Московский физико-технический институт (государственный университет)

Факультет радиотехники и кибернетики

Лаборатория суперкомпьютерных технологий для биомедицины, фармакологии и малоразмерных структур

Виртуальная платформа Simics для разработки многопроцессорных систем

Москва 2010

Содержание

Список иллюстраций

1 Введение

Массовый переход промышленности к многоядерным процессорам и многопроцессорным системам требует нового программного обеспечения и средств разработки для того, чтобы помочь разработчикам сделать свой код пригодным для параллельного исполнения и таким образом повысить его производительность. В наше время необходимо знать, как создавать такое программное обеспечение и проектировать системы, эффективно использующие параллелизм аппаратуры.

Системы виртуальной разработки предоставляют новую методологию, в которой настоящее оборудование систем дополняется виртуальной платформой Wind River Simics [2], которая имитирует разрабатываемое оборудование и работает на обычной рабочей станции или ПК. Виртуальная платформа может исполнять тот же двоичный код программного обеспечения, что и физическое оборудование, и делать это достаточно быстро для того, чтобы использоваться в качестве альтернативы и дополнять аппаратные средства при разработке программного обеспечения.

2 Виртуальные платформы

Виртуальная платформа предоставляет целый ряд преимуществ, такие как свобода от физических ограничений реального оборудования, широкие возможности конфигурирования, сохранение состояния и возможность перезапуска системы в любой момент, бóльшая стабильность, возможность работы с целевой системой задолго до готовности аппаратных прототипов [1], возможность проверки модели на ошибки и анализ граничных случаев с полным контролем и точностью.

Виртуальные платформы могут обеспечить контроль и повторяемость при отладке программного обеспечения и процесса анализа систем путём создания косвенности между аппаратным и программным обеспечением. Традиционные методы отладки работают не очень хорошо на принципиально недетерминированных системах, таких как многопроцессорные системы или многоядерные процессоры.

Разрабатываемые системы выглядят так, как показано на рис. 1. Здесь стек программного обеспечения, работающего на малых (или больших) узлах с общей памятью, общается с другими узлами с помощью различных типов соединений. Программная абстракция будет строиться поверх многоядерного и распределенного оборудования.

Обратим внимание, что такая структура программы может использоваться и на одиночном многоядерныом устройстве. Для многопроцессорных аппаратных средств это позволяет несколько изолированных групп ядер запускать на удаленных узлах сети и обеспечивать взаимодействие между ними по сети, даже если они физически должны находиться на одном кристалле.

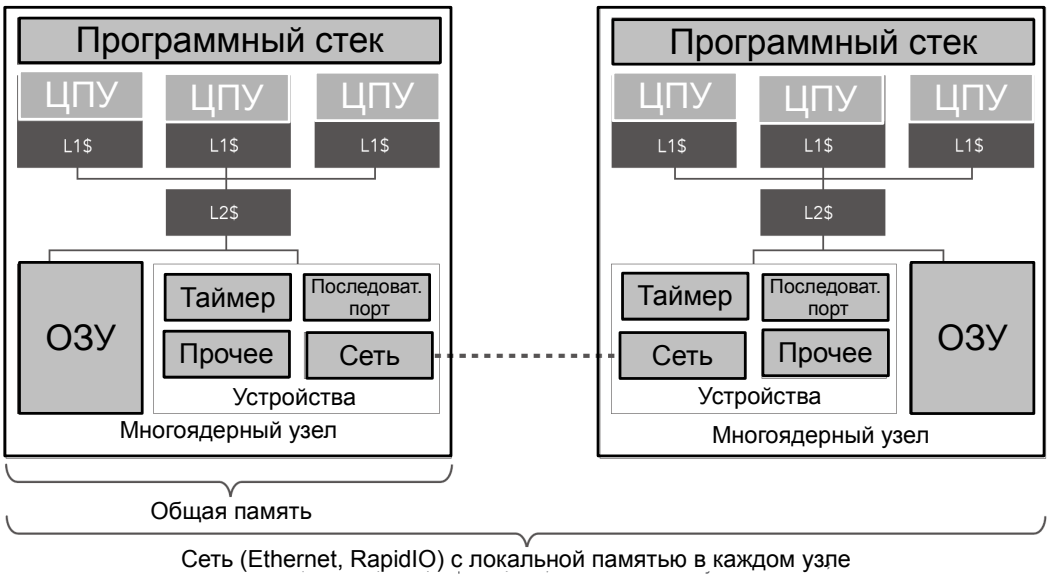

Рис. 1: Схема компьютерного комплекса.

3 Виртуальная разработка систем

Проектирование многопроцессорных систем не является простым делом. Однако и это легче, чем создавать программное обеспечение для таких систем. Существует три основные задачи:

- Обеспечение того, что существующее программное обеспечение продолжит корректно работать на новой системе, пусть и без получения преимуществ от её многоядерного характера.
- Распараллеливание существующего программного обеспечения для получения выигрыша в производительности и энергопотреблении благодаря параллельному выполнению.
- Создание нового программного обеспечения с нуля, использующего параллельность с самого начала.

Виртуальная разработка систем с помощью Wind River Simics предлагает новый подход к созданию и портирование многоядерных приложений, который обладает многими убедительными преимуществами по сравнению с простыми попытками запуска программ на таких системах с непосредственным тестированием и отладкой [3]. Simics предоставляет ряд важнейших функций для тестирования и отладки параллельных кодов, в особенности многоядерные реализации параллельных систем:

Повторяемость. Каждый запуск программного обеспечения в Simics даёт в точности повторяемые результаты исполнения, даже при моделировании нескольких

ядер и нескольких процессов на каждом ядре. Таким образом, самая большая проблема многопроцессорной отладки устраняется: трудность воспроизведения условий, в которых возникла ошибка. Это позволяет отлаживать мультипроцессорные системы так же просто, как отлаживать одну программу на одном процессоре.

Многоядерная отладка. Simics можете присоединить отладчик одновременно к каждому ядру процессора в многопроцессорных машинах.

Отладка с поддержкой ОС. Использование знаний об исполняемой ОС[1] позволяет различать и отлаживать отдельные процессы и потоки, а также код операционной системы.

Глобальная остановка и пошаговое исполнение. При пошаговом исполнении кода одного процессора в многопроцессорной системе все остальные устройства в системе также работают пошагово. Такое невозможно сделать на реальном оборудовании.

Обратимость. Обратимость исполнения и отладки безупречно работает в многопроцессорных и многоядерных системах. Это обеспечивает мощные возможности для отслеживания труднодоступных ошибок, такие как блокировки, тупики и состояния гонки, более характерные именно для многоядерных сред.

4 Обнаружение ошибок в ПО

Полностью параллельная многопроцессорная система обладает рядом новых типов потенциальных ошибок ПО, к тому же усиливается серьёзность классических ошибок. В частности, существующие программы, которые прекрасно работают на одноядерных машинах, часто дисфункционируют на многоядерных. Рис. 2 показывает простой пример того, как переход от одноядерного к двухъядерному процессору результирует тем, что система с большей вероятностью будет проявлять ошибки параллельного программирования. График приведён для тестирования программы для ряда частот ядра на одно- и двухъядерных конфигурациях.

Известно, что данная программа имеет проблему состояния гонки. Видно, что на одноядерном процессоре эта проблема проявляется лишь изредка. На высоких тактовых частотах она проявляется крайне редко, поскольку операционная система выполняет больше инструкций в программе между переключениями потоков, и шансы попасть на критическую секцию в момент переключения ниже.

При использовании же двухъядерной системы *каждое* исполнение вызывает ошибку. Если эта ошибка была бы менее агрессивной, на однопроцессорном устройстве она проявлялась бы один раз в год, что представляется лишь редким случайным сбоем. После того, как система будет переведена на несколько ядер, ошибка будет

[1] Операционной системы.

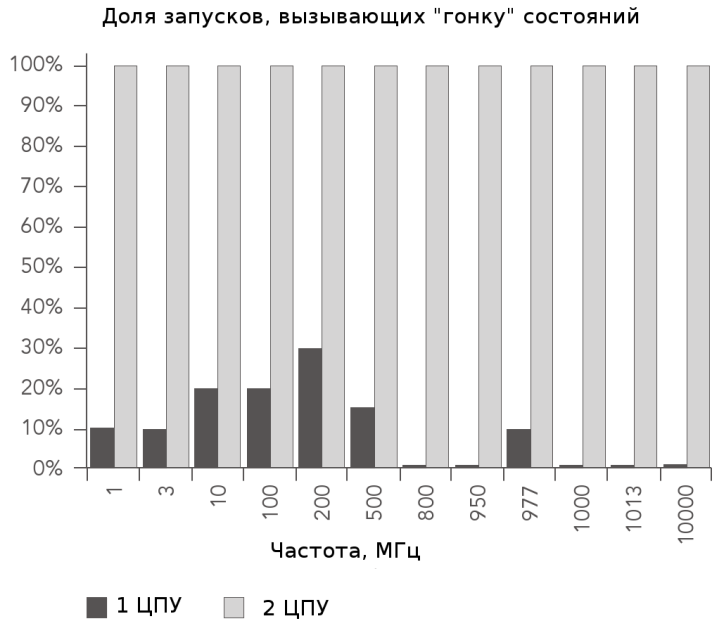

Рис. 2: Исполнение на одно- и многоядерной системе.

проявляться раз в неделю, что сделает её реальной проблемой. На виртуальной платформе такие особенности можно быстро обнаружить с помощью тестирования программного обеспечения на различных конфигурациях, в том числе ещё не доступных в реальной аппаратуре.

5 Отладка параллельных программ

Основным преимуществом виртуальной платформы является то, что она предоставляет средства отладки и анализа лучшие по сравнению с физическим оборудованием. Любой, кто когда-либо разрабатывал код для встраиваемых платформ, по достоинству оценит удобства виртуальной среды. Вы получаете систему, которая не ведёт себя случайно время от времени, гораздо лучше контролирует объект работы. Также вы можете поставить неограниченное количество точек останова для инспектирования состояния. Если система полностью зависнет, вы можете остановить её и понять, что произошло.

Давайте разбем принципы работы в такой среде на реальном примере отладки многоядерной ошибки с помощью Simics.

6 Пример отладки

Simics был использован для проверки первого прототипа ОС VxWorks на двухъядерный процессор Freescale 8641D [4]. В одном тесте, когда тактовая частота целе-

вой системы была изменена с 800 МГц до 833 МГц, она внезапно застыла в начале процесса загрузки. Был полностью потерян контакт с ОС.

Было установлено, что проблема возникает только между 829.9 МГц и 833.3 МГц. Этот факт не был обнаружен раньше, так как тактовая частота реальной аппаратуры всегда была 800 МГц. Из-за повторяемости симуляции в Simics, ошибку было тривиально воспроизвести. Каждое раз, когда виртуальная платформа загружалась с «плохой» тактовой частотой, происходило зависание в один и тот же момент времени. В отличие от настоящего оборудования, которое дало бы в таком случае немой «кирпич», виртуальная платформа позволила изучить состояние процессора, памяти и программного обеспечения в том месте, где происходила неисправность .

Для выяснения причин этой проблемы были использованы обратимое исполнениие и трассировка прерываний последовательного порта, контроллера прерываний и процессорных ядер. Это дало возможность точно определить номер цикла и конкретную инструкцию, где возникла проблема, и последовательность событий, ведущих к ней.

Было установлено, что проблема была вызвана процедурой обслуживания прерываний, которая пыталась блокировать ядро до повторного включения прерываний. Когда система застывала, блокировка была уже сделана в момент входа в процедуру обработки, и без включения прерываний не было никакой возможности запустить другой код, чтобы снять эту блокировку.

Ошибка была найдена только потому, что виртуальная платформа исполняла полный реальных программный стек, в том числе обработчики прерываний и драйверы аппаратных средств. Проблема была вызвана изменением конфигурации системы, что свидетельствует о ценности гибкой конфигурируемости виртуальной среды. Благодаря повторяемости ошибка легко воспроизводилась. Способность отслеживать и контролировать полное состояние имеет решающее значение для понимания того, что произошло и в каком порядке.

7 Преимущества отладки на виртуальной платформе

Главное преимущество виртуальной платформы — повторяемость и обратимость, как показано на рис. 3. Ключевой проблемой в поиске и фиксации ошибок в ПО параллельного программного обеспечения является отсутствие детерминизма исполнения системного программного обеспечения. Каждый прогон программы создаёт различный порядок событий, и даже очень малые изменения в состоянии системы или её временных характеристиках даёт в результате самых разные сценарии выполнения. Это значительно усложняет отладку, потому что сам факт отладки параллельных программ меняет задержки, и состояния гонки могут исчезать или появляться в разных местах.

Это, однако, не означает, что исполнение программы всегда идентично. Оно лишь означает, что запуск на идентичной конфигурации с одниаковыми начальными

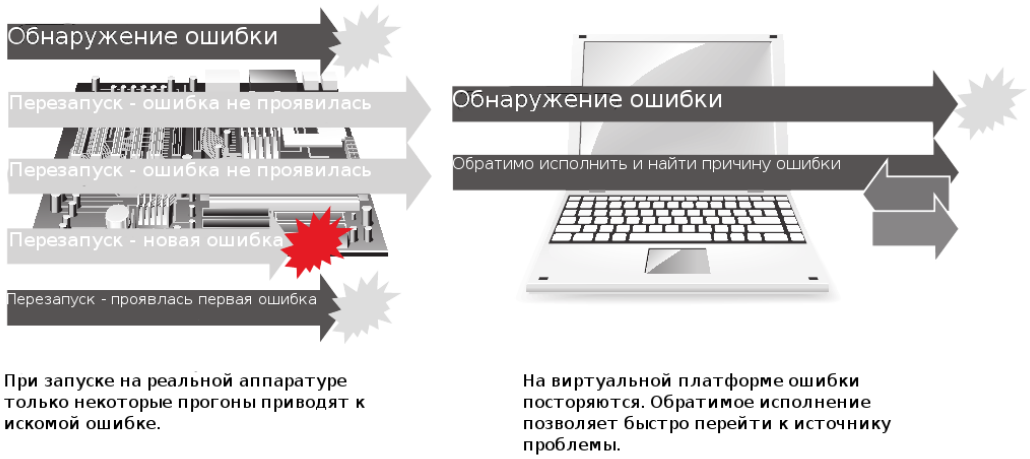

Рис. 3: Повторяемость и обратимое исполнение.

условиями даст в точности одинаковые результаты. Если же что-то изменилось, то и результаты будут различны. Рис. 4 показывает пример того, что одна ошибка проявляется для нескольких запусков одной программы на двух различных моделируемых многоядерных машинах. Каждый запуск даёт разные результаты, поскольку она выполняется из разных начальных состояний. Симулятор может вернуться обратно во времени и воспроизвести каждый запуск точно, что невозможно на физическом оборудовании.

Дополнительное преимущество виртуальной платформы для многоядерной отладки — это то, что симулятор может остановить выполнение всей системы в любой момент времени. Это означает, что можно поинструкционно исполнять код, в котором процессоры взаимодействуют друг с другом, без изменения их поведения. Код, выполняемый на других процессорах, не будет перегружать отлаживаемый процессор данными.

8 Заключение

Создание надёжных и высокопроизводительных систем с использованием многоядерных процессоров и параллельного программирования является сложной задачей для разработчиков всего мира. Системы виртуальной разработки являются ключевым инструментом для того, чтобы делать это проще, быстрее и уменьшить вероятность ошибки. В процессе разработки могут быть полученны более точные данные и достигнуто более глубокое понимание процеесов, происходящих в системе. Отладка происходит быстрее, а аппаратура и ПО различных архитектур могут быть изучены с помощью виртуальной платформы для улучшения характеристик реального оборудования.

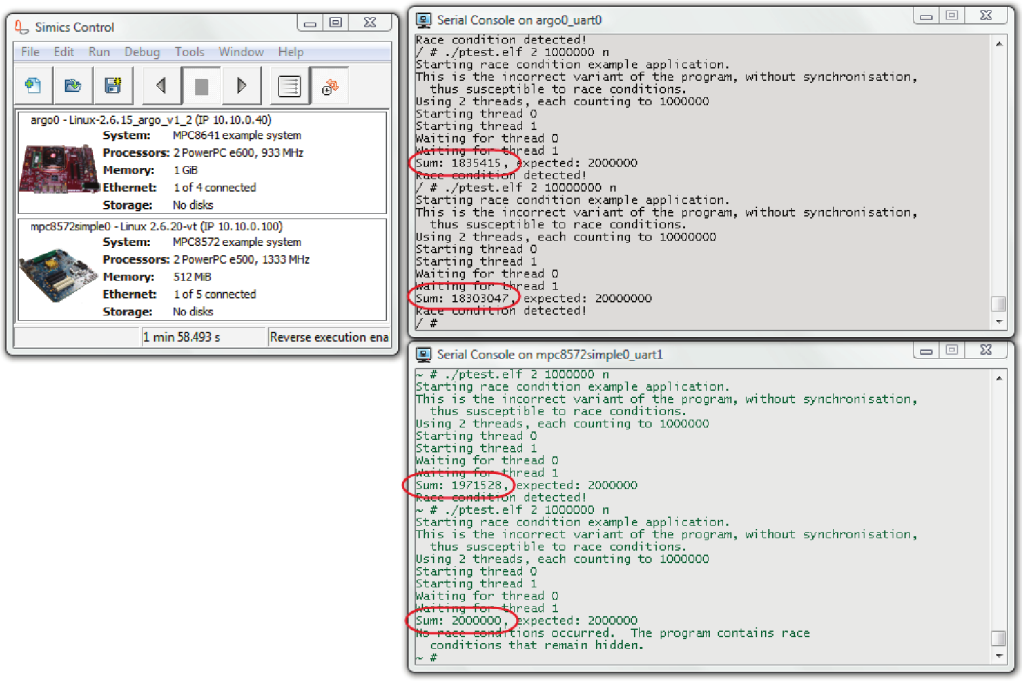

Рис. 4: Повторяемость в многоядерной системе.

Список литературы

[1] *Uhlig R., Fishtein R., Gershon O., Hirsh I., Wang H.* SoftSDV: A Presilicon Software Development Environment for the IA-64 Architecture // Intel Technology Journal. — 1999. — P. 112 — 126.

[2] Wind River Simics for Multi-core Systems Development. Whitepaper http://www.windriver.com/whitepapers/

[3] *Jakob Engblom.* Debugging Real-Time Multiprocessor Systems. Class #264, Embedded Systems Conference, Silicon Valley 2006.

[4] *Freescale Semiconductor.* MPC8641 and MPC8641D Integrated Host Processor Hardware Specifications. Document Number: MPC8641DEC. Доступен с http://freescale.com.