОС показали свою работоспособность в качестве гостевой системы внутри QEMU. В этот список входят настольные версии Windows 1.x, 2.x, 3.x и 95, GEM от Digital Research, GEOS от Berkeley Softworks, IMB OS/2. Продемонстрировали свою совместимость и ряд мобильных ОС: Pen Windows, Maemo, Android, WebOS (не в последнюю очередь благодаря тому, что эмулятор на базе QEMU часто включался производителями в комплект разработчика).

Однако эксперименты с рядом ОС показали, что несмотря на возможность эмуляции множества процессоров, степень поддержки QEMU внутренних и внешних периферийных устройств, характерных для устаревших платформ, бывает недостаточной. В то же время для поддержки практически всех устаревших Intel-несовместимых ОС, которые не могут быть запущены непосредственно в QEMU, либо разработаны свободные эмуляторы, либо имеются эмуляторы из комплекта разработчика (SDK). Первый вариант более характерен для ОС настольных компьютеров, благодаря чему становится возможным включение в электронную выставку таких ОС, как Xerox Alto, Amiga, RiscOS, Apple Lisa, MacOS версий 1.x, 7.x. Коммерческие эмуляторы ОС настольных компьютеров редки и обычно принадлежат производителю самой ОС, как в случае графической оболочки Xerox GlobalView. В случае ОС мобильных устройств наоборот преобладают эмуляторы из состава SDK: Psion EPOC16 и EPOC32, PalmOS, Magic Cap, Windows CE. Исключением из списка является свободный эмулятор Open Einstein, позволяющий запускать NewtonOS [3].

Однако перечисленные эмуляторы не поддерживают снапшоты, а кроме того (в отличие от QEMU) не могут прозрачно обеспечивать сеанс работы с ВМ по сетевому протоколу VNC. В результате запуск значительной части демонстрируемых ОС осуществляется по схеме вложенной виртуализации (рис. 1), где QEMU играет роль внешнего контейнера.

В схеме вложенной виртуализации помимо эмулятора QEMU демонстрируемой ОС и вложенного эмулятора присутствует еще один дополнительный компонент – сервисная гостевая ОС, необходимая для запуска вложенного эмулятора. В каждом случае на ее выбор влияют требование минимального потребления памяти, возможность использо-

вания циклов бездействия процессора, а также поддержка шины USB для возможности эмуляции позиционирования в абсолютных координатах. Последнее требование важно для комфортного управления мышью в виртуальной машине [1].

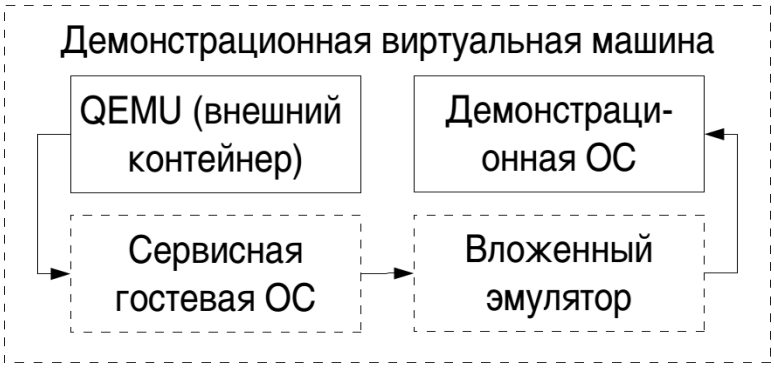

Рисунок 1 – Вложенная виртуализация

Таким образом, при организации электронной выставки по истории графических ОС возможны три типовых варианта виртуализации:

1. Запуск демонстрируемой ОС в эмуляторе QEMU.
2. Использование стороннего эмулятора, воспроизводящего поведение демонстрируемой ОС и позволяющего запускать ее приложения.
3. Запуск оригинала демонстрируемой ОС во вложенном эмуляторе.

В качестве примера нативной работы демонстрируемой ОС в QEMU можно рассматривать версии ОС Windows. В схеме рис. 1 в этом случае отсутствует вложенный эмулятор; однако сервисная гостевая ОС может присутствовать в Windows 1.x, 2.x, 3.x, поскольку данные ОС на самом деле являются операционными оболочками и требуют для своего запуска какую-либо версию DOS. Windows 95 в отличие от них является полноценной ОС и потому запускается без дополнительной прослойки.

Рис. 2 показывает пример использования такой схемы для запуска Windows 2.0, разработанной в 1987 г. и известной как первая версия Windows, в которой были реализованы метафора рабочего стола, возможность перекрытия окон и изменения их размеров.

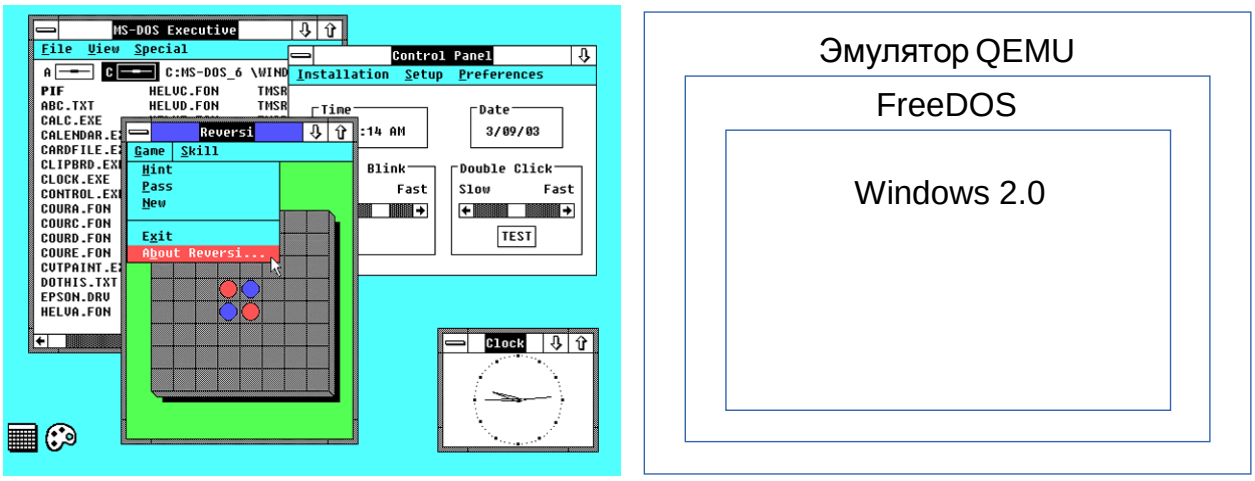

Рисунок 2 – Рабочий экран и схема запуска ОС Windows 2.0

В качестве сервисной гостевой ОС для первых трех версий Windows применена система FreeDOS. Обоснованием выбора служит то, что она является свободным ПО

(уменьшая тем самым лицензионную нагрузку по юридической легализации электронной выставки), а кроме того эффективно использует циклы простоя процессора, что положительно сказывается на потреблении ресурсов хост-системы.

Примером схемы №2, предполагающей наличие вложенного эмулятора, является запуск ранних ОС фирмы Apple: Lisa, MacOS версий 1.x, 7.x.

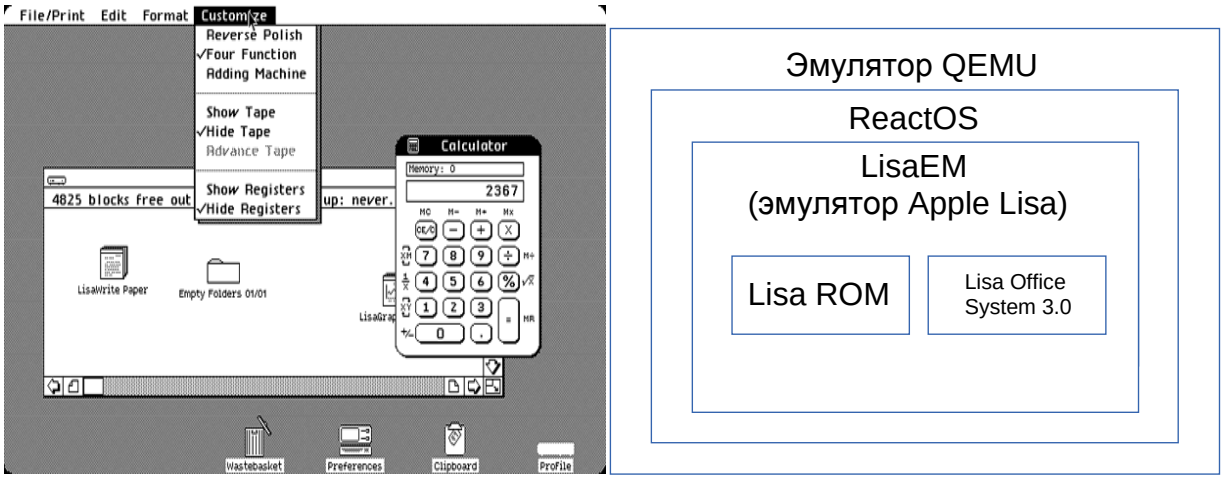

Рисунок 3 – Рабочий экран и схема запуска ОС Apple Lisa 3.0

На рис. 3 демонстрируется запуск ОС Apple Lisa Office System 3, которая была создана в 1983 г. компанией Apple с намерением сделать компьютер для работы с документами и имела первый коммерчески-доступный графический интерфейс на базе метафоры рабочего стола. Для запуска ОС использован эмулятор LisaEM, разработанный в свое время энтузиастами платформы и работающий под управлением Windows-совместимых ОС. В качестве сервисной гостевой ОС использована свободная ОС ReactOS, развиваемая как хобби-проект и имитирующая поведение WindowsNT-совместимых систем. В отношении ReactOS можно дополнительно заметить, что она идеально отвечает всем трем требованиям (поддержка USB, малое потребление памяти, использование циклов простоя процессора), и таким образом опыте это первый случай ее удачного применения. Внутри эмулятора LisaEM использован образ загрузочного диска с демонстрируемой ОС, а также дамп ПЗУ оригинального компьютера Apple, обозначенный на схеме запуска как Lisa ROM.

Пример использования схемы №3, где нет четкого разделения между эмулятором аппаратной платформы и гостевой ОС, можно видеть на рис. 4.

Xerox Alto был разработан в исследовательском центре Xerox PARC в 1973 году и является первым компьютером, использующим оконный графический интерфейс. Программы для Alto являются графическими и поддерживают метафору неперекрывающихся окон, однако они работают в полноэкранном режиме и запускаются из командной строки, с помощью специальной оболочки Alto Executive shell. Для запуска в виртуальной машине использован свободно доступный эмулятор salto, а также образ дискеты, содержащей в себе демонстрируемое программное обеспечение (на рис. 4 можно наблюдать пример запуска такой программы – файлового менеджера Neptune). Роль сервисной ОС снова успешно исполняет ReactOS.

В целом приведенные примеры (как и большинство аналогичных случаев) демонстрируют устойчивую обратную зависимость между возрастом демонстрируемой ОС и

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

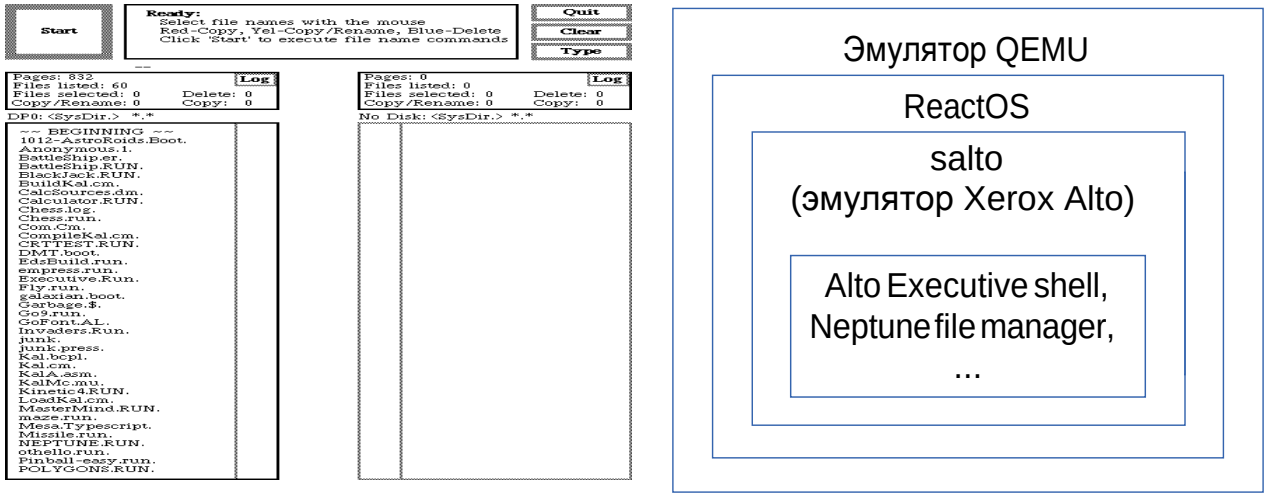

Рисунок 4 – Рабочий экран и схема запуска программ платформы Xerox Alto

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].