

УДК 004.4

Выбор между виртуальными машинами и машинной компиляцией для разработки программного обеспечения: сравнительный анализ

Титов Александр Юрьевич

titovayu@student.bmstu.ru

Чухраев Игорь Владимирович

chukhraev@bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Россия

Рассмотрены подходы к разработке программного обеспечения для промышленных контроллеров, акцентируя внимание на сравнении виртуализации и машинной компиляции. Обсуждаются ключевые аспекты, такие как производительность, эффективность использования ресурсов и соответствие международным стандартам. Приводятся примеры реализации каждого из методов, а также их преимущества и недостатки в контексте создания надежного и быстродействующего ПО для контроллеров серии ПЛК 500. Статья предлагает рекомендации по выбору оптимального подхода на основе специфики задач разработки.

Ключевые слова: программное обеспечение, виртуализация, машинная компиляция, производительность, контроллеры, ПЛК 500

Разработка программного обеспечения для промышленных контроллеров требует особого внимания к быстродействию и экономии вычислительных ресурсов, что особенно важно в условиях жестких требований к стабильности и эффективности систем. Предприятие АО «Алгонт» столкнулось с необходимостью создания ПО для контроллеров серии ПЛК 500, где ключевыми условиями стали высокая производительность и соответствие стандартам программирования по МЭК. Этот стандарт, широко используемый в автоматизации, требует от программного обеспечения не только надежности, но и совместимости с широким спектром оборудования, что увеличивает важность оптимального выбора технологии разработки [1].

На этапе выбора подхода к разработке рассматривались два основных метода: использование виртуальных машин и машинной компиляции. Первый подход, виртуализация, позволяет создать платформенно-независимое окружение, но может наложить дополнительные требования к ресурсам и привести к незначительному снижению быстродействия [2]. Второй подход, машинная компиляция, ориентирован на использование аппаратных ресурсов максимально эффективно, однако ограничивает совместимость и требует более глубокого учета особенностей конкретного оборудования.

Целью данной работы является сравнительный анализ этих двух методов для определения их сильных и слабых сторон применительно к разработке ПО для контроллеров, где критически важны производительность, компактность и надежность. Это должно позволить разработчикам выбрать оптимальный метод в зависимости от требований к программному обеспечению, заданных особенностями его применения на практике.

Метод разработки ПО с использованием виртуальной машины. Разработка программного обеспечения для промышленных контроллеров с использованием виртуальной машины подразумевает особый подход, обеспечивающий универсальность и переносимость программ между различными аппаратными платформами. В первую очередь, программный код пишется на языке ST (Structured Text), который компилируется в универсальный набор инструкций. На этом этапе компилятор ST преобразует код, соответствующий стандарту МЭК, в промежуточный формат, не зависящий от конкретного типа процессора. Этот промежуточный код облегчает совместимость с разными аппаратными средствами и делает систему независимой от архитектуры конкретного контроллера. Общая схема работы с виртуальной машиной представлена на рис. 1.

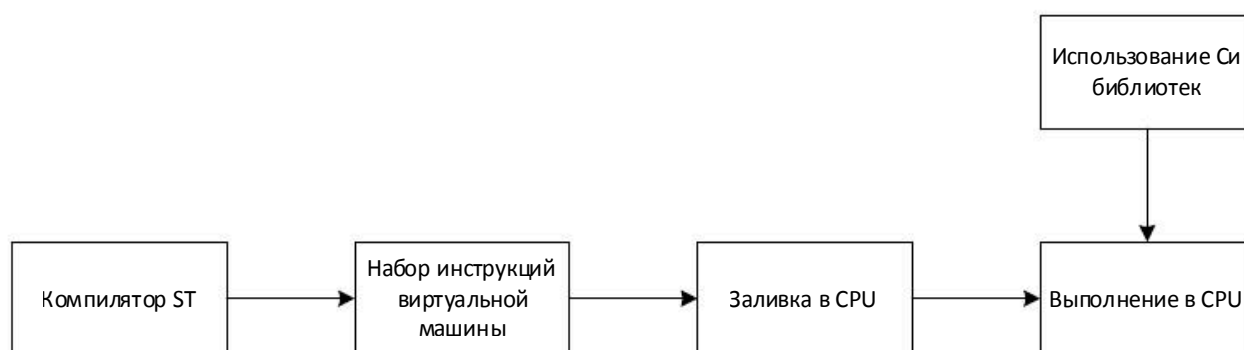


Рис. 1. Схема работы виртуальной машины

Скомпилированный код, представляющий собой универсальный набор инструкций, затем загружается в центральный процессор контроллера. Внутри процессора виртуальная машина берет на себя задачу интерпретации и выполнения инструкций. Важно отметить, что для выполнения определенных функций виртуальная машина может задействовать встроенные библиотеки на языке C. Это позволяет применять более низкоуровневые операции, расширяя возможности программного обеспечения [3].

Процесс компиляции для виртуальной машины включает несколько важных этапов анализа. Сначала происходит удаление комментариев и разбивка исходного кода на лексемы, чтобы устранить избыточные данные и упростить структуру кода для интерпретатора. Далее следует анализ директив и команд, что помогает оптимизировать код и проверять его на наличие ошибок. Системный анализатор проверяет количество команд в блоках, предотвращая возможные проблемы переполнения и упрощая управление ресурсами. Последовательность выполнения также имеет значение: математические операции обрабатываются слева направо, а команды присвоения — справа налево, что позволяет рационально распределить нагрузку и ресурсы.

Использование виртуальной машины имеет свои плюсы и минусы. Среди преимуществ — сокращение количества операций, так как виртуальная машина оптимизирует код для его эффективного исполнения. Также за счет платформенной независимости виртуальная машина предоставляет универ-

сальный код, подходящий для различных процессоров, что особенно удобно при переносе ПО между различными контроллерами. К тому же, код в таком формате становится проще, поскольку интерпретатор берет на себя значительную часть обработки [4].

Однако виртуальная машина также имеет некоторые недостатки. Виртуализация требует значительных накладных расходов ресурсов процессора, что может снижать общую производительность системы. Кроме того, сам процесс интерпретации приводит к снижению скорости исполнения по сравнению с машинной компиляцией, где код напрямую работает с аппаратными ресурсами, обеспечивая максимальную производительность.

Этот подход был реализован, например, в системе ISaGRAF, которая поддерживает языки стандарта МЭК и применяет виртуальную машину для повышения переносимости и упрощения разработки [5].

Метод разработки ПО с использованием машинной компиляции. Метод машинной компиляции опирается на создание промежуточного представления исходного кода, используя дерево абстрактного синтаксиса (AST), и дальнейшую компиляцию с помощью стандартных инструментов, таких как компилятор GCC. Компиляция начинается с того, что код, написанный на языке ST (Structured Text), подвергается анализу и преобразуется в дерево AST, которое отражает иерархию кода, обеспечивая структурированное представление данных. Это дерево упрощает обработку и манипуляцию программными компонентами, что особенно полезно при генерации кода на C++, поскольку C++ предоставляет больше возможностей для управления памятью и доступа к данным на низком уровне.

Преобразование кода на C++ происходит посредством транспилатора, что позволяет использовать функционал, который поддерживается большинством процессоров, и создавать динамические библиотеки (.so файлы), готовые для загрузки в процессор (CPU). Бинарный код, сформированный на выходе, обладает всеми необходимыми оптимизациями для эффективного исполнения.

Кроме того, наряду с AST также формируется глобальная структура, включающая конфигурационные данные, которая также компилируется в бинарный файл. Эта структура содержит все сведения о программе, ее ресурсах и конфигурации устройств, упрощая процесс взаимодействия с аппаратными компонентами. Общая схема работы с машинным компилятором приведена на рис. 2.

Главные преимущества метода машинной компиляции заключаются в том, что он позволяет генерировать код, который исполняется непосредственно на процессоре, минуя виртуальную машину. Это позволяет обеспечить значительно более высокую производительность по сравнению с интерпретацией инструкций виртуальной машиной, так как программа может выполнять команды на уровне процессора. Однако, в этом подходе есть и свои минусы. Так, машинная компиляция требует значительных ресурсов на этапе компиляции, что может усложнить внесение изменений в код. Кроме

того, такая система будет менее гибкой и более зависимой от конкретной архитектуры оборудования, что усложняет переносимость программы на другие устройства.

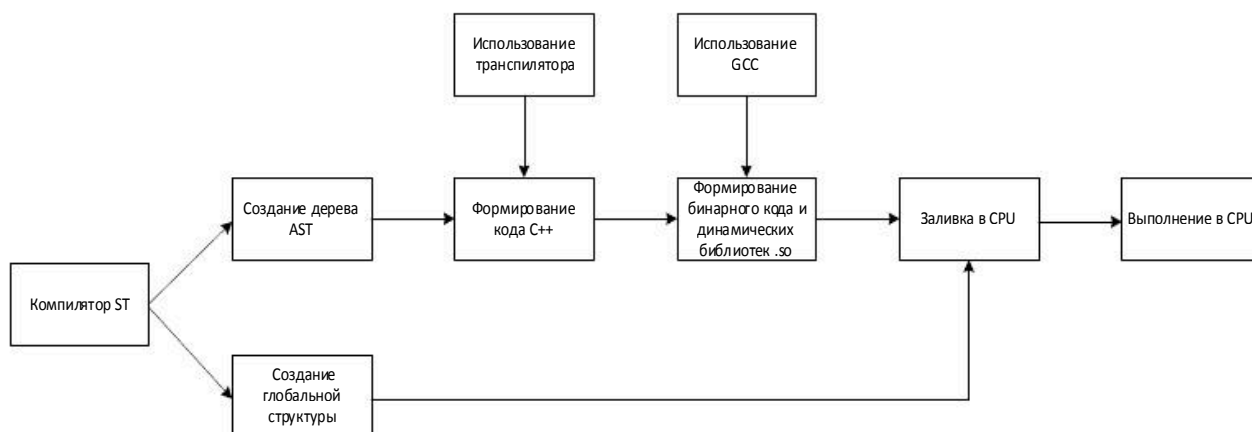


Рис. 2. Схема работы машинного компилятора

Преимущества машинной компиляции можно наблюдать на примере ПО, разрабатываемого в таких платформах, как CoDeSys и TwinCAT, где реализация программного обеспечения выполняется с минимальным количеством промежуточных слоев между кодом и аппаратным обеспечением, обеспечивая высокую производительность и экономию ресурсов.

Заключение. Выбор между рассмотренными методами зависит от специфики задач и требований приложения. Виртуализация обеспечивает универсальную совместимость с различными аппаратными платформами, что упрощает переносимость и повторное использование кода. Этот метод оптимален в ситуациях, где важны кроссплатформенность и эффективность обслуживания. Однако из-за дополнительного уровня абстракции виртуализация может накладывать ресурсоемкие требования и несколько снижать производительность. С другой стороны, машинная компиляция позволяет более тесно взаимодействовать с оборудованием, что обеспечивает высокую скорость выполнения и оптимизацию использования ресурсов. Это особенно важно для приложений, где критичны время отклика и производительность, как в случае контроллеров ПЛК 500, применяемых в сложных промышленных условиях. Несмотря на ограниченную кроссплатформенную гибкость, машинная компиляция предпочтительна, когда требуется надежность и максимальная производительность в реальном времени. Следовательно, для задач промышленной автоматизации, где стабильность и эффективность критичны, машинная компиляция чаще оказывается лучшим выбором. Однако в проектах, требующих адаптивности, при допустимых компромиссах по скорости выполнения, предпочтение можно отдать виртуализации.

- [1] Скоробогатко А.А. Методика объектного моделирования и разработки приложений для промышленных контроллеров по стандарту МЭК 61131. *Инновационные информационные технологии*, 2013, № 2–3, с. 373–381.
- [2] Галагуз Е.С., Пивненко М.Ю., Панов Р.С. Применение виртуальных машин. *Инновационное развитие техники и технологий в промышленности. Всерос. науч. конф. молодых исследователей с междунар. уч.: сб. матер.* Москва, Российский государственный университет имени А.Н. Косыгина (Технологии. Дизайн. Искусство), 2024, с. 231–234.
- [3] Зацаринная Ю.Н., Староверова Н.А. Виртуализация и виртуальные машины в подготовке современных ИТ специалистов. *Вестник Технологического университета*, 2015, т. 18, № 9, с. 247–250.
- [4] Феоктистов А.Г., Сидоров И.А., Горский С.А. Автоматизация разработки и применения распределенных пакетов прикладных программ. *Проблемы информатики*, 2017, № 4 (37), с. 61–78.
- [5] Колтунцев А.В., Золотарев С.В. ISAGRAF 5 — основа для создания распределенных приложений в среде программируемых контроллеров на базе стандарта IEC61499. *Промышленные АСУ и контроллеры*, 2008, № 12, с. 40–43.

Keywords: software, virtualization, machine compilation, performance, controllers, PLC 500