ВИРТУАЛИЗАЦИЯ ВСТРАИВАЕМЫХ ПРОМЫШЛЕННЫХ СИСТЕМ

СЕРГЕЙ ЖУКОВ, инженер

В статье описаны системы виртуализации разных типов, рассмотрены их достоинства и недостатки. В числе прочих рассматриваются системы, не привязанные к реальному времени.

Виртуальное облако позволяет решать гораздо более широкий круг задач, чем виртуализованная система управления предприятием (virtualized enterprise system). Кроме того, оно выполняет задачи, далеко выходящие за пределы возможностей виртуализованных встраиваемых систем.

Несмотря на то, что виртуализованные системы бывают разных типов и предназначены для разных конечных приложений, они имеют одно общее свойство: с помощью ограниченного набора аппаратного обеспечения запускается несколько виртуальных сред путем создания разделов (partitioning).

Виртуализация может работать на уровне сервера, отдельной платформы или сети. Она позволяет запускать одновременно независимые сессии для нескольких пользователей или заданий на одном и том же сервере (как облачные вычислительные приложения). С их помощью легко создать на одном компьютере две виртуальные машины, как, например, на ноутбуке с двумя установленными ОС – Windows и Linux. Это разделение задач и расширение ресурсов востребовано конечными пользователями, однако разные схемы виртуализации не работают одинаково в рамках ограничений, действующих в отношении встраиваемых промышленных систем с учетом объемов памяти, потребляемой мощности и других параметров.

Встраиваемые системы имеют приоритеты, отличающиеся от приоритетов систем инфраструктуры предприятия или серверов центров обработки данных (ЦОД). Если системы предприятия отвечают, например, определенным требованиям к среднему значению пропускной способности, то встраиваемые системы, в первую очередь, обеспечивают работу в реальном времени. Для многих встраиваемых систем, особенно промышленной автоматизации, выполнение требований к времени выполнения является частью корректной работы программы, т. е. детерминизм не приносится в жертву ради достижения других целей.

Распространенными ОС для промышленной автоматизации в течение уже многих лет являются операционные системы реального времени (ОСРВ) и контуры периодического управления, работающие без ОС (bare-metal-based periodic control loops).

Последние рыночные тренды побуждают производителей искать новые решения для реализации не привязанных к реальному времени функций в системах реального времени и выполнения других задач, к которым относятся облачная связь для загрузки машинных данных и диагностическое обслуживание. Это одно из наиболее востребованных приложений для Industry 4.0, или промышленных сетей интернета вещей (IIoT). Именно в таких случаях виртуализация обеспечивает взаимодействие узлов без ущерба критическим задачам реального времени.

Виртуализация идет на шаг вперед, однако, учитывая широкое разнообразие решений, требуется сделать правильный выбор, тщательно проанализировав ситуацию.

ТИПЫ СИСТЕМ ВИРТУАЛИЗАЦИИ

Системы виртуализации делятся на два типа: системы с полной виртуализацией и статическим разделением (виртуализация ядра). Системы полной виртуализации симулируют аппаратную среду, так что программные разделы, создающие виртуальную ОС, не привязаны напрямую к аппаратным разделам. Например, на одном физическом ядре можно реализовать два виртуальных ядра, где работают две разные ОС.

Схемы статического разделения позволяют поместить программы или задачи в конкретные разделы существующей аппаратной платформы,

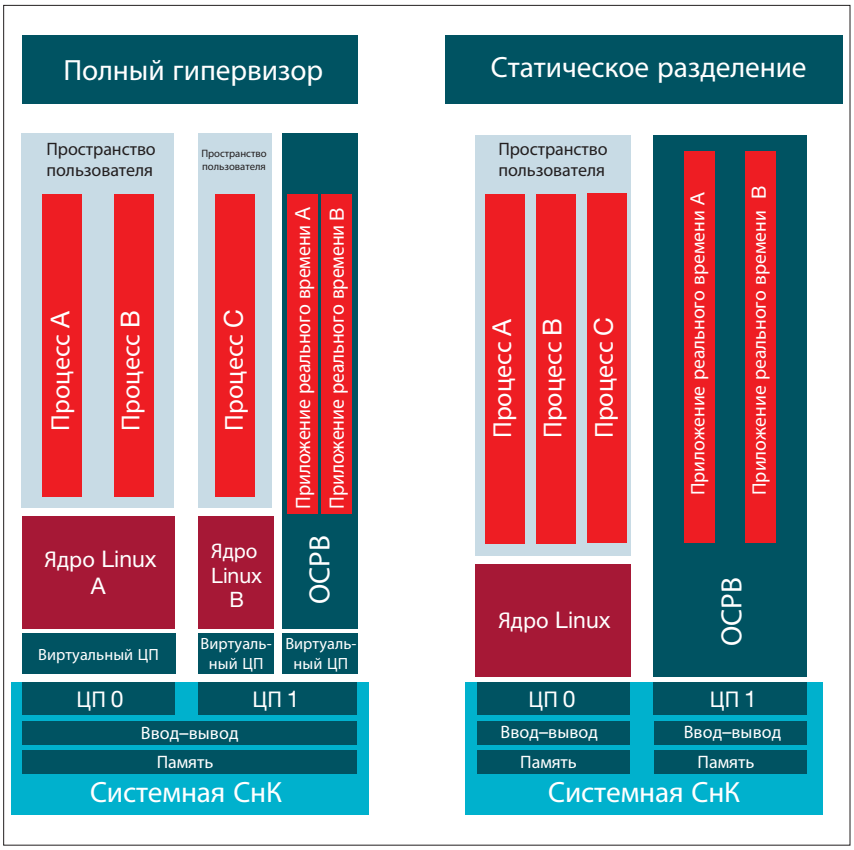

Рис. 1. **Сравнение двух основных типов виртуализации**

чтобы симулировать отдельные системы. Поскольку программные разделы привязаны к аппаратным разделам, на каждое физическое ядро приходится не более одной ОС, как показано на рисунке 1. Недостатком такого подхода является отсутствие возможности использования ресурсов, к которым относятся центральные процессоры (ЦП) или оперативная память (ОЗУ), выполняющие другие задачи.

Обе схемы позволяют разделить изолированные задачи (например, для совместного использования системы несколькими пользователями) или отделить критические задачи от менее важных (например, отделение защищенной области от области общего назначения). Полная виртуализация позволяет остановить ОС или даже перенести ее с одного физического процессора на другой путем сетевого подключения, не прекращая работу. Статичное разделение не обеспечивает такую гибкость, однако гарантирует детерминированную работу.

Поскольку разные схемы виртуализации по-разному относятся к аппаратной платформе, у каждой из них есть свои преимущества в конкретных приложениях. Так, полная виртуализация является мощным инструментом, позволяющим сделать из одного сервера сотню. Однако есть риск, что несколько программ одновременно попытаются обратиться к одному и тому же набору ресурсов. Конфликты разрешаются за счет программного управления, однако при этом возрастает задержка.

Другой недостаток заключается в объеме необходимых для виртуализованных систем вычислительных ресурсов, например памяти. Соответственно, полная виртуализация более распространена в облачных или промышленных системах, у которых приоритетом является административная и служебная масштабируемость. Поскольку память является менее ограниченным ресурсом и оборудование, как правило, находится в зоне с климат-контролем, разработчику проще подобрать необходимый размер устройств или обеспечить требуемое рассеяние тепла.

Напротив, имеющиеся на платформе физические ресурсы ограничивают количество виртуализованных сред, которые можно реализовать в схеме со статичным разделением. Статически разделенные системы обеспечивают те же преимущества по разграничению смешанных задач, т. е. позволяют отделить задачи, критичные к безопасности, защищенности или работе в реальном времени. Однако поскольку физические ресурсы привязаны напрямую к среде виртуализации, время выполнения задачи практически не изменяется. Это делает разделенные системы более подходящими для встраиваемых систем с ограниченными вычислительными ресурсами.

ИСПОЛНЕНИЕ

Как при полной виртуализации, так и при статичном разделении необходимо программным образом создать схему виртуализации. Это задача выполняется гипервизором, который, имея самый высокий приоритет исполнения, управляет гостевыми машинами или гостевыми ОС. Гипервизор также контролирует совместное использование имеющихся физических ресурсов виртуальными системами.

По аналогии с уровнями виртуализации существуют гипервизоры двух типов. Гипервизоры первого типа представляют собой особый уровень ПО, запущенного на аппаратной платформе. Оно содержит ОС и управляет ресурсами, а также назначением памяти между виртуальными машинами.

В гипервизорах второго типа нижний уровень запущенного ПО является хостовой ОС, которая предоставляет гипервизору драйверы и сервисы для виртуализации гостевых ОС (см. рис. 2). Гостевые ОС «не знают», что они запущены не напрямую с аппаратной части системы. Гипервизор второго типа исполняет роль уровня абстракции. Гостевые ОС представлены как процессы хостовой ОС. Доступ к аппаратным ресурсам предоставляется хостовой ОС. Преимуществами гипервизоров второго типа является отсутствие необходимости внесения изменении ни в хостовую, ни в гостевую ОС. Недостаток заключается в том, что из-за дополнительных слоев абстракции может снизиться общая производительность системы.

Другим широко применяемым программным решением для разделения приложений являются контейнеры. Если супервизоры виртуализуют аппаратную платформу для запуска нескольких ОС, контейнеры виртуализуют ОС для запуска нескольких приложений (см. рис. 3). Контейнеры занимают меньше ресурсов, могут работать с большим количеством приложений, однако их недостатком является то, что используется только одна ОС.

Хотя разделение приложений является важной функцией для встраиваемого промышленного оборудования, из-за него становится сложнее отделять задачи реального времени от некритичных к этому режиму, поскольку система не может запускать одновременно ОСРВ

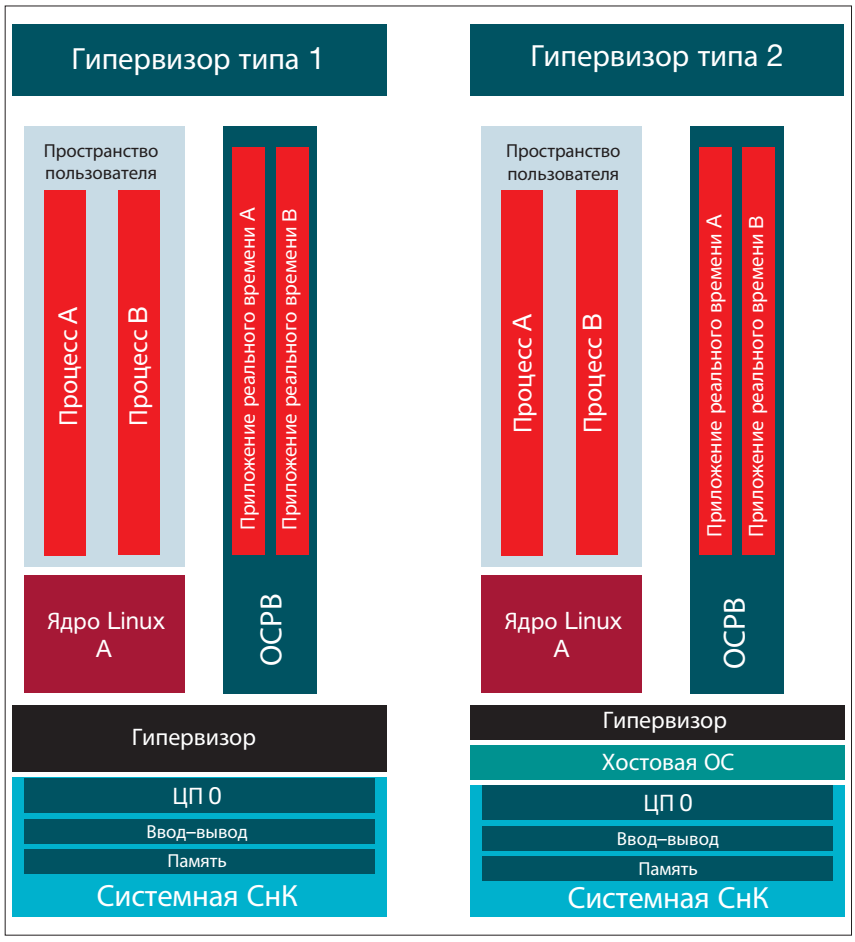

Рис. 2. **Сравнение гипервизоров типов 1 и 2**

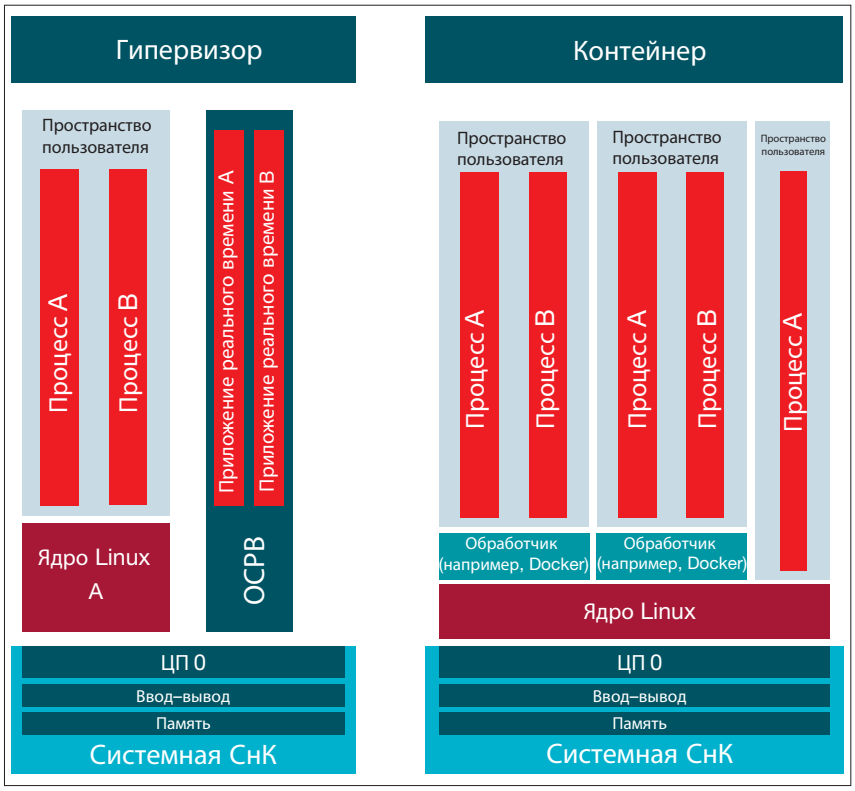

Рис. 3. **Сравнение контейнера и гипервизора на СнК**

и ОС высокого уровня. Контейнеры не обеспечивают детерминизм, однако их вполне можно использовать во встраиваемых системах для управления другими программными стеками высокого уровня без ограничений по реальному времени со стороны Linux.

РЕШЕНИЯ ПО ВИРТУАЛИЗАЦИИ

Наиболее распространенными решениями по виртуализации являются KVM, Xen, Jailhouse и Docker. Многие сторонние ОС обеспечивают решения по виртуализации, например профиль виртуализации Wind River VxWorks или мультивизор Green Hills Integrity. Все эти примеры подпадают под описанную выше классификацию: гипервизоры полной виртуализации или статическое разделение, типы 1–2, контейнеры.

KVM – это полный гипервизор с открытым кодом на базе Linux. На базовом уровне ядро Linux превращается в гипервизор; при этом можно реализовать гипервизор и типа 1, и типа 2. Хостовая ОС обязательно должна быть Linux. К типу гостевой ОС это ограничение не относится.

Решение Xen также имеет открытый код, но, в отличие от KVM, оно поддерживает гипервизоры обоих типов: полную виртуализацию и статическое разделение. В качестве хостовой ОС не обязательно выступает Linux. Поскольку Xen поддерживает паравиртуализацию, такое решение работает на платформах без расширений виртуализации. Гипервизор KVM не обладает этой функцией.

Гипервизор первого типа Jailhouse с открытым кодом предназначен для создания разделов. В отличие от других гипервизоров, работающих на аппаратном уровне, его загрузку и настройку можно произвести через ОС Linux. Интерфейс управления основан на инфраструктуре Linux. Одним

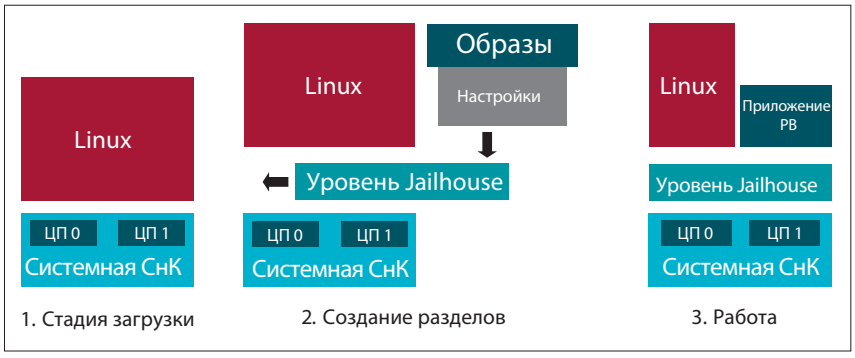

Рис. 4. **Этапы работы системы при использовании виртуализации Jailhouse**

из ее преимуществ является то, что все аппаратные и конфигурационные настройки Linux используются повторно – они не прописываются в гипервизоре. Гостевые ОС могут быть любого типа.

Green Hills Integrity и Wind River VxWorks являются двумя распространенными промышленными ОСРВ для промышленных приложений. Обе обеспечивают виртуализацию. На текущий момент Wind River имеет профиль виртуализации для VxWorks только типа 1, при этом хостовая ОС должна быть VxWorks. Мультивизор Green Hills Integrity является гипервизором типа 1, поддерживающим полную виртуализацию.

Docker является коммерческим программным решением для контейнеров и может работать и под Linux, и под Windows.

JAILHOUSE – ГИПЕРВИЗОР ДЛЯ ВСТРАИВАЕМЫХ ПРОМЫШЛЕННЫХ СИСТЕМ

Хотя каждое из упомянутых решений можно запускать на процессоре AM572x Sitara от Texas Instruments (TI) и процессоре AM65x этой же компании на ядрах Arm Cortex-A, TI в настоящее время напрямую поддерживает гипервизор Jailhouse с открытым кодом от Siemens как часть стандартного ПО, поставляемого в пакете разработки для Linux. У Jailhouse для промышленных встраиваемых систем имеются следующие преимущества.

1. Поддержка описанного выше разделения. Оно подходит для встраиваемых систем больше, чем полная виртуализация.
2. Простота за счет отсутствия дополнительных функций.
3. Запуск на аппаратном уровне при активации. Это означает, что гипервизор берет полный контроль над аппаратной частью и не требует внешней поддержки. В отличие от других гипервизоров, работающих без ОС, он загружается и конфигурируется через Linux, как упоминалось. Это упрощает использование и делает адаптацию быстрее.

Jailhouse конфигурирует свойства ЦП и параметры виртуализованных устройств аппаратной платформы так, чтобы ни один из этих доменов, называемых ячейками, не взаимодействовал нежелательным образом с остальными. Гипервизор не осуществляет планирование – только виртуализует программным образом ресурсы, которые существенны для платформы и аппаратно не разделяются.

На рисунке 4 показаны разные фазы работы системы с использовани-

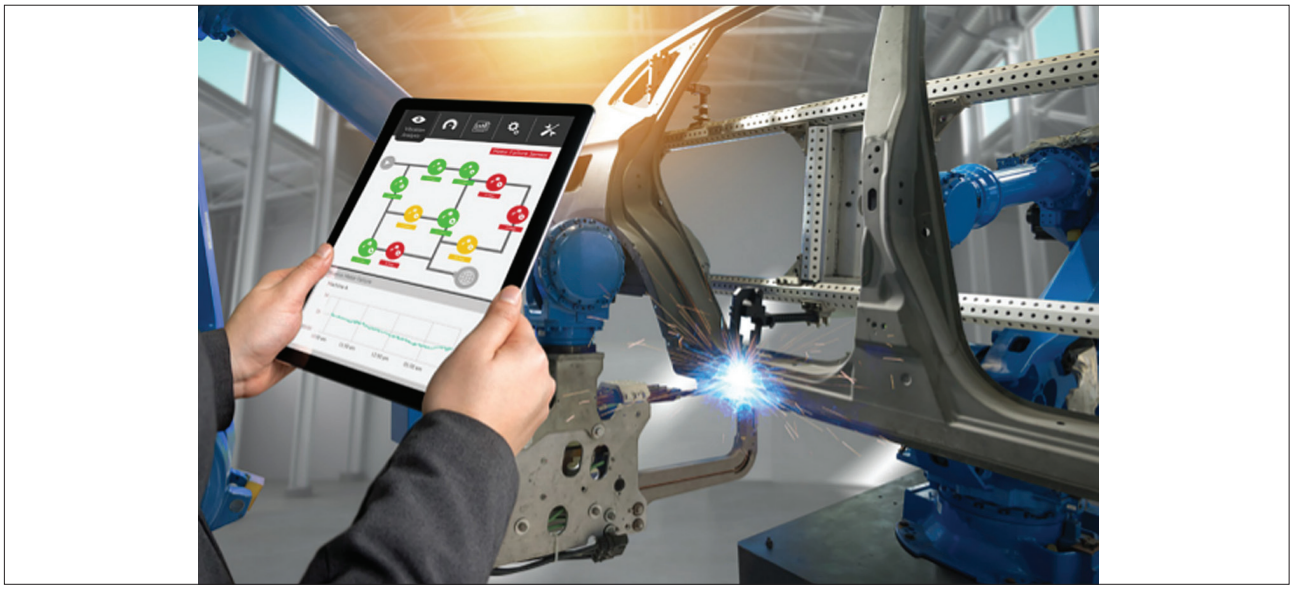

Рис. 5. **Представление на планшете модели автоматизации**

ем виртуализации Jailhouse. Сначала загружается Linux, затем активируется Jailhouse, а на стадии создания разделов назначаются дополнительные ячейки. Поскольку гипервизор Jailhouse не требует высокой вычислительной мощности или больших объемов памяти, системы виртуализации с его использованием могут работать в реальном времени, как невиртуализованные приложения. Таким образом, его можно применять во встраиваемых системах, особенно когда одновременно выполняются два приложения с разными уровнями работы в режиме реального времени.

Вернемся к примеру с Industry 4.0, когда для выгрузки данных для диагностического обслуживания осуществляется подключение к серверу. Понятно, что большое количество задач не требует работы в реальном времени. В Jailhouse всегда имеется, по крайней мере, одна запущенная ОС Linux. Это удобно для промышленных встраиваемых систем, поскольку у Linux большое количество доступных сетевых стеков и протоколов. Например, можно включить патч безопасности в сетевые протоколы, что не скажется на работе приложений реального времени.

ВИРТУАЛИЗАЦИЯ В ПРОМЫШЛЕННЫХ ВСТРАИВАЕМЫХ СИСТЕМАХ

Диагностическое обслуживание не является единственным приложением IIoT, которое способствует развитию решений для виртуализации. Проводные сети, не требующие работы в режиме реального времени, а также облачные подключения позволяют реализовать большое количество приложений для промышленной автоматизации.

Одним из таких них является удаленный мониторинг и удаленная загрузка. Во многих областях, к которым относятся производство, автоматизация процессов, нефтегазовая сфера, оборудование нередко устанавливается в труднодоступных или опасных средах, куда затруднен доступ операторов для проверки или обновления вручную. Подключенная машина может обновляться дистанционно с сервера.

Другим примером возможностей облачных сервисов в промышленной автоматизации является предоставление оборудования как услуги (hardware-as-a-service, HaaS). Поставщики HaaS предоставляют машины конечным пользователям в соответствии с контрактом, в который также входят обслуживание и другие сервисы. Поскольку промышленное оборудование является дорогостоящим, эта модель выгодна как производителям (обеспечивает прогнозируемый доход), так и потребителям (предоставляется доступ к оборудованию с меньшими предварительными капиталовложениями).

Другим преимуществом для конечных пользователей является то, что надежность оборудования выше, поскольку контракт предусматривает ситуации, когда оно функционирует неправильно. Однако такая модель работает лучше в случае такого подключения машины, что ее владелец, не находясь поблизости от нее, может отслеживать ее статус и, например, осуществлять обслуживание до ее выхода из строя.

Ожидается, что будет реализовано приложение, моделирующее весь процесс в облаке, например сборку на линии или химическую реакцию в фармацевтическом производстве (см. рис. 5). Такой углубленный анализ требует наличия связи в масштабе практически всего предприятия. Функцию связи можно встроить в крупногабаритное оборудование, в то время как для малых устройств целесообразнее иметь шлюзы для сбора информации.

Функция связи в промышленном оборудовании получит широкое распространение только тогда, когда станет дешевой. Это произойдет не так скоро, как хотелось бы, однако ведущие производители уже готовятся к изменениям в этом направлении, разрабатывая облачные платформы Predix (GE) или Mindsphere (Siemens).

Необходимы экономичные системы связи, использование которых не препятствует осуществлению главной задачи оборудования. Хотя приложения, требующие подключения к интернету или серверу, выиграли бы от уменьшения задержки, их работа в режиме реального времени не является большим преимуществом в сравнении с теми критичными ко времени задачами ОСРВ, которые она решает, управляя промышленными программируемыми логическими контроллерами, станками с ЧПУ, приводами и т. д. Следует учитывать это обстоятельство при оценке стоимости, т. е. анализировать не только перечень аппаратных компонентов (bill of materials, BOM), но и сложность разработки ПО. Наиболее простым решением для предотвращения взаимодействия этих двух приложений друг с другом является разнесение их на две разные СнК, но он увеличивает стоимость системы.

Оптимальным с точки зрения расходов методом управления разработкой программной части является сбалансированное использование проверенных решений, существующих в ОСРВ для критичных ко времени приложений

и в Linux для сетевых приложений. Компромисс между этими двумя методами заключается в запуске ОС на одном многоядерном процессоре со статичным гипервизором, позволяющим запускать каждую ОС на отдельном ядре, чтобы критичные ко времени задачи были выполнены в отведенный для них срок.

Хотя использование подключенного к облаку оборудования и сетей часто дискутируется, виртуализация находит применение и вне совместной эксплуатации критичных ко времени приложений и приложений, не критичных ко времени. Например, к стандартному ПЛК, машинному контроллеру или защитному реле можно добавить более современный человеко-машинный интерфейс, чтобы упростить работу операторов (см. рис. 6). В этом случае станет проще воспользоваться сравнительно большой дисплейной панелью со стандартной интегрированной графической средой под Linux, чем поддерживать панель с помощью ОСРВ, особенно учитывая, что графический пользовательский интерфейс (GUI), как правило, настраивают под свои нужды конечные пользователи оборудования.

Заметим, что работа всей системы под Linux не является самым подходящим выбором в отношении критичных ко времени выполнения функций оборудования, к которым относятся логические операции ПЛК, осуществление контроля движения или алгоритмов защиты. Одним из решений может стать назначение функций человеко-машинного интерфейса и управления двумя процессорам на плате, однако при этом увеличивается размер, стоимость и общая потребляемая мощность приложения.

Более рациональным решением является запуск обеих задач на одном ЦП при сохранении разделения между ними и оптимизированными под них операционными системами с помощью гипервизора Jailhouse, не требующего большого объема ресурсов для своей работы.

ВЫВОДЫ

Потребность в виртуализации будет увеличиваться, поскольку приложения, подключаемые к облачным сервисам или требующие простого человеко-машинного интерфейса, побуждают производителей использовать больше некритичных ко времени функций в оборудовании, работающему в режиме реального времени.

Производителям промышленного оборудования, пытающимся найти программные решения для виртуализации, придется искать компромиссы между быстрой реализацией решений, затратами на проведение исследований и разработки, величиной лицензионных отчислений на ПО, экономически эффективными аппаратными решениями, не требующими большого объема памяти, и, что важнее всего, возможностью установления приоритета операций, выполняющихся в реальном времени.