УДК 510.635 DOI: 10.30748/soi.2018.152.13

Н.В. Борисова, Л.В. Шабанова-Кушнаренко

Национальный технический университет "ХПИ", Харьков

ЭФФЕКТИВНОЕ УПРАВЛЕНИЕ РЕСУРСАМИ ВСТРОЕННЫХ СИСТЕМ ДЛЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ РЕАЛЬНОГО ВРЕМЕНИ

Современные встроенные системы должны работать во все возрастающих динамических средах, где прогнозирование вычислительной нагрузки на эти системы невозможно. В дополнение к ограничениям, предъявляемым к строгим ограничениям времени, большинство встроенных систем ограничены размером, весом, потреблением энергии и ограничениями по стоимости. В результате эффективное управление ресурсами является критическим аспектом во встроенных системах, которые необходимо учитывать на нескольких архитектурных уровнях. В статье проанализированы определяющие основные направления исследований в области встроенных вычислений в реальном времени. Предложен подход к динамическому распределению ресурсов, который может обеспечить улучшенную адаптивность и надежность системы.

Ключевые слова: *встроенные системы, динамическая среда, вычисления в реальном времени, динамическое распределение ресурсов.*

Введение

Современные встроенные системы [1–2] должны работать во все возрастающих динамических средах, где прогнозирование вычислительной нагрузки на эти системы невозможно. Тем не менее, своевременные ответы на события должны предоставляться в рамках точных временных ограничений, для гарантирования требуемого уровня производительности [3–4]. Следовательно, встроенные системы по самой своей природе демонстрируют характеристики в реальном времени, которые налагают дополнительный набор ограничений, кроме тех, которые применяются в типичной системе общего назначения. В дополнение к ограничениям, предъявляемым к строгим ограничениям времени, большинство встроенных систем ограничены размером, весом, потреблением энергии и ограничениями по стоимости [5]. В результате эффективное управление ресурсами является критическим аспектом во встроенных системах, которые необходимо учитывать на нескольких архитектурных уровнях.

В статье проанализированы определяющие основные направления исследований в области встроенных вычислений в реальном времени. Встроенные системы развиваются из одноцелевых высокоспециализированных средах обработки на более общие вычислительные платформы с открытым исходным кодом. Рассматриваются проблемы существующих подходов и обсуждаются новые направления исследований в операционных системах, планируются новые пути их преодоления.

1 Встроенные вычисления в реальном времени

Встраиваемые системы реального времени [6] следующего поколения развиваются из вычислительных автономных компонентов в большую часть сети взаимодействующих элементов между как вычислительными, так и физическими объектами. Эти новые системы, известные как кибер-физические системы (CPS), содержат набор вычислительных компонентов, которые контролируют или контролируют физические компоненты, такие как датчики. Поэтому для реализации удобства использования CPS необходимы дальнейшие исследования для повышения адаптивности и надежности существующих встроенных систем [7].

Чтобы повысить адаптивность и надежность, необходимо будет выделить различные ресурсы во время выполнения, а не во время разработки [8]. Ресурсы, такие как пропускная способность процессора или использование памяти, могут выиграть от стратегии динамического распределения ресурсов, которая основывает распределение на рабочем режиме или изменения окружающей среды в системе во время выполнения. Существует ряд сценариев, в которых встроенная система реального времени может извлечь выгоду из более динамичного подхода к распределению ресурсов. Рассмотрим автономный беспилотный аппарат (autonomous unmanned vehicle, AUV), где режимы работы определяются как полностью автономные, полуавтономные или дистанционные. Полностью автономный – это режим работы, в котором не требуется вмешательства человека. Полуавтономный режим требует вмешательства определенного уровня или оператора. Режим пульта дистанционного управления требует почти непрерывного ввода от оператора. В старых системах AUV, которые разработаны в соответствии со статическими методами выделения ресурсов, оператор должен переключаться между режимами работы для поддержки этих функций. Основная стратегия раз-

работки этих устаревших систем заключается в оптимизации распределения ресурсов в соответствии с режимом работы. Проблема с этим подходом заключается в том, что для принятия решений о размещении в компонентах приложения требуется принятие решений о распределении, а не общая система, которая препятствует адаптации. Результатом является обширная интеграция, и тестирование требуется при изменении системных возможностей или требований.

Динамическое распределение ресурсов [9] может обеспечить улучшенную адаптивность и надежность системы. Например, изменение режима от пульта до полностью автономного режима, где рабочие нагрузки сенсорного процесса могут значительно увеличиться. Использование динамического распределения может быть использовано для корректировки вычислительных ресурсов для управления повышенной обработкой, необходимой для обслуживания сенсорных данных. Таким образом, качество обслуживания может быть обеспечено для более важных задач, сохраняя при этом уровень обслуживания для менее важных задач.

Будущие встраиваемые системы, такие AUV – всего лишь один из примеров общей эволюции встроенных систем из статически настроенной, тесно связанной системы с более открытой исходной компонентой, работающей в менее детерминированной ситуации. В целом, киберфизические системы, такие как аэрокосмическая, транспортная, энергетическая, развлекательная, медицина и производство, становятся все более взаимосвязанными с другими системами реального времени и даже в режиме реального времени. Из-за желания реагировать на эту переменную нагрузку встраиваемая система реального времени развивается, чтобы быть более адаптивной во время выполнения. Эта эволюция мотивирует наши исследования адаптивными системами встроенного программного обеспечения реального времени, которые динамически распределяют ресурсы для обеспечения уровня обслуживания, основанного на изменениях окружающей среды.

2 Цель исследования

Цель наших исследований – разработать адаптивную схему планирования программного обеспечения, ориентированную на следующие аспекты адаптации [8]. Во-первых, необходимо определить подход к планированию на основе ресурсов, который управляет операциями в условиях перегрузки. Во-вторых, добиться более высокого разрешения существующих вычислений за счет изменения производительности ЦП, возникающих в результате изменения рабочей нагрузки. Наконец, увеличить способность обрабатывать смешанные задачи критичности с детерминированным и недетерминированным временем выполнения.

Системы с неоднородной критичностью в реальном времени должны работать в суровых условиях, где ожидается, что они будут выполнять подмножество своих критических функций в условиях сбоя. Появление сбоев классифицируется как разделяемые ресурсы, такие как процессоры, перегруженные и, следовательно, некоторые задачи будут упускать из виду их крайние сроки. Основная цель работы в этом тезисе – разработать подход, который управляет задачами систем реального времени, в частности, в условиях перегрузки, когда ошибка одной задачи (задач) не вызывает катастрофического коллапса системы в целом. То есть, во время номинальных условий система будет работать так, как если бы не было управления перегрузкой, в то время как адаптивная способность вступила бы в силу при условии перегрузки. Затем этот адаптивный подход сможет реагировать на возникновение события сбоя или перегрузки, а затем распределять ресурсы для задач на основе некоторых параметров качества обслуживания (Quality of Service, QoS). Использование этих параметров QoS – это то, что позволяет планировать задачи на основе их важности.

3 Состояние проблемы

Процессоры встроены во многие устройства, которые мы используем каждый день, такие как мобильные телефоны, планшеты, телевизоры, бытовая техника, автомобили и почти все, что вы можете подключить.

Учитывая распространение встроенных устройств следующего поколения в киберфизические системы, такие как интеллектуальная энергетическая сетка, ожидается, что тенденция сохранится и в будущем. В то время как встроенные устройства следующего поколения продолжают развиваться, они по-прежнему обладают многими свойствами, идентифицированными авторами в [10].

– *Ограниченные ресурсы*. Многие встроенные системы разработаны с ограничениями ресурсов, такими как требования к размеру, весу и мощности (SWaP). В результате многие встроенные системы имеют ограниченную вычислительную мощность и объем памяти по сравнению с системами общего назначения. Из-за этих типов ограничений еще более важно, чтобы доступные пугающие ресурсы управлялись как можно более эффективно.

– *Ограничения в режиме реального времени*. Встроенные системы обычно взаимодействуют с физической средой, как правило, в форме чувствительных устройств. Обработка в этих системах имеет строгие ограничения времени, чтобы реагировать на изменения в физической среде. Операционная

система отвечает за соблюдение этих ограничений времени и обеспечение предсказуемого поведения процесса.

– *Динамическое поведение*. Встраиваемые системы следующего поколения развиваются из однопоточной обработки в гораздо более сложные многопоточные приложения, где несколько процессов взаимодействуют и конкурируют за общие ресурсы. Для дальнейшего увеличения этой сложности приложения более тесно связаны с физической средой, что значительно затрудняет прогнозирование их обработки. Кроме того, современные аппаратные оптимизации, предназначенные для повышения общей производительности, такие как кеширование, конвейерная обработка, предварительная выборка и DMA, могут вводить недетерминированное поведение, делая оценку наихудшего времени выполнения очень непредсказуемой. В результате сочетание вариаций, возникающих в результате аппаратной оптимизации и изменяющейся рабочей нагрузки, делает статическое предсказание системного поведения намного сложнее.

Поскольку встроенное устройство следующего поколения разрабатывается больше как платформы общего назначения в сочетании с тем фактом, что они по-прежнему обладают теми же свойствами, что и унаследованные системы, представляют дополнительные проблемы на нескольких архитектурных уровнях. В прошлом типичный подход заключался в разработке систем, основанных на наихудших временах исполнения, но в новых высокодинамичных средах этот метод мог бы сбрасывать ограниченные ресурсы и увеличивать общую стоимость.

Более разумный подход состоял бы в том, чтобы система могла реагировать на изменяющуюся физическую среду, а не просто статически конструировать для наихудшего случая. Однако для этого требуется, чтобы система теперь была адаптивной, чтобы адаптироваться к динамической среде и изменить внутренние ответы, чтобы поддерживать производительность системы на желаемом уровне или, по меньшей мере, снизить производительность до приемлемого уровня. Чтобы обеспечить такой тип адаптации, изменения требуются на нескольких уровнях стека программного обеспечения. Модификации потребуются на уровне операционной системы, особенно в ядре, на уровне промежуточного программного обеспечения и, возможно, на уровне приложений. Однако, поскольку ядро ближе всего к оборудованию, было бы наиболее эффективным реализовать адаптацию на этом уровне. Кроме того, реализуя детали адаптации на ближайшем к слою ресурсов, уровни абстракционной парадигмы поддерживаются на более высоких уровнях, таких как прикладной уровень.

4 Требования к приложениям

Внедренные приложения развиваются из специальных однопоточных задач в многоцелевые многопоточные приложения со сложными взаимодействиями. Фактически, когда дело доходит до обработки рабочей нагрузки, встроенные системы становятся все более похожими на приложения общего назначения, за исключением того, что они, как ожидается, будут работать более надежно, с ограниченными ресурсами и вовремя. Одним из наиболее распространенных примеров является смартфон с большим количеством приложений и несколькими параллельными процессами. Поэтому крайне важно, чтобы программное обеспечение оптимизировало доступные ограниченные аппаратные ресурсы для повышения эффективности и поддержания надежности.

На рынке потребительской электроники авторы [10] классифицируют основные программные действия в отношении планирования и распределения ресурсов.

– *Программное обеспечение для управления*, которое обычно реализуется как периодические задачи и использует небольшую часть общих ресурсов. Примеры включают задачи управления, которые подвержены жестким временным ограничениям и должны быть гарантированы от линии во всех рабочих условиях.

– *Программное обеспечение для обработки мультимедиа*, как правило, управляется данными и основным потребителем доступных ресурсов (процессор или объем памяти). Основными примерами этого типа программного обеспечения являются мультимедийные приложения. Эти приложения из-за их больших потребностей в ресурсах и ограничений в режиме реального времени обычно предлагаются на уровне обслуживания. Например, если текущие доступные ресурсы недостаточны для воспроизведения видео высокой четкости, требования к качеству обслуживания (QoS) могут указывать на то, что видео можно обрабатывать с более низким разрешением. Таким образом, вместо того, чтобы не воспроизводить видео по требуемому уровню обслуживания, предоставляется хотя бы некоторый уровень обслуживания.

– *Интерактивное программное обеспечение* описывает приложения, которые реагируют на взаимодействие пользователя или операции. Эти приложения представлены как спорадические задачи, поэтому требования к срокам задач больше зависят от времени ответа, а не от времени. Примеры включают в себя видеоигры, интернет или фото / музыку, а также руководства по программированию по требованию для DVD-плеера.

Задача заключается в планировании такой разнообразной деятельности, чтобы эффективно управлять доступными ресурсами, так что каждое приложение, по крайней мере, обеспечивает минимальный уровень обслуживания. Кроме того, сложно масштабировать эти системы, поскольку простое добавление или удаление функций может привести к неустойчивости системы или даже к сбою. В результате в любое время добавляется новая функция или удаляется старая функция с исчерпывающим редизайном и тестированием. Нецелесообразно проектировать системы для всех возможных вариантов использования. Чтобы устранить чрезмерную переработку, необходимы новые инструменты для представления и управления потребностями в ресурсах в динамических системах.

В традиционных устаревших встроенных системах решение этих проблем заключалось в распространении разнообразных приложений на нескольких процессорах. Одному процессору могут быть назначены задачи управления; другому процессору назначаются задачи обработки мультимедиа, а другому назначены интерактивные задачи. Идея заключается в физическом разделении различных функциональных возможностей программного обеспечения. Распределение ресурсов и планирование упрощаются, в то время как обеспечивается защита, так что поведение приложения обработки мультимедиа не влияет на поведение приложения управления. Проблема с этими распределенными приложениями заключается в том, что им требуются специализированные механизмы синхронизации и связи, которые потребляют больше энергии и расширяют физический охват.

Другой альтернативой является создание единой системы ЦП (возможно, с несколькими ядрами), которая поддерживает защиту памяти и временную изоляцию, где можно безопасно комбинировать различные жесткие задачи реального времени и мягкие задачи реального времени. Результатом будет гораздо более масштабируемая система, в которой функциональная проверка может быть упрощена за счет устранения необходимости тщательной реорганизации и повторного тестирования.

5 Ограничения к существующим подходам

Приоритет задач является основным механизмом определения важности или критичности в системах реального времени. Существует ряд причин, по которым один параметр неадекватен при настройке сложных динамических встроенных систем. Одна из причин заключается в том, что динамические системные ограничения не могут быть сопоставлены с набором уровней приоритета.

Другая проблема с использованием приоритета для представления важности – это те же приложения, которые могут играть различную роль в зависимости от сценария. Например, рассмотрим систему AUV, упомянутую в предыдущей главе. Задачи руководства, навигации и управления (guidance, navigational and control, GNC) играют существенно разные роли в зависимости от того, находится ли автомобиль в полностью автономном или полуавтономном режиме. В этом сценарии гораздо сложнее группировать приложения по их режиму или функциональности, потому что глобальное свойство системных приоритетов будет нарушать эту группировку.

Текущий подход, используемый для обеспечения некоторой гибкости в динамических системах, заключается в изменении приоритетов задач во время выполнения. Проблема в том, что трудно предсказать, как система будет работать во время выполнения, когда приоритеты задач изменяются динамически. Например, увеличение приоритета задачи с длительным исполнением может привести к голоданию любых задач с более низким приоритетом. И наоборот, уменьшение приоритета задачи с высоким приоритетом может отрицательно повлиять на поведение задач с более низким приоритетом, неявно повышая приоритет задач с более низким приоритетом, что приводит к возможным условиям перегрузки. Задачи, совместно использующие ресурсы (например, семафоры), создают дополнительные проблемы в том, что приоритет задачи также может влиять на общее время блокировки или ресурса. Рассмотрим ситуацию, когда задача с высоким приоритетом и задача с низким приоритетом совместно используют ресурс, а ресурс был определен с использованием протокола приоритета наследования, чтобы избежать инверсии приоритета. Предположим, что приоритет задач, связанных с ресурсом, изменяется в неподходящее время, что может привести к серьезным побочным эффектам, таким как инверсия приоритета [11]. Эти примеры иллюстрируют, почему традиционные механизмы планирования не подходят для динамических адаптивных систем, поскольку операционные системы не обеспечивают явной поддержки управления качеством обслуживания (QoS).

6 Обеспечение контроля в реальном времени

Если приложение имеет различные требования к рабочей нагрузке или различное поведение выполнения, то теория управления обратной связью может использоваться для оценки текущей рабочей нагрузки, а затем соответствующим образом настраивать параметры задачи или резервирования [11]. Интеграция теории планирования и управления

в режиме реального времени является относительно новым подходом, который обеспечил некоторые перспективные исследования. Преимущество заключается в том, что управление обратной связью может использоваться ядром для настройки параметров приложения, так что резервирование ресурсов приложения может быть более адаптировано к непредсказуемым изменениям.

7 Иерархическое планирование

Иерархическое планирование – это структура, которая повышает производительность на современных компьютерах для разделения нескольких приложений. Этот подход заключается в разделении процессора на группу виртуальных машин, где каждому приложению (подсистеме) выделяется часть процессора. Мотивация заключается в том, что одна схема планирования не может быть идеальным отображением для разнообразного набора приложений, поэтому для одного и того же процессора может потребоваться использование разных алгоритмов планирования. Тем не менее, несколько алгоритмов планирования делают анализ планирования более сложным и могут потребоваться дополнительные теоретические работы для обеспечения гарантий планирования. Иерархическая структура планирования означает, что для каждой подсистемы существует только один алгоритм планирования. Подсистемы или приложения сгруппированы в соответствии с их критичностью или функциональностью и представлены как листья. Общая система представлена как корень иерархии. Поэтому каждая подсистема может иметь свой собственный планировщик, поэтому система состоит из иерархии планировщиков. Таким образом, корневой планировщик может обеспечить гарантированное резервирование ресурсов, временную изоляцию и легкость сложности анализа планируемости, поскольку каждая подсистема может рассматриваться как работающая на своем изолированном процессоре.

8 Управление перегрузкой

Для обеспечения предсказуемости в динамических системах необходимо контролировать входящую рабочую нагрузку, чтобы предотвратить условия перегрузки. Когда рабочая нагрузка превышает доступную пропускную способность процессора, система может быстро стать нестабильной и вызвать серьезную деградацию производительности. Вычислительная перегрузка может управляться с использованием различных методов.

– *Уровни QoS*. Некоторое программное обеспечение может быть выполнено с использованием разных алгоритмов с различными средами исполнения. Например, в некоторых случаях приложение может не требовать точных вычислений, поэтому вместо операций с плавающей запятой может использоваться целочисленная арифметика. В других случаях некоторые вычисления могут выполняться с использованием алгоритмов с более интенсивным вычислением, что приводит к различным уровням качества. Другими словами, рабочую нагрузку можно контролировать, используя разные уровни QoS.

– *Регулируемые временные ограничения*. В режиме реального времени рабочая нагрузка контролируется не только временем выполнения, но и временными ограничениями задачи. Поэтому рабочую нагрузку можно облегчить, ослабляя временные ограничения задачи. В случае условия перегрузки крайний срок или период задания могут быть уменьшены или увеличены [12].

– *Контроль приема*. Другим способом управления рабочей нагрузкой во время перегрузки является контроль количества задач, которые фактически разрешены для выполнения. Этот метод облегчает условие перегрузки, отклоняя задачу или задачи, которые планируется выполнить.

Выводы

За последние 30 лет встроенные системы демонстрировали экспоненциальный рост во многих областях приложений как по количеству, так и по сложности. Удивительно, однако, что такой рост сложности не сопровождался соответствующей эволюцией программного обеспечения управления, используемого для управления вычислительными ресурсами, что существенно похоже на то, которое было принято в начале 70-х годов. Фактически, действия приложений по-прежнему обрабатываются циклическими руководителями или, в лучшем случае, ядрами с фиксированным приоритетом. Проблема заключается не в отсутствии альтернатив, а в том, что никто не смог сделать убедительный пример для перехода. Каждая попытка повысить уровень абстракции включала неприемлемые наказания с точки зрения памяти и скорости. Также с точки зрения промышленности поддержка устаревшего кода часто была слабой.

Возможности для встроенных систем развиваться и становиться более надежными, хотя и более сложными, в какой-то степени зависят от того, что могут предложить операционные системы следующего поколения в реальном времени и инструменты реализации. Задача состоит в том, как реализовать приложения, которые могут эффективно выполняться на ограниченных ресурсах, для удовлетворения нефункциональных требований, таких как своевременность.

Надлежащее управление ресурсами и качеством обслуживания позволит внедрить внедренные системы, которые являются более гибкими, но более детерминированными, чем это возможно сегодня.

Поскольку такие системы будут лучше определены, их свойства также будут проверяться более легко. Поддерживая явное распределение ресурсов и качество обслуживания, разработчики системы вернут контроль над системой, которую они настроили на проектирование.

Чтобы эффективно распределять системные ресурсы между приложениями и обеспечивать предсказуемость и гибкость, вопросы следует дополнительно исследовать. На более высоком уровне абстракции для получения гибких систем следует использовать протоколы для управления уровнями качества и подходящими архитектурами. На более низком уровне необходимо продолжить работу над алгоритмами управления ресурсами, новыми целевыми моделями, алгоритмами управления доступом, мониторинга и адаптации.

Список литературы (References)

1. Davis, R.I. and Burns, A. (2006), Resource Sharing in Hierarchical Fixed Priority Pre-emptive Systems, *Proc. of the 27th IEEE International Real-Time Systems Symposium (RTSS'06)*.

2. Lipari, G. and Baraugh, S.K. (2000), Efficient scheduling of real-time multi-task applications in dynamic systems, *Proc. 6th IEEE Real-Time Technol. Appl. Symp. (RTAS'00)*, pp. 166-175.

3. Behnam, M., Nolte, T., Sjodin, M. and Shin, I. (2007), SIRAP: A synchronization protocol for hierarchical resource sharing real-time open systems, *Proc. 7th ACM and IEEE Int. Conf. Embedded Software (EMSOFT 07)*.

4. Behnam, M., Nolte, T., Sjodin, M. and Shin, I. (2010), Overrun Methods and Resource Holding Times for Semi-Independent Real-Time Systems, *IEEE trans. on Indus. Informatics*.

5. Buttazzo, G.C. (2011), *Hard Real-Time Computing Systems, Predictable Scheduling Algorithms and Applications*. Springer, Real-Time System Series, 2011.

6. Chandra, A. and Shenoy, P. (2013), Hierarchical Scheduling for Symmetric Multiprocessor, *IEEE Trans. on Parallel and Distributed Systems*.

7. Buttazzo, G., Lipari, G., Abeni, L. and Caccamo, M. (2005), *Soft Real-Time Systems: Predictability vs. Efficiency*, Springer.

8. Kelly O., Aydin H. and Zhao, B. (2011), On Partitioned Scheduling of Fixed-Priority Mixed Criticality Task Set, *TrustCom 2011*.

9. Buttazzo, G., Lipari, G., Caccamo, M. and Abeni, L. (2002), Elastic Scheduling for Flexible Workload Management, *IEEE Transactions on Computers*, Vol. 51, No. 3, pp. 289-302.

10. Buttazzo, G. (2006), Research trends in real-time computing for embedded systems, *ACM SIGBED Review 3.3*.

11. Stankovic, J.A., Lu, C., Son, S.H. and Tao, G. (1999), The Case for Feedback Control Real-Time Scheduling, *Proceedings of the 11th EuroMicro Conference on Real-Time Systems (ECRTS 1999)*, York, UK.

12. Lu, C., Stankovic, C.J., Son, S. and Tao, G. (2002), Feedback control real-time scheduling: Framework, modeling and algorithms, *Real-Time Systems*, Vol. 23, pp. 85-126.

Поступила в редколлегию 15.01.2018
Одобрена к печати 20.02.2018

Відомості про авторів:

Борисова Наталя Володимирівна

кандидат наук доцент кафедри
Національного технічного університету
"Харківський політехнічний інститут"
Харків, Україна
https://orcid.org/0000-0002-8834-2536
e-mail: borisova_nv@mail.ru

Шабанова-Кушнаренко Любов Володимирівна

кандидат наук асистент кафедри
Національного технічного університету
"Харківський політехнічний інститут"
Харків, Україна
https://orcid.org/0000-0002-2080-7173
e-mail: l.v.shabanova.kushnarenko@gmail.com

Information about the authors:

Natalya Borisova

Candidate of Sciences Associate Professor
of Department of National Technical University
"Kharkiv Polytechnic Institute"
Kharkiv, Ukraine
https://orcid.org/0000-0002-8834-2536
e-mail: borisova_nv@mail.ru

Lyubov Shabanova-Kushnarenko

Candidate of Sciences Assistant Lecturer
of Department National Technical University
"Kharkiv Polytechnic Institute"
Kharkiv, Ukraine
https://orcid.org/0000-0002-2080-7173
e-mail: l.v.shabanova.kushnarenko@gmail.com

ЕФЕКТИВНЕ УПРАВЛІННЯ РЕСУРСАМИ ВБУДОВАНИХ СИСТЕМ ДЛЯ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ РЕАЛЬНОГО ЧАСУ

Н.В. Борисова, Л.В. Шабанова-Кушнаренко

Сучасні вбудовані системи повинні працювати у все зростаючих динамічних середовищах, де прогнозування обчислювального навантаження на ці системи неможливо. На додаток до обмежень, що пред'являються до суворих обмежень часу, більшість вбудованих систем обмежені розміром, вагою, споживанням енергії і обмеженнями за вартістю. В результаті ефективне управління ресурсами є критичним аспектом у вбудованих системах, який необхідно враховувати на кількох архітектурних рівнях. У статті проаналізовано визначають основні напрямки досліджень в області вбудованих обчислень в реальному часі. Запропоновано підхід до динамічного розподілу ресурсів, який може забезпечити поліпшену адаптивність і надійність системи.

Ключові слова: *вбудовані системи, динамічне середовище, обчислення в реальному часі, динамічний розподіл ресурсів.*

EFFICIENT MANAGEMENT OF RESOURCES OF BUILT-IN SYSTEMS FOR COMPUTER TECHNOLOGY OF REAL TIME

N. Borisova, L. Shabanova-Kushnarenko

Modern embedded systems must work in ever increasing dynamic environments, where it is impossible to predict the computational load on these systems. In addition to the limitations imposed on strict time constraints, most embedded systems are limited in size, weight, energy consumption and cost constraints. As a result, efficient resource management is a critical aspect in embedded systems that need to be considered at several architectural levels. The article analyzes the main directions of research in the field of embedded computing in real time. Embedded systems evolve from single-purpose highly specialized processing environments to more general open source computing platforms. To increase the adaptability and reliability, it will be necessary to allocate various resources at runtime, rather than during development. Resources such as processor bandwidth or memory usage can benefit from a dynamic resource allocation strategy that bases distribution on the operating mode or changes in the environment in the system at runtime. The problems of existing approaches are considered and new directions of research in operating systems are discussed, new ways of their overcoming are planned. An approach is proposed to the dynamic allocation of resources, which can provide improved adaptability and reliability of the system. Proper management of resources and quality of service will allow the introduction of deployed systems that are more flexible, but more deterministic than is possible today.

Keywords: *embedded systems, dynamic environment, real-time calculations, dynamic resource allocation.*